CHAPTER 1

Architecting Information Infrastructures for Security

Cliff Wang

The current Internet started with an implicit assumption of trust among the parties making up the infrastructure of the network. This assumption might have been valid during the earlier days of the Internet when it was a self-policing organization of academic, military, and corporate users, but it began to falter when millions of new users were added to the Internet on a yearly basis and the Internet became an unregulated territory. Today, hackers of various levels of sophistication freely exploit the vulnerabilities and weakness of the Internet. Numerous large-scale attacks and intrusions have resulted in billions of dollars of damage. Maintaining the security of our information infrastructure has become a top priority in recent years.

There have been constant efforts to retrofit security into the existing Internet. However, such approaches have largely been hampered by the fundamentally open nature of the Internet architecture and the lack of good security principles in the existing Internet architecture. There is a critical need to develop a novel architecture that allows robust, thorough security to be easily implemented on a network designed with different assumptions than those of the existing Internet. The next generation information and communications infrastructure must provide authentic, accurate, secure, and reliable services, even under a full range of threats or attacks. These qualities must be assured across a heterogeneous information infrastructure that provides interconnectivity via both wired and wireless links, at a wide range of link speeds. This new architecture will enable many emerging applications such as mobile computing or virtual office. In this new architecture, an individual user could rely on networking technologies to perform various tasks (from business transactions to research experiments) at any place and time, with the confidence that a high level of information assurance will be guaranteed.

For the next generation communication infrastructure, managing security and maintaining trust are critical challenges. Contrary to the initial assumption of trust in place, the new network architectures will operate on an implicit assumption of mistrust. The architecture will enable the trust level to be dynamically modified over time based upon the characteristics and behaviors of the network itself. The network will have the capability to operate in degraded mode at all times, while trading off performance with security continually in response to the threat environment.

In the past decade, we have experienced an explosive growth in both the scale of the infrastructure and the speeds of the networks. In addition, mobile devices are widely in use today with wireless connections to the Internet. Maintaining a high level of security for the infrastructure is an enormous technical challenge that demands the development of new capabilities in many areas. For example, innovative approaches and techniques to defend against sophisticated intrusions have always been a top priority. Currently available techniques have severe limitations. For example, they:

- Can detect only the most common attacks,
- Do not adapt to changing conditions,
- Inadequately detect sophisticated multi-stage, multi-level attacks against network infrastructure elements,
- Do not perceive indirect attacks,
- Exhibit high false alarm rates.

We need both host-based and network-based defenses against attacks and intrusions. These require the development of breakthrough technologies in the areas of anomaly based detection, correlation, fusion methods, adaptive response mechanisms, and automatic generation of responses. Furthermore, new techniques and tools are required to scale to very high-speed,

dynamic networks and to provide a real-time or near real-time, network-wide capability to accurately detect, analyze, and respond to intrusions, intrusion attempts, suspicious network activities, and anomalies.

Military tactical communication relies on a dynamic and mobile wireless infrastructure to support a heterogeneous mixture of individual soldiers, ground vehicles, airborne platforms, unmanned aerial vehicles, robotics, and unattended sensor networks. A tactical mobile ad hoc network works in a challenging environment of noisy wireless channels, high mobility of individual nodes, and a mobile network infrastructure. Tactical networks also need to perform self-configuration and dynamic addressing while maintaining interoperability with non-wireless infrastructures. Techniques designed for fixed infrastructures may not work for highly mobile, wireless, high-bit error-rate communications. Protection techniques for this type of wireless infrastructure must be resilient and fault tolerant against communication channel interruption, data loss, or even malicious attacks or intrusions.

To support critical missions of network-centric operations and to achieve information superiority, the next generation Internet must be a high-performance network infrastructure that can serve as a reliable, high-capacity information backbone to make information resources readily available. This requires that the infrastructure provide a quality-of-service (QoS) guarantee in terms of timeliness, precision, and accuracy. To establish and maintain a high level of QoS assurance, the next generation architecture needs to incorporate advanced techniques and technologies that can effectively and intelligently monitor and allocate the available resources and meet QoS requirements dynamically and at different levels.

In this chapter, we have contributed technical papers from six Critical Infrastructure Protection (CIP) MURI teams working on the following aspects of next-generation Internet and secure information infrastructure:

- Communication infrastructure protection, defense, and response,
- Service assurance of communication infrastructure,
- Resiliency and robustness of communication infrastructure,
- Highly assured mobile communication systems,
- Distributed system security and assurance.

The paper "Overview of the ASRDI (Architectures for Secure and Robust Distributed Infrastructures) Project" discusses the development of new theoretical tools for analysis of the dynamics of large-scale networked infrastructures. The Stanford MURI team focused on unifying mathematical principles, ideas, and frameworks from communications, controls, computation, and dynamical systems theory, with an emphasis on attempting to reduce the use of ad hoc methods and capture more fundamental notions of limits on performance. Another paper, "Quality of Service Assurance for Dependable Information Infrastructures," summarizes the work carried out by the Arizona State University researchers on providing QoS service assurance at different levels, ranging from local service assurance to global service assurance, along with security assurance for the next-generation communication infrastructure.

The paper "Summary of the Hi-DRA Project: A System for High-Speed, Wide-Area Network Detection, Response, and Analysis" summarizes the University of California Santa Barbara MURI project on developing a network surveillance, analysis, and response infrastructure for high-speed, wide-area networks, whereas the paper "Anomaly and Misuse Detection in Network Traffic Streams—Checking and Machine Learning Approaches" presents the University of Pennsylvania MURI team's latest work on building effective defenses against malware, including both network-based and host-based approaches.

The MURI team at University of Maryland has been working on developing innovative distributed methods and algorithms to secure wireless communication. The paper "Distributed Immune Systems for Wireless Networks Information Assurance" summarizes their scientific achievements. The MURI project at Carnegie Mellon University has been focusing on advancing security in distributed systems. The paper "Distributed System Security via Logical Frameworks" provides a detailed description of the application of logical frameworks to distributed systems.

Overview of the ASRDI (Architectures for Secure and Robust Distributed Infrastructures) Project*

Sanjay Lall, Aeronautics and Astronautics, Stanford
Carolyn Beck, General Engineering, UIUC
Stephen Boyd, Electrical Engineering, Stanford
John Doyle, Control and Dynamical Systems, Caltech
Geir Dullerud, Mechanical Engineering, UIUC
Chris Hadjicostis, Electrical and Computer Engineering, UIUC
Muriel Medard, Electrical Engineering and Computer Science, MIT
Balaji Prabhakar, Electrical Engineering and Computer Science, Stanford
Rayadurgam Srikant, General Engineering UIUC
George Verghese, Electrical Engineering and Computer Science, MIT

Abstract

The major barrier constraining the successful management and design of large-scale distributed infrastructures is the conspicuous lack of knowledge about their *dynamical* features and behaviors. Up until very recently analysis of systems such as the Internet, or the national electricity distribution system, have primarily relied on the use of non-dynamical models, which neglect their complex, and frequently subtle, inherent dynamical properties. These traditional approaches have enjoyed considerable success while systems are run in predominantly cooperative environments, and provided that their performance boundaries are not approached. With the current proliferation of applications using and relying on such infrastructures, these infrastructures are becoming increasingly stressed, and as a result the incentives for malicious attacks are heightening. The stunning fact is that the fundamental assumptions under which all significant large-scale distributed infrastructures have been constructed and analyzed no longer hold; the invalidity of these non-dynamical assumptions is witnessed with the greater frequency of catastrophic failures in major infrastructures such as the Internet, the power grid, the air traffic system, and national-scale telecommunication systems.

This project is about network, reliability and robustness in large-scale systems. The major vision of this program is ubiquitous: we have distributed computing and information and would like to link these via secure communications to allow coordination of limited resources to achieve

*This research was supported by AFOSR DoD award number 49620-01-1-0365

global objectives that can be both predicted and guaranteed. The objective is to ensure that incorrect local decisions, due to dynamical effects, do not cause large-scale failures.

To address the challenges posed by dynamical behavior of large-scale network infrastructures, we bring to bear the tools and techniques of control theory together with those from communication networks and queuing theory. In particular, the algorithms and analytical approaches of control used for developing control strategies and logic are combined with protocol design methods to construct new, secure architectures for distributed networks. We focus on the dominant issues of complex dynamic behavior, local rather than global information and state, distributed rather than centralized decision making, secure, robust performance in an uncertain environment, and dynamic network connectivity.

1 Introduction

The objective of this project is to advance the security and reliability of large-scale network infrastructures. The research is specifically targeted at the most critical medium- and long-term security and performance issues facing current and future military networks, as well as homeland installations. The program is focused both on fundamental scientific understanding of the mechanisms by which single attacks can lead to catastrophic cascading failures caused by dynamic effects of propagation, misprioritization and instability, as well as development of new protocols which eliminate such vulnerabilities. The research of this URI is already having significant impact on the commercial sector, affecting the router designs of Cisco, and the protocol designs of Microsoft and Nokia, as well as the open protocols behind TCP and AQM.

The program 1) analyzes the decisions made by routers and layer protocols to see how they lead to network-level consequences; 2) studies the propagation dynamics of the network to characterize instabilities; 3) has developed protocols that eliminate these instabilities; 4) studies the measurable indicators of high-throughput data streams that can be used to detect attempted attacks in real-time and monitor network performance; 5) formulates new methods, such as combining routing and coding over multiple paths, that make certain classes of attack much more difficult; 6) develops information-based randomized algorithms that prevent attacks which depend on distributed, multi-source simultaneous attack and response; 7) re-examines the basic protocols of the network to suggest modifications that can be incrementally deployed, without requiring all users to simultaneously change software systems; 8) understands the parallels and distinctions between data networks and transportation and energy networks, noting that each of these three infrastructures has witnessed large-scale catastrophic failures of similar nature in recent years; and 9) has worked on a fundamental rethinking of how such networks function, to guide designers of the next generation of systems.

Traditional approaches to increasing network security have primarily focused on protecting the integrity of nodes at the edge of the network rather than systematic and robust design of the network

itself. In these traditional approaches the emphasis has been on improving encryption, key-exchange protocols and intelligent attack detection, so as to achieve nodes with extremely fortified defenses. This does not however protect against the catastrophic failures we have witnessed in recent years on the Internet and the National Power Grid, where nodes misinform each other, or under- or over-react to remote events, causing large-scale cascading failures. In current networks knowledge of which nodes to rely upon to make mission-critical decisions is being passed through and fed back via increasingly long chains of intermediaries, each of which inherently decreases the reliability of the global system. A secure layer is important, but in order to have more than a secure layer on top of a fragile foundation layer it is necessary to design security and robustness into the system interactions. The Internet requires end-user protocol-based cooperation, and the Power Grid requires multiple control centers; the Stanford URI is working on architectures which require neither of these limitations. In particular, this URI program is developing sophisticated mathematical models and systematic tools, and using them not only for post-mortem analysis of attacks and failures but also to design and implement new protocols which are resistant to these modes of failure.

This work is undergoing transition to current communication networks, and will have significant application to future military mixed wired and wireless command and control networks. The focus is on attacks at the network infrastructure level, not on attacks on the computers at the edge of the network.

2 Congestion and Buffering in Wired networks

We have studied the problem of designing globally stable, scalable congestion control algorithms for the Internet. Prior work primarily used linear stability as the criterion for such a design. Global stability has been studied only for single node, single source problems. In our work, we have obtained conditions for a general topology network accessed by sources with heterogeneous delays. We obtain a sufficient condition for global stability in terms of the increase and decrease parameters of the congestion control algorithm and the price functions used at the links.

The key idea in our recent work is to first show that the source rates are both upper and lower bounded, and then use these bounds in Razumikhin's theorem to derive conditions for global stability. However, a stumbling block in extending earlier results to a general network is the difficulty in obtaining reasonable bounds on the source rates and in finding an appropriate Lyapunov-Razumikhin function. We take a significant step in this direction by finding a Lyapunov-Razumikhin function that provides global stability conditions for a general topology network with heterogeneous delays.

The global stability condition derived thereby is delay-independent, and is given in terms of the increase and decrease parameters and a parameter of the price function. When the condition holds, the network is globally stable for all values of fixed communication delays and controller gains. It is

different from most prior works, where the conditions are given in term of the gains and the delays. Since our global stability condition is delay-independent, the network is robust to the delays and the gains used by users in the network. On the other hand, our stability condition restricts the possible choices for the utility functions and the price functions, whereas earlier stability conditions like work for general utility functions. Characterizing the stability region when our condition is violated, but the local stability condition still holds, is still an open problem. Our simulation results indicate that the region of attraction could be large under such a scenario.

We show that one can obtain conditions for global stability that relate the parameters of the congestion algorithm to the parameters of the price functions used at the links of the network. We further considered a two-phase algorithm, with a slow-start phase followed by a congestion-avoidance phase, as in today's version of TCP-Reno, and showed that a three-phase approximation of this two-phase algorithm is still globally, asymptotically stable under the same conditions on the congestion control parameters.

2.1 Sizing issues in buffer routers

Large buffers in Internet routers often limit achievable throughput, requiring the use of off-chip DRAMs. A standard guideline used for buffer size design is $B = RTT \times C$, where C is the capacity of the link and RTT is the round trip time. Recently, this design rule has been questioned and the use of small buffer routers was validated based on statistical multiplexing effects.

However, these results have been based on *static* network simulations with *fixed number of flows*. In our work, we have completed far more extensive network simulations evaluating the accuracy of these results in a *dynamic* environment where file flows arrive and depart, i.e., flow numbers are not fixed. More specifically, we assess the performance of dynamic networks with very small buffers, with the end-user in mind. As flows arrive and depart, link utilization should not be considered as the most important factor in the design of the network. In a static network, where the number of flows is fixed, utilization and goodput have a direct correspondence as each user sees an average throughput of $\frac{C \times U}{N}$. In a dynamic network, the number of flows is time-varying: there is no such correspondence between the link utilization and the end-to-end throughput. Therefore, we directly calculate end-to-end throughput seen by the users and use this as a metric for evaluating performance.

For completeness we first completed simulations with fixed number of long flows. We then showed that in this case smaller buffers can indeed be used without any significant effects on throughput. We then further showed that Poisson pacing of TCP is not necessary. In fact, our simulations demonstrate that the effect of short flows and RTT variations create sufficient randomness to ensure high link utilization.

The current Internet consists of extremely fast core routers and slow edge routers. The edge routers switch packets at a rate which is several orders of magnitude smaller than the core routers.

Our simulations show that in this case, very small buffers can be used without affecting the throughput, even in a dynamic network.

We have also considered the case where edge routers switch packets at a rate comparable to that of core routers, that is, there are no access bandwith constraints. In this case, our simulation results demonstrate that if the network is moderately congested, then increasing the buffer size will in fact result in a substantial increase in overall throughput. As an example, when the load on a 100Mbps link is approximately 80%, we find an increase in average throughput of between 60%-100% as the buffer size is increased from 20 to 1000 packets. We have now showed that TCP-pacing does not improve average throughput significantly.

Considering the same architecture as in the preceding discussion, we show that under mildly loaded conditions (load less than 50%), increased buffer sizes do not lead to increased throughput. That is, small buffers are adequate only when the core router is guaranteed to operate at 50% load, or less. In summary, in contrast to previous work, our simulations show that actual performance of routers with small buffers depends on the type of Internet architecture assumed.

2.2 Fluid model development and analysis of priority processing schemes in the Internet

Previous simulation studies have shown that providing a simple priority to short flows in Internet routers can dramatically reduce their mean delay while having little impact on the long flows that carry the bulk of the Internet traffic. We have proposed simple fluid models that can be used to quantify these observations. These fluid models are justified by showing that stochastic models of resource-sharing among TCP flows converge to these fluid models when the router capacity and the number of users is large.

We showed that a Shortest Remaining Processing Time (SRPT) scheme dramatically improves short flow performance, while having little impact on long flows. This scheme requires the router to estimate whether the flow is short or long, which is not feasible given that routers do not have access to per-flow information. Alternatively, using simple sampling techniques, it can be determined fairly accurately whether a flow is long or short. Assuming such a mechanism exists and can be easily implemented, we evaluate the performance of such priority processing schemes analytically, and further strengthen our conclusions via simulations.

Without priorities, the nature of bandwidth sharing in the Internet favors long flows. A more equitable sharing discipline can be approximated by discriminatory processor sharing (DPS). Stochastic analysis of DPS is extremely difficult and closed-form solutions exist only for exponentially distributed service times. However, in a system with a large number of files and a large server capacity, such as the Internet, some form of the law of large numbers can be applied and the resulting stochastic system can be approximated by a deterministic system that can be modeled by a set

of differential equations. We have proposed such fluid flow models to capture the resource sharing character of TCP flows in the Internet. These models consider the impact of access bandwidth constraints. Using these models, we showed analytically that a stochastic model for DPS converges to the fluid limit in a large system. This further characterizes the speed of convergence of the fluid limit to equilibrium.

2.3 Connection-level stability analysis in the Internet

In this work, we have studied connection-level models of file transfer requests in the Internet, where connection arrivals to each route occur according to Poisson processes and the file-sizes have phase-type distributions. We use Sum-of-Squares techniques to construct Lyapunov functions statisfying Foster's condition for stochastic stability.

3 Scheduling and Resource Allocation in Wireless Networks

3.1 A Large Deviations Analysis of Scheduling

In [37] we consider a cellular network consisting of a base station and N receivers. The channel states of the receivers are assumed to be identical and independent of each other. The goal is to compare the throughput of two different scheduling policies (a queue-length-based policy and a greedy scheduling policy) given an upper bound on the queue overflow probability or the delay violation probability. We consider a multi-state channel model, where each channel is assumed to be in one of L states. Given an upper bound on the queue overflow probability or an upper bound on the delay violation probability, we show that the total network throughput of the queue-length-based policy is no less than the throughput of the greedy policy for all N. We also obtain a lower bound on the throughput of the queue-length-based policy. For sufficiently large N, the lower bound is shown to be tight, strictly increasing with N, and strictly larger than the throughput of the greedy policy. Further, for a simple multi-state channel model (on-off channel), we prove that the lower bound is tight for all N.

Multiuser wireless scheduling has received much attention in recent years. Consider a cellular network consisting of a base station and N users (receivers), where the base station maintains N separate queues, one corresponding to each user. Assume time is slotted and the channel states of the receivers at each time slot are known at the base station. Then, the base station can decide which queues to serve according to their channel states. We considered the case where the base station operates in a TDMA fashion, i.e., the base station can serve only one queue in each time slot. Two scheduling policies have been widely studied in the literature: (i) the base station serves the user with the best (weighted) channel state (opportunistic scheduling) [34, 19]; or (ii) serve the one with the best queue-length-weighted channel state (queue-length based (QLB) scheduling)

[31, 11, 26, 27, 7, 4, 21]. While the QLB scheduling is throughput optimal (i.e., can stabilize any set of user throughputs that can be stabilized by any other algorithm), opportunistic scheduling maximizes the total network throughput if all queues are continuously backlogged. If the arrival rates to the users are identical and the channel state distributions to the receivers are identical, then these two scheduling policies have the same stability region.

While stability is the first concern of scheduling policies, quality-of-service (QoS) is equally important in applications. For example, we may require the queue overflow probability to be small or require small delays. The performance of different scheduling policies under QoS constraints has received much attention recently. For reasons of analytical tractability, much of the prior work assumes that the channels to all the receivers are independent and statistically identical. Under this assumption, and assuming identical user utilities, opportunistic scheduling policies become greedy policies in which the base station transmits to the receiver with the best channel state. In [25], the author studies a simple network consisting of two users where the channels are assumed to be independent, identically distributed ON-OFF channels. Using large-deviations techniques, it is shown that the total network throughput of the QLB policy is larger than the throughput of the greedy policy under the queue overflow constraint. In [11], a wireless network with N users and ON-OFF channels is considered. It is assumed that the arrivals are identical and Poisson, and the capacity when the channel is ON is one packet per time slot. It is then shown that, when the number of users increases from N to $2N$, the expected sum of queue lengths is non-increasing under the QLB policy, while it increases linearly under the greedy policy. Further, in [8], the behavior of the greedy policy for Rayleigh fading channels is studied and it is shown that under a delay constraint, the total network throughput of the greedy policy increases initially with the number of users, but eventually decreases and goes to zero when the number of the users is sufficiently large.

Motivated by these prior results, in our work reported in [37], we study the performance of the two scheduling policies (greedy and QLB) for a wireless network with multi-state channels and constant arrivals. Using sample-path large-deviations techniques that have been used in [5], [25] and [29], we obtain the following results:

1. Assuming a multi-state channel model and a constant arrival rate in each time slot, under the QLB policy, we compute a lower bound on the large-deviations exponent of the probability that at least one queue in the network exceeds a large threshold. We obtain lower bounds on the maximum network throughput under the QoS constraints, and for large N, the lower bounds are tight, strictly increasing, and strictly greater than the throughput of the greedy policy. For the ON-OFF channel model, we prove that the lower bounds are tight for all N. It was conjectured that in [25] that, for the ON-OFF channel model, the complexity of the calculation of the large-deviations exponent increases exponentially with increasing N, but we show here that a simple closed-form expression can be obtained.

2. Consider ON-OFF channels and the QLB policy. In [11], under the assumption that the channel capacity is one packet per time slot, for a different model, it is shown the expected sum of the queue lengths is nondecreasing when the number of users increases from N to $2N$. For the ON-OFF channel model, we show that the maximum network throughput is strictly increasing in N under the delay-violation constraint or queue overflow constraint. Our result does not only compare performance with N users and $2N$ users, but at all intermediate values as well. Our result also holds even when the capacity of the network is greater than one packet-per-slot. Further, for the general multi-state channel model, the maximum throughput is shown to be strictly increasing with N for large N.

3. For the greedy policy, we analytically show that the throughput goes to a constant under the queue overflow constraint, and decreases to zero under the delay violation constraint. This result holds for the general multi-state channel model, and is consistent with the numerical results for Rayleigh fading channels in [8].

4. Under the QoS constraints, we show that the throughput of the QLB scheduling policy is no less than the throughput of the greedy policy. This conclusion was also obtained in [25] for a two-user system and under the queue overflow constraint. Here, we prove that it is true for networks with N users ($N \geq 2$) and multi-state channels.

3.2 Distributed Fair Resource Allocation in Cellular Networks in the Presence of Heterogeneous Delays

In [38] we consider the problem of allocating resources at a base station to many competing flows, where each flow is intended for a different receiver. The channel conditions may be time-varying and different for different receivers. It has been shown in [8] that in a delay-free network, a combination of queue-length-based scheduling at the base station and congestion control at the end users can guarantee queue-length stability and fair resource allocation. We extended this result to wireless networks where the congestion information from the base station is received with a feedback delay at the transmitters. The delays can be heterogenous (i.e., different transmitters may have different round-trip delays) and time-varying, but are assumed to be upper-bounded, with possibly very large upper bounds. We showed that the joint congestion control-scheduling algorithm continues to be stable and continues to provide a fair allocation of the network resources.

We have studied the problem of fair allocation of resources in the downlink of a cellular wireless network consisting of a single base station and many receivers (see Figure 1). The data destined for each receiver is maintained in a separate buffer. The arrivals to the buffers are determined via a congestion control mechanism. We assume that the time is slotted. The channels between the base station and the receivers are assumed to have random time-varying gains which are independent from

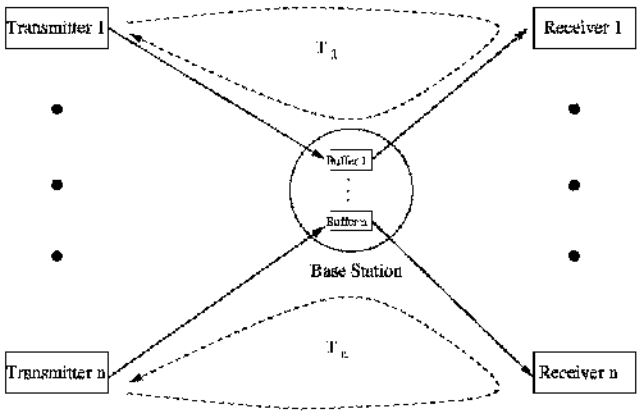


Figure 1: Network with feedback delays. The channel from the base station to the receivers is time-varying.

one time-slot to the next. The independence assumption can be relaxed easily, but we use it here for ease of exposition. The goal is to allocate the network capacity fairly among the users, in accordance with the needs of the users, while exploiting the time-variations in the channel conditions. We associate a utility function with each user that is a concave, increasing function of the mean service that it receives from the network. In an earlier paper [8], it was shown that a combination of Internet-style congestion control at the end-users and queue-length based scheduling at the base station achieves the goal of fair and stabilizing resource allocation. This result is somewhat surprising since the resource constraints in the case of a wireless network are very different from the linear constraints in the case of the Internet [28]. The relative merits of congestion control-based resource allocation scheme as compared to other resource allocation schemes for cellular networks are discussed in [8]. Several other works in the same context are [30, 18, 20]. However, none of these works explicitly include the effect of feedback delay in their analysis. One of the reasons that delay is not important in these other works is that a specific scheduling algorithm is used in the network which allows the congestion control to be based only on the queue length at the entry node of each source. However, we considered a situation where such scheduling is not used and where the bottleneck is at the cellular network while the sources may be located far away from the base station. An example of such a situation is a file transfer from a remote host over the downlink of a cellular network. We aim to consider the effect of this essential parameter on the fairness and stability properties of the algorithm presented in [8].

In [8], it is assumed that there are no delays in the transmission of packets from an end-user (transmitter) to the base station and in the transmission of congestion information from the base station back to the end users. But if we consider the case where the end users are connected to the base station through the Internet, then delays exist in both directions: there is a propagation delay τ_i^f from the end user i to the base station — we call it the forward delay of the end user i, and a propagation delay τ_i^b from the base station to the end user i — we call it the backward delay. It

is well-known that the presence of delays may affect the performance of the network. For example, Internet congestion controllers which are globally stable for the delay-free network may become unstable if the feedback delays are large [28]. In our problem, when delays exist, the information the end users obtain will be "outdated" information. So the congestion information the users obtain at time t does not reflect the queue status at the base station at time t. So it is interesting to study a wireless network with delays and ask whether the conclusions of [8] still hold for wireless networks with heterogeneous delays. We answer this question by showing that for a network with uniformly-bounded delays, which are potentially heterogeneous and time-varying, the algorithm of [8] is stable and can be used to approximate weighted-m fair allocation arbitrarily closely. We emphasize that the results hold for networks with arbitrarily large, but bounded time-varying delays. So even if the end users can only get very old feedback information from the base station, the network is still stable and can eventually reach the fair resource allocation.

3.3 Simultaneous Routing and Resource Allocation

In wireless data networks the optimal routing of data depends on the link capacities which, in turn, are determined by the allocation of communications resources (such as transmit powers and signal bandwidths) to the links. The optimal performance of the network can only be achieved by simultaneous optimization of routing and resource allocation.

The paper [120] studies the simultaneous routing and resource allocation problem and exploits problem structure to derive efficient solution methods. We use a capacitated multicommodity flow model for the data flows in the network. We assume that the capacity of a wireless link is a concave and increasing function of the communications resources allocated to the link (TDMA and FDMA systems), and the communications resources for groups of links are limited. These assumptions allow us to formulate the simultaneous routing and resource allocation problem as a convex optimization problem over the network flow variables and the communications variables. These two sets of variables are coupled only through the link capacity constraints. We exploit this separable structure by dual decomposition. The resulting solution method attains the optimal coordination of data routing in the network layer and resource allocation in the radio control layer via pricing on the link capacities.

In [118], we generalize the simultaneous routing and resource allocation formulation to include CDMA wireless systems. Although link capacity constraints of CDMA systems are not jointly convex in rates and powers, we show that by using coordinate projections or transformations, the simultaneous routing and power allocation problem can still be formulated as (in systems with interference cancellation) or approximated by (in systems without interference cancellation) a convex optimization problem which can be solved very efficiently. We also propose a heuristic link-removal procedure based on the convex approximation to further improve the system performance.

4 Fault Diagnosis over Packet Dropping Networks

There are several challenges that arise when trying to perform management and control over unreliable, possibly heterogeneous networks. The major concern is the fact that, due to the nature of network links, observations may be delayed, lost or received out of order. In such case, diagnosers or controllers that use this underlying network infrastructure need to be able to cope with the unreliability of the communication links in an effective and reliable manner. Within the context of this URI project, we have been exploring a number of directions that aim to achieve this ultimate goal.

4.1 Probabilistic Fault Detection and Identification

Another direction that we have been pursuing is along the lines of our work on systematic and efficient methodologies for fault management in dynamic systems. For example, we have developed probabilistic schemes for detecting permanent or transient functional changes (faults) in large-scale discrete event systems that can be modeled as finite-state machines. In one particular setup, the detector observes the frequencies with which states are occupied and detects faults by analyzing the deviation between the expected frequencies and the actual measurements. These features can be useful in distributed or networked settings where the input-state order may not be known and, at this point, we are considering applications of these ideas in the context of statistical methods for network security and intrusion detection. This work appeared as an invited paper during the 2002 Conference on Decision and Control; an extended version of it also appeared in the IEEE Transactions on Automatic Control.

More recently, we have began the investigation of schemes for observing/diagnosing/controlling systems or networks under unreliable information that might arise due to permanent or transient faults in the system sensors. More specifically, we have developed a probabilistic methodology for failure diagnosis in finite state machines given a sequence of unreliable (possibly corrupted) observations. Assuming prior knowledge of the input probability distribution but no knowledge of the actual input sequence, the core problem we considered aimed at choosing from a pool of known, deterministic finite state machines (FSMs) the one that most likely matches a given (output) sequence of observations. The main challenge is that errors, such as symbol insertions, deletions, and transpositions, may corrupt the observed output sequence; the cause of these errors could be a faulty sensor or problems encountered in the communication channels or network links connecting the system sensors with the diagnoser/observer. Given the possibly erroneous output sequence of observations, we have proposed an efficient recursive algorithm for obtaining the most likely underlying FSM. We have illustrated the proposed methodology using as an example the diagnosis (identification) of a communication protocol.

Along these lines, we have also been able to make connections with the literature on hidden

Markov models. To this end, we are currently trying to understand the role of reduced-order models and the role of modeling methodologies such as hidden Markov models or the influence model. We believe that this work will have important practical implications because it relates directly to the issue of sensor reliability and cost (i.e., the task of determining the required levels of reliability for the system sensors in order to guarantee a certain level of performance).

4.2 Distributed Symmetric Function Computation in Noisy Wireless Sensor Networks with Binary Data

With the wide availability of inexpensive wireless technology and sensing hardware, wireless sensor networks are expected to become commonplace because of their broad range of potential applications. A wireless sensor network consists of sensors that have sensing, computation and wireless communication capabilities. Each sensor monitors the environment surrounding it, collects and processes data, and when appropriate transmits information so as to cooperatively achieve a global detection objective. We have considered the common situation where there is a single fusion center, and the network goal is to cooperatively provide information to this fusion center so it can compute some function of the sensor measurements.

We have investigated this problem in multi-hop networks with noisy communication channels where the measurement of each sensor consists of one bit. We consider a sensor network consisting of n sensors, each having a recorded bit, the sensor's measurement, which has been set to either "0" or "1". The goal of the fusion center is to compute a symmetric function of these bits; i.e., a function that depends only on the number of sensors that have a "1". Specifically, distributed symmetric function computation with binary data, which is also called a counting problem, is as follows: each node is in either state "1" or "0", and the fusion center needs to decide, using information transmitted from the network, the number of sensors in state "1".

The sensors convey information to the fusion center in a multi-hop fashion to enable the function computation. The problem studied is to minimize the total transmission energy used by the network when computing this function, subject to the constraint that this computation is correct with high probability.

When nothing is known about the structure of the function to be computed, all bits must to be transmitted to the fusion center, and this is purely a routing problem when the channels are reliable. When the wireless channels are unreliable, the use of channel coding (see, for example, [9]) makes it possible to convey information in a point-to-point fashion with arbitrarily small amounts of error. However, the use of point-to-point error-correction coding without any in-network processing may result in high energy cost and delay. Our focus is computation of symmetric functions in a noisy wireless sensor network when total energy consumption *is* a major concern.

The algorithms considered are related to the algorithms for distributed computation over noisy

networks, which are studied in [10, 23, 24, 22, 17], and references within. In both problems, the goal is to compute the value of some function based on the information of the nodes. Our work is closely related to parity computation and threshold detection in noisy radio networks studied in [10] and [17], respectively, where a broadcast network is assumed, in which all nodes can hear all transmissions, and each node has a "1" or a "0". The goal in [10, 17] was to investigate the minimum number of transmissions required to compute the parity or decide whether the number of nodes in state "1" has exceeded the threshold value. Note that parity and threshold detection are special cases of counting, since both of these are determined if we know how many nodes have "1".

While the problems considered in [10] and [17] are similar to our problem, a major difference is that in our model, each node may not be able to hear every other node in the network. The reason for this is that energy consumption can be an important consideration in wireless networks and it is well-known that it can be reduced significantly if the transmissions are carried out in a multi-hop fashion. This is a consequence of the well-known propagation model used to model wireless communication channels, whereby the energy required to transmit over a distance of r is proportional to r^{α}, where $\alpha \geq 2$ is a constant depending upon the environment. Thus, instead of each sensor sending its information to the fusion center directly, it is more efficient from an energy consumption point of view to send the information through relay nodes. It may be possible to reduce energy consumption even further by using some form of in-network data processing. This may have further benefits; for instance, if all the sensor measurements are to be transmitted from the sensors to the fusion center, then relay nodes closer to the fusion center would be depleted of their energy faster than nodes that are further away from the fusion center. Thus, in-network processing to reduce the number of transmissions could be beneficial for eliminating hotspots. Fundamentally, this is the distinction between multi-hop wireless networks used for communication and multi-hop wireless networks used for sensing. In multi-hop wireless communication networks, the protocols are designed so that they are not application-specific, and therefore the network can support a constantly evolving set of applications. Contrasting this, in multi-hop sensor networks, the architecture and protocols can be designed for each specific application, exploiting its structure, to reduce the energy usage within the network. This is the motivation for the recent works reported in [12] and [15]. In [12], the authors have designed a block coding scheme to compress the amount of information to be transmitted in a sensor network computing some functions. In [15], the authors investigate the optimal computation time and the minimum energy consumption required to compute the maximum of the sensor measurements. However, the in-network processing that we consider is different from the processing considered in [12] and [15], where the communication channels are assumed to be reliable, and the processing is to primarily exploit the spatial correlation [15] or the spatio-temporal correlations [12]. In our problem, processing is required not only to reduce the redundancy in the information to be conveyed in the fusion center, but also to introduce some redundancy to

combat the effect of the noisy channels in the sensor network. Our results show that the additional redundancy required to combat channel errors does not significantly negate the benefits of in-network computation used to eliminate redundancy in the information, and the combination of in-network computation and channel coding could reduce the number of transmissions required in multi-hop networks to the same order as the number required in single-hop networks.

We use the routing protocol in [12] along with ideas from distributed parity computation in noisy networks ([10]) to devise near energy-optimal algorithms for counting in sensor networks. A key difference between our work and the work in [10] is that, in the case of sensor networks, the fusion center does not communicate directly with each of the sensors. Thus, local computation is necessary before conveying some aggregate information in a multi-hop fashion to the fusion center. The local computation in our case is not a simple parity computation as in [10] but as we will see later, the network needs to compute the number of sensors in each local neighborhood (called a *cell*) that have seen a "1". Further, we require that the computation be accurate uniformly over all cells. In addition, we will show that error-correction coding is required in the algorithms to minimize the energy required for counting.

We assume the wireless channels are binary symmetric channels with a probability of error p, and that each sensor uses r^α units of energy to transmit each bit, where r is the transmission range of the sensor. Using the above ideas, we first study the case where each sensor has only one observation to report, and show that the amount of energy required for counting (i.e., detecting the number of sensors seeing a "1") is $O\left(n(\log\log n)\left(\sqrt{\frac{\log n}{n}}\right)^\alpha\right)$, where n is the number of sensors in the network. We also show that any algorithm satisfying the performance constraints must necessarily have energy usage $\Omega\left(n\left(\sqrt{\frac{\log n}{n}}\right)^\alpha\right)$. Then, we consider the case where the sensor network observes N events, and each node records one bit per event, thus having N bits to convey. The fusion center now wants to compute N symmetric functions, one for each of the events. We then extend to the case where each sensor has N binary observations, and the symmetric function needs to be computed for each observation. We show that the total transmission energy consumption can be reduced to $O\left(n\left(\max\left\{1, \frac{\log\log n}{N}\right\}\right)\left(\sqrt{\frac{\log n}{n}}\right)^\alpha\right)$ per observation. When $N = \Omega(\log\log n)$, the energy consumption is $\Theta\left(n\left(\sqrt{\frac{\log n}{n}}\right)^\alpha\right)$ per observation, which is a tight bound. If we only want to know roughly (a notion made precise in [39]) how many sensors have "1". The answer can be obtained with the transmission energy consumption $\Theta\left(n\left(\sqrt{\frac{\log n}{n}}\right)^\alpha\right)$.

5 Decentralized Control

5.1 Control over Networks

The first line of research we have been pursuing relates to the study of the fundamental performance limitations of control methodologies that use existing network infrastructure as their communications backbone. For instance, by modeling a packet dropping network as an erasure channel and by focusing on bounded variance stabilization schemes, one can study the problem of plant stabilization despite message delays, packet drops, quantization noise and measurement noise. Our initial work on this problem appeared as an invited paper in the 2002 Conference on Decision and Control and focused on the case when the system to be stabilized is a discrete-time linear time-invariant system, the communication links (between sensors, controller(s) and actuators) are part of a packet dropping network in which transmissions are independent, and the network packets are large enough so that the effect of measurement quantization can be modeled by an additive white noise process. This work, which has been referenced extensively by many researchers, has also been extended to settings where the system to be stabilized is a continuous-time system, in which case one additional parameter that needs to be chosen is the sampling rate at which the sample-data controller is operating. We have been able to determine ways to optimally choose this and other parameters of this control problem, and our results are currently under revision in the IEE Proceedings on Control Theory and Applications.

Related to the task described above is our study of the effects of roundoff noise on our ability to detect and identify transient faults that affect the operation of control systems. This roundoff noise could arise due to finite precision limitations of our controllers or due to quantization that takes place when sending information of the underlying communication network (e.g., the Internet). Our analysis has provided insight that allows us to handle roundoff noise via explicit bounds on the precision needed to guarantee the correct identification of the number of errors. Our analytical bounds can be very tight for certain choices of design parameters and can be used to provide guidance about the design of fault-tolerant systems.

More recently, our group has been focusing its efforts in extending these ideas to settings where the network delays between different packets are not independent. To this end, we have been trying to make connections with work on linear jump Markov systems. We are also interested in understanding how different network protocols (e.g., forward error correction or path diversity techniques) can be used to enable more effective controllers.

5.2 Decentralized Observation and Monitoring

Another direction that we have been pursuing within the context of this project relates to the construction of observers for switched systems under unknown or partially known inputs. This is

a situation that arises frequently in practice as unknown inputs are used to represent uncertain system dynamics and faults or, in the case of decentralized systems, control signals generated by other controllers. Within this line of work, we have obtained methods for constructing reduced-order state observers for linear systems with unknown inputs. Apart from making connections with existing work on system invertibility and fault detection and identification, our approach provides a characterization of observers with delay, which eases the established necessary conditions for existence of unknown input observers with zero-delay. Our techniques are quite general in that they encompass the design of full-order observers via appropriate choices of design matrices.

Our work has also looked at challenges that arise in monitoring and controlling discrete event systems over unreliable networks. For instance, we have looked at decentralized failure diagnosis schemes for systems that can be modeled as finite state machines. The specific scenario we considered consists of multiple local diagnosers, each with partial access to the outputs of the system under diagnosis. Our focus has been on designing a global coordinator which synchronizes with the local diagnosers at unspecified time intervals, and combines the local estimates in order to reach a final diagnosis. Under the assumption that the system and the local diagnosers are known, we have been able to analyze the effectiveness of simple types of global coordinators that operate without knowledge of the functionality of the system or the local diagnosers. In each case, we were able to derive conditions for finite-delay and zero-delay diagnosability, and to explore the trade-offs between the various schemes in terms of processing power and memory requirements on the global coordinator.

5.3 Decentralized Control

In order to develop systems that can coordinate with each other via communication the research in this program targets many new issues which are not present in the traditional feedback control scenario. These include distributed control decision based on local rather than global information, asynchronous information transmission, dynamic network topology, and scalability of algorithms to networks with large-numbers of nodes. Separate idealizations of the first two of these aspects of the problem have been possible in the past, although even these model problems lead to substantial difficulties in analysis. For example, one may use an idealized model of communication, and consider the simplest *decentralized control* problem. A general formulation of this problem can be reduced to one of structured control synthesis, a problem for which a systematic approach is lacking in the current literature.

Conversely, one may assume a centralized information pattern and consider the centralized control problem subject to *asynchronous communication*. In this case, instead of data consisting of continuous signals, data is now transmitted in packets. Furthermore, packets are subject to loss or delay, large packets may be fragmented and require reassembly, and packet streams may be received

out of order. Control systems must be designed which are robust to these occurrences.

Given a particular plant and a constraint set of allowable decentralized controllers, one would like to determine if the associated control problem is, in a certain sense, easily solvable. The paper [121] develops a clear and precise characterization of when this is so. The notion of *quadratic invariance* is introduced in that paper, and is outlined here.

We suppose we have a linear plant G, and a subspace of admissible controllers S, which captures any sparsity constraints on the controller. The set S is called quadratically invariant if KGK is an element of S for all K in S. The paper shows that, if the constraint set has this property, then a controller which minimizes any norm of the closed-loop system may be efficiently found.

The area of decentralized control systems has been a source of challenging problems for many years. Starting with work on team theory, there have been many results showing that problems with certain information structures may be solved, and recently many papers have developed specific optimization methods to address these problems. Examples include decentralized control where the systems are chained in a particular way, as well as arrays of systems where information satisfies certain delay requirements. Our work provides a unifying framework in which to analyze these systems. So far, all of the known solvable problems to which we have applied this theory have been found to be quadratically invariant.

There are many links here to other active areas of research within this project, in particular to the work on the decentralized congestion control mechanisms used by TCP in the Internet. Further work remains, as there are important questions of computational complexity, since many decentralized control problems are known to be intractable. It is also known that many decentralized linear control systems have optimal controllers which are nonlinear. Our work addresses some extremely fundamental issues which are a central concern, and produces results which are both theoretically important and practically relevant.

5.4 Monitoring and Control of Power System Dynamics

We have addressed monitoring and control of system-wide electromechanical (or 'swing') dynamics in power systems, as well as the dynamics of auction-based electricity markets. We showed that observer-based power system monitors can be used to estimate the full state of the system as well as identify and isolate a number of events (e.g., faults) using only sparse local measurements, all in the presence of various system disturbances. This work also develops and exploits a spatio-temporally integrated view of electromechanical dynamics. This contrasts with the traditional approach of either studying temporal variations at fixed spatial points or investigating spatial variations of specified (e.g., modal) temporal behavior. We use a continuum model of the swing dynamics to expose the wave-like propagation of electromechanical disturbances and to gain insight for the design of controls. This leads to strategies for decentralized control of these electromechanical waves, drawing

on prototype controllers found in electromagnetic transmission line theory (e.g., matched-impedance terminations) and active vibration damping (e.g., energy-absorbing controllers and vibration isolators). Finally, we have proposed various controllers to realize quenching or confining-and-quenching strategies, and tested these in simulations of a 179-bus reduced-order representation of the power grid of the western US and Canada.

6 Complexity and robustness in complex networks

Recent progress in systems biology and network-based technological systems, together with new mathematical theories, has revealed generalized principles that shed new light on complex networks, and confirmed the observations that an inherent feature of complex multiscale systems is that they are "robust yet fragile" (RYF). They are both intrinsically robust under most normative conditions and yet can be extremely sensitive to certain perturbations in their environment and component parts. This RYF feature provides a new paradigm for thinking about complexity and evolution across a broad range of phenomena and scales from computer networks to immune systems, from power grids and cancers to ecosystems, financial markets and human societies. While this research draws heavily on systems and control theory, most papers have appeared in biology, networking, and physics journals. In this section, we will review recent progress in theory and applications aimed at an engineering audience.

This research builds on insights about the fundamental nature of complex biological and technological networks that can now be drawn from the convergence of three research themes. 1) Molecular biology has provided a detailed description of much of the components of biological networks, and with the growing attention to systems biology the organizational principles of these networks are becoming increasingly apparent. 2) Advanced technology has provided engineering examples of networks with complexity approaching that of biology. While the components differ from biology, we have found striking convergence at the network level of architecture and the role of layering, protocols, and feedback control in structuring complex multiscale modularity. Our research is leading to new theories of the Internet and to new protocols that are being tested and deployed for high performance scientific computing. 3) Most importantly, there is a new mathematical framework for the study of complex networks that suggests that this apparent network-level evolutionary convergence both within biology and between biology and technology is not accidental, but follows necessarily from the requirements that both biology and technology be efficient, adaptive, evolvable, and robust to perturbations in their environment and component parts. This theory builds on and integrates decades of research in pure and applied mathematics with engineering, and specifically with robust control theory.

Through evolution and natural selection or by deliberate design, such systems exhibit highly func-

tional and symbiotic interactions of extremely heterogeneous components, the very essence of "complexity". At the same time this resulting organization allows, and even facilitates, severe fragility to cascading failure triggered by relatively small perturbations. Thus robustness and fragility are deeply intertwined in biological systems, and in fact the mechanisms that create their extraordinary robustness are also responsible for their greatest fragilities. Our highly regulated and efficient metabolism evolved when life was physical challenging and food was often scarce. In a modern lifestyle, this robust metabolism can contribute to obesity and diabetes. More generally, our highly controlled physiology creates an ideal ecosystem for parasites, who hijack our robust cellular machinery for their own purposes. Our immune system prevents most infections but can cause devastating autoimmune diseases, including a type of diabetes. Our complex physiology requires robust development and regenerative capacity in the adult, but this very robustness at the cellular level is turned against us in cancer. We protect ourselves in highly organized and complex societies which facilitate spread of epidemics and destruction of our ecosystems. We rely on ever advancing technologies, but these confer both benefits and horrors previously unimaginable. This universal "robust yet fragile" (RYF) nature of complex systems is well-known to experts such as physicians and systems engineers, but has been systematically studied in any unified way only recently. It is now clear that it must be treated explicitly in any theory that hopes to explain the emergence of biological complexity, and indeed is at the heart of complexity itself.

These RYF features appear on all time and space scales, from the tiniest microbes and cellular subsystems up to global ecosystems, and also -we believe- to human social systems, and from the oldest known history of the evolution of life through human evolution to our latest technological innovations. Typically, our networks protect us, which is a major reason for their existence. But in addition to cancer, epidemics, and chronic auto-immune disease, the rare but catastrophic market crashes, terrorist attacks, large power outages, computer network virus epidemics, and devastating fires, etc, remind us that our complexity always comes at a price. Statistics reveal that most dollars and lives lost in natural and technological disasters happen in just a small subset of the very largest events, while the typical event is so small as to usually go unreported. The emergence of complexity can be largely seen as a spiral of new challenges and opportunities which organisms exploit, but lead to new fragilities, often to novel perturbations. These are met with increasing complexity and robustness, which in turn creates new opportunities but also new fragilities, and so on. This is not an inexorable trend to greater complexity, however, as there are numerous examples of lineages evolving increasing simplicity in response to less uncertain environments. This is particularly true of parasites that rely on their hosts to control fluctuations in their microenvironment, thus shielding them from the larger perturbations that their hosts experience.

It is only fairly recently, and particularly the last few decades, that human technology has become focused not just on robustness, but on architectures that facilitate the evolution of new

capabilities and the scaling to large system sizes. Protocol-based multilayer modular design is permeating advanced technologies of all kinds, but the Internet remains perhaps the most well-known example. It is also particularly suitable for our purposes for several reasons. The Internet, and cybertechnology generally, are unprecedented in the extent to which their features parallel biology. Their most salient features are often hidden from the user and thus as metaphors are often terribly misleading, yet are extremely useful when right. Only cybertechnology has the potential to rival biotechnology in accelerating the human/technology evolution, and the combined RYF spiral could have profound consequence. The most consistent, coherent, and salient features of all complex technologies are their protocols. To engineers, the term "protocol" is the set of rules by which components interact to create system-level functionality. Indeed, in advanced technologies, and we believe in the organization of cells and organisms, the protocols are more fundamental than the modules whose interconnection they facilitate, although they often are obscured by the overwhelming details that now characterize experimental results in biology. A central feature of efficient, protocol-based systems is that, provided they obey the protocols, modules can be exchanged. The details are less important here than the consequences, which are the system-level robustness and evolvability that these protocols facilitate. New and even radically different hardware is easily incorporated at the lowest physical layers, and even more radically varying applications are enabled at the highest layers. Ironically as in biology, it is these transient elements of hardware and application software that are most visible to the user, while the far more fundamental and persistent infrastructure is the core protocols, which by design remain largely hidden from the user.

A protocol-based organization facilitates coordination and integration of function to create coherent and global adaptation to variations in their components and environments on a vast range of time scales despite implementation mechanisms that are largely decentralized and asynchronous. The parallels here between the Internet and biology are particularly striking. The TCP/IP protocol suite enables adaptation and control on time scale from the sub-microsecond changes in physical media, to the millisecond-to-second changes in traffic flow, to the daily fluctuations in user interactions, to evolving hardware and application modules over years and decades. The remarkable robustness to changing circumstances and evolution of Internet-related technology could only have come about as the result of a highly structured and organized suite of relatively invariant and universally-shared, well-engineered protocols.

Similarly, a protocol-based architecture in biology and its control mechanisms facilitate both robustness and evolvability, despite massive impinging pressures and variation in the environment. With the most obvious example involving the table of codons, biology's universally shared set of protocols are more fundamental and invariant than the modules whose control and evolution they facilitate. Allostery, a huge suite of post-translational modifications, and the rapid changes in location of macromolecular modules enable adaptive responses to environmental signals or alterations

on rapid time scales. Translational and transcriptional control and regulation of alternative splicing and editing act on somewhat longer time scales. On still longer time scales within and across generations, the sequences of the DNA itself can change, not only through random mutation, but also through highly structured and evolved mechanisms that facilitate the generation of adaptive diversity. Furthermore, as biologists dig deeper past the superficiality of sequence data into the complexity of regulation, they unearth additional layers of control that are fundamentally similar to those in advanced technologies. There is seemingly no limit to the ingenuity that biology uses in creating additional layers of sophisticated control. Now familiar examples range from RNA editing and alternative splicing to transposons, mismatch repair, and repetitive sequences to the cutting and pasting in the "arms race" of the immune system versus spirochete and trypanosome coat proteins.

Perhaps the most familiar example of lateral gene transfer in bacteria is possible because bacteria have a shared set of protocols that have even been quite appropriately described by some as the "bacterial Internet." Bacteria can simply grab DNA encoding new genes from other bacteria and incorporate it into their genome, just like computer users can buy a new computer and plug it into home or office networks. This "plug and play" modularity works because there is a shared set of protocols that allow even novel genetic material to be functional in an entirely new cellular setting. Plug and play DNA mobility and expression is further facilitated by integrons and plasmids. Thus, for example, bacteria can acquire antibiotic resistance on time-scales that would be vanishingly improbable by point mutations, an example of how rapid evolution of complexity is possible by Darwinian mechanisms. Natural selection can favor the evolution of whole protocol suites, and their interactions, which in turn massively accelerate the acquisition and sharing of functional adaptive change. Thus evolvability itself can be seen as the robustness of lineages, rather than organisms, on long time scales and to possibly large changes in the environment, indeed ones that would be lethal to organisms if they occurred rapidly. An important insight is that robustness and evolvability are generally not in conflict, and both are the product of systematic and organized control mechanisms.

The framework being developed here is radical in both its methodology and philosophical implication. Methodologically it draws on mathematics that is often not well known outside expert circles and in many cases had not traditionally been thought of as "applied." The mathematics tells us that robustness and fragility have conserved quantities, and we believe these will ultimately be of as much importance to understanding biological complexity as energy and entropy were to understanding the steam engine and mitochondria. The above views of "organized complexity" motivated by biology and engineering contrast sharply with that of "emergent complexity" that is more popular within the physical sciences. Highly Optimized/Organized Tolerance/Tradeoffs (HOT) has aimed to explain the issues of organized complexity, but emphasizing models and concepts such as lattices, cellular automata, spin glasses, phase transitions, criticality, chaos, fractals, scale-free networks, self-organization, and so on, that have been the inspiration for the "emergent" perspective. A side

benefit of this largely pedagogical effort is it has led to apparently novel insights into RYF aspects of longstanding mysteries in physics, from coherent structures in shear flow turbulence and coupled oscillators, to the ubiquity of power laws, to the nature of quantum entanglement, to the origin of dissipation. Finally, the underlying mathematics may offer new tools to explore other problems in physics where RYF features may play a role, particularly involving multiple scales and organized structures and phenomena.

7 Network Reliability

We have investigated the reliability of networks operated in a distributed manner to changes in the network. Our work is based on the use of network coding, which allows both distributed coding and distributed implementations of cost optimization for the subgraphs used for network coding. We have considered two main issues: 1) the reliability of networks to packet losses 2) the cost efficiency of distributed coding and optimization under changes of topology, cost and traffic.

Packet losses in networks result from a variety of causes, which include congestion, buffer overflows, and, in wireless networks, link outage due to fading. Thus a method to ensure reliable communication is necessary, and the prevailing approach is for the receiver to send requests for the retransmission of lost packets over some feedback channel. There are, however, a number of drawbacks to such an approach, which are evident most notably in high-loss environments and for multicast connections. In both instances, many requests for retransmissions are usually required, which place an unnecessary load on the network and which may overwhelm the source. In the latter instance, there is the additional problem that retransmitted packets are often only of use to a subset of the receivers and are therefore redundant to the remainder. An approach that overcomes these drawbacks is to use erasure-correcting codes. Under such an approach, the original packets are reconstructed from those that are received and little or no feedback is required. This approach has been recently exemplified by digital fountain codes, which are fast, near-optimal erasure codes. Such codes can approach the capacity of connections over lossy packet networks, provided that the connection as a whole is viewed as a single channel and coding is performed only at the source node. But in lossy packet networks where all nodes have the capability for coding, such as overlay networks using UDP and wireless networks, there is no compelling reason to adopt this view, and a greater capacity can in fact be achieved if we do not.

We have developed a capacity-approaching coding scheme for unicast or multicast over lossy packet networks. In the scheme, all nodes perform coding, but do not wait for a full block of packets to be received before sending out coded packets. Rather, whenever they have a transmission opportunity, they form coded packets with random linear combinations of previously received packets. All coding and decoding operations in the scheme have polynomial complexity. Our analysis of the

scheme has shown that it is not only capacity-approaching, but that the propagation of packets carrying innovative information follows that of a queueing network where every node acts as a stable MM1 queue. We are able to consider networks with both lossy point-to-point and broadcast links, allowing us to model both wireline and wireless packet networks.

In the area of distributed optimization, we have presented decentralized algorithms that compute minimum-cost subgraphs for establishing multicast connections in networks that use coding. These algorithms, coupled with existing decentralized schemes for constructing network codes, constitute a fully decentralized approach for achieving minimum cost multicast. Our approach is in sharp contrast to the prevailing approach based on approximation algorithms for the directed Steiner tree problem, which is suboptimal and generally assumes centralized computation with full network knowledge. We also have developed extensions beyond the basic problem of fixed-rate multicast in networks with directed point-to-point links, and consider the case of elastic rate demand as well as the problem of minimum energy multicast in wireless networks. For the case of optimization under changing conditions, we have given a formulation of the dynamic multicast problem for coded networks that lies within the framework of dynamic programming. Our formulation addresses the desired objective of finding minimum-cost time-varying subgraphs that can deliver continuous service to dynamic multicast groups in coded networks and, because it lies within the framework of dynamic programming, can be approached using methods developed for general dynamic programming problems. The solution that we propose uses such methods to approximate the optimal cost function, which is used to modify the objective function of an optimization that determines the multicast subgraph to use during each time interval. Depending upon the approximation that is used for the optimal cost function, this optimization conducted every time interval may be tractable and may even be amenable to decentralized computation.

7.1 Graph Structure and Similarity

We have investigated measures of graph similarity, developed a new measure, and applied it to the matching of graph fragments to their original locations in a parent graph. Measures of graph similarity have a broad array of applications, including comparing chemical structures, navigating complex networks like the World Wide Web, and more recently, analyzing different kinds of biological data. The research focuses on an interesting class of iterative algorithms that use the structural similarity of local neighborhoods to derive pairwise similarity scores between graph elements. Our new similarity measure uses a linear update to generate both node-node and edge-edge similarity scores, and has desirable convergence properties. The research also explores the application of our similarity measure to graph matching. We attempt to correctly position a subgraph within a parent graph using a maximum-weight matching algorithm applied to the similarity scores between the two graphs. Significant performance improvements are observed when the 'topological' information pro-

vided by the similarity measure is combined with additional information — such as partial labeling — about the attributes of the graph elements and their local neighborhoods. Further work is needed to explore various extensions of these ideas, including to the case of dynamically evolving graphs.

In other work, we study synchronization of complex random networks of nonlinear oscillators, with identical oscillators at the nodes interacting through 'diffusive' coupling across edges of the interconnection graph. Our random network is constructed by a generalized Erdos-Renyi method, so as to have specifiable expected-degree distribution. We present a sufficient condition for synchronization and a sufficient condition for desynchronization, stated in terms of the coupling strength and the extreme values of the distribution of nontrivial eigenvalues of the graph Laplacian. We then determine the Laplacian eigenvalue distribution for the case of large random graphs through computation of the moments of the eigenvalue density function. The analysis is illustrated using a random network with a power-law expected-degree distribution and chaotic dynamics at each node. The mathematical structure of our problem is closely related to that of consensus problems in networks of agents, as well as the task of analyzing flocking/swarming conditions in a group of autonomous agents.

8 Impact of research and transitions

1) The first important algorithm to be transitioned is Approximate Fair Dropping (AFD). This is a new randomized algorithm that partitions the bandwidth of a link among the flows traversing the link. It builds on the CHOKe (choose and keep or choose and kill) algorithm, which is a simple algorithm for protecting TCP flows from UDP flows and enables the detection of flows which attempt to take up a disproportionate share of resources. The randomized nature of the algorithms not only make them simpler to implement, but also prevents users from predicting and attempting to spoof their behavior. The AFD algorithm is in discussion for implementation in the CISCO GSR12000 series of core routers.

2) Secondly, the FAST (Fast AQM, Scalable TCP) protects the TCP protocol from instabilities which currently occur at high link speeds. These instabilities cause network throughput to drop to an extremely low level, and affect fast networks dramatically. Building on research from both the controls and networking community has led to this new protocol, which is both provably robust and scalable as well as incrementally deployable. In an experiment in November 2002, a speed of 8,609 megabits per second (Mbps) was achieved by using 10 simultaneous flows of data over routed paths, which is the largest aggregate throughput ever accomplished in such a configuration. FAST has been developed in significant part by Caltech, and has been transitioned through the Stanford Linear Accelerator Center (SLAC), working in partnership with the European Organization for Nuclear Research (CERN), and DataTAG, StarLight, TeraGrid, Cisco, and Level3.

3) This program has developed Distributed Random Coding (DRC), which combines the benefits of coding and routing into a single protocol. Here, instead of simply forwarding packets, nodes construct and forward algebraic combinations of inputs. This results in a network which has both significantly increased throughput as well as making it impossible for an observer to decode data transmitted by simply observing the network at a single point. This protocol has been implemented in a large-scale network testbed by Microsoft, working with Sprint.

All of the above protocols are backed up by theoretical analysis, with, for example, associated proofs of stability and convergence. The program has developed several further protocols for network security, including mechanisms for covert message transmission via timing channels, protection against SYN flooding, the SIFT algorithm for prioritization in caches and buffers, and the SHRiNK method for monitoring extremely large-scale networks.

References

[1] R. Pan, B. Prabhakar, K. Psounis: "CHOKe - A stateless active queue management scheme for approximating fair bandwidth allocation". *INFOCOM 2000.*

[2] R.Pan, L.Breslau, B.Prabhakar, S.Shenker: "Approximate fairness through differential dropping," *ACM Computer Communications Review,* January 2002.

[3] R.Pan, L.Breslau, B.Prabhakar, S.Shenker: "Approximate fair allocation of link bandwidth," *IEEE Micro,* January 2003.

[4] M. Andrews, K. Kumaran, K. Ramanan, A.L. Stolyar, R. Vijayakumar, and P. Whiting. CDMA data QoS scheduling on the forward link with variable channel conditions. Bell Laboratories Tech. Rep., April 2000.

[5] D. Bertsimas, I. Ch. Paschalidis, and J. N. Tsitsiklis. Asymptotic buffer overflow probabilities in multiclass multiplexers: An optimal control approach. In *IEEE Transactions on Automatic Control,* 43:315-335, March 1998.

[6] S. Deb and R. Srikant. Global stability of congestion controllers for the internet. *IEEE Transactions on Automatic Control,* 48(6):1055–1060, June 2003.

[7] A. Eryilmaz and R. Srikant, and J. Perkins. Stable scheduling policies for fading wireless channels. In *IEEE/ACM Transactions on Networking,* April 2005, pp. 411-424.

[8] A. Eryilmaz and R. Srikant. Scheduling with Quality of Service Constraints over Rayleigh Fading Channels In *Proceedings of IEEE Conference on Decision and Control,* Dec. 2004.

[9] R. G. Gallager. Information Theory and Reliable Communication. John Wiley & Sons, New York, 1968.

[10] R. G. Gallager. Finding parity in a simple broadcast network. In *IEEE Transactions on Information Theory,* vol. 34, pp 176-180, 1988.

[11] A. Ganti, E. Modiano and J. Tsitsiklis. Optimal Transmission Scheduling in Symmetric Communication Models with Intermittent Connectivity, 2004 Preprint.

[12] A. Giridhar and P. R. Kumar. Computing and communicating functions over sensor networks. In *IEEE Journal on Selected Areas in Communications*, pp. 755–764, vol. 23, no. 4, April 2005.

[13] P. Gupta and P. Kumar. Critical power for asymptotic connectivity in wireless network. In *Stochastic Analysis, Contrtol, Optimization and Applicaitons: a Volume in Honor of W.H.Fleming,* W. McEneaney, G. Yin and Q. Zhang, Eds., 1998

[14] P. Gupta and P. Kumar. The capacity of wireless networks. In *IEEE transactions of Information Theory,* vol. 46, no.2, pp. 388-404, 2000.

[15] N. Khude, A. Kumar and A. Karnik. Time and Energy Complexity of Distributed Computation in Wireless Sensor Networks. In *Proceedings of the IEEE Infocom,* 2005.

[16] S. R. Kulkarni and P. Viswanath. A Deterministic Approach to Throughput Scaling in Wireless Networks. In *IEEE Trans. on Information Theory,* Vol. 50, No.6, pp. 1041-1049, June 2004.

[17] E. Kushilevitz and Y. Mansour. Computation in Noisy Radio Networks In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms,* pp. 236-243, 1998.

[18] X. Lin and N. Shroff. The impact of imperfect scheduling on cross-layer rate control in wireless networks. In *Proceedings of IEEE Infocom 2005*, Miami, Florida, March 2005.

[19] X. Liu, E. Chong, and N. Shroff. Opportunistic transmission scheduling with resource-sharing constraints in wireless networks. In *IEEE Journal on Selected Areas in Communications,* 19(10):2053 – 2064, October 2001.

[20] M. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. In *Proceedings of IEEE Infocom 2005*, Miami, Florida, March 2005.

[21] M. J. Neely, E. Modiano, and C. E. Rohrs. Power and server allocation in a multi-beam satellite with time varying channels. In *Proceedings of IEEE Infocom,* New York, NY. June 2002

[22] S. Rajagopalan and L. J. Schulman. A Coding Theorem for Distributed Computation. In Proc. 26th STOC 790-799, 1994.

[23] L. J. Schulman. Communication on Noisy Channels: A Coding Theorem for Computation. In *Proceeding of 33rd FOCS,* pp. 724-733, 1992.

[24] L. J. Schulman. Deterministic Coding for Interactive Communication. In *Proceeding of the 25th Annual Symposium on Theory of Computing,* pp. 747-756, 1993.

[25] S. Shakkottai. Effective Capacity and QoS for Wireless Scheduling, 2004 Preprint.

[26] S. Shakkottai, R. Srikant, and A. Stolyar. Pathwise optimality of the exponential scheduling rule for wireless channels. In *Advances in Applied Probability,* 1021-1045, Dec. 2004.

[27] S. Shakkottai and A. Stolyar. Scheduling for multiple flows sharing a time-varying channel: The exponential rule. In *Translations of the American Mathematical Society,* Series 2,A volume in memory of F. Karpelevich, Yu. M. Suhov, Editor, Vol. 207, 2002.

[28] R. Srikant. *The Mathematics of Internet Congestion Control.* Birkhauser, 2004.

[29] A. Stolyar and K. Ramanan. Largest weighted delay first scheduling: Large deviations and optimality. In *Ann. Appl. Probab.* 11:1-48, 2001.

[30] A. Stolyar. Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm. Submitted.

[31] L. Tassiulas and A. Ephremides. Dynamic server allocation to parallel queues with randomly varying connectivity. In *IEEE Transactions on Information Theory*, 39:466-478, March 1993.

[32] S. Toumpis and A. J. Goldsmith Large wireless network under fading, mobility, and delay constraints. In *Procedings of IEEE INFOCOM*, 2004.

[33] G. Vinnicombe. On the stability of networks operating TCP-like congestion control. In *Proceedings of the IFAC World Congress*, Barcelona, Spain, 2002. University of Cambridge Technical Report CUED/F-INFENG/TR.398. Available at http://www.eng.cam.ac.uk/~gv.

[34] P. Viswanath, D. Tse, and R. Laroia. Opportunistic beamforming using dumb antennas. In *IEEE Transactions on Information Theory*, 48(6):1277C1294, June 2002.

[35] F. Xue and P. R. Kumar. The number of neighbors needed for connectivity of wireless networks. Wireless Networks, pp. 169 181, vol.10, no. 2, March 2004.

[36] Ying, L., G. E. Dullerud, and R. Srikant, "Global Stability of Internet Congestion Controllers with Heterogeneous Delays," IEEE Transactions on Networking; to appear June 2006.

[37] Ying, L. , R. Srikant, A. Eryilmaz, and G. E. Dullerud, "A Large Deviations Analysis of Scheduling in Wireless Networks", submitted to IEEE Transactions on Information Theory; conference version in *Proceedings of the IEEE Conference on Decision and Control*, 2005.

[38] Ying, L. , R. Srikant, A. Eryilmaz, and G. E. Dullerud, "Distributed Fair Resource Allocation in Cellular Networks in the Presence of Heterogeneous Delays", submitted to IEEE Transactions on Automatic Control; conference version in *Proceedings of the Intl. Symposium on Modelling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, 2005.

[39] Ying, L. , R. Srikant, and G. E. Dullerud, "Distributed Symmetric Function Computation in Noisy Wireless Sensor Networks with Binary Data", *Proceedings of the Intl. Symposium on Modelling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, 2006.

[40] C. N. Hadjicostis, "Coding Approaches to Fault Tolerance in Combinational and Dynamic Systems." Kluwer Academic Publishers, 2002.

[41] S. Sundaram and C. N. Hadjicostis, "Non-Concurrent Error Detection and Correction in Switched Linear Controllers." International Workshop on Hybrid Systems: Computation and Control (Series Lecture Notes in Computer Science, vol. 2993), pp. 585-599, Springer-Verlag, 2004.

[42] S. Sundaram and C. N. Hadjicostis, "Error Detection and Correction in Switched Linear Controllers via Periodic and Non-Concurrent Checks." To appear in Automatica.

[43] S. Sundaram and C. N. Hadjicostis, "Comments on "Time-Delayed State Estimator for Linear Systems with Unknown Inputs"," International Journal on Control, Automation and Systems, vol. 3, no. 4, pp. 646-647, December 2005.

[44] C. N. Hadjicostis, "Probabilistic Fault Detection in Finite-State Machines Based on State Occupancy Measurements," IEEE Transactions on Automatic Control, vol. 50, no. 12, pp. 2078-2083, December 2005.

[45] Y. Wu and C. N. Hadjicostis, "Algebraic Approaches for Centralized and Distributed Fault Identification in Discrete Event Systems," IEEE Transactions on Automatic Control, vol. 50, no. 12, pp. 2048-2053, December 2005.

[46] C. N. Hadjicostis, "Aliasing Probability Calculations for Arbitrary Compaction under Independently Selected Random Test Vectors," IEEE Transactions on Computers, vol. 54, no. 12, pp. 1614-1627, December 2005.

[47] E. Athanasopoulou and C. N. Hadjicostis, "Probabilistic Approaches to Fault Detection in Networked Discrete Event Systems," IEEE Transactions on Neural Networks, vol. 16, no. 5, pp. 1042-1053, September 2005 (Special Issue on Adaptive Learning Systems in Communication Networks).

[48] C. N. Hadjicostis, "Finite-State Machine Embeddings for Non-Concurrent Error Detection and Identification," IEEE Transactions on Automatic Control, vol. 50, no. 2, pp. 142-153, February 2005.

[49] C. N. Hadjicostis and G. C. Verghese, "Coding Approaches to Fault Tolerance in Linear Dynamic Systems," IEEE Transactions on Information Theory, vol. 51, no. 1, pp. 210-228, January 2005.

[50] C. N. Hadjicostis, "Non-Concurrent Error Detection and Identification in One-Hot Encoded FSMs," Automatica, vol. 40, pp. 1665-1676, 2004.

[51] C. N. Hadjicostis, "Coding Techniques for Fault-Tolerant Parallel Prefix Computations in Abelian Groups," The Computer Journal, vol. 47, no. 3, pp. 329-341, 2004.

[52] C. N. Hadjicostis, "Non-Concurrent Error Detection and Correction in Fault-Tolerant Linear Finite-State Machines," IEEE Transactions on Automatic Control, vol. 48, no. 12, pp. 2133-2140, December 2003.

[53] C. N. Hadjicostis, "Non-Concurrent Error Detection and Correction in Fault-Tolerant Discrete-Time LTI Dynamic Systems," IEEE Transactions on Circuits and Systems (I), vol. 50, no. 1, pp. 45-55, January 2003.

[54] C. N. Hadjicostis and G. C. Verghese, "Encoded Dynamics for Fault Tolerance in Linear Finite-State Machines," IEEE Transactions on Automatic Control, vol. 47, no. 1, pp. 189-192, January 2002.

[55] R. Touri, P. G. Voulgaris and C. N. Hadjicostis, "Time-Varying Power-Limited Preprocessing for Perfect Reconstruction of Binary Signals." To appear in the Proceedings of ACC 2006, the 2006 American Control Conference.

[56] E. Athanasopoulou and C. N. Hadjicostis, "Decentralized Failure Diagnosis in Discrete Event Systems." To appear Proceedings of ACC 2006, the 2006 American Control Conference.

[57] P. G. Voulgaris, C. N. Hadjicostis and R. Touri, "Encoder-Decoder Design for Perfect Reconstruction: A Robust Control Perspective." Proceedings of CDC/ECC 2005, the joint 44th IEEE Conference on Decision and Control and the European Control Conference 2005 (invited), pp. 2536-2541, Seville, Spain, 2005.

[58] S. Sundaram and C. N. Hadjicostis, "On Delayed Observers for Linear Systems with Unknown Inputs." Proceedings of CDC/ECC 2005, the joint 44th IEEE Conference on Decision and Control and the European Control Conference 2005, pp. 7210-7215, Seville, Spain, 2005.

[59] Athanasopoulou and C. N. Hadjicostis "Maximum Likelihood Diagnosis in Partially Observable Finite-State Machines." Proceedings of MED 2005, the 13th IEEE Mediterranean Conference on Control and Automation (invited), Limassol, Cyprus, 2005.

[60] G. Takos and C. N. Hadjicostis, "Hierarchical Decentralized Fusion from Correlated Sensor Measurements." Proceedings of ICNSC 2005, the 2005 IEEE International Conference on Networking, Sensing and Control.

[61] E. Athanasopoulou and C. N. Hadjicostis "Synchronization-Based Fault Detection in Discrete Event Systems." Proceedings of CDC 2004, the 43rd IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas, 2004.

[62] Lingxi Li, C. N. Hadjicostis and R. Sreenivas, "Fault Detection and Identification in Petri Net Controllers." Proceedings of CDC 2004, the 43rd IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas, 2004.

[63] C. N. Hadjicostis, "Finite-State Machine Embeddings for Non-Concurrent Error Error Detection and Identification," Proceedings of CDC 2003, the 42nd IEEE Conference on Decision and Control, vol. 4, pp. 3215-3220, Maui, Hawai, 2003.

[64] P. G. Voulgaris, C. N. Hadjicostis and R. Touri, "A Perfect Reconstruction Paradigm for Digital Communication," Proceedings of CDC 2003, the 42nd IEEE Conference on Decision and Control, vol. 4, pp. 4196-4201, Maui, Hawaii, 2003.

[65] Y. Wu and C. N. Hadjicostis, "Distributed Non-Concurrent Fault Identification in Discrete Event Systems." Proceedings of CESA 2003, the Multiconference on Computational Engineering in Systems Applications (invited), Lille, France, 2003.

[66] E. Athanasopoulou and C. N. Hadjicostis, "Aliasing Probability Calculations in Testing Sequential Circuits." Proceedings of MED 2003, the 11th IEEE Mediterranean Conference on Control and Automation, Rhodes, Greece, 2003.

[67] C. N. Hadjicostis, "Encoded Finite-State Machines for Non-Concurrent Error Detection and Identification." Proceedings of ISCAS 2003, the 2003 IEEE International Symposium on Circuits and Systems (invited), vol. 3, pp. 858-861, Bangkok, Thailand, 2003.

[68] C. N. Hadjicostis, "Aliasing Probability Calculations in Nonlinear Compactors." Proceedings of ISCAS 2003, the 2003 IEEE International Symposium on Circuits and Systems, vol. 5, pp. 529-532, Bangkok, Thailand, 2003.

[69] C. N. Hadjicostis and R. Touri, "Feedback Control utilizing Packet Dropping Network Links." Proceedings of CDC 2002, the 41st IEEE Conference on Decision and Control (invited), vol. 2, pp. 1205-1210, Las Vegas, NV, 2002.

[70] Y. Wu and C. N. Hadjicostis, "Non-Concurrent Fault Identification in Discrete Event Systems using Encoded Petri Net States." Proceedings of CDC 2002, the 41st IEEE Conference on Decision and Control (invited), vol. 4, pp. 4018-4023, Las Vegas, Nevada, 2002.

[71] C. N. Hadjicostis, "Probabilistic Fault Detection in Finite-State Machines Based on State Occupancy Measurements." Proceedings of CDC 2002, the 41st IEEE Conference on Decision and Control (invited), vol. 4, pp. 3994-3999, Las Vegas, Nevada, 2002.

[72] Papachristodoulou, A; Li, L; Doyle, JC, 2004. Methodological frameworks for large-scale network analysis and design, Computer Communication Review, Jul 2004

[73] Li, L., D. Alderson, J. Doyle, and W. Willinger. 2004. A First-Principles Approach to Understanding the Internet's Router-level Topology. Proc. ACM Sigcomm, Computer Communication Review, Oct. 2004.

[74] Csete M.E. and J.C. Doyle, 2004, Bow ties, metabolism, and disease, Trends in Biotechnology, Vol 22, Issue 9, pg. 446-450

[75] J. Stelling, U. Sauer, Z. Szallasi, F. J. Doyle III, and J. Doyle, 2004, Robustness of cellular functions, Cell, October, 2004.

[76] Willinger, W, D. Alderson, J.C. Doyle, and L.Li., 2004. More "Normal" Than Normal: Scaling Distributions and Complex Systems. Proceedings of the 2004 Winter Simulation Conference.

[77] Shapiro, BE; Hucka, M; Finney, A; Doyle, J. 2004. MathSBML: a package for manipulating SBML-based biological models, Bioinformatics, Nov 1, 2004

[78] Doyle, J; Csete, M, 2004. Imitation of life: How biology is inspiring computing, Nature. Oct 21 2004

[79] Kitano, H; Oda, K; Kimura, T; Matsuoka, Y; Csete, M; Doyle, J; Muramatsu, M, 2004 .Metabolic syndrome and robustness tradeoffs, DIABETES 53: S6-S15 Suppl. 3, Dec 2004

[80] H. El-Samad, H. Kurata , J.C. Doyle , C.A. Gross, and M. Khammash, 2005, Surviving Heat Shock: Control Strategies for Robustness and Performance, PNAS (8): 2736-2741 Feb 22, 2005

[81] Jin C, Wei D, Low SH, Bunn J, Choe HD, Doyle JC,et al, FAST TCP: From theory to experiments IEEE Network 19 (1): 4-11 Jan-Feb 2005

[82] Paganini F, Wang ZK, Doyle JC, et al. Congestion control for high performance, stability, and fairness in general networks, IEEE-ACM Transactions On Networking 13 (1): 43-56 Feb 2005

[83] W. Willinger, J Doyle, Robustness and the Internet: Design and evolution, in Robust Design: A Repertoire of Biological, Ecological, and Engineering Case Studies (Santa Fe Institute Studies on the Sciences of Complexity), Erica Jen, Editor

[84] L. Chen, S. H. Low and J. C. Doyle. Joint congestion control and media access control design for wireless ad hoc networks, IEEE Infocom, Miami, FL, March 2005.

[85] Wang JT, Li L, Low SH, Doyle JC. (2005) Cross-layer optimization in TCP/IP networks, IEEE-ACM Transactions On Networking 13 (3): 582-595 Jun 2005

[86] Manning M, Carlson JM, Doyle J (2005) Highly optimized tolerance and power laws in dense and sparse resource regimes Physical Review E 72 (1): Art. No. 016108 Part 2 Jul 2005

[87] R. Tanaka, T-M Yi, and J. Doyle (2005) Some protein interaction data do not exhibit power law statistics, FEBS letters, 579 (23): 5140-5144 Sep 26 2005

[88] Doyle et al, (2005), The "Robust Yet Fragile" Nature of the Internet, P Natl Acad Sci USA. vol. 102 no. 41, October 11, 2005

[89] Zhou T, Carlson JM, Doyle J (2005) Evolutionary dynamics and highly optimized tolerance, Journal Of Theoretical Biology 236 (4): 438-447 Oct 21 2005

[90] L Li, D Alderson, JC Doyle, W Willinger (2005) Towards a Theory of Scale-Free Graphs: Definition, Properties, and Implications, Internet Math, to appear

[91] D Alderson, L Li, W Willinger, JC Doyle (2005) Understanding Internet Topology: Principles, Models, and Validation, IEEE/ACM Transactions On Networking, Dec. 2005.

[92] MA Moritz, ME Morais, LA Summerell, JM Carlson, J Doyle (2005) Wildfires, complexity, and highly optimized tolerance, P Natl Acad Sci USA, December 13, 2005; 102 (50),

[93] T Brookings, JM Carlson, and J Doyle (2005) Three mechanisms for power laws on the Cayley tree, Phys. Rev. E 72, 056120

[94] S. Boyd, P. Diaconis, P. Parrilo, and L. Xiao. Symmetry analysis of reversible Markov chains. To appear in *Internet Mathematics*, 2005.

[95] S. Boyd, P. Diaconis, J. Sun, and L. Xiao. Fastest mixing Markov chain on a path. To appear in *The American Mathematical Monthly*, 2005.

[96] S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing Markov chain on a graph. *SIAM Review*, 46(4):667–689, 2004.

[97] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. Presented at *INFOCOM 2005*.

[98] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Mixing times for random walks on geometric random graphs. Presented at *SIAM ANALCO 2005*.

[99] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Analysis and optimization of randomized gossip algorithms. In *Proceedings of IEEE Conference on Decision and Control (CDC)*, volume 5, pages 5310–5315, Bahamas, December 2004.

[100] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. Submitted to *IEEE Transactions on Information Theory*, March 2005.

[101] C. Langbort, L. Xiao, R. D'Andrea, and S. Boyd. A decomposition approach to distributed analysis of networked systems. In *Proceedings of IEEE Conference on Decision and Control (CDC)*, volume 4, pages 3980 3985, Bahamas, December 2004.

[102] J. Sun, S. Boyd, L. Xiao, and P. Diaconis. The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem.

[103] L. Xiao and S. Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53(1):65 78, 2004.

[104] L. Xiao and S. Boyd. Optimal scaling of a gradient method for distributed resource allocation. Revised for publication in *Journal of Optimization Theory and Applications*, February 2005. Available at `http://www.stanford.edu/~boyd/fast_redstb.html`.

[105] L. Xiao, S. Boyd, and S.-J. Kim. Distributed average consensus with least-mean-square deviation. Submitted to *Journal of Parallel and Distributed Computing*, May 2005. Available at `http://www.stanford.edu/~boyd/lms_consensus.html`.

[106] L. Xiao, S. Boyd, and S. Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN)*, pages 63–70, Los Angeles, CA, April 2005.

[107] D. S. Lun, M. Medard, and R. Koetter. Efficient operation of wireless packet networks using network coding.. In Proc. International Workshop on Convergent Technologies (IWCT) 2005, June 2005. Invited paper.

[108] S. Deb, M. Effros, T. Ho, D. R. Karger, R. Koetter, D. S. Lun, M. Medard, and N. Ratnakar. Network coding for wireless applications: A brief tutorial. In Proc. International Workshop on Wireless Ad-hoc Networks (IWWAN) 2005, May 2005. Invited paper.

[109] D. S. Lun, N. Ratnakar, R. Koetter, M. Medard, E. Ahmed, and H. Lee. Achieving minimum cost multicast: A decentralized approach based on network coding. In Proc. IEEE Infocom 2005, March 2005.

[110] D. S. Lun, M. Medard, and M. Effros. On coding for reliable communication over packet networks. In Proc. 42nd Annual Allerton Conference on Communication, Control, and Computing, September-October 2004. Invited paper.

[111] Laura Zager, Graph Similarity and Matching, Masters thesis, EECS Department, MIT, May 2005.

[112] Victor Preciado and George Verghese, "Synchronization in Generalized Erdos-Renyi Networks of Nonlinear Oscillators," IEEE Conf. on Decision and Control, Sevilla, Spain, December 2005.

[113] Ernst Scholtz, Observer-Based Monitors and Distributed Wave Controllers for Electromechanical Disturbances in Power Systems, PhD thesis, EECS Department, MIT, August 2004.

[114] Ernst Scholtz, Bernard Lesieutre and GeorgeVerghese, "Decentralized Controllers for Electromechanical Waves in Power Networks," Proc. 36th North American Power Symposium, August 2004, Moscow, Idaho.

[115] Ernst Scholtz, George Verghese and Bernard Lesieutre, "Observer-Based Monitors for Electromechanical Dynamics in Power Networks," Proc. 15th Power System Computation Conference, August 2005, Liege, Belgium.

[116] Teruo Ono, Game Theoretic Analysis and Agent-Based Simulation of Electricity Markets, Masters thesis, EECS Department, May 2005.

[117] Teruo Ono and George Verghese, "Replicator Agents for Electricity Markets," 13th International Conf. on Intelligent Systems Application to Power Systems (ISAP 2005), Washington DC, November 2005.

[118] M. Johansson, L. Xiao, and S. Boyd. Simultaneous routing and resource allocation in CDMA wireless data networks. Submitted to *The 2003 IEEE International Conference on Communications*, August 2002.

[119] S. Lall and C. Beck. Error bounds for balanced model reduction of linear time-varying systems. *IEEE Transactions on Automatic Control*, 48(6):946–956, June 2003.

[120] L. Xiao, M. Johansson, and S. Boyd. Simultaneous routing and resource allocation via dual decomposition. Submitted to *IEEE Transactions on Communications*, August 2002.

[121] M. Rotkowitz and S. Lall. Decentralized control information structures preserved under feedback. In *Proceedings of the IEEE Conference on Decision and Control*, pages 569–575, December 2002.

[122] R. Cogill and S. Lall. Topology independent controller design for networked systems. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 1788–1793, 2004.

[123] M. Rotkowitz and S. Lall. On computation of optimal controllers subject to quadratically invariant sparsity constraints. In *Proceedings of the American Control Conference (ACC)*, pages 5659–5664, June 2004.

[124] B.-D. Chen and S. Lall. Control of distributed discrete-time systems on graphs. In *Proceedings of the American Control Conference (ACC)*, pages 2251–2256, June 2004.

Approximate Fairness through Differential Dropping

Rong Pan, Stanford University
Lee Breslau, AT&T Research Labs
Balaji Prabhakar, Stanford University
Scott Shenker, ACIRI

Abstract—**Many researchers have argued that the Internet architecture would be more robust and more accommodating of heterogeneity if routers allocated bandwidth fairly. However, most of the mechanisms proposed to accomplish this, such as Fair Queueing [16], [6] and its many variants [2], [23], [15], involve complicated packet scheduling algorithms. These algorithms, while increasingly common in router designs, may not be inexpensively implementable at extremely high speeds; thus, finding more easily implementable variants of such algorithms may be of significant practical value. This paper proposes an algorithm that – similar to FRED [13], CSFQ [24], and several other designs [17], [14], [5], [25] – combines FIFO packet scheduling with *differential dropping* on arrival. Our design, called *Approximate Fair Dropping* (AFD), bases these dropping decisions on the recent history of packet arrivals. AFD retains a simple forwarding path and requires an amount of additional state that is small compared to current packet buffers. Simulation results, which we describe here, suggest that the design provides a reasonable degree of fairness in a wide variety of operating conditions. The performance of our approach is aided by the fact that the vast majority of Internet flows are slow but the fast flows send the bulk of the bits. This allows a small sample of recent history to provide accurate rate estimates of the fast flows.**

I. Introduction

Since the pioneering observations of Nagle [16], many researchers have argued that the Internet architecture would be more robust (in the face of ill-behaved flows) and more accommodating of heterogeneity (by no longer requiring adherence to a single congestion control algorithm) if routers allocated bandwidth fairly. This viewpoint is not universally shared,[1] but even among adherents of this approach the question of feasibility has been a major concern. This is because most of the proposed mechanisms, such as Fair Queueing [16], [6] and its many variants [2], [23], [15], involve complicated packet scheduling algorithms. These algorithms, while increasingly common in router designs, may not be inexpensively implementable at extremely high speeds; thus, finding more easily implementable variants of such algorithms may be of significant practical value. This paper focuses on the design of a router mechanism that combines approximately fair bandwidth allocations with relatively low implementation complexity.

We have three basic requirements of our design. First, it should achieve reasonably fair bandwidth allocations, and by "fair" we mean the max-min definition of fairness [6]. We make no pretense at being able to match the packet-by-packet fairness of Fair Queueing or other schemes that use intricate packet scheduling algorithms to ensure that perfect fairness is achieved on very short time scales. Instead, we aim for approximate fairness over longer time scales, on the order of several roundtrip times.

Second, we require that the design be easily amenable to high speed implementations. To this end we limit ourselves to FIFO packet scheduling algorithms with probabilistic drop-on-arrival; as in RED [8], when a packet arrives either the packet is dropped or placed on a FIFO queue. The dropping decisions must be simple with $O(1)$ complexity (that is, the complexity must not increase with the number of flows or packets). The amount of state required to make these dropping decisions must be small; the point of reference we choose to define "small" is the amount of memory which is already devoted to the packet buffers.

Third, the algorithm must employ some form of active queue management (AQM). It need not mimic RED or any other form of AQM precisely, but it should embody AQM's fundamental principles of responding early (and gently) to congestion while absorbing small bursts of traffic.[2]

To achieve these goals, we propose an algorithm called Approximate Fair Dropping (AFD). Its structure is similar to RED in that it uses a FIFO queue with probabilistic

[1] We discuss other viewpoints in Section VII.

[2] There is a fourth requirement that, for lack of space, we do not discuss in this paper; this is the requirement that the scheme be able to *punish* unresponsive flows. That is, the scheme should not just allocate bandwidth fairly, but should be able to shut down (by dropping all their packets) flows that incur high drop rates for long periods of time. The rationale for this is persuasively described in [10]; [24] discusses the role of fair bandwidth allocation in implementing this. We are easily able to augment our design to meet this goal and, as we see in Section V, one of the designs we present here already (but unintentionally!) achieves this goal.

Reprinted with permission from *ACM SIGCOMM Computer Communication Review*, Vol. 33, No. 2 (April 2003), 23–39.

drop-on-arrival. In contrast to RED, however, these probabilistic dropping decisions are based not only on the past measurements of the queue size but also on the recent history of a packet's flow.[3] AFD uses the history to estimate the flow's current sending rate, and then drops with a probability designed to limit that flow's throughput to the fair share.

Thus, dropping is not applied uniformly across flows (as in RED) but is applied differentially based on an estimate of the flow's current sending rate. The exact form of differential dropping is chosen to approximate fair bandwidth allocations. Several recent designs have advocated such FIFO-based *differential dropping* designs, including FRED [13], CSFQ and its several extensions [24], [5], [25], and RED-PD [14]. AFD adopts many aspects of these proposals.

Our design might initially appear to be seriously misguided. The state required to keep enough history to accurately estimate each flow's rate is quite large, and grows linearly with the number of flows. This is clearly not a feasible approach. However, note that our design only needs to estimate the rates of flows whose packets are likely to be dropped. Thus, we need only keep enough state to estimate rates when those rates are comparable to (or larger than) the fair share rate. This is a crucial difference.

It is well known that the distribution of the sizes of flows (the total number of bytes transferred) has a long tail, and that most flows are small – the mice – but a large fraction of the bytes are sent by large flows – the elephants.[4] As we show using trace data in Section VI, and has been observed in [14], the distribution of flow *rates* has a similar property. While perhaps not as long-tailed as the size distribution, initial evidence suggests that the flow rate distribution (measured on the time scale of a second) is long-tailed. Most flows are *slow* (in bits/sec) – we will call them *turtles* – but most of the bytes are sent by fast flows – we will call them *cheetahs*.

In AFD, we only need state for the fast flows and can ignore the slow flows because they won't be dropped; the amount of state required is roughly proportional to the number of fast flows, which is manageable. However, the challenge is how to keep state only on the fast flows when one doesn't know, in advance, which flows are fast or slow. A record of the very recent packet arrivals contains mostly packets from the fast flows (because they send most of the bytes), while most of the slow flows won't show up in the recent history. This is the approach we take.

This paper has VII sections. We describe a conceptual version of our design in Section II, and then use analysis to develop guidelines for the various parameters in Section III. We then present the practical design in Section IV. We evaluate this design using simulation in Section V, and use trace data and other measurements to estimate the state requirements for the design in Section VI. We conclude in Section VII with a discussion of related work.

II. Conceptual Design

We first present a high level conceptual design of AFD. Consider a link of speed C shared by n flows, each sending at rate r_i. Assume, for convenience, that all packets are the same size P. The total load on the link is $R = \sum_i r_i$. The fair share r_{fair} is given by the solution to $C = \sum_i \min(r_i, r_{fair})$. We can use differential dropping to accomplish our goal of fair bandwidth allocations if we use dropping probabilities for each flow, d_i, given by the relation $d_i = (1 - \frac{r_{fair}}{r_i})_+$.[5] The resulting throughput of each flow is bounded by the fair share: $r_i(1 - d_i) = \max(r_i, r_{fair})$.

The key design question is: how can we estimate r_i and r_{fair}. To estimate r_i we keep a *shadow buffer* of b recent packet headers. When a packet arrives, with probability $\frac{1}{s}$, where s is the sampling interval, we copy its header and insert it into the shadow buffer; thus, we sample roughly 1 in s packets. When a packet is inserted into the shadow buffer, we remove another packet at random to keep the total number of packets at b.[6] Thus, at all times the shadow buffer has a record of b recent packet arrivals. Note that the shadow buffer contains copies of the packet headers, and the insertion and deletion of packets from the shadow buffer is parallel to the main forwarding of packets. The shadow buffer is used to guide the dropping decisions in the following manner. When a packet (from, say, flow i) arrives (regardless of whether its header is copied into the packet buffer) we compute the number of packets from that flow in the shadow buffer; we call this the number of *matches*, m_i. We then estimate the rate of that flow to be $r_i^{est} = R\frac{m_i}{b}$ (recall that R is the aggregate arrival rate).

To avoid having to scan the shadow buffer each time a packet arrives, we keep a table of the flows and their current packet counts; we call this the *flow table*. A hash table is one natural data structure for the flow table that has $O(1)$ lookup time. Each time a packet is inserted into the shadow buffer, the appropriate counter in the flow ta-

[3] In this paper, we define a flow as a stream of packets with the same source-destination addresses. However, one could use any definition of flow discernible from an IP packet header.

[4] As we discuss later in Section VI, we use the term "long-tail" to describe distributions that decay slower than exponentially.

[5] We use the notation that $z_+ = \max(0, z)$.

[6] Random removal avoids synchronization problems. We could also remove packets in a FIFO manner; it turns out that this does not affect performance greatly.

ble is incremented, each time a packet is removed the appropriate count is decremented, and every time a packet arrives the match count is looked up in the flow table. Insertions and deletions can be somewhat slow (because we need only insert and delete every s packets) but the lookup has to done at linespeed (because it is done on every packet arrival).

Estimating the rate of a single flow requires only the recent activity of that flow and so is fairly straightforward. Estimating r_{fair} is more complicated because the definition of r_{fair} depends on all the r_i. We use an approach borrowed from [24]. Upon each packet arrival we apply the dropping probability $d_i = (1 - \frac{r_{fair}}{r_i^{est}})_+$, which can be rewritten as $d_i = (1 - \frac{m_{fair}}{m_i})_+$ with $m_{fair} = b\frac{r_{fair}}{R}$. Note that if we vary m_{fair} the total throughput, $\sum_i r_i d_i$, varies from R when $m_{fair} = \infty$, to C when $m_{fair} = b\frac{r_{fair}}{R}$, to 0 when $m_{fair} = 0$. Thus, we can approximate r_{fair} by varying m_{fair} so that the link is fully utilized but not overloaded.

In our approach, we vary m_{fair} to ensure that the average queue length stabilizes around the target value of $0.5(min_{th} + max_{th})$ where min_{th} and max_{th} are the threshold values defined in RED. We borrow the AQM approach described in [12], which employs a proportional-integral controller. Since this form of AQM is very effective at keeping the links fully utilized, the resulting values of $m_{fair}\frac{R}{b}$ will be a good approximation to r_{fair}.

Our basic approach has borrowed liberally from CSFQ [24], FRED [13], and CHOKe [18], and is quite similar to the concurrently developed RED-PD [14].[7] FRED uses the number of packets from each flow in the packet buffer to guide its dropping decisions; thus, our design can be seen as an extension of FRED to use more history and a different dropping function (which is the same as used in CSFQ).

Does AFD achieve the goals we laid out in the Introduction? AFD's probabilistic drop-on-arrival can be implemented with a very simple forwarding path. The dropping decision has complexity $O(1)$ and involves few computations. In addition, AFD incorporates active queue management. Fairness and feasibility (in terms of the state required) are the two remaining questions. If the rate estimates are accurate, we expect AFD to allocate bandwidth fairly. While we delay the bulk of our simulation results, and all of the simulation details, until Section V, we now show initial evidence that this expected fairness is actually achieved. Figure 1 shows the result of 50 CBR sources

[7] In fact, we had several design discussions with the RED-PD designers while both algorithms were being developed. While the two schemes have some similarities, they are targeted at slightly different goals. See Section VII.

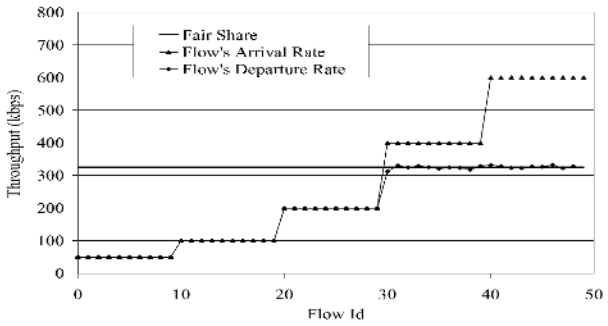


Fig. 1. Offered Load and Throughput for 50 CBR Flows under AFD

sending at five different rates (some above the fair share, some below) over a single link. The bandwidth allocations are quite fair; flows sending below the fair share incur no drops, and flows sending above the fair share have their excess packets dropped.

Fairness depends on the accuracy of the rate estimations, and the rate estimations depend on the size of the shadow buffer. Without sufficient history the rate estimations will fluctuate wildly and lead to quite unpredictable (and unfair) results. Rate estimation improves with larger b (the size of the shadow buffer) and, to some extent, with larger s (because s increases the time interval represented by the contents of the shadow buffer). The question is how large do b and s have to be in order to achieve reasonable fairness. This is crucial because we want the state required for this algorithm to be small (at least compared to the packet buffers). The viability of our scheme comes down to whether fairness can be achieved without too much state, and we explore that issue in the next few sections.

In Section III we develop three guidelines for how to set b and s. After describing a more practical (as opposed to the conceptual version just presented) version of AFD in Section IV, we evaluate the fairness that results from such parameter settings in Section V. Finally, in Section VI we estimate the state requirements of AFD when using the parameter-setting guidelines.

III. Setting the Basic Parameters

In this conceptual model, there are two basic parameters that we need to set: the shadow buffer size b and sampling interval s. Our goal is to achieve a reasonable degree of fairness, and in this section we develop three *guidelines* on what values of b and s achieve that goal. To do so, we consider three scenarios and analyze them using very simple

models. These models are unrealistic but our hope is that the resulting guidelines provide useful order-of-magnitude estimates for the parameters of interest.

A. Static Scenario

We first consider the static case of n sources sharing a single link of bandwidth C. All flows are Poisson sources sending at a constant rate r_i and all packets are the same size. The overall drop rate D is given by $D = \frac{(R-C)_+}{R}$ where, as above, $R = \sum_i r_i$. Let $P_i(m)$ be the probability that when a packet from flow i arrives, there are m packets from flow i in the shadow buffer. In our simple model,

$$P_i(m) = \binom{b}{m}(1 - \frac{r_i}{R})^{b-m}(\frac{r_i}{R})^m$$

For the purposes of this model, we assume that m_{fair} is fixed to be the appropriate value: $m_{fair} = b^{\frac{r_{fair}}{R}}$. We denote the *ideal* dropping rate by $\tilde{d}_i$: $\tilde{d}_i = (1 - \frac{r_{fair}}{r_i})_+$. $d_i(b)$ is the average dropping rate of the i'th flow for a given size of shadow buffer:

$$d_i(b) = \sum_{m=0}^{b} P_i(m)(1 - \frac{m_{fair}}{m})_+$$

The relative error in the throughput of the i'th flow is

$$\Delta_i(b) = |\frac{r_i(1 - d_i(b)) - r_i(1 - \tilde{d}_i)}{r_i(1 - d_i)}| = |\frac{\tilde{d}_i - d_i(b)}{1 - d_i}|$$

Note that the sampling interval s drops out of all of these quantities. This is not surprising given that a random sampling of a Poisson process remains a Poisson process.

Our goal is to choose the size of the shadow buffer so that we achieve some reasonable degree of fairness. We seek to determine how large the shadow buffer must be so that the maximal error, $\max_i \Delta_i(b)$, is below some target error tolerance E. Analysis and numerical calculations (which we do not have space for here) indicate the maximal error occurs when $r_i \approx r_{fair}$. Consider a flow with $r_i = r_{fair}$; we can approximate the relative error, call it Δ, in the limit of large b for this flow as follows (noting that the ideal dropping rate for this flow is zero, $\tilde{d}_i = 0$, and so $\Delta(b) = d_i(b)$). Letting $\alpha = \frac{r_{fair}}{R}$, we have:

$$\Delta(b) = \sum_{m=0}^{b} \binom{b}{m}(1-\alpha)^{b-m}\alpha^m(1 - \frac{b\alpha}{m})_+$$

$$= \sum_{m=\lceil b\alpha \rceil}^{b} \binom{b}{m}(1-\alpha)^{b-m}\alpha^m(1 - \frac{b\alpha}{m})$$

For sufficiently large b, we can approximate the binomial form as a normal distribution with average αb and variance $b\alpha(1-\alpha)$. Using an integral formulation, we find:

$$\Delta(b) \approx \frac{1}{\sqrt{2\pi b\alpha(1-\alpha)}} \int_{b\alpha}^{\infty} dx\, e^{-\frac{(x-b\alpha)^2}{2b\alpha(1-\alpha)}}(1 - \frac{b\alpha}{x})_+$$

$$< \frac{1}{\sqrt{2\pi b\alpha(1-\alpha)}} \int_{0}^{\infty} dy\, e^{-\frac{y^2}{2b\alpha(1-\alpha)}} \frac{y}{b\alpha}$$

$$= \sqrt{\frac{2(1-\alpha)}{\pi b\alpha}} \int_0^{\infty} dz\, z e^{-z^2} \approx .40\sqrt{\frac{(1-\alpha)}{b\alpha}}$$

Thus, we can bound the asymptotic expression for the relative error from above by the formula

$$\Delta(b) \leq .40(b\alpha)^{-0.5}$$

A choice of b that will always meet the error tolerance of E (in this asymptotic limit) is:

$$b^{\frac{r_{fair}}{R}} > (\frac{E}{0.40})^{-2}$$

Note that $b^{\frac{r_{fair}}{R}} = m_{fair}$ where, as defined above, m_{fair} is the expected number of shadow buffer matches for a flow with $r_i = r_{fair}$. One can meet the error tolerance merely by making the shadow buffer big enough to make sure that $m_{fair} > (\frac{E}{0.40})^{-2}$. For our purposes, we choose $E \approx .15$. This might seem quite unambitious, but recall that we are calculating an upper bound on the worst-case error (that is, of flows sending at a constant rate right at the fair share); the errors for rates well above and well below the fair share are significantly smaller. Setting $E = 0.15$ results in $m_{fair} \approx 10$. This is our first guideline.

Guideline 1: $b \gtrapprox 10\frac{R}{r_{fair}}$ or, equivalently, $m_{fair} \gtrapprox 10$

B. Dynamic Scenario

Even if routers allocate bandwidth fairly on the time scale over which they measure usage, flows that respond slowly when excess bandwidth is available are at a disadvantage when competing with flows that respond more rapidly. We can illustrate this with a simple model involving flows with different roundtrip times (RTTs).

Consider flows competing for bandwidth with differing roundtrip times τ_i. All packets are the same size P. Each flow adjusts its window size w_i in a TCP-like fashion (with all the details of TCP omitted); the window (measured in terms of packets) increases by one in time τ_i when there is no packet loss during that RTT, and the window is halved if there is a packet loss (and we neglect the case of multiple

packet losses, or of timeouts, in an RTT). The rates r_i of the flows are roughly $r_i = \frac{w_i P}{\tau_i}$. The router has a shadow buffer of size b and a sampling interval of s; the time interval T over which the shadow buffer measures usage is $T = \frac{bsP}{R}$. We assume that the fair share rate r_{fair} is constant and that the fair share expected number of matches $m_{fair} = b\frac{r_{fair}}{R}$ is roughly 10 (as suggested by guideline 1). Consider a flow i and let m_i denote the average number of its matches in the shadow buffer. Because the fair share is fixed, we can approximate the dynamics of each flow as independent.[8] We consider a series of time slots $t = 1, 2, \ldots$, each of length τ_i. For convenience, we assume that $T = k_i\tau_i$ for some integer k_i. The average number of matches m_i in the shadow buffer at some time t is roughly $m_i(t) = \frac{1}{s}\sum_{j=0}^{k_i-1} w_i(t-j)$. Thus, the window dynamics can be written as:

$$w_i(t+1) = w_i(t) + 1 \ \ (m_i \leq m_{fair})$$

$$w_i(t+1) = \frac{w_i(t)}{2} \ \ (m_i > m_{fair})$$

$$m_i(t) = \frac{1}{s}\sum_{j=0}^{k_i-1} w_i(t-j)$$

If we start off with $w_i(t=0) = 1$, the first drop for flow i occurs (approximately) when $w_i(t) = s\frac{m_{fair}}{k_i} + \frac{k_i}{2}$ (which we assume is an integer). At that point, the window is halved to $w_i(t) = \frac{sm_{fair}}{2k_i} + \frac{k_i}{4}$. If this halving of the window brings the number of matches down below m_{fair} in the next time period then the increasing process starts again. If not, the window is halved again (and again) until $m_i(t)$ is below m_{fair}. Let's assume that the window is halved only once; in this case $w_i(t)$ becomes a repeating series of values starting at $w_i(t) = \frac{sm_{fair}}{2k_i} + \frac{k_i}{4}$ and increasing by 1 until the value $w_i(t) = \frac{sm_{fair}}{k_i} + \frac{k_i}{2}$ is reached. The average window over this periodic cycle of flow i, $\bar{w}_i$, is:

$$\bar{w}_i = \frac{3}{4}\left(\frac{sm_{fair}}{k} + \frac{k_i}{2}\right)$$

The average window size that yields the fair share is $\frac{sm_{fair}}{k_i}$. w_i reaches that level when $sm_{fair} = \frac{3}{2}k_i^2$. Thus, if we set $m_{fair} = 10$ we have, approximately,

$$k_i \approx \sqrt{\frac{20s}{3}}$$

For $s = 1$, $k_i \approx 2.6$, and for $s = 10$, $k_i \approx 8$.

[8] The error occurs because the timescale τ_i has r_i in the denominator; if we replace that by C then the dynamics are completely independent in this simple model.

This model is vastly oversimplified – in particular, the impact of multiple drops per round-trip times and possible time-outs are not modeled and it is assumed that dropping starts as soon as the number of matches is over m_{fair} – and it probably significantly underestimates the hardship imposed on flows with long RTTs. We therefore choose to be conservative and increase (almost double) the ratio to be closer to $k_i \approx 5\sqrt{s}$. Moreover, we choose to accommodate roundtrips on the order of $150msec$. Recalling that $T = k_i\tau$ and $T = \frac{bsP}{R}$, we adopt as our guideline 2 the following inequality:

Guideline 2: $b\sqrt{s}\frac{P}{R} \gtrapprox 750msec$

C. Burst Scenario

Consider a flow that, after being quiescent, sends h packets (of size P) back to back. If h is smaller than $m_{fair}s$ then it is likely that none of the incoming packets will be dropped, and if h is significantly larger than this amount then the latter packets will probably be dropped. We want the quantity $m_{fair}s$ to be large enough to absorb the packet bursts typical of TCP and other window-based congestion control algorithms. If we take τ to be a typical RTT, and assume that a flow might send an entire window's worth of packets back-to-back (where the window size corresponds to the fair share), then we would want $m_{fair}sP > r_{fair}\tau$. For a link of capacity C with $R \approx C$, this becomes $bs > \frac{C\tau}{P}$.

We must also ensure that $m_{fair}sP$ is not so large that bursts of that size dominate the packet buffer. If the packet buffers are sized to be $C\tau$ where C is the link speed and τ is some nominal delay (on the order of 250msec) used to size the packet buffers, and as above we assume the dropping rate is low (so $R \approx C$), then we require that $m_{fair}sP \ll C\tau$ or, equivalently, $bs \ll \frac{C^2\tau}{Pr_{fair}}$.

The previous two guidelines presented lower bounds for b and s. Our third guideline yields both upper and lower bounds on the product bs.

Guideline 3: $\frac{C\tau}{P} < bs \ll \frac{C^2\tau}{Pr_{fair}} = \frac{C\tau}{P}\frac{b}{m_{fair}}$

We treat these three guidelines not as hard-and-fast rules but rather as general rules-of-thumb to guide how the parameters are set. There is no need for getting the parameters b and s exactly right. If they are too small, the fairness will be less than ideal; if they are too big, the network may be more vulnerable to bursts. But in both cases, as we see in Section V, the degradation is gradual. We also note that compliance with these guidelines can be easily checked by operators. For the first guideline, one need only record the historical values of m_{fair} (which, as we describe in Section IV, the algorithm sets automatically) to ensure that it is well above 10. The second guideline is computable

from the link capacity. The third can be checked by using the values of m_{fair}.

With these guidelines, we now see how the algorithm works in practice. In the next section we describe two practical versions of the algorithm, one that follows directly from our conceptual model and one that requires less state.

IV. Practical Designs

Our description of the conceptual design glossed over several aspects of the algorithm. First, the algorithm for adjusting m_{fair} is borrowed from [12]. We sample the queue length at a rate f_s; upon each sample we adjust the value of m_{fair} according to the following equation:

$$m_{fair}(t) = m_{fair}(t-1) + \alpha(q(t-1) - q_{target})$$
$$-\beta(q(t) - q_{target})$$

where $q(t)$ is the queue length at the t'th sample, $q(t-1)$ is the queue length at the previous sample, and $q_{target} = 0.5(min_{th} + max_{th})$ is the target queue size. This procedure not only stabilizes the average queue length around the target value and provides active queue management, but also estimates m_{fair} implicitly. We choose $f_s = 160Hz$, which is the same as in [12], and α , β are 1.7 and 1.8, respectively. Our algorithm does not seem terribly sensitive to small changes in the parameter values, but we have not done a systematic study of these parameters.[9]

Second, we will use a hash table for the flow table. In so doing, we need not store entire packet headers in the shadow buffer but merely enough bits to accurately identify the flow in the hash table. In practice we will store a k-bit hash of the source-destination addresses (or whatever flow signature is chosen). Third, the design must be modified to accommodate variable size packets. This entails keeping byte counts in the shadow buffer, and measuring matches in bytes rather than packets in the hash table. When removing packets from the shadow buffer (which are picked randomly), we seek to keep the total number of bytes in the shadow buffer constant (but tolerate some jitter). We call this design AFD-SB, with the SB standing for *shadow buffer*.

AFD-SB stores flow information in two places, the shadow buffer and the flow table. A natural way to reduce the state requirements of AFD would be to eliminate one of these. We can't eliminate the flow table, since it is required to make the packet lookups $O(1)$, so we seek instead to eliminate the shadow buffer. We argue in Section VI that if a hash table is used, the hash table and shadow buffer require similar amounts of state; eliminating the shadow buffer would thereby reduce the state requirements by half. However, if we used CAM memory for the flow table the state reduction would be far greater (since hash tables need to be sized an order of magnitude larger than the number of flows to avoid collisions).

The question is whether we can perform the correct operations on the flow table without the shadow buffer. Incrementing the flow table upon packet insertions is straightforward. However, decrementing is hard. When we eliminated packets from the shadow buffer, we picked one at random (or in a FIFO order); all packets had equal chance to be eliminated. Choosing a *flow* at random with uniform probability in the flow table is easy, but it isn't clear how to pick a *packet* at random with uniform probability without linearly traversing the flow entries.

To solve this problem, we adopt an approximation. Note that one could choose a packet with uniform probability if one chose flows (from which to eliminate a packet) according to probabilities $\frac{m_i}{\sum_j m_j}$. However, choosing among all n flows in this manner requires a linear search. We propose picking k flows at random, where $k \ll n$ and choosing among them with the same probability function: $\frac{m_i}{\sum_{j \in S} m_j}$ where S is the set of flows chosen at random. We use $k = 5$ in our simulations.[10]

We call this AFD-FT (with the FT standing for *flow table*). We propose AFD-FT as an example of how one could introduce approximations into AFD in order to make it more easily implementable (by requiring less state). However, we expect that there are many other ways to implement the flow table; in particular, each router design will have its own hardware constraints and to maximize implementability it will be important to adapt AFD to each situation. We offer AFD-FT as an existence proof that AFD's performance is reasonably robust to approximations.

We now proceed to explore the performance of our two designs – AFD-SB and AFD-FT – through simulation.

V. Simulation

A. Simulation Preliminaries

We used the *ns* [28] simulator to evaluate our two designs in a variety of scenarios. We compare our designs to two other schemes: RED [8] and FRED [13]. RED provides a baseline comparison with an algorithm that makes

[9]Our particular AQM design choice, borrowed from [12], seemed particularly easy to adapt to the AFD algorithm. However, we assume that there are countless ways to incorporate AQM in our differential dropping design.

[10]To accommodate varying size packets, we decrement the chosen flow by the number of bytes in the arriving packet. If the chosen flow doesn't have enough bytes, we remember the deficit and subtract more upon the next packet.

no attempt to allocate bandwidth fairly.[11] FRED provides a useful guidepost of how much fairer the allocations are when the dropping decisions are informed by the current packet buffer occupancy.[12] AFD keeps a longer history than FRED, and so the relative performance can be seen as an indication of how important this additional state is. FRED is easier to implement than AFD and requires less state, and so our simulation results should be taken as illustrating the fairness vs. complexity/state tradeoffs provided by the two algorithms.[13]

We envision AFD operating in an environment where flows use many forms of congestion control. In our simulations, we used the following different congestion control algorithms:

TCP: We use TCP-sack as provided in the *ns* release. The TCP flows sometimes have different RTTs, or different file sizes, as described below.

AIMD: We modify the increase parameter a and decrease parameter b in TCP's Additive-Increase/Multiplicative-Decrease algorithm. Standard TCP has $(a, b) = (1, 0.5)$. In addition, we use sources with the following parameters: $(1, 0.9)$, $(0.75, 0.31)$, $(2.0, 0.5)$. We refer to these as AIMD1, AIMD2 and AIMD3, respectively. The first and third are more aggressive than normal TCP, and the second is slightly less aggressive than TCP.

Binomial: Recently Bansal and Balakrishnan [1] introduced a more general family of increase/decrease algorithms. Increases are of the form $w + aw^{-k}$ and decreases are of the form $w - bw^{\iota}$ for constants (a, b, k, ι). We used two versions: $(1.5, 1.0, 2, 0)$ and $(1.0, 0.5, 0, 1)$. The first is roughly comparable to TCP and the second is much more aggressive than TCP. We refer to these as Binomial1 and Binomial2.

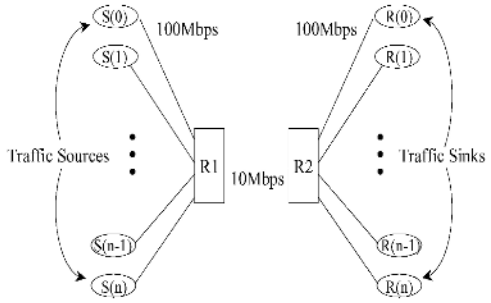
CBR: These are constant rate flows that do not perform congestion control.

Most of our simulations are run on the topology depicted in Figure 2(a), and a few are run on the topology in Figure 2(b). The congested links have transmission latencies of 20 msec, and unless otherwise stated the uncongested links have latencies of 2 msec. The routers on the congested link(s) have buffers that hold 600 packets. The simulations are run for 10 minutes of simulation time, with the first 100 seconds discarded for warmup. Unless otherwise stated, all data packets are 1000 bytes, and AFD is run with $b = 500$ and $s = 5$.

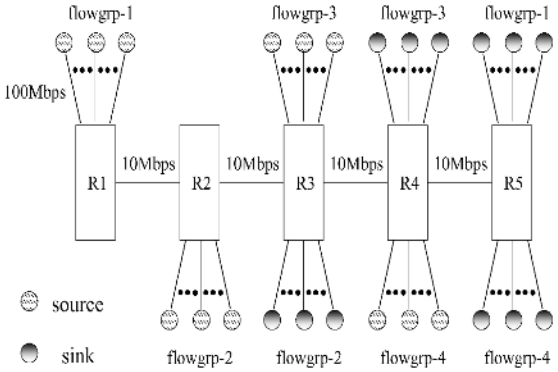
[11] We use RED in *gentle* mode with the parameter settings of $min_{th} = 25$ and $max_{th} = 125$ for a 10Mbps link. The thresholds are scaled proportionaly for higher speed links.

[12] We simulated FRED with $min_q = 4$ and other parameters as in the RED simulations.

[13] Other algorithmic differences, such as the manner in which the drop probabilities are computed, differentiate AFD from RED.



(a) Single Link Topology

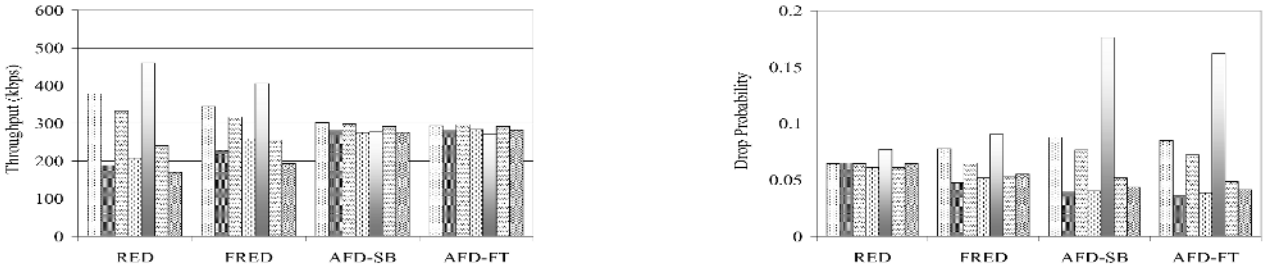


(b) Multiple Link Topology

Fig. 2. Topologies

B. Simulation Results

Mixed Traffic: In Section II we showed that AFD provides equal bandwidth shares to CBR flows sending at different rates. We now consider the more challenging case of a mixture of flows using different congestion control algorithms. The basic scenario we present consists of 7 groups of flows, each with five flows, sharing the bottleneck link in Figure 2(a). The 7 groups are AIMD1, AIMD2, AIMD3, Binomial1, Binomial2, TCP, and TCP with a higher latency of 24.5msec on the 100Mbps access links (yielding a roundtrip latency, without queueing, of 100msec for this flow group, as compared to 10msec for the other flows). Figure 3(a) shows the throughput received by each group of flows (we average within each group so that the data is more easily presentable; the fluctuations within groups are quite small, as was seen in Figure 1 for the CBR simulation). The bar chart shows the bandwidth allocations for the flow groups in the following order: AIMD1, AIMD2, AIMD3, Binomial1, Binomial2,



(a) Throughput

(b) Drop Rates

Fig. 3. Mixed Traffic

TCP, and TCP with increased latency.[14]

The throughput allocations for RED and FRED are substantially uneven, showing that this set of flows includes several rather aggressive congestion control algorithms. Therefore, this presents a reasonable test of the fairness properties of the various algorithms. AFD-SB and AFD-FT both have reasonably fair allocations. Figure 3(b) shows the average drop rate for each group of flows; RED has uniform drop rates, FRED has somewhat uneven drop rates, and both AFD algorithms have extremely uneven drop rates. This demonstrates that applying differentiated drop rates can lead to fairness for different flows.

We now check how well this scenario compares with the guidelines. With $b = 500$ and 35 flows, the average share of the shadow buffer is roughly 14 packets which is in line with the recommendation of 10 packets in Guideline 1. It turns out that due to the fluctuations induced by the window based flow control, in which packets tend to be bunched, the average of m_{fair} is closer to 1. Guideline 2 sets bounds on $b\sqrt{s}\frac{P}{R}$; the guideline calls for this quantity to be larger than 750msec and our scenario has 890msec. Lastly, Guideline 3 calls for $300 < bs \ll 15,000$ if we choose $\tau = 250$msec. Our scenario has $bs = 2,000$.

High Speed Link: To ensure that the fairness in this initial simulation scales, we increased the speed of the congested link, the size of the shadow buffer, and the number of flows by a factor of 10. Figure 4 shows the throughput levels that result.

Increased Multiplexing: Next we tested what happens if the number of flows is increased by a factor of three (retaining the same mixture of flows and the same b). Results for AFD-SB are shown in Figure 5; results for AFD-FT are similar. Even though this scenario violates Guideline 1 (because now the fair number of matches is less than 5), the fair throughput levels are preserved. The conservative assumptions used in deriving the guidelines provide a margin for error.

Reduced Shadow Buffer: If we return to our canonical scenario and reduce the size of the shadow buffer, does fairness suffer? Figure 6 shows the throughput levels for AFD-SB (similar results hold for AFD-FT) for buffer sizes of $b = 250$ and $b = 125$ (the latter violates Guideline 1 significantly) in addition to the canonical case of $b = 500$. In this case, fairness degrades slightly as the shadow buffer size decreases, but AFD still outperforms the baseline algorithms.

While these cases establish that AFD-SB and AFD-FT can provide fair bandwidth allocations when faced with a diverse set of sources, there are cases when the AFD algorithms do not perform as perfectly. We now present three such cases.

Long RTTs: First, consider the case where there are 4 flow groups, each with 10 TCP flows. The flow groups have different RTTs, as determined by the latencies of the connecting 100Mbps links. The latency of the central congested link remains the same (20msec), and the latencies of the connecting links are (followed in parentheses by the resulting total latency of the path, not counting queueing delays): 2msec (48msec), 7msec (68msec), 12msec (88msec), 17msec (108msec). The congested link has queueing delays of roughly 60msec which should be added to the total latency numbers to compute estimates of the RTTs. For each algorithm, the resulting bandwidth allocations for each flow group are shown in the bar chart in Figure 7(a) ordered from smallest to largest latency. The AFD designs provide much better fairness than RED or FRED. The throughput values differ from the fair share by as much as 4%. We chose parameters that were designed (according to Guideline 2) to accommodate RTTs of up to 150msec, and this scenario is just a bit over that limit. If we increase the latency values to 2msec (48msec), 22msec

[14] Unless otherwise noted, we use the same ordering of flow groups in the remaining bar charts.

(111msec), 42msec (175msec) and 62msec(238msec), the fairness is even further reduced, as shown in Figure 7(b). Here the throughputs vary from the fair share by as much as 20%. The largest RTT here (roughly 300msec including queueing delay) is larger than our b and s can accommodate (by guideline 2), and the resulting decrease in fairness is evident.[15]

Unresponsive Flow: Next, we return to the latency values and traffic mix used in Figure 3. To this traffic mix we add a CBR flow sending at 15% of the link capacity. Figure 8 shows that the throughput allocations in AFD-SB are not perfectly fair, with the CBR receiving roughly 15% more than the fair share. The CBR under AFD-FT receives very little throughput at all, only about 15% of the fair share. This was not our intention when designing AFD-FT. However, we are not unhappy with this result; as we observed in Section I, punishing unresponsive flows that tolerate high persistent high drop rates is an important design goal (see [10] for the rationale). We have to (and can easily) add additional mechanism (which we do not discuss in this paper) to AFD-SB to accomplish this goal but AFD-FT does this by default. This occurs because the CBR flow occupies 15% of the shadow buffer (packets are selected to be inserted into the shadow buffer whether or not they are destined to be dropped), which is 5 times larger than the share of any other flow. The approximate packet deletion scheme in AFD-FT is biased against larger flows; when the differences between flows is small this bias is negligible, but when the difference is a factor of five, the bias dominates the results. We believe this phenomenon is also responsible for AFD-FT performing better than AFD-SB in the RTT tests (see Figure 7).

Multiple Congested Links: The third challenging scenario for AFD is when there are multiple congested links. This scenario used the topology in Figure 2(b). There are four different source-destination pairs: three that traverse a single congested link and one that traverses the long path through 3 congested links. Each link has a latency of 5ms. In the first scenario we consider, each of these flow groups consists of 20 standard TCP flows, so that every congested link is traversed by the same number (i.e., 40) of flows. Figure 9(a) shows that the flows that traverse the long path (the fourth group shown for each algorithm in the bar chart) receive substantially less bandwidth than the others, roughly 83% of the fair share. While the throughputs are much fairer than RED or FRED, these results are troubling. If we double the number of flows in the group that traverses the link between R4 and R5, which reduces the fair share on that link, then the results become much fairer. These results are shown in Figure 9(b), with the first group of two bars showing the throughput for the flow groups traversing each of the first two congested links, respectively, and the second group of two bars showing the throughput of the group traversing R4-R5 and the throughput of the group traversing the long path, respectively. Bandwidth allocation for pairs of flow groups that share the same bottleneck link is much fairer for the AFD algorithms relative to the previous scenario. These results suggest, and simple analysis confirms, that AFD does not perform well when there are multiple congested links with *exactly* the same fair share. Flows that are sending right at their fair share experience probabilistic dropping at multiple hops, and therefore receive less than flows that traverse only one congested link. However, if flows pass through multiple congested links that have different fair shares, then when sending at around the fair share the multi-hop flow only suffers drops at one link. We conjecture (not based on any special insight but just based on common sense) that traversing paths with multiple congested links with approximately the same fair shares will be an unusual case.

We tested many other scenarios in simulation, including different traffic mixes and variable size packets, that we do not present here. In all cases, the fairness properties of AFD were maintained. Before concluding the description of our simulations, we briefly mention 3 additional experiments.

Many Small Flows: In the simulations described above, all of our sources had an unlimited amount of data to send. In the next scenario, keeping our basic mixture of 7 flow groups (with infinite sources), we introduce approximately 22,000 short TCP flows representing, for example, shorter web transactions. We use a Pareto distribution of flow lengths. The average number of packets per flow is 15 with a shape parameter of 1.2. We ask two questions: does the presence of many short flows interfere with the fairness properties of the longer-lived flows, and does AFD allow the shorter flows to finish sooner? Figure 11(a) shows the throughput received by the infinite source flow groups under RED, AFD-SB and AFD-FT. AFD-SB and AFD-FT still provide very good fairness in this scenario. Figure 11(b) shows a histogram of the finishing times of the short-lived flows under AFD-SB and RED. Note that this is on a logarithmic scale, so that while the difference in heights between the bars representing the fastest finishing times is small, it represents a difference of approximately 2000 flows. In all the other histogram bins, RED has more flows. Thus, AFD-SB (and the same applies to AFD-FT) aids short flows by lowering their drop rates and allowing

[15] The analysis leading to guideline 2 suggests that $\alpha\sqrt{s}\frac{P}{R}$ should be greater than 5 times the maximal RTT. In this case, that would call for $b\sqrt{s}\frac{P}{R} \approx 1.55$sec whereas in this scenario $b\sqrt{s}\frac{P}{R} = 890$msec.

them to finish sooner. With RED, the short flows see the same ambient drop rate as all other flows.

Two-Way Traffic: When two-way traffic is present, the traffic can become burstier due to ACK-compression [27]. In this scenario, we have the 7 flow groups from our basic scenario sending in both directions. The congested link has a latency of 5ms. Figure 10 shows the throughput for each flow group. AFD-SB and AFD-FT both provide much better fairness than RED and FRED. AFD-FT punishes one particularly aggressive group of flows (for reasons we discussed in the Unresponsive Flow section above). The additional queueing delay incurred in the reverse direction is enough to cause the performance of the TCP flow group with longer RTT (the 7th group) to suffer under AFD-SB (their RTT with queueing delay is roughly 180msec, which is larger than the 150msec our parameters were intended for).

Changing Traffic: We also tested the performance of AFD under changing traffic conditions. In this scenario, three flow groups (AIMD1, AIMD2, TCP) begin sending data at the start of the simulation. After 100 seconds, three additional groups (of the same kinds) start transmitting, doubling the offered load. At time 200 seconds, the first 3 groups stop sending. Figure 12 shows that the AFD-SB algorithm is able to adapt to the changing traffic, cutting the allocations in half when the offered load doubles, and increasing them again when the load is reduced.

Bursty Flows: We also examined the performance of AFD in the presence of on-off sources. These sources exhibit maximally bursty behavior by sending a burst at the speed of the access link (100Mbps) for a brief period and then going idle. The burstiness of these on-off sources is varied by adjusting their burst times while holding the burst rate and idle period constant. An on-off source can be characterized by the ratio of its average sending rate, $\frac{r_{burst} \times l_{burst}}{l_{burst}+l_{idle}}$, to the fair share rate, r_{fair}.

The results of these experiments are shown in Figure 13. Each group of bars in the figure represents one experiment. In each experiment, there are 5 flows in each of the original 7 groups of flows (i.e., those used in the Mixed Traffic experiments above), and a single on-off source. The ratio of the average sending rate of the on-off flow to the fair share rate is indicated on the X-axis (ranging from .5 to 16.) For each experiment, the first seven bars show the average throughput for the flows in each of the 7 flow groups. The eighth bar shows the throughput of the on-off source.

There are three things to note in this figure. First, the bursty flows are not punished for their burstiness. When the average rate of the on-off source is one-half the fair share, its packets are not dropped. When its average sending rate is equal to the fair share, it experiences some packet drops and its throughput is approximately 90% of the fair share rate.[16] The second point to note is that as the sending rate of the bursty flow increases (up to 16 times the fair share rate), it does not receive more than the fair share. Finally, as the sending rate of the bursty flow increases, the fairness among the other flow groups is maintained. When the on-off source sends at 16 times the fair share rate, all the other flow groups receive between 95% and 100% of the fair share. Hence, these results indicate that AFD performs well in the face of very bursty flows, neither punishing nor rewarding flows for their burstiness while preserving the basic fairness among all flows.

C. Discussion of Simulation Results

We now review the general conclusions that can be drawn from our set of simulations. In most scenarios, as long as the guidelines are followed AFD-SB and AFD-FT provide very good approximations to fair bandwidth allocations. The level of fairness is far superior to RED (which does not attempt to provide fairness) and FRED (which uses a very restricted amount of state to guide the dropping decisions). The performance of AFD-SB and AFD-FT seemed fairly robust to varying parameters and conditions. In those scenarios where b was not large enough, or the RTTs were too large, the level of fairness degraded gracefully. Thus, AFD appears to not need precise parameter tuning, and fails softly when the parameters are far out of alignment.

As we saw with the unresponsive CBR flow, and in many of our other simulations not shown here, AFD-FT responds quite punitively when flows are not responsive. We judge this a good thing, but ultimately this is a policy question. With AFD-SB we can turn this policy on or off depending on the desires of the network operator;[17] in AFD-FT this policy is hardwired in. Moreover, the choice of $k = 5$ in our version of AFD-FT (where k is the number of other flows that must be compared before a packet is eliminated from the flow table) will probably need to be made larger when the number of persistent fast flows is larger. AFD-FT dealt quite well with the many small flows, but if the number of fast flows is on the order of thousands the number of comparisons will probably have to be on the order of 10 or more. We have not yet investigated how this parameter will scale.

[16] Recall that deviations from the fair share are most likely to occur for flows sending at or near the fair share rate.

[17] Again, we don't present the algorithm which implements this policy in AFD-SB, but it is straightforward and requires only that the shadow buffer and flow table keep separate track of matches that were dropped and matches that were not dropped.

VI. State Requirements and Rate Distributions

These simulations show that when using the guidelines to set the parameters, AFD provides reasonably fair bandwidth allocations. However, can this algorithm be feasibly implemented? All the operations on the forwarding path are $O(1)$, so the main barrier to implementation would be if AFD required an impractical amount of state. This feasibility requirement is economic, not technical. Clearly routers could be built with vast amounts of additional storage; however, that would come at a steep price, and we are looking to achieve fairness without greatly increasing the complexity or the cost of future routers. We use, as our standard, the requirement that the additional state required by AFD should be small compared to the memory already required by the packet buffer. We compute the state requirements, with the aid of two measurements, in Section VI-A. In Section VI-B we discuss how these state requirements depend crucially on the rate distribution and, in Section VI-C, present some trace data on current rate distributions.

A. Calculation of State Requirements

We compute the state requirements of our two designs – AFD-SB and AFD-FT – separately. We start with AFD-SB, whose state consists of two parts: the shadow buffer and the hash table.

Guideline 1 (Section III-A) states that the shadow buffer should hold roughly $10\frac{R}{r_{fair}}$ packets. This estimate assumed that all packets were the same size. To adjust this for variable packet sizes, we first define $\tilde{R}$ as the packet arrival rate in terms of packets/sec, and $\tilde{r}_{fair}$ as the arrival rate of a flow sending at the fair share rate; also, let P_{max} and P_{ave} denote the maximal and average packet sizes. The proper sizing of the shadow buffer in terms of packets, when you have variable sized packets, is then $b = 10\frac{\tilde{R}}{\tilde{r}_{fair}}$. The worst case, the lowest value, for $\tilde{r}_{fair}$ is when a fair share flow is sending maximal sized packets: $\tilde{r}_{fair} \geq \frac{r_{fair}}{P_{max}}$. The expression for R is simply $R = \frac{R}{P_{ave}}$. To represent each packet in the shadow buffer we use, roughly 6 bytes.[18] Assuming that $P_{max} = 1500$ bytes and $P_{ave} = 300$ bytes,[19] we then find that b_{bit}, the shadow buffer size in bits, should be roughly:

$$b_{bit} = 10 * 48 * \frac{\frac{R}{300}}{\frac{r_{fair}}{1500}} = 2400\frac{R}{r_{fair}}$$

Similarly, we can estimate the storage required for a hash table to keep track of the number of bytes sent by each flow. We describe our trace data in Section VI-C, but for now we use the fact that, in the various traces we have seen and for the shadow buffer sizes we are proposing, the number of flows in the hash table is typically less than a fourth the number of packets in the shadow buffer (See Figure 14). Let's assume that we need 2 bytes per entry for counting (assuming we count matches at the granularity of 40 byte units and ignore roundoff errors), and that we use a hash table 12 times larger than the expected number of flows to reduce the chance of collisions in the hash table. The size of the hash table h_{bit}, in bits, is then:

$$h_{bit} = 10 * 16 * 12 * \frac{1}{4}\frac{\frac{R}{300}}{\frac{r_{fair}}{1500}} = 2400\frac{R}{r_{fair}}$$

Thus, to a first approximation, these two data structures require similar amounts of memory. Many router vendors recommend having on the order of 250 msec's worth of memory in the packet buffers. We compare the memory required by AFD to the memory already recommended for the packet buffer, which is roughly $\frac{C}{4}$ where C is the speed of the link (in bps). The ratio, call it ρ_{SB}, of AFD-SB's state requirement to the size of the packet buffers is given by:

$$\rho_{SB} = \frac{4800\frac{R}{r_{fair}}}{\frac{1}{4}C} = \frac{19.2kbits/sec}{(1-D)r_{fair}} \approx \frac{19.2kbits/sec}{r_{fair}}$$

where the last approximation is because $C = R(1-D)$ and we suspect the aggregate dropping rate will typically be low. Thus, the fraction of extra memory, ρ_{SB}, required by AFD-SB depends on the typical size of the fair share rate on a link.

We now calculate the state requirements of AFD-FT. If the flow table is implemented using a hash table, then we merely use the state requirements above (only the hash table part) and so the state requirements are roughly half that of AFD-SB. If the flow table is implemented using CAM then we no longer need a table 12 times the number of flows (as we did with the hash table). The CAM-based design needs roughly 64 bits per flow (48 bits for the hashed flow-id and 16 bits for a counter). Modifying the calculations above to reflect these changes, we find that the AFD-FT ratio is:

$$\rho_{FT} \approx \frac{3.2kbits/sec}{r_{fair}}$$

[18] We need not store the full header in the shadow buffer, merely the hash of the source-destination addresses. 6 bytes is a sufficiently large hash to comfortably accommodate roughly 10^8 flows with small chance of collision.

[19] See [3] for measurements of average packet size. We've used a fairly conservative estimate, as the average packet size reported in [3] is over 400bytes. The average packet size over our three traces that we report on in Section VI-C is almost 500 bytes.

which is one-sixth of the AFD-SB requirements.

Note that these estimates of the ratios ρ depend only on the fair share of a link and not directly on its bandwidth. For slow links, the fair share will be quite small (and thus ρ could be 1 or larger), but the amount of memory required for these slow links is insignificant; if the fair share is 19.2kbps, so $\rho = 1$, on a T3 link, the required extra memory for AFD-SB is roughly 1.5Mbytes. Thus, we care mostly about these ratios ρ on faster links where the absolute amount of memory devoted to the packet buffers is quite large. We assume that faster links will have larger fair shares, and these ratios ρ estimating the relative memory requirements of AFD will be smaller on such links.

We are not aware of many studies of the typical fair shares on current Internet links.[20] One can't infer the fair share merely by taking a packet trace; without fair bandwidth allocation one can't tell which flows are constrained by that link, and which flows are constrained elsewhere. One would have to find a link whose router has Fair Queueing (or some equivalent algorithm) on which to take traces (and even these traces would be misleading because of the lack of fair bandwidth allocations elsewhere along in the network). Lacking a solidly grounded method for measuring typical fair shares, we turned to an available dataset that has some bearing on the question.

One way to obtain a very rough estimate of the fair share on a path is to measure the end-to-end throughput obtained by a TCP connection that traverses that path. This is a very imperfect measure, since TCP throughput varies as a function of RTT and not all traffic is TCP-friendly. We obtained a dataset of measured end-to-end throughputs of transfers between NIMI measurement sites.[21] 47 sites participated in this measurement by periodically transferring a 1MB file with another randomly chosen NIMI site. The dataset contains roughly $35,000$ transfers. For lack of space we do not show the distribution here, but in over 90% of the transfers the rate is greater than 250kbps. These measurements are not focused on high-speed links. However, most of the NIMI boxes are at universities and other sites that are well-connected to the Internet, and none of them are behind very slow modems, so the measurements are probably indicative of moderate speed links. If we take 250kbps as a rough estimate of the fair share then $\rho_{SB} \approx \frac{1}{13}$ and $\rho_{FT} \approx \frac{1}{78}$. That is, the extra memory required by AFD-SB is less than a tenth of the memory already devoted to packet buffers. We view these estimates as being a very conservative lower bound, in that the fair shares on very fast links may be much larger, and thus the ratios ρ_{SB} and ρ_{FT} much smaller. As one moves up the network hierarchy towards the backbone links, it is likely that a larger fraction of flows are bottlenecked somewhere else in the network. That means that the fair share available to flows unconstrained elsewhere could be quite large. If backbone links had $r_{fair} \approx 2Mbps$ then $\rho_{SB} \approx \frac{1}{100}$ and $\rho_{FT} \approx \frac{1}{600}$. However, the level of fair shares remains an open question, and we hope in the future to find better ways to estimate the fair share on high-speed links.

This paper does not address detailed implementation issues. However, since we've used the storage requirements of the packet buffer as a yardstick for the state requirements of AFD, we would be remiss if we did not point out that this comparison is somewhat misleading. Packet buffers make heavy use of slower (and cheaper) DRAM with a smaller amount of faster (and more expensive) SRAM. We can do likewise with the shadow buffer state. However, the flow table will probably require SRAM or CAM. Thus, we cannot conclude that AFD imposes a cost that is a fraction ρ of the packet buffer cost. We leave the detailed implementation issues for future work (by others); our goal here is to provide a rough estimate of the state requirements.

The estimates above are very rough in nature. On any particular choice, such as the 250msecs for packet buffers or the numbers of bits for counters, one could argue that we are off by a factor of two. But we believe these calculations give us an order-of-magnitude estimate of the state requirements of our design, and suggest that AFD, in either of its incarnations, is likely able to achieve reasonable levels of fairness without exorbidant amounts of extra state.

We now argue that the distribution of rates greatly aids the AFD approach.

B. *Impact of the Rate Distribution*

Our state calculations depended critically on the value of r_{fair}, and in particular on the ratio $\frac{r_{fair}}{R}$. One intuitive way understanding this ratio is the following. Let's call a flow a *cheetah* if it is sending above or near the fair share, and let n_c denote the number of such flows. We call the other flows, the slow ones, *turtles*. Let $\gamma = \sum_{turtles} \frac{r_i}{R}$ denote the fraction of the offered load that comes from these slow flows. The equation defining the fair share $\sum_i \min(r_i, r_{fair}) = C$ becomes: $R\gamma + r_{fair} n_c = C$. We can write this as

$$\frac{R}{r_{fair}} = \frac{n_c}{(1-D) - \gamma}$$

[20] The *available bandwidth*, which is a somewhat different concept, was studied in [19].

[21] NIMI is the National Internet Measurement Infrastructure; see [20] for a more detailed description. See [26] for details of the measurement process.

where D is the overall drop rate. The quantity $\frac{R}{r_{fair}}$ is minimized when both the number of cheetahs is small *and* the fraction of bandwidth consumed by the turtles is quite small: in short, when most of the bandwidth is being sent by a very few fast flows. It is in this regime that AFD requires very little state to perform well.

To make this more precise, we now look at how r_{fair} varies under different rate distributions in a simple analytical model. We consider a continuum of flows whose rates are given by a density function $h(r)$. We will consider four continuum distributions that all have the same average rate (of 1 in arbitrary units) and normalize the distribution by n (representing the number of flows), so the total load offered by each distribution is n. For each distribution we have $n = \int_0 \ h(r)dr$ and $1 = \frac{1}{n}\int_0 \ h(r)rdr$. We assume that the router allocates bandwidth fairly, and that flows respond by restraining their flow to the fair share if their assigned rate (by the distribution) is larger than the fair share. As we vary the capacity C between 0 and n we compute the fair share $r_{fair}(C)$ from the constraint: $C = \int_0^\infty h(r)\min(r, r_{fair}(C))dr$. The question is how $r_{fair}(C)$ compares for the various distributions we consider.

We consider four distributions, in order of increasing variance. For a *point* distribution, where all flows are the same rate, the fair share is merely $r_{fair}^{int} = \frac{C}{n}$. For a distribution where rates are uniformly distributed between 0 and 2, the fair share is $r_{fair}^{uni} = 2(1 - \sqrt{1 - \frac{C}{n}})$. For an exponential distribution, $h(r) = ne^{-r}$, the fair share is $r_{fair}^{ex} = -\ln(1 - \frac{C}{n})$. Finally, for a power-law distribution $h(r) = \frac{n}{2}r^{-3}$ for $r \geq \frac{1}{2}$ (and 0 otherwise; this restriction is to avoid the divergence at the origin), we have $r_{fair}^{pl} = \frac{1}{4}(1 - \frac{C}{n})^{-1}$. The point of this exercise is that as the tail of the distributions got larger (that is, the number of very fast flows increases) the fair share as a function of C became larger. While the fair share diverges for both the exponential and the cubic power-law distributions as $\frac{C}{n}$ approaches 1, the fair share is dramatically larger in the power-law case.

Guideline 1 calls for the buffer size to be roughly $b = 10\frac{R}{r_{fair}}$. If the rate distribution is such that r_{fair} is larger (for a given R) then the shadow buffer size, and all the state requirements, become smaller. Note that if the fair share remains the same then the state requirements increase linearly with the speed of the link (and the offered load R). However, if we keep the same offered load and let the fair share increase to its natural level as we increase C, and study $b(C) = 10\frac{C}{r_{fair}(C)}$, then we find two cases. For the point and uniform distributions, $b(C)$ increases; this is what we expect, more state is required on faster links. For the exponential and power-law distributions, however, $b(C)$ decreases; as the speed of the link increases the state requirements of the algorithm go down! Note this is not a decrease in the ratio ρ, this is a decrease in the absolute amount of state.

We have no way of judging with any certainty what the fair share rates will be in the future on high-speed links. However, the arguments above suggest that AFD will operate best when the distribution of flow rates has a long tail. We now turn to empirical evidence to see if this is currently the case.

C. Trace Data

We obtained traces of traffic from three separate locations: a 100Mbps link at a national laboratory, a T3 peering link between two providers, and a FDDI ring connecting a modem bank at a dial up provider to a backbone network.[22] We refer to these traces as Trace 1, Trace 2, and Trace 3, respectively. Trace 1 consists of approximately 22 million packets collected during a two hour interval. Trace 2 consists of 34 million packets over a 45 minute span. Trace 3 collected approximately 6 million packets in 70 minutes.

Guideline 2 calls for keeping state of about 1 second in order to estimate the rates. For a variety of different time intervals, Figure 14 shows the average ratio between the number of packets in the shadow buffer and the number of distinct flows represented in the shadow buffer; this ratio was used in Section VI-A to estimate the state requirements for AFD.

For each of the traces we estimated individual flow rates based on the packets seen in the last second; see [14] for similar data and related discussions. Figure 15 shows the complementary distribution of flow rates for each of the three traces. Trace 3 is not obviously inconsistent with a slowly decaying exponential distribution, but the other two clearly have long tails.[23] This is even more pronounced on a log-linear plot, which for space reasons we do not show here. Figure 16 shows the cumulative distributions of the 1-second flow rates. Note 10% or less of the flows represent roughly 60% of the bytes in the worst case, and roughly 90% of the bytes in the best case. This is in contrast to the fastest 10% of the flows carrying 33% of the bytes for an exponential distribution. Thus, based on the very preliminary evidence presented here and in [14] we

[22] To preserve anonymity, we omit further details about these sites.

[23] We use the term *long tailed* to refer to distributions that decay slower than exponentially. Power-law distributions are examples of this, but so are Weibull and other distributions. We do not intend to enter the rather lengthy and subtle debates about whether or not a particular distribution is a power law or Weibull or some other form. We merely observe that it obviously decays slower than an exponential.

conjecture that rate distributions are usually long-tailed. In the language of Section VI-B, we expect that the number of cheetahs will typically be small, but they will represent the bulk of the bytes.

Similar statements have long been made about the distribution of flow sizes. Figures 17 and 18 show the analogous data for flow sizes as measured by the total number of bytes transferred. These distributions are significantly more skewed than the rate distributions. Figure 19 shows the total flow size plotted against the flow's rate (as measured by total bytes over completion time) for flows in Trace 2 that are larger than 10,000 bytes (the other traces have similar results). No correlation is visually apparent, and when we compute the correlation it is quite small; the correlations between rate and size are .0075, .0031 and .00088 for the three traces, and are .11, .22 and −.0045 when we correlate the logarithms of the rate and size. This shows that the distinction between cheetahs and turtles is different from the historical distinction between elephants and mice. The long-tailed distribution of rates is clearly not directly driven by the long-tailed distribution in sizes. Their underlying mechanisms are presumably different.

We are not aware of work that analyzes the mechanisms responsible for the distribution of flow rates. We assume that the rates on a link, at least for longer-lasting flows, reflects the available bandwidth at that flow's bottleneck link. However, it is not clear to us why the distribution of these bottleneck rates should have a long tail. Much more work remains to be done to characterize these rate distributions, and explain their origin.

VII. Context, Related Work, and Discussion

This paper starts with the assumption that routers should allocate bandwidth fairly. While this is a familiar and oft-told story, for context we once again briefly review the rationale for fairness. We then discuss related work and conclude with some comments on AFD's underlying design principles.

Congestion control is one of the Internet's most fundamental architectural problems. In the current Internet, most routers do not actively manage per-flow bandwidth allocations, and so the bandwidth received by a flow depends on the congestion control algorithms used by other competing flows. To achieve relatively equitable bandwidth allocations, all flows must be TCP-compatible [4] (also known as TCP-friendly); that is, they must use a congestion control algorithm that results in bandwidth allocations similar to TCP's. This approach requires uniformity and cooperation, both of which might be problematic.

By restricting the world of congestion control algorithms to those that are TCP-compatible, some applications may be impaired. This would be the case if some applications required radically different forms of congestion control that are inherently unfriendly to TCP. However, recent advances in *Equation-Based Congestion Control* (EBCC) [11] and other TCP-compatible algorithms [22], [21] gives hope that a very wide variety of application requirements could be fulfilled within the sphere of TCP-compatibility. Thus, the uniformity requirement, while potentially a problem, may in fact be tolerable.

As for cooperation, any host can obtain more bandwidth simply by using a more aggressive congestion control algorithm. Thus, the current approach relies on end-hosts (and their users) voluntarily adopting the TCP-compatible guidelines. There has been some initial work to penalize flows that violate the rules [9], but so far the goal of reliably identifying *ill-behaved* flows has proved elusive.[24]

In response to these problems, there has been a long history (dating back to Nagle [16]) of proposing that routers play a more active role in allocating bandwidth. If the routers ensure fair bandwidth allocations[25] then end-hosts are no longer required to adhere to any particular form of congestion control and the problems of uniformity and cooperation mentioned above no longer exist.[26] The proposals to accomplish this fair bandwidth allocation, such as Fair Queueing [6] and related algorithms [15], [23], [2], all involve complicated packet scheduling algorithms and require per-flow state. High-speed implementations of these algorithms are just now becoming available. If such implementations continue to scale to the highest speeds without causing undue costs, then the contents of this paper are probably moot. However, it isn't yet clear that these implementations will scale *inexpensively* to increasingly higher speeds. If the cost of adding this extra functionality is significant, designs with lower complexity but similar functionality may be preferable in commercial designs. This was our goal in designing AFD, to provide fairness at a significantly reduced level of complexity.

Core-Stateless Fair Queueing (CSFQ) [24] and the subsequently proposed variations [5], [25] share the same goal. While they achieve high degrees of fairness with

[24]There has been more success with identifying high-bandwidth flows, and unresponsive flows, but that still leaves flows a large leeway to cheat.

[25]To give users an incentive to use some form of responsive congestion control we add the requirement that flows are punished if they incur persistent high drop rates. See [10], [24] for discussions of this topic.

[26]Some claim that such fair allocation mechanisms open the door to denial-of-service attacks through flow-spoofing; while we do not believe such arguments are sufficient to nullify the desirability of fair bandwidth allocations, and that this paper is not the place to delve into such arguments at length, we did want to note the existence of objections to the fair bandwidth allocation paradigm.

scalable mechanisms in the core routers, these schemes require a change in the packet format (to accommodate another field) and the careful configuration of routers into core, edge, and peripheral regions. Thus, while CSFQ delivers a high degree of fairness, it faces significant deployment hurdles.

There are several other proposals for low-complexity approximations to fairness, such as FRED, [13], SRED [17], SFB [7], CHOKe [18] and RED-PD [14]. These present a spectrum of possible designs, differing in the extent to which they carefully manage the bandwidth allocation. The extremes of the spectrum are complete fairness (Fair Queueing) on one end and unmanaged allocation (RED) on the other; in between, some algorithms carefully manage the bandwidth of a few flows (*e.g.* RED-PD) while others attempt to manage all flows at the same time, but do a less careful job on each one (*e.g.* AFD). Different choices along this spectrum embody different expectations about the set of congestion algorithms deployed. One possible view is that in the future almost all flows will use a TCP-compatible congestion control algorithm, and that there will only be a very few malicious (or broken) flows that are substantially more aggressive. In this scenario, routers only need to detect these few outlying flows and restrain their usage to an appropriate level. RED-PD is an example of an algorithm well-suited to this task. Another possible view is that flows will use a very wide variety of congestion control algorithms, not necessarily TCP-compatible, and that routers will need to allocate bandwidth to flows as the common case. AFD is designed for this scenario. Which of these two approaches – identifying a few outlying flows and treating them as special or treating allocation as the common case – depends on how the future of congestion control unfolds.[27] Our purpose in this paper is not to argue that one vision is more likely than another, merely to design an algorithm that would be suitable if the second scenario comes to pass.

The spectrum of approximate fairness designs also presents us with different tradeoffs between increased fairness and reduced complexity. The desirability of one spot on the spectrum versus another depends greatly on the nature of Internet traffic and on router design constraints, both of which change over time. We based AFD on the assumption that while traffic characteristics and hardware capabilities will evolve, there will be three enduring truths. First, FIFO packet scheduling will be significantly easier to implement than certain non-FIFO scheduling algorithms; it uses cheaper hardware that scales to faster speeds. Second, the distribution of rates on high-speed links will be long-tailed. We don't expect this to necessarily be a power-law or any other particular form, but we do expect that the majority of flows will be slow but the fast (high-rate) flows will send the bulk of the bytes. Third, we assume that memory – fast, slow, and CAM – will continue to decrease in price relative to the special logic needed to implement non-FIFO packet scheduling. AFD's main implementation burden is additional memory; we assume that the marginal cost of equipping routers with AFD will shrink (relative to packet scheduling designs) as these commodity products become cheaper over time. If these three assumptions hold, then we believe AFD may be a cost-effective scheme for providing approximately fair bandwidth allocations.

[27] That future will depend in part on the choices made for router allocation mechanisms, so the dependencies are circular.

REFERENCES

[1] Bansal, D., and Balakrishnan, H., "Binomial Congestion Control Algorithms" *Proceedings of Infocom '01*.

[2] Bennett, J. and Zhang, H., "Hierarchical Packet Fair Queueing Algorithms", *SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 143–156, Aug. 1996.

[3] `http://www.caida.org/analysis/AIX/plenhist/`

[4] Braden, B., Clark, D., Crowcroft, J., Davie, B., Deering, S.,Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L.,Ramakrishnan,K., Shenker,S., Wroclawski, J., Zhang, L., "Recommendations on queue management and congestion avoidance in the internet", *IETF RFC (Informational) 2309*, April 1998.

[5] Cao, Z., Wang, Z. and Zegura, E., "Rainbow Fair Queueing: Fair Bandwidth Sharing Without Per-Flow State", *Proceedings of INFOCOM'00* March 2000.

[6] Demers, A., Keshav, S. and Shenker, S., "Analysis and simulation of a fair queueing algorithm", *Journal of Internetworking Research and Experience*, pp 3-26, Oct. 1990. Also in Proceedings of ACM SIGCOMM'89, pp 3-12.

[7] Feng, W., Shin, K., Kandlur, D. and Saha, D., "Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness", *Proceedings of INFOCOM'2001 (to appear)*, April, 2001.

[8] Floyd, S. and Jacobson, V., "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transaction on Networking*, 1(4), pp 397-413, Aug. 1993.

[9] Floyd, S., and Fall, K., "Router Mechanisms to Support End-to-End Congestion Control", *LBL Technical report*, February 1997.

[10] Floyd, S., and Fall, K., "Promoting the Use of End-to-End Congestion Control in the Internet", To appear in *IEEE/ACM Transactions on Networking*, August 1999.

[11] Floyd, S., Handley, M., Padhye, J., and Widmer, J., "Equation-Based Congestion Control for Unicast Applications", *Proceedings of ACM SIGCOMM'2000*, August 2000.

[12] Hollot, C.V., Misra, V., Towsley, D. and Gong, W., "On Designing Improved Controllers for AQM Routers Supporting TCP Flows", *Proceedings of Infocom '01*.

[13] Lin, D. and Morris, R., "Dynamics of random early detection", *Proceedings of ACM SIGCOMM'97*, pp 127-137, Oct. 1997.

[14] Mahajan, R. and Floyd, S. "Controlling High-Bandwidth Flows at the Congested Router", ACIRI, Berkeley, California, Nov. 2000.

[15] McKenny, P., "Stochastic Fairness Queueing", *Proceedings of INFOCOM'90*, pp 733-740.

[16] Nagle, J., "On packet switches with infinite storage", *Internet Engineering Task Force*, RFC-970, December, 1985.

[17] Ott, T., Lakshman, T. and Wong, L., "SRED: Stabilized RED", *Proceedings of INFOCOM'99*, pp 1346-1355, March 1999.

[18] Pan, R., Prabhakar, B. and Psounis, K., "CHOKe - A Stateless Active Queue Management Scheme For Approximating Fair Bandwidth Allocation", *Proceedings of INFOCOM'00* March 2000.

[19] Paxson, V., "End-to-End Internet Packet Dynamics", *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 277-292, 1999.

[20] Paxson, V., Mahdavi, J., Adams, A. and Mathis, M., "An Architecture for Large-Scale Internet Measurement", *IEEE Communications Magazine*, vol. 36, no. 8, pp. 48-54, August 1998.

[21] Rajaie, R., Handley, M. and Estrin, D., "An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet", *Proceedings of INFOCOM'99*, March, 1999.

[22] Rhee, I., Ozdemir, V. and Yi, Y., "TEAR: TCP emulation at receivers – flow control for multimedia streaming", *Technical Report, Department of Computer Science, NCSU*, April, 2000.

[23] Shreedhar, M., and Varghese, G., "Efficient Fair Queueing using Deficit Round Robin", *ACM Computer Communication Review*, vol. 25, no. 4, pp. 231–242, October, 1995.

[24] Stoica, I., Shenker, S. and Zhang, H., "Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks", *Proceedings of ACM SIGCOMM'98*.

[25] Venkitaraman, N., Mysore, J., Srikant R.,and Barnes, R. "Stateless Prioritized Fair Queueing", *Internet Engineering Task Force* July 2000.

[26] Zhang, Y., Paxson, V., and Shenker, S., "The Stationarity of Internet Path Properties: Routing, Loss, and Throughput", ACIRI Technical Report, May 2000.

[27] Zhang, L., Shenker, S. and Clark, D., "Observations on the Dynamics of a congestion control Algorithm: The Effects of Two-Way Traffic", *Proceedings of ACM SIGCOMM'91*.

[28] ns - Network Simulator (Version 2.1b6).

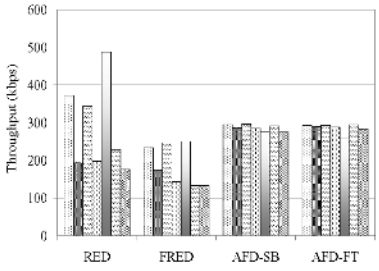


Fig. 4. High-Speed Link

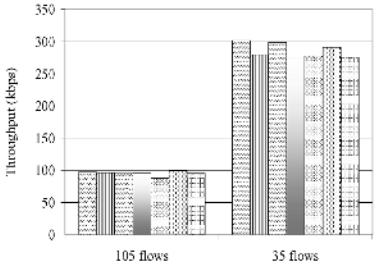


Fig. 5. Increased Multiplexing

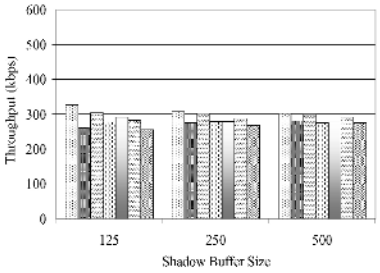
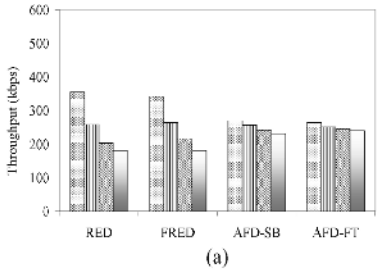


Fig. 6. Reduced Shadow Buffer



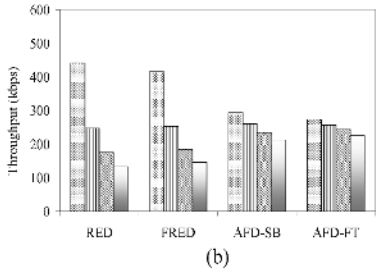


Fig. 7. Long RTTs (a) Max Latency = 200 msec (b) Max Latency = 300 msec

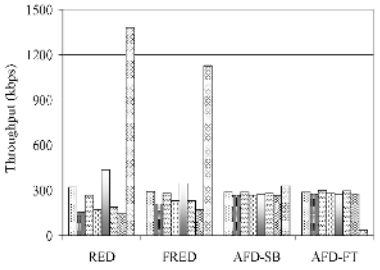
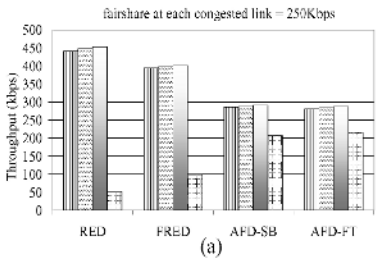


Fig. 8. Unresponsive Flow



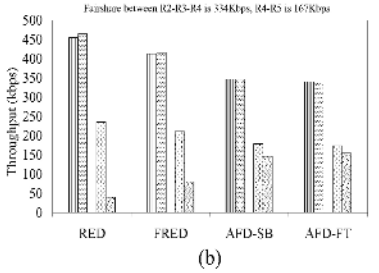


Fig. 9. Multiple Congested Links (a) same Fair Share (b) different Fair Share

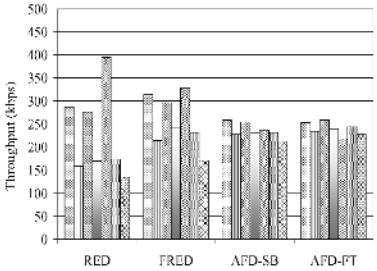
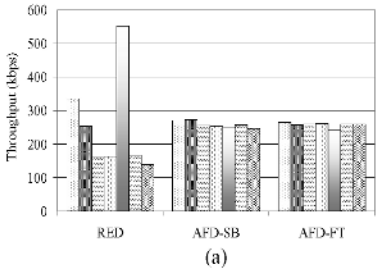


Fig. 10. Two-Way Traffic



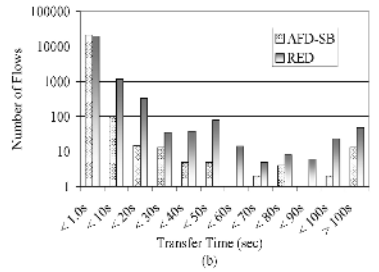


Fig. 11. Web Traffic – (a) throughput of large flows (b) transfer time of small flows

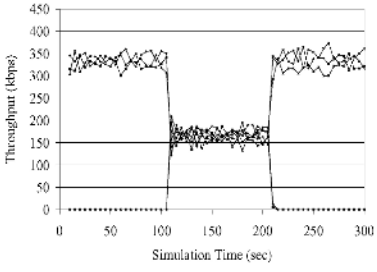


Fig. 12. Changing Traffic

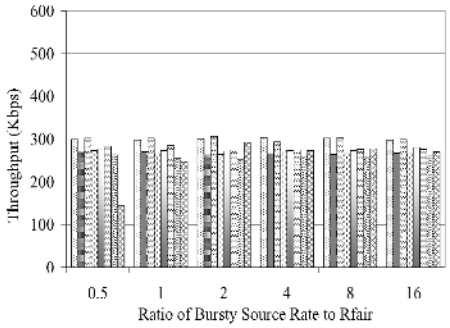


Fig. 13. Bursty Traffic

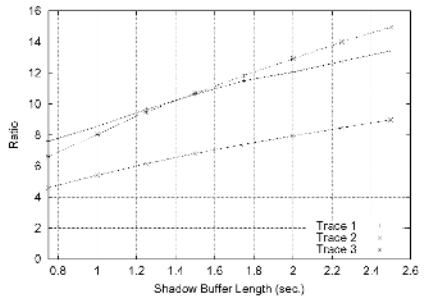


Fig. 14. Ratio of Packets to Flows in Shadow Buffers

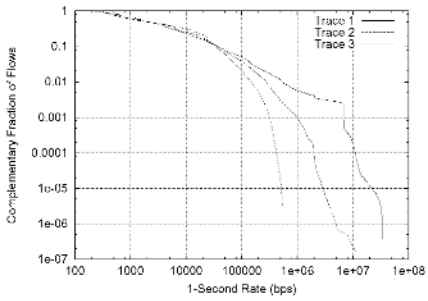


Fig. 15. Complementary Distribution of 1-Second Rates

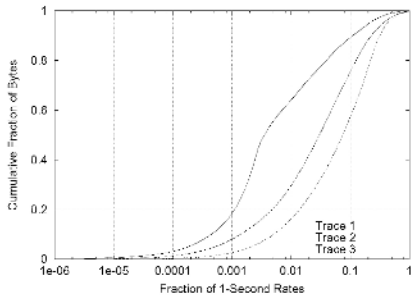


Fig. 16. Cumulative Distribution of 1-Second Rates

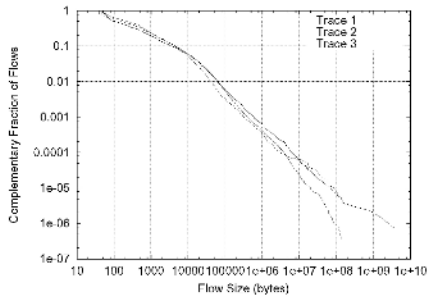


Fig. 17. Complementary Distribution of Flow Sizes

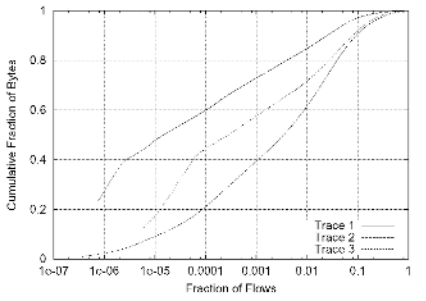


Fig. 18. Cumulative Distribution of Flow Sizes

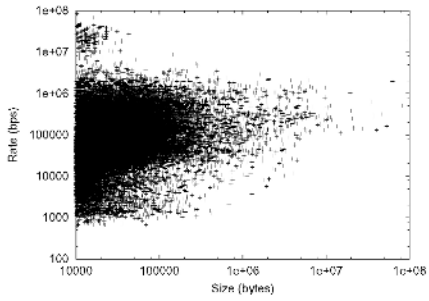


Fig. 19. Rates vs Size

Quality of Service Assurance for Dependable Information Infrastructures

Nong Ye, Ying-Cheng Lai and Toni Farley
ARIZONA STATE UNIVERSITY
{nongye@asu.edu}

Abstract

The information infrastructure has enabled a networked computing and communication environment. Although many DoD organizations depend on network-centric information operations to support critical missions, the current information infrastructure provides little guarantee of their dependability. In this paper, we first discuss some drawbacks of existing information infrastructures, such as the Internet and Computational Grids, in Quality of Service (QoS) assurance. Next, we present the objective of our project in the Critical Infrastructure Protection (CIP) program to assure end-to-end QoS for individual high-priority jobs on the information infrastructure with minimum traffic congestion through local, regional, and global level QoS models. In each section, we summarize our research accomplishments to date, present the outcomes of some representative work, and point to published papers for more details.

Introduction

Information infrastructures enable a networked computing and communications environment, such as those used by many DoD organizations, to perform network-centric information operations for critical missions. Network-centric information operations are characterized by information moving between computational nodes on the network with the goal of information superiority -- 100% relevant content, along with data accuracy and timely delivery. Such operations depend on network-centric operational architectures that provide desired QoS assurance. For example, a high-performance information infrastructure that provides a speedy, dependable backplane sharing of various resources (including data, computation, communication, and visualization) on the computing and communications network. QoS has three attributes: timeliness, precision, and accuracy (Ye, 2002; Chen, et al., 2003). This project focuses on the timeliness of information operations and time management of computational resources on the network.

Existing information infrastructures, such as the Internet and Computational Grids, provide little dependability guarantees, with regards to timeliness, for networked computing and communication. The Internet supports the sharing of computational resources on a network based on the "best-effort" model, in which trust among participating parties is assumed. That is, participant A will satisfy the need of other participants as long as the current capacity of participant A's computational resources can do so. Hence, resources are made available for use by anyone regardless of the state of the resources until those resources are depleted. This "best-effort" model has provided an environment for crafty exploits and denial-of-service attacks that have occurred and presented a significant threat to the realm of information superiority. Moreover, QoS for a user's application is not guaranteed because other users may emerge at any time to compete for and share computational resources, causing time delay. Thus, little resource management and QoS guarantees exist for dependable information operations on the Internet.

Computational Grids (Foster and Kesselman, 1999) address the resource management problem by using centralized brokers or mating agents to assign networked computational resources to users' applications. The centralized authority of resource management (top-down, command-directed synchronization of network-centric information operations) works only within a small-scale networking environment of closely-coupled administrative domains, such as a network of several national supercomputing centers, as currently demonstrated (National Science Foundation, 2000). Centralized authority of resource management does not scale to an information infrastructure where many independent administrative domains exist, as these domains may not necessarily obey this authority. First of all, administrative domains for different organizations likely act on their own to rapidly respond to local situations and meet local contingencies as they see them. Secondly, information operations through interactions among different administrative domains are typically complex and non-linear. These two factors make even local influences from closely-coupled, friendly administrative domains hard to predict. The top-down, centralized synchronization of resource management in Computational Grids is not scalable to large-scale, complex, adaptive information infrastructures in dynamically changing environments.

Most existing research work on computer network QoS is aimed towards population-based performance objectives concerning the mean, worst-case, or statistically bounded performance of time delay, execution time, or other timeliness measures; with little consideration for the end-to-end timelines assurance of an individual job on the network. For example, consider a message that needs to be sent from a command center to a battle field with a certain timeliness requirement. The mean time delay property of a computer network or the statistical bound on the percentage of jobs experiencing extended delay on the network tells little about the specific time delay that this particular message experiences. The lowest bound set by the worst-case time delay property of the network may be short of meeting the timeliness requirement of this message. To achieve information superiority, it is important to assure end-to-end QoS for each individual high-priority job. This project focuses on end-to-end performance objectives for individual high-priority jobs on information infrastructures.

Integrated Service (IntServ) and Differentiated Service (DiffServ) are two existing models that address QoS on information infrastructures (Almquist, 1992; Blake et al., 1998; Braden et al., 1994). The IntServ model relies on bandwidth reservation for each flow on a computer network. This model has an extremely high overhead from managing the reservations made for many flows on large-scale computer networks, such as information infrastructures, and thus is not scalable. The DiffServ model classifies jobs on computer networks into groups of different priorities and, given these groups, different QoS treatments. However, the DiffServ model does not consider the performance objectives of individual jobs.

The objective of this project is to achieve end-to-end QoS assurance for individual high-priority jobs on an information infrastructure. To accomplish this objective, we work on QoS models at the local, regional, and global levels of the information infrastructure. A local level QoS model aims at providing service stability and dependability on an individual resource, of an information infrastructure, by minimizing the waiting time variance (WTV) of jobs admitted for service by that resource. A regional level QoS model aims at providing service stability and dependability from a collection of multiple resources on the information infrastructure under a centralized control environment, (e.g., computational resources shared within the administrative domain of one organization). A global level QoS model is built on service stability and dependability at the local and regional levels of the information infrastructure to provide

assurance for the end-to-end QoS requirement of an individual high-priority job that flows across multiple administrative domains, without a centralized control authority.

In the following sections of this paper, we describe our major research for the local, regional and global level QoS models. In these sections we reference only some of our publications, but still include all publications related to this project in the reference section.

Local Level QoS Models

The local level QoS model manages a computer or network system with a single resource that services processes (jobs) initiated by user requests for service. Timeliness is an important attribute of Quality of Service (Ye, 2002; Chen et al., 2003). Timeliness measures the time it takes a computer or network system to respond to, process, and complete a service request (job). In general, when a job arrives at a system requesting service, the system admits the job and places it in a waiting queue if the resource is busy processing another job. The time that a job spends in the system includes waiting time and processing time, which is the time it takes the resource to process the job after it is taken from the waiting queue. Processing time is usually determined by the size of a job and thus out of our control. Waiting time depends on admission control (which determines if a job is admitted into the system) and job scheduling (which determines job servicing order).

We aim to reduce the variance of job waiting times for service stability and dependability. Ultimately we would like to see the waiting time of any job in a computer or network system, at any given time, remain the same. That is, the objective of our local level QoS model is service stability and dependability by minimizing job WTV. Service stability of individual resources at the local level of an information infrastructure makes them become "standard parts", enabling predictable performance of each job passing through each individual resource, which in turn will greatly simplify QoS assurance problems at the regional and global levels. Otherwise regional and global level QoS assurance becomes extremely difficult to plan and manage.

Our local level QoS model focuses on admission control, job scheduling and their effects on job waiting time. Our research in this area has led to the following:

1) Batch Scheduled Admission Control (BSAC) – a method that controls the admission of jobs into a local level system in batches
2) Verified Spiral (VS) and Balanced Spiral (BS) - job scheduling methods to determine the sequence of admitted jobs for receiving service one by one, leading to:
    - Job scheduling method for the weighted WTV problem in which each job has a priority weight
    - Inventory control method that reserves the capacity of a resource to further stabilize job WTV
    - Mathematical relationship of WTV with buffer size and job processing time distribution such that a system administrator can use this relationship to set the buffer size according to the desired WTV and job processing time distribution for desired service stability and dependability
3) Implementation of BSAC and BS on a working hardware router
4) Discovery and geometric interpretation of an eye-shape pattern revealing the relationship between job waiting time variance and mean, leading to:

- Mathematical method for deriving an optimal job sequence for minimizing the waiting time variance of a given problem
- Mathematical derivation of the optimal job sequence for the WTV problem

5) Weighted Shortest Processing Time – Adjusted (WSPT-A) - another job scheduling method to stabilize service on an individual resource (Ye et al., 2005)
6) Feedback control mechanism for service stability and dependability from an individual resource (Ye et al., 2003)

In the remainder of this section, we briefly overview items 1-4, and point to publications on this research for further details. We refer the reader to the respective references for details on items 4 and 5.

BSAC for Admission Control

Most computer and network systems (e.g., web servers and routers) on the Internet currently do not use any type of admission control for jobs requesting service. For example, a router usually admits all arriving data packets (jobs) and places them in a waiting queue if the bandwidth (resource) is being used by other data packets, or drops it if the waiting queue is full. Since a job's waiting time depends on the number of other jobs waiting, it varies greatly. With no admission control, the number of jobs waiting in a system is unpredictable, waiting time is unstable, and the timeliness of service undependable. Additionally, the job waiting time variable has a negative impact on QoS and user satisfaction.

There is little existing work that investigates the impact of admission control on the variance of job waiting times. We develop an admission control method, called Batch Scheduled Admission Control (BSAC), to reduce WTV (Ye et al, in Review). A computer/network system with BSAC makes a decision of whether or not to admit an incoming job. Using the BSAC method, dynamically arriving jobs are admitted into the system in batches. Many criteria can be considered to determine the size of each batch. For example, the number of jobs in the batch can be used to define the batch size if the processing time does not vary much among jobs. The total load of the processing times of all the jobs in the batch can also be used to define the batch size if there is a large variance in the processing times of jobs. System administrators can set the batch size based on memory capacity, desired upper bound on waiting time (since a larger batch causes more jobs waiting for longer time), and so on.

To assure QoS on computers and networks, we can classify jobs into two groups with different priorities: high and low (Almquist, 1992; Blake et al., 1998; Braden et al., 1994). The system we design is for high priority jobs. We assume that lower priority jobs are processed whenever the system is not busy, or by another resource. At any given time, the system maintains current batch and a waiting batch. The resource is taking jobs in the current batch one by one for service according to their scheduled order. During the processing of jobs in the current batch, arriving jobs, if admitted, are placed in the waiting batch. For an incoming job, the system admits it if adding it to the waiting batch does not produce a batch whose length exceeds the batch size, and rejects it otherwise. A rejected job may be advised by the system to come back at a later time or turn to another system with a similar resource.

A time slot, T, for processing one batch can be set according to the maximum time required to process all jobs in any batch of a given size. At times T, $2T$, $3T$, etc. the resource has finished processing all jobs in the current batch, and moves jobs in the waiting batch into the

current batch for processing. The length of the current batch may be less than the batch size since there may not have been enough jobs that arrived during a time interval of T to fill it up. When this occurs, low-priority jobs can fill up the current batch. So, any residual time is spent processing lower priority jobs, maintaining a time synchronized schedule for high priority jobs.

Since current computer and network systems admit all jobs, waiting time depends on the variable job arrival times. As a result the number or load of jobs waiting is unbounded. We expect that BSAC reduces WTV because it bounds and stabilizes the number or total load of jobs in each batch processed for service. Waiting time consists of waiting time in the waiting and current batches. Time in the waiting batch is the same for all jobs. Time in the current batch is bounded by the batch size. Hence, the total waiting time of each job is bounded and stabilized.

We test and compare WTV between BSAC and no admission control for two test problems with 30 and 3000 jobs respectively that arrive in two different arrival patterns: bursty and steady. All jobs must be scheduled for service by the resource. Among existing job scheduling methods in the literature, we select five representative methods to implement in this study: First Come First Serve (FCFS), Shortest Processing Time first (SPT), Weighted SPT first (WSPT), Earliest Due Date (EDD), Highest Levels First with Estimated Times (HLFWET) and Smallest Co-Levels First with Estimated Times (SCFWET).

The results in Table 1 show that BSAC provides a smaller variance and thus better stability of job waiting times when compared to no admission control. This holds for both bursty and steady job arrival conditions, regardless of the job scheduling method or problem size.

Table 1. Variance of job waiting times for BSAC and no admission control.

Scheduling method	No admission control and bursty pattern	No admission control and steady pattern	BSAC and both arrival patterns
FCFS, Problem 1	148.8	89.06	21.16
FCFS, Problem 2	240.5122	30.9732	11.9975
SPT, Problem 1	10257	3080.62	282
SPT, Problem 2	93.0607	19.193	3.752
WSPT, Problem 1	11973.67	4095.51	355.28
EDD, Problem 1	10944.73	5092.29	361.23
HLFWET, Problem 1	11291.42	4619.90	450.54
SCFWET, Problem 1	10941.06	3806.60	323.30

Spiral Methods for Job Scheduling

Although jobs arrive dynamically at an individual computer or network resource, the application of BSAC allows job admission into a resource in batches. Hence, we investigate the problem of scheduling a batch of jobs, with given processing times, to minimize WTV for service stability and dependability. When scheduling jobs, we may take into account several factors of individual jobs, such as their processing times, priorities, due dates, and so on (Pinedo, 1995). In this project, we develop two job scheduling methods, Verified Spiral and Balanced Spiral, to reduce the WTV of jobs without priorities in a given batch on a single computer or network resource. (Li, 2005; Ye, in Review). This project also consider the factors influencing WTV (e.g. sum and distribution of job processing times, and scheduling methods), as well as the

weighted WTV problem, in which jobs have different weights (priorities), related job scheduling methods are Weighted Verified Spiral and Weighted Simplified Spiral (Li, 2005).

In this section, we first describe the mathematical formulation of the scheduling problem. Next we outline the scheduling methods: Verified Spiral and Balanced Spiral. We include some results comparing our Spiral methods to other scheduling methods, with regards to minimizing WTV. We refer the reader to respective references for details and work not outlined here.

Given n jobs in a batch all available at time zero, there are $n!$ possible job sequences. Assume that there is no set-up time for each job to be processed by a resource. We formulate a WTV problem as an Integer Programming problem. The decision variables are s_{ij}'s, for $i = 1, \ldots, n$ and $j = 1, \ldots, n$, representing a job sequence as well as the position of each job in the sequence. The binary integer, s_{ij}, is 1 if job j is scheduled at position i, and 0 otherwise. There are n positions in the job sequence since there are n jobs. The job to be scheduled and thus processed first is placed at position 1, the job to be scheduled second is placed at position 2, and so on. The processing time of job j is p_j, which is given. The waiting time of the job at position i is w_i.

Objective Function:

$$\text{Minimize: } \frac{1}{n-1}\sum_{k=1}^{n}\left(w_k - \frac{1}{n}\sum_{i=1}^{n} w_i\right)^2 \tag{1}$$

Subject to:

$$\sum_{j=1}^{n} s_{ij} = 1, i = 1,\ldots,n; \tag{2}$$

$$\sum_{i=1}^{n} s_{ij} = 1, j = 1,\ldots,n; \tag{3}$$

$$s_{ij} = 0 \text{ or } 1;\ \text{i, j} = 1,\ldots,\text{n}; \tag{4}$$

$$w_1 = 0; \tag{5}$$

$$w_i = w_{i-1} + \sum_{j=1}^{n} s_{i-1,j} * p_j,\ \text{i} = 2,\ldots,\text{n}. \tag{6}$$

The objective function of the WTV problem (Equation 1) is to minimize the sample variance of waiting times of n jobs. Equation 2 describes the constraint that there can be only one job assigned to each position. Equation 3 indicates that one job can be placed at only one position. Equation 4 gives the integer constraint. The waiting time of the first job to be processed is 0, which is given in equation 5. Equation 6 defines the waiting time of the job at position i ($i \geq 2$), which is the waiting and processing time of the job at position i-1.

Since the WTV problem is NP-hard (Kubiak, 1993), there are no efficient search algorithms to find the optimal sequence(s). Among four heuristic job scheduling methods for the WTV problem developed by Eilon and Chowdhury, two methods produce better performance for a number of small data sets: E&C1.1 and E&C1.2 (Eilon and Chowdhury, 1977). We select these two methods for further testing on large data sets and comparison with our scheduling methods, which make further improvement of E&C1.1 and E&C1.2. In addition to these methods, we compare our methods to that of SPT.

Verified Spiral (VS)

In Verified Spiral we modify E. &. C. 1.1 by first incorporating Schrage's conjecture and Hall and Kubiak's proof about the placement of the first, second and third longest jobs (Schrage, 1975; Hall and Kubiak, 1991). And then, modifying the spiral placement of remaining jobs by adding a selection of two positions to place the next job, either before the tail or after the head of the job sequence, based on which position produces a smaller variance of waiting times for jobs already in the sequence, as follows.

Suppose an arbitrary job set $P = \{p_1, p_2, \mathrm{K}, p_n\}$ needs to be scheduled for a single resource. Assume that the jobs are numbered such that $p_1 \le p_2 \le \Lambda \le p_n$.

1. To start, first place job p_n in the last position, job p_{n-1} in the last-but-one position, job p_{n-2} in the first position, and job p_1 in the second position. Now the job sequence becomes $\{p_{n-2}, p_1, p_{n-1}, p_n\}$. The job pool has the remaining jobs $\{p_2, \mathrm{K}, p_{n-3}\}$.
2. Remove the longest job from the job pool, place the job either exactly before job p_1 or exactly after job p_1 in the job sequence, depending on which position produces a smaller WTV of the job sequence so far.
3. Repeat Step 2 until the job pool is empty.

Balanced Spiral (BS)

To reduce the computational cost associated with the VS method, the BS method replaces the verification of WTV during the placement selection (Step 2). In BS, we maintain the balance of the total processing time of jobs in the left (L) and right (R) side of the sequence, while placing a job from the job pool, as follows:

1. To start, first place job p_n in the last position, and job p_{n-1} in the last-but-one position, and then job p_{n-2} in the first position. Let sequence $L = \{p_{n-2}\}$ and sequence $R = \{p_{n-1}\}$. Note that p_n is not included in R. We denote the sum of the processing times of the jobs in L and R as SUM_L and SUM_R respectively. The job pool has the remaining jobs $\{p_1, p_2, \mathrm{K}, p_{n-3}\}$.
2. If $SUM\_L < SUM\_R$, remove the largest job from the job pool, append the job to the last position of L, and update SUM_L; else if $SUM\_L \ge SUM\_R$, remove the largest job from the job pool, add the job to the first position of R, and update SUM_R.
3. Repeat Step 2 until the job pool is empty.

We develop the BS method based on an observation of balanced L and R in the optimal sequence of a special case, and the near-balanced L and R that we obtain at each step when placing a job to construct the optimal sequence for some small-size WTV problems.

Comparison Testing

We test and compare VS and BS with three other job scheduling methods: SPT, E&C1.1 and E&C1.2. Table 2 shows the percentage deviation from the lower bound job WTV on nine

small problems (S1-S9) and four large problems (L1-L4). The results demonstrate that VS and BS give comparable or better performance in minimizing WTV for these problems.

Table 2. Deviation from lower bound WTV for SPT, E&C1.1, E&C1.2, VS and BS.

Test Problem	SPT	E& C1.1	E&C1.2	VS	BS
S1	13.72%	0.36%	0.00%	0.00%	0.00%
S2	19.15%	0.27%	0.00%	0.00%	0.00%
S3	31.07%	0.23%	0.03%	0.00%	0.00%
S4	34.08%	0.19%	0.09%	0.00%	0.00%
S5	48.07%	0.11%	0.09%	0.00%	0.00%
S6	66.45%	2.41%	0.01%	0.00%	1.37%
S7	17.90%	0.02%	0.01%	0.00%	0.00%
S8	35.31%	1.30%	0.01%	0.00%	0.01%
S9	51.14%	0.40%	0.17%	0.01%	0.01%
L1	15.7%	0.00%	0.00%	0.00%	0.00%
L2	99.3%	0.02%	0.01%	0.00%	0.00%
L3	62.8%	0.00%	0.00%	0.00%	0.00%
L4	49.7%	0.11%	0.07%	0.07%	0.41%

Hardware Implementation of BSAC and BS

To show the feasibility and performance of our BSAC and BS algorithms, we implement them on a research router using the Intel IXP1200 network processor. The router is able to run both algorithms for minimizing the WTV of jobs arriving for service. In initial experiments we process 1,000 packets in the router grouped in batches of size 10. Our results show that the WTV for a batch under the FCFS scheme with no admission control is 44,645. With BSAC added for admission control, and still using FIFO scheduling, we get a better WTV of 43,433. When we add BSAC and BS together, the WTV reduces even more to 36,770.

Thus, we find that our initial tests of implementing our methods on real hardware show that the algorithms are feasible and can improve performance with respect to minimizing job WTV. The details of our implementation, extended experimental results, and further analysis on performance metrics, such as WTV and running time, will be reported in future publications.

A Universal Pattern in the Optimization of Waiting Time Variance

During our investigation of WTV minimization, we discover an interesting eye shape pattern when we plot the waiting time variance over mean of all possible job sequences for a given problem, as shown in Figure 1. The eye shape pattern has several important implications for the WTV minimization problem. This pattern allows us to evaluate the sacrifice of the mean waiting time while pursuing the minimum WTV, and possibly develop a mathematical method to estimate the minimum WTV and derive the optimal job sequence.

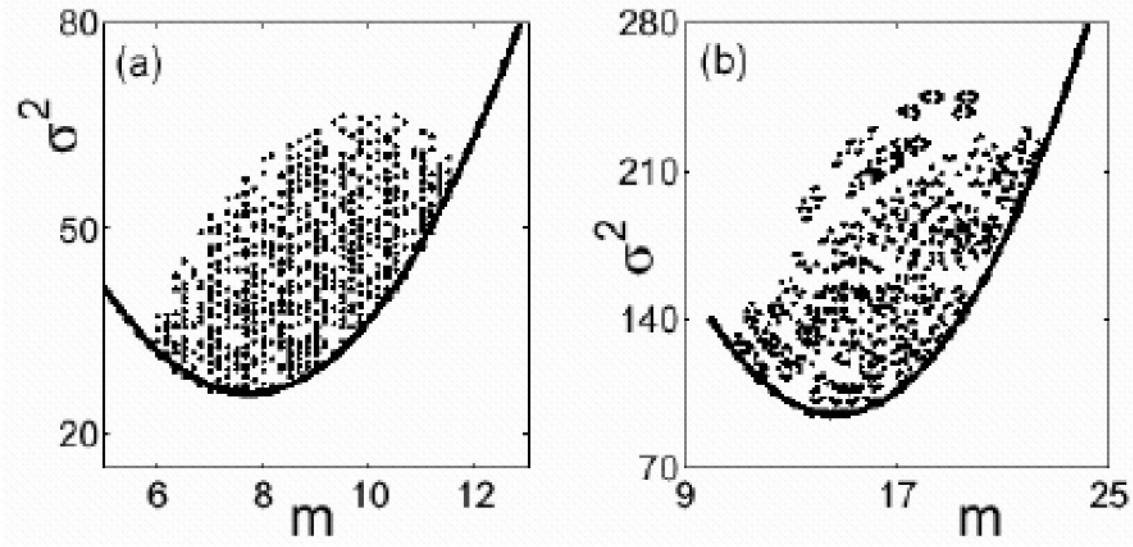


Figure 1. Examples of the eye-shape pattern: (a) P = (1,2,3,4,5,6), and (b) P= (2,3,5,9,10,11). The lower contour of the eye shape can be fit by a quadratic curve.

To evaluate the sacrifice of the mean waiting time while pursuing the minimum WTV, for the nine small-size problems in Table 2, S1-S9, we examine and compare the mean waiting times of three data points in each of the eye-shape plots for these problems:

1. the data point with the minimum WTV that is located at the lowest point on the y-axis of the eye shape pattern (if there are multiple data points with that minimum WTV value, the data point with the smallest mean waiting time among these data points is examined)
2. the data point with the minimum mean waiting time that is located at the left corner of the eye shape pattern
3. the data point with the maximum mean waiting time that is located at the right corner of the eye shape pattern

It is shown that data point 2 corresponds to the job sequence produced by SPT and 3 corresponds to the job sequence produced by the job scheduling method of placing the job with the Longest Processing Time (LPT) first (Pinedo, 1995). Data points 2 and 3 define the range of the mean waiting times from all the possible job sequences. From Figure 1, we observe that data point 1 is closer to 2 (minimum mean) than to 3 (maximum mean) on the x-axis. We examine the distances between the data points on the x-axis and the ratio of this distance to the range. We find that, on average, for these nine problems the optimal job sequence with minimum WTV (data point 1) sacrifices the mean waiting time by about 28.73% of the entire range of mean waiting times. Hence, while pursuing WTV minimization, we do not sacrifice much from the optimal waiting time mean.

Two other important implications of the eye shape pattern lie in the lower contour and the use of this contour to find the minimum WTV and derive the job sequence that produces it. If we can mathematically derive the minimum WTV using the lower contour, one implication is that we can compute the minimum WTV without knowing the exact job sequence that produces it. This allows us to evaluate the WTV of a job sequence produced by a heuristic job scheduling

method. Further deriving the job sequence that produces the minimum WTV leads to another implication; we will have a mathematical method of deriving the optimal sequence for a given problem, rather than using a space search or heuristic method to find a sequence with good or near optimal WTV. In our study, we have provided a geometric interpretation of the eye shape pattern. We have also inferred the mathematical form of the function for the lower contour of the eye shape pattern to be a quadratic function of waiting time variance over mean, and have verified this function through a number of examples.

Regional Level QoS Models

The regional level QoS model is similar to the local level QoS model in that the objective is to minimize WTV for service stability and dependability. However, the regional level QoS model deals with a network of computers with shared computational resources in the centralized control environment, like that for computational grids. Since existing research work on computational grids does not address the objective of service stability through minimizing WTV, we work on the multiple-resource job scheduling problem to schedule jobs on multiple resources of same or different capacities for minimizing WTV. Our research has produced the following:

1) Mathematical proof of the equivalence of the completion time variance (CTV) problem and the WTV problem involving multiple shared resources of the same capacity; the identical parallel machine problem (Xu)
2) Mathematical proof of certain properties of the optimal sequence(s) for the multiple-resource WTV problem (Xu)
3) Five heuristic algorithms for the identical parallel machine problem: First in First Out (FIFO)+VS, SPT+VS, LPT+VS, Dynamic VS (DVS) and Dynamic BS (DBS) (Xu).
4) Job scheduling methods for the WTV problem involving multiple resources of different capacities.

In this section we overview item 3 above. Items 1 and 2, add additional information on 3, can be found in the associated reference. Item number 4 is in progress.

Heuristic algorithms for the Identical Parallel Machine WTV Problem

The identical parallel machine CTV and WTV problems are NP-complete (Cai and Cheng, 1998; Xu). Hence, the use of a heuristic algorithm for computational efficiency is justified. In this section, five heuristic algorithms are presented for the identical parallel machine WTV problem: FIFO+VS, SPT+VS, LPT+VS, DVS and DBS (Xu). We compare these five algorithms with FIFO, SPT and LPT alone. These 8 algorithms are described in this section.

1. FIFO (First-In-First-Out)
   FIFO is considered in this study because it is one of the most commonly used dispatching rules in scheduling and is widely used for a variety of Internet services. In FIFO, we assume the jobs arrive in a random order and all jobs have arrived. All machines are idle at the beginning. The first job is served by an idle machine. The next job will be served by another idle machine. If all machines are busy, then the next job will be served on a

machine that becomes free next. That is, in FIFO both job dispatching to machines and job scheduling on each machine follow the FIFO order.

2. SPT (Shortest Processing Time)
   SPT is presented here because it is optimal to a related measure as presented in Pinedo, 1995. In the SPT heuristic, jobs are first sorted in increasing order of their processing times. The smallest m jobs are assigned to the first position on each machine, and then whenever a machine is freed, the next smallest job is assigned to that machine. That is, both job dispatching to machines, and scheduling on each machine, follow the order of SPT first.
3. LPT(Longest Processing Time)
   LPT is considered in this study because it is also optimal to a related measure. In LPT jobs are sorted in a decreasing order of their processing times. The largest m jobs will be assigned to the first position on each machine, and then whenever a machine is freed, the next largest job will be assigned to that machine. Hence, in LPT, job dispatching to machines and job scheduling on each machine follows the order of LPT first.
4. FIFO+VS (FIFO + Verified Spiral)
   FIFO+VS is shown in Figure 2.
5. SPT+VS (SPT + Verified Sprial)
   This is similar to FIFO + VS except that FIFO is replaced by SPT for job dispatching to machines in Step 1.
6. LPT+VS (LPT + Verified Sprial)
   This is similar to FIFO + VS except that FIFO is replaced by LPT for job dispatching to machines in Step 1.
7. DVS (Dynamic Verified Spiral)
   DVS checks and compares the waiting time variances from the possible assignments of a given job to a possible machine. The DVS heuristic is presented in Figure 3.
8. DBS (Dynamic Balance Spiral)
   DBS is similar to DVS except that VS is replaced by BS to schedule the jobs assigned to each machine $i \in M$. The BS method is shown in Figure 4.

FIFO+VS

1 $\overline{w}_{max} \leftarrow 0$
2 Use FIFO to assign jobs to m machines.
3 **for** $i \leftarrow 1$ **to** m
4 **do**
5 Apply Verified Spiral (VS) [4] to schedule the jobs assigned to machine i
6 Calculate $\overline{w_i}$
7 **if** $\overline{w_i} > \overline{w}_{max}$
8 **then** $\overline{w}_{max} = \overline{w_i}$
9 **for** $i \leftarrow 1$ **to** m
10 **do**
11 $r_i = \overline{w}_{max} - \overline{w_i}$
12 Let the current schedule be $\lambda_{current}$
13 Calculate the variance $var(\lambda_{current})$
14 PRINT $\lambda_{current}$, $var(\lambda_{current})$

Ye et al. [4] show that the schedule from Verified Spiral (VS) for a single machine is very close to the optimal solution of $1||WTV$. VS is briefly described below.

VERIFIED SPIRAL

1 $J \leftarrow \{p_1, p_2, \ldots, p_n\}$, where $p_1 \leq p_2 \leq p_3 \leq \ldots \leq p_n$
2 $L \leftarrow [p_{n-2}]$, $R \leftarrow [p_{n-1}, p_n]$
3 Place the smallest job, p_1, in between L and R.
4 $J \leftarrow J - \{p_1, p_{n-2}, p_{n-1}, p_n\}$
5 $j \leftarrow n - 3$
6 **while** $J \neq \emptyset$
7 **do**
8 $S_R \leftarrow [L, p_1, p_j, R]$, calculate $Var(S_R)$.
9 $S_L \leftarrow [L, p_j, p_1, R]$, calculate $Var(S_L)$
10 **if** $Var(S_R) \leq Var(S_L)$
11 **then**
12 $R \leftarrow [p_j, R]$
13 **else**
14 $L \leftarrow [L, p_j]$
15 $j \leftarrow j - 1$
16 $J \leftarrow J - \{p_j\}$

Figure 2. FIFO+VS Heuristic

DYNAMIC VERIFIED SPIRAL(DVS)

1 $J \leftarrow \{p_1, p_2, \ldots, p_n\}$, where $p_1 \leq p_2 \leq p_3 \leq \ldots \leq p_n$
2 $j \leftarrow 1$, $\lambda \leftarrow \emptyset$, $\lambda_{current} = \emptyset$
3 **while** $J \neq \emptyset$
4 **do**
5 **for** $i \leftarrow 1$ **to** m
6 **do**
7 $VAR_{min} \leftarrow \infty$, $\overline{w}_{max} \leftarrow 0$
8 Add p_j to machine i under schedule λ
9 **for** $i \leftarrow 1$ **to** m
10 **do**
11 Apply Verified Spiral (VS) to schedule the jobs assigned to machine i
12 Calculate $\overline{w_i}$
13 **if** $\overline{w_i} > \overline{w}_{max}$
14 **then**
15 $\overline{w}_{max} \leftarrow \overline{w_i}$
16 **for** $i \leftarrow 1$ **to** m
17 **do**
18 $r_i \leftarrow \overline{w}_{max} - \overline{w_i}$
19 Let $\lambda_{current}$ be the current schedule, calculate the variance $var(\lambda_{current})$
20 **if** $var(\lambda_{current}) < VAR_{min}$
21 **then**
22 $VAR_{min} \leftarrow var(\lambda_{current})$
23 $\lambda_{min} \leftarrow \lambda_{current}$
24 $\lambda \leftarrow \lambda_{min}$
25 $J \leftarrow J - \{p_j\}$
26 $j \leftarrow j + 1$
27 PRINT λ_{min}, VAR_{min}

Figure 3: Dynamic Verified Spiral Heuristic

```
BALANCED SPIRAL
1  J ← {p_1, p_2, ..., p_n}, where p_1 ≤ p_2 ≤ p_3 ≤ ... ≤ p_n
2  Place job p_n in the last position
3  L = [p_{n-2}], R = [p_{n-1}]
4  J ← J − {p_{n-2}, p_{n-1}, p_n}
5  j ← n − 3
6  SUM_L = Σ_{k∈L} p_k, SUM_R = Σ_{k∈R} p_k
7  while J ≠ ∅
8        do
9           if SUM_L < SUM_R
10            then
11                  L ← [L, p_j]
12                  Update SUM_L
13            else
14                  R ← [p_j, R]
15                  Update SUM_R
16          J ← J − {p_j}
17          j ← j − 1
```

Figure 4. Scheduling Method used in Dynamic Balanced Spiral

Comparison of Heuristics

Table 3 shows the results for six small-size WTV problems, where WTVD is the Waiting Time Variance Deviation from the optimal solution and WTMD is the Waiting Time Mean deviation from optimal. We find the optimal solution by enumerating all possible schedules. For each problem, FIFO, SPT and LPT are the worst among all the heuristics in waiting time variance. However, a significant improvement is made when VS is added to these three heuristics. DVS has the best performance in waiting time variance among all heuristics, and in 2 out of 6 problems it gives the optimal solution. The performance of DBS is very close to DVS, and DBS gives the optimal solution for one out of six problems. Further results on larger problems and varying distributions are given by Xu.

Table 3: Comparison Results for 8 methods on 6 small-size WTV problems

Heuristic	Measurement	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	Problem 6
FIFO	WTVD	29.40%	50.35%	705.91%	12.57%	338.41%	51.99%
	WTMD	13.33%	62.85%	196.25%	18.65%	91.62%	41.53%
SPT	WTVD	20.90%	28.97%	36.54%	10.01%	14.32%	31.66%
	WTMD	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
LPT	WTVD	115.30%	143.48%	838.49%	49.48%	569.12%	180.99%
	WTMD	88.89%	117.71%	521.61%	39.45%	282.20%	142.37%
FIFO+VS	WTVD	4.70%	0.65%	29.34%	0.91%	24.61%	3.03%
	WTMD	22.22%	51.61%	96.38%	13.58%	58.56%	54.66%
SPT+VS	WTVD	0.82%	1.91%	5.91%	0.42%	0.79%	1.77%
	WTMD	28.89%	43.20%	58.21%	15.53%	23.56%	42.37%
LPT+VS	WTVD	0.15%	0.03%	11.57%	0.02%	53.06%	0.00%
	WTMD	22.22%	24.96%	108.93%	9.90%	107.18%	25.00%
DVS	WTVD	0.15%	0.03%	3.58%	0.00%	0.13%	0.00%
	WTMD	22.22%	24.43%	81.92%	9.67%	19.37%	25.00%
DBS$_\lambda$	WTVD	0.15%	0.03%	4.39%	0.00%	0.13%	0.14%
	WTMD	22.22%	24.43%	87.32%	9.67%	19.37%	25.71%

Global Level QoS Assurance

At the global level, we investigate providing end-to-end QoS on general networks. Our objectives are to assure end-to-end QoS of high-priority individual jobs and minimize traffic congestion. In this area we have achieved the following:

1) Review of existing decentralized resource management methods (Wu et al., 2005) and existing work on QoS in the application fields of computers and networks, air traffic management, postal mail delivery, and so on.
2) Metrics of QoS requirements for various applications on the information infrastructure towards QoS standards, using human factors and technology data.
3) Network simulation experiments to reveal sensitive points of data collection and measures, and emergence of network hierarchy.
4) A job reservation and execution protocol and supporting algorithms to assure the end-to-end QoS of an individual high-priority job, and minimize traffic congestion.
5) Mathematical theories of scale free networks, probabilistic methods of generating scale free networks, and empirical results on attack resistance and infection.

In this section we overview the above items and refer to our publications for further reading and research results. Research that is ongoing is briefly overviewed based on goals and initial directions.

QoS Metrics

Concise specification of QoS requirements is vital to realizing QoS assurance on the information infrastructure. Since different applications have different service features, each application should specify its explicit QoS requirements to a computer network in order to achieve the desired QoS. If there are no requirements given, the network will take for granted

that any level of service is acceptable, and therefore can provide any level of QoS support. The types of QoS support provided, such as bandwidth and priority in a router's queue, may lead to delay and jitter that can then render QoS unacceptable. Consequently, if we want to satisfy QoS of various applications, the first step is to identify the QoS metrics of each application on the information infrastructure.

In this study, we first analyze the influence of two key factors, human factors and technology attributes, on QoS requirements. Next, we classify an application based on the application's different service characteristics resulting mainly from these two key factors. Finally, based on existing literature, we systematically propose numerical measures, which can quantitatively represent the QoS requirements of various applications (including web browsing, enhanced web browsing, email, ftp, telnet, Internet relay chat, audio broadcasting, video broadcasting, interactive audio on demand, interactive video on demand, telemetry, audio conferencing, audiographic conferencing, video conferencing, videophony, and voice over IP) with different service characteristics in Multi Service Networks, along with the rational behind these choices. More details of the QoS metrics for various applications are presented in (Chen et al, 2003).

Table 4. QoS metrics for web browsing

Appl. Class	Technology Attributes	QoS Metrics						
		Timeliness			Preciseness			Accuracy
		Response time Expected by Users	Delay (ms)	Jitter (ms)	Data Rate (bps)	Required Bandwidth (bps)	Loss Rate	Error Rate
Web Browsing	Non Real Time and Asymmetric	2-5 Seconds	< 400	N/A	< 30.5 K	< 30.5 K	Zero	*Zero*

Data Collection and Measures

At the global level, we investigate detecting emergent behavior on a network of networks modeled as the Internet. We determine that the Internet is a scale free network, and build our internet model using the two key characteristics: preferential attachment and growth. We run experiments using various levels and types of attacks and failures, and collect data to determine which metric best describes the state of the system, and at which points to collect this metric. Details and results of this research work are currently in the publishing process (Ye, Farley and Aswath, accepted).

End-to-end QoS Assurance Problem

We first define and formalize the end-to-end QoS assurance problem. Next, we introduce a simulation design to investigate the application of our research work at the local and regional levels to this global QoS problem.

Definition and formulation of the end-to-end QoS assurance problem

We define end-to-end QoS assurance as a fixed path problem of self-interest only. Given the following, determine the arrival time of each flow at each hop along the path of the flow:

a. n flows
b. a fixed end-to-end path of each flow
c. end-to-end timeliness target of each flow
d. m resources required by all n flows
e. service capacity and waiting time at each resource from local-level and regional-level QoS models

We make the fixed path assumption based on existing evidences of a stable primary path of an end-to-end flow. For more than 50% of destinations there is only one dominant path, and for 25% of destinations there are exactly two domain-level paths (Govindan and Reddy, 1997). The reason for this is that routing policy is usually set by bi-lateral transit agreements. In most cases, a domain keeps a primary and a back-up transit to a collection of destinations. It is also shown that the likelihood of observing a dominant route is 82% at the host level, 97% at the city level and 100% at the autonomous system (AS) level (Paxson, 1996). In EGP (Exterior Gateway Protocol) for inter-AS routing, almost 90% of recorded updates contain close to 0% new information, indicating stable routes (Chinoy, 1993). The Border Gateway Protocol (BGP), contains only incremental updates. Injecting just 10% of the total inter-AS reachability information (about 200 entries) into the inter-AS routing permits the forwarding of at least 85% of the transit traffic without resorting to encapsulation (Rekhter and Chinoy, 1992) More than 80% of prefixes are reachable through a primary path for more than 95% of time.

In our ongoing work we consider subproblems of the fixed path problem of self-interest only. First, we allow each flow to determine its own arrival time by considering shortest possible time along with slack time. Next, we resolve timing conflicts at each resource while making the slack time of each flow $\geq$ zero. And finally, coordinate the resolutions at various resources. We also consider finding solutions for the open path problem. Such as assuring end-to-end QoS of some flows, expanding the set of m resources by pursuing alternative paths for those flows. Subproblems here include selecting alternative resources for those flows and solving a fixed path problem with an expanded set of resources. For both fixed and open path problems, we consider self-interest and global interest of minimizing global traffic congestion (e.g., minimizing traffic bottlenecks).

A Job Reservation and Execution Protocol

To address the end-to-end QoS assurance problem, we incorporate our work at the local and regional levels. Consider a network where variance in job waiting times is minimized. The processing time of a job depends on its size and can be calculated. By minimizing waiting time variance, we can predict the completion time (waiting time + processing time) of a job at each point in a network. Our network model considers only high priority jobs and assumes low priority jobs are handled whenever resources are idle.

In addition to incorporating our work at the local and regional levels, our experimental framework uses the concept of path reservation, as seen in the Resource ReSerVation Protocol (RSVP) proposal (Braden et al., 1997). However, we aim to overcome some problems with RSVP. We briefly overview RSVP and our protocol.

RSVP reserves a path for a flow on the Internet. This reservation is made at intermediate routers along the path. After a reservation for a flow is made, the jobs (in the form of a set of individual packets) in that flow travel along the reserved path. The idea behind this method is that by reserving resources to manage a flow, its QoS requirements can be assured. Some of the key points to RSVP are:

1. RSVP is receiver oriented, i.e., the receiver initiates the reservation request.
2. RSVP does not perform its own routing; it uses underlying routing protocols to determine where it should carry reservation requests, i.e., RSVP runs on top of the Internet Protocol.
3. Once the path is determined, the receiver sends a RESV packet with the bandwidth requirement along the determined path.
4. Each intermediate node on the path then makes a decision about accepting or rejecting the RESV request.
    a. Fail – NACK or error sent to the originator of the RESV packet.
    b. Success – set parameters in packet classifier and packet scheduler to achieve required QoS.

Our protocol is also based on path reservation and incorporates our work at the local and regional levels to minimize the variance of job waiting times at each point along the reserved path. We briefly outline the two phases (probe and job) of our protocol.

1. Probe Phase:
    a. At the source node, find the best path among n possible paths based on historic information and current state information. The source node subscribes to the historic performance and current state information from intermediate routers along n possible paths.
    b. Source initiated: source sends a probe packet along the best path. The probe packet carries the parameters (job_id, start_time, J, D_{ee}), where J is the job (packet) size, D_{ee} is the end to end delay requirement.
    c. Every intermediate router i has three parameters (B_{ij}, P_i, D_i), where B_{ij} is the size of batch j at the router, P_i is the processing power of the router, D_i is the max (worst) possible delay that a packet can experience at this router. Each router also maintains a variable $B_{residual}$ that keeps track of how much resource is left at the router as reservations are made.
    d. For each router i, upon receiving a probe packet:

        If ($B_{residual} \geq J$),
            $D_{ee} = D_{ee} - D_i$
            If ($D_{ee}>0$)
                Forward probe packet
                $B_{residual}=B_{residual}-J$
                Add *(job_id, probe_reply_timeout)* to reservation list of batch j.
            Else
                drop probe packet

        (probe_reply_timeout = arrival time + timeout based on the avg roundtrip time)
    e. Upon destination receiving a probe packet:

If (D_{ee}–transportation time of the probe packet per job size unit * job size > 0)
Return *probe_reply* packet
Else
Drop probe packet

f. For each router *i*,
If (*probe_reply* packet not received by the time *probe_reply_timeout*)
Drop the corresponding job from the list of jobs scheduled for batch *j*.
If (*probe_reply* packet arrives without a corresponding *job_id* in the list)
Drop probe_reply packet

g. Source receives the probe_reply packet and enters Job Phase

2. Job Phase:
   a. Source sends the job (packets) along the reserved path
   b. Each intermediate router *i*, receiving the job checks to see if the job is in the job list for batch *j*
      a. If yes
         i. If the complete job arrives within its corresponding batch start time, schedule the job using BS
         ii. If the job does not arrive before its corresponding batch's start time, drop the job from high priority queue
      b. If no
         i. Keep the job in the best-effort queue

From the brief outline given, observe that our protocol assumes admission control in batches. For this we use the BSAC method from our local and regional level work. During the job phase, jobs are scheduled using the BS algorithm from our local and regional level work to minimize the waiting time variance of jobs, which gives us the ability to determine the amount of time it will take a job to travel along a path, because we can compute its completion time at each router along the path (waiting time+processing time). Without stabilizing the waiting time variance, it would not be possible to make close predictions of the completion time at each point, thereby complicating the issue of maintaining timing along the path.

Our method uses a type of resource reservation that is different than RSVP. We compare our method to that of RSVP and list some of the main advantages of our method:

1. Non-Persistent reservation. RSVP reserves a path for an entire flow, we make the reservation for a specific job (set of packets) in a flow. Some advantages of this are:
   a. Resources are not wasted if the flow is not active. Consider the case of Voice over IP (VoIP) where there are distinct active and inactive periods. If we consider an active period as a job, then RSVP reserves the path for the duration of the call, whereas our protocol only reserves the path during the active periods.
   b. We target all types of applications, and the reservations are therefore valid only as long as they are needed irrespective of whether it's a short-term or a long-term connection.
   c. We may know all of the characteristics of a job, but not of a flow which has characteristics that may change over time.

2. Less State Info: Unlike RSVP, we store less state information. We store state information corresponding to two batches, the current batch and the next batch. The state information includes only job ids and the residual capacity of a batch.
3. Parallels Routing Algorithm: RSVP runs on top of IP. Our solution is integrated parallel to the routing algorithm to incorporate adjustments based on network dynamics. For example, a path that was good at the beginning of a VoIP session may at some point become congested. In our solution, a new path may be selected mid-session since path selection is done on a per job (active period) basis.
4. Light weight and Distributed: Our solution is light weight as it does not carry much state information and distributed as each router makes its own independent decision about accepting or rejecting a job.

One of the key benefits of our method is reducing resource wastage in a reservation. There are 3 ways to look at reserving paths. Reservation per packet, reservation per job (our method) and reservation per flow (RSVP). The first case, per packet, is clearly impractical as the reservation mechanism would significantly slow down network traffic. The last case, per flow, wastes too much resource along a path when a flow is inactive. We attempt to find a middle ground between the two by defining a job (set of packets) and making the reservation for that job.

Since we consider a network based on BSAC, we investigate batch sizes with respect to the number of packets in a job. The size of a batch is a variable that can be changed, adding flexibility to the framework. Another variable is the number of batches to reserve (persistence of reservation). For this, we consider the following optimization problem:

1. Let $C_{probe,i}$ be the cost of sending probe i
2. Let $C_{resv,i}$ be the cost of holding reservation for batch i
3. Let y be the optimum reservation persistance
4. Let a_i be the number of packets in job i
5. Minimize $\sum (a_i / y) * C_{probe,i} + (y - a_i \%_0 y) * C_{resv,i}$
6. s.t. $y>1$

Our proposed solution is currently in development. The ideas presented here are preliminary. We do not claim this as a complete solution for solving end-to-end QoS. However, we aim to provide a solution that is flexible, proactive, and secure. Flexibility is inherent in the variable parameters we allow (path selection, batch size, persistence of reservation) and the ability to change these parameters dynamically. As an example of proactive, consider sending packets in a flow, and stopping the flow after finding its QoS cannot be met (reactive), we do not release packets for a job until we know the QoS can be met (proactive). Our solution is more secure in that various "pieces" of a flow may not travel along the same path, thereby increasing the difficulty in eavesdropping. The use of BSAC and BS allows the overall benefits of time synchronization on a network, for which the implications are numerous.

In addition to the benefits we are continuing to explore, we also consider the tradeoffs. Obviously adding such synchronization and reservation to a network will consume resources and add processing time. Our ongoing efforts consider the advantages of our solution and weigh them against the shortcomings. We view this problem from a framework point of view, and are not currently actively trying to integrate it into existing protocols on the Internet.

Dynamics and Security of Computer Networks

Complex networks arise in natural systems and they are also an essential part of modern society. Many real complex networks were found to be heterogeneous with power-law degree distribution (Barabasi and Albert, 1999, Albert and Barabasi, 2002, Newman, 2003): $P(k): k^{-\gamma}$, where k is the number of links of a randomly chosen node in the network and γ is the scaling exponent. This power-law, or algebraic, distribution means that the probability for a subset of nodes to possess a large number of links is not exponentially small, in contrast to random networks. Mathematically, the power-law distribution means that statistical moments of the degree variable are generally not defined, hence the name of scale-free networks. Because of the ubiquity of scale-free networks in natural and man-made systems, the security of these networks, i.e., how failures or attacks affect the integrity and operation of the networks, has been of great interest since the discovery of the scale-free property. The work by Albert et al. demonstrated that scale-free networks possess the robust-yet-fragile property, in the sense that they are robust against random failures of nodes but fragile to intentional attacks (Albert et al, 2000). However, the term fragility here means that a scale-free network can become disintegrated under attacks on a small but still appreciable set of nodes that include a substantial fraction of links in the network (Cohen, et al, 2000). An attack on a single or very few nodes will in general not bring down the network. This interesting result was actually obtained based purely on the scale-free architecture of the network. In other words, dynamics in the network, i.e., how information or load is distributed in the network, was not taken into account.

An intuitive reasoning based on the load distribution would suggest that, for a scale-free network, the possibility of breakdown triggered by an attack on or failure of even only a single node cannot be ignored. Imagine such a network that transports some physical quantities, or load. Nodes with large numbers of links receive relatively heavier load. Each node, however, has a finite capacity to process or transport load. In order for a node to function properly, its load must be less than the capacity at all time; otherwise the node fails. If a node fails, its load will be directed to other nodes, causing a redistribution of load in the network. If the failing node deals with a small amount of load, there will be little effect on the network because the amount of load that needs to be redistributed is small. This is typically the situation of random failure of nodes. However, if the failing node carries a large amount of load, the consequence could be serious because this amount of load needs to be redistributed and it is possible that for some nodes, the new load exceeds their capacities. These nodes will then fail, causing further redistributions of load, and so on. As a consequence, a large fraction of the network can be shutdown.

Cascading failures can occur in many physical systems. In a power transmission grid, for instance, each node (a generator) deals with a load of power. Removal of nodes in general can cause redistribution of loads over all the network, which can trigger a cascade of overloading failures. The recent massive power blackout caused by a series of seemingly unrelated events on August 14, 2003 in the northeastern United States and Canada seemed to have the characteristics of cascading breakdown. Another example is the Internet, where the load represents data packets a node (router) is requested to transmit and overloading corresponds to congestion (Arenas et al, 2001). The rerouting of data packets from a congested router to another may spread the congestion to a large fraction of the network. Internet collapses caused by congestion have been reported (Jacobson, 1988). With the possibility of cascading failures, a realistic concern is attacks on complex networks. In particular, for a scale-free network, the majority of the nodes deal with small amount of load, so the probability for a node with a large amount of load to fail

randomly is small. This, of course, will not be the case of intentional attacks that usually target one or a few of the most heavily linked nodes.

There have been a few recent studies on cascading failures in complex networks (Motter and Lai, 2002, Holme and Kim, 2002). In Motter and Lai, 2002, a simple mechanism was proposed to incorporate the dynamics of load in both random and scale-free networks. The model generates results that are completely consistent with the above intuition on cascading failures. For instance, it was demonstrated that random networks are robust against cascading breakdown but it can be easily triggered by intentional attacks in scale-free networks. The existing results are, however, largely descriptive and qualitative. We have addressed theoretically and numerically the fundamental mechanism of cascading breakdown. To make analysis amenable, we focused on scale-free networks, use the load model in Motter and Lai, 2002 that captures the essential features of cascading failures, and investigated cascades triggered by attack on a single node. Our finding is that cascading breakdown in scale-free networks can be understood in terms of a phase transition. In particular, let α be the tolerance parameter characterizing the capacity of nodes in the network. Cascading breakdown due to attack on a single node is possible only when α is below a critical value α_c. By making use of the degree distribution of scale-free networks and the concept of betweenness (Newman, 2001) to characterize the load distribution, we were able to derive a theoretical formula for estimating the phase-transition point α_c, which was verified by numerical experiments. In terms of practical utility, our result enables a possible implementation of predicting and preventing mechanism for cascading breakdown in scale-free networks.

The load dynamics in scale-free networks can be modeled, as follows. For a given network, suppose that at each time step one unit of the relevant quantity, which can be information, energy, etc., is exchanged between every pair of nodes and transported along the shortest path. To characterize the load distribution, the concept of betweenness is useful (Newman, 2001). The load (or betweenness) at a node i is defined as the total number of shortest paths passing through this node. The capacity of a node is the maximum load that the node can handle. In man-made networks, the capacity is severely limited by cost. Thus, it is natural to assume that the capacity C_i of node i is proportional to its initial load L_i (Motter and Lai, 2002),

$$C_i = (1+\alpha)L_i, \tag{7}$$

where the constant $\alpha \geq 0$ is the tolerance parameter. When all nodes are on, the network operates in a free flow state insofar as $\alpha \geq 0$. But, the removal of nodes in general changes the distribution of shortest paths. The load at a particular node can then change. If it increases and becomes larger than the capacity, the node fails. Any failure leads to a new distribution of load and, as a result, subsequent failures can occur. The failures can stop without affecting too much the connectivity of the network but it can also propagate and shutdown a considerable fraction of the whole network. Cascading failures can be conveniently quantified by the relative size of the largest connected component

$$G = \frac{N'}{N}, \tag{8}$$

where N and N' are the numbers of nodes in the largest component before and after the cascade, respectively. The integrity of the network is maintained if $G \approx 1$, while breakdown occurs if $G \approx 0$.

To obtain an analytic estimate of the critical value of the tolerance parameter, we focus on the situation where cascading failures are caused by attack on the node with the largest number of links and the failures lead to immediate breakdown of the network. That is, G becomes close to zero after one redistribution of the load. For a node in the network, its load is a function of the degree variable k. For scale-free networks, we have (Goh et al, 2001, Park et al, accepted),

$$L(k): \ k^{\eta}, \tag{9}$$

where $\eta > 0$ is a scaling exponent. To proceed, we write the degree distribution as $P(k) = ak^{-\gamma}$ and the load distribution as $L(k) = bk^{\eta}$, where a and b are positive constants. Let k_{max} be the largest degree in the network. Before the attack, we have

$$\int_1^{k_{max}} P(k)dk = N \text{ and} \tag{10}$$
$$\int_1^{k_{max}} P(k)L(k)dk = S,$$

where S is the total load of the network. These two equations give

$$a = \frac{(1-\gamma)N}{[k_{max}^{1-\gamma} - 1]} \text{ and} \tag{11}$$
$$b = \frac{\beta S}{a(1-k_{max})^{-\beta}},$$

where $\beta \equiv \gamma - \eta - 1$. After the removal of the highest degree node (it is only the first step of the whole cascading process), the degree and load distributions become $P'(k) = a'k^{-\gamma'}$ and $L'(k) = b'k^{\eta'}$, respectively. Since only a small fraction of nodes are removed from the network, we expect the changes in the algebraic scaling exponents of these distributions to be negligible. We thus write $P'(k) \approx a'k^{-\gamma}$ and $L'(k) \approx b'k^{\eta}$, where the proportional constants a' and b' can be calculated in the same way as for a and b. We obtain $a' = (1-\gamma)(N-1)/[k_{max}^{1-\gamma} - 1]$ and $b' = \beta S'/a'(1-k_{max})^{-\beta}$, where S' is the total load of the network after the attack. For nodes with k links, the difference in load before and after the attack can be written as $\Delta L(k) \approx (b'-b)k^{\eta} = (\frac{b'}{b} - 1)L(k)$. Given the capacity $C(k)$, the maximum load increase that the nodes can handle is $C(k) - L(k) = \alpha L(k)$. The nodes still function if $\alpha > (\frac{b'}{b} - 1)$ but they fail if $\alpha < (\frac{b'}{b} - 1)$. The critical value α_c of the tolerance parameter is then

$$\alpha_c = \frac{b'}{b} - 1 \tag{12}$$
$$\approx (\frac{{k_{max'}}^{1-\gamma} - 1}{{k_{max}}^{1-\gamma} - 1})(\frac{1-{k_{max}}^{-\beta}}{1-{k_{max'}}^{-\beta}})(\frac{S'}{S}) - 1$$
$$\approx (\frac{1-{k_{max}}^{-\beta}}{1-{k_{max'}}^{-\beta}})(\frac{S'}{S}) - 1$$
$$\approx \{1 - ({k_{max}}^{-\beta} - {k_{max'}}^{-\beta})\}(\frac{S'}{S}) - 1$$

$$= \{1 - k_{max'}{}^{-\beta}(-1 + (\frac{k_{max}}{k_{max'}})^{-\beta})\}(\frac{S'}{S}) - 1,$$

where the third line of Eq. (12) is obtained from the second line by using the fact $(k_{max'}{}^{1-\gamma} - 1)/(k_{max}{}^{1-\gamma} - 1) \approx 1$. This is so because both $k_{max'}{}^{1-\gamma}$ and $k_{max'}{}^{1-\gamma}$ approach zero when $N \to \infty$ and $\gamma > 1$. In the limit $N \to \infty$, we have $k_{max'}^{-\beta} : 0$, $k_{max}/k_{max'} :$ constant, and $S'/S \to 1$, so $\alpha_c \approx 0$, indicating that an infinite scale-free network cannot be brought down by a single attack if $\alpha > 0$. On the other hand, for finite size network, since $k_{max'}^{-\beta} > 0$, we have $\alpha_c > 0$, suggesting that breakdown can occur for $\alpha < \alpha_c$. The practical usage of Eq. (12) is that it provides a way to monitor the state of a (finite) network to assess the risk of cascading breakdown. In particular, the critical value α_c can be computed in time and comparison with the pre-designed tolerance parameter value α can be made. If α_c shows a tendency of increase and approaches α, early warning can be issued to signal an immediate danger of network breakdown.

References

R. Albert and A.-L. Barabsi. Statistical mechanics of complex networks. Rev. of Mod. Phys., 74(47), 2002.

R. Albert, H. Jeong, and A.-L. Barabasi. Error and attack tolerance of complex networks. Nature, 406:378–382, 2000.

P. Almquist. Type of service in the internet protocol suite. Request for Comments 1349, Internet Engineering Task Force. URL: http://www.ietf.org/rfc.html, July 1992.

A. Arenas, A. Das-Guilera, and R. Guimer. Communication in networks with hierarchical branching. Phys. Rev. Lett., 86:3196–3199, 2001.

A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. Science, 509, 1999.

S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service. Request for Comments (Informational) 2475, Internet Engineering Task Force. URL: http://www.ietf.org/rfc.html, Dec 1998.

R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: an overview. RFC 1633, IETF, proposed standard, June 1994.

R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin. Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification. RFC 2205, IETF, proposed standard, September 1997.

X. Cai and T.C.E. Cheng, Multi-machine scheduling with variance minimization. Discrete Applied Mathematics, 84: 55–70, 1998.

Y. Chen, T. Farley, and N. Ye. Qos requirements of network applications on the internet. Information, Knowledge, Systems Management, 4(1):55–76, 2003.

B. Chinoy. Dynamics of internet routing information. Proc. ACM SIGCOMM, 45-52, 1993.

R. Cohen, K. Erez, D. b Avraham, and S. Havlin. Resilience of the internet to random breakdowns. Phys. Rev. Lett., 85:4626–4628, 2000.

S. Eilon and I.G. Chowdhury. Minimizing waiting time variance in the single machine problem. Management Science, 23:567–574, 1977.

Ian Foster and C. Kesselman. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco, California, 1999.

K.-I. Goh, B. Kahng, and D. Kim. Universal behavior of load distribution in scale-free networks. Phys. Rev. Lett., 87:278701, 2001.

R. Govindan and A. Reddy. An analysis of inter-domain topology and route stability. Proc. IEEE INFOCOM, 1997.

N.G. Hall and W. Kubiak. Proof of a conjecture of schrage about the completion time variance problem. Operations Research Letters, 10:467–472, 1991.

P. Holme and B. J. Kim. Vertex overload breakdown in evolving networks. Phys. Rev. E, 65:066109, 2002.

V. Jacobson. Congestion avoidance and control. Comput. Commum. Rev, 18:314–329, Aug 1988.

W. Kubiak. Completion time variance minimization on a single machine is difficult. Operations Research Letters, 14:49–59, 1993.

Y.-C. Lai, Z. Liu, and N. Ye. Infection dynamics on growing networks. International Journal of Modern Physics B, 17:4045–4061, 2003.

Y.-C. Lai and N. Ye. Recent developments in chaotic time series analysis. International Journal of Bifurcation and Chaos, 13(6):1383–1422, 2003.

X. Li. Minimizing waiting time variance for stable quality of service on local computer and network resources. PhD Dissertation, Arizona State University, Tempe, AZ, April, 2005.

Z. Liu, Y.-C. Lai, and N. Ye. Statistical properties and attack tolerance of growing networks with algebraic preferential attachment. Physical Review E, 66:036112/1–7, 2002.

Z. Liu, Y.-C. Lai, and N. Ye. Propagation and immunization of infection on general networks with both homogeneous and heterogeneous components. Physical Review E, 67:031911/1–5,

2003. This work was selected by the Virtual Journal of Biological Physics Research for the April 1, 2003 issue.

Z. Liu, Y.-C. Lai, N. Ye, and P. Dasgupta. Connectivity distribution and attack tolerance of general networks with both preferential and random attachments. Physics Letters A, 303:337–344, 2002.

A. E. Motter, A. P. S. de Moura, Y.-C. Lai, and P. Dasgupta. Topology of the conceptual network of language. Physical Review E (Rapid Communications), 65:065102, 2002. This work was featured in New Scientists (July 13, page 22) AND Nature Science Update AND Wissenschaft in German AND in newspapers in Europe, Brazil, and China. This work was also selected by the Virtual Journal of Biological Physics Research for the July 1, 2002 issue.

A.E. Motter and Y.-C. Lai. Cascade-based attacks on complex networks. Phys. Rev. E, 66:065102(R), 2002.

National Science Foundation. New grid portal to improve u.s. researchers' access to advanced computing resources. URL: http://www.nsf.gov/cgi-bin/getpub?pr0088, 2000.

M.E.J. Newman. The structure of scientific collaboration networks. Proc. Natl. Acad. Sci., 98:404–409, 2001.

M.E.J. Newman. Who is the best connected scientist? A study of scientific coauthorship networks. Phys. Rev. E, 64:016131/32, 2001.

M.E.J. Newman. The structure and function of complex networks. SIAM Review, 45:167–256, 2003.

K. Park, Y.-C. Lai, and N. Ye. Self-organized scale-free networks. Physical Review Letters, accepted.

K. Park, Y.-C. Lai, and N. Ye. Characterization of weighted complex networks. Physical Review E, 70(2):026109/1–4, 2004.

K. Park, L. Zhao, Y.-C. Lai, and N. Ye. Jamming in complex gradient networks. Physical Review E, accepted.

V. Paxson. End-to-end routing behavior in the internet. Proc. ACM SIGCOMM, 1996.

M. Pinedo. Scheduling Theory, Algorithms, and Systems. Prentice-Hall, Inc., 1995.

Y. Rekhter and B. Chinoy. Injecting inter-autonomous system routes into intro-autonomous system routing: a performance analysis. Computer Communcations Review, January, 1992.

L. Schrage. Minimizing the time-in-system variance for a finite jobset. Management Science, 21:540–543, 1975.

T. Wu, N. Ye, and D. Zhang. Comparison of distributed methods for resource allocation. International Journal of Production Research, 43(3):515–536, 2005.

X. Xu and N. Ye. Minimization of Job Waiting Time Variance on Identical Parallel Machines. In review.

Z. Yang, N. Ye, and Y.-C. Lai. Qos model of a router with feedback control. Quality and Reliability Engineering International, accepted.

N. Ye. Qos-centric stateful resource management in information systems. Information Systems Frontiers, 4(2):149–160, 2002.

N. Ye. Network security and quality of service. In McGraw-Hill Yearbook of Science & Technology 2005, New York, New York: McGraw Hill, 232-235, 2005.

N. Ye, T. Farley, and D. Aswath. Data measures and collection points to detect traffic changes on large-scale computer networks. Information, Knowledge, Systems Management, accepted.

N. Ye, E. Gel, X. Li, T. Farley, and Y.-C. Lai. Web-server qos models: Applying scheduling rules from production planning. Computers & Operations Research, 32(5):1147–1164, 2005.

N. Ye, B. Harish, X. Li, and T. Farley. Batch scheduled admission control for service dependability of computer and network systems. In review.

N. Ye, Y.-C. Lai, and T. Farley. Dependable information infrastructures as complex adaptive systems. Systems Engineering, 6(4):225–237, 2003.

N. Ye, X. Li, and T. Farley. Job scheduling methods to reduce the variance of job waiting times on computers and networks. In review.

N. Ye, Z. Yang, Y.-C. Lai, and Toni Farley. Enhancing router qos through job scheduling with weighted shortest processing time-adjusted. Computers & Operations Research, 32(9):2255–2269, 2005.

L. Zhao, Y.-C. Lai, K. Park, and N. Ye. Onset of traffic congestion in complex networks. Physical Review E, 71(2):026125/1–8, 2005.

L. Zhao, K. Park, Y.-C. Lai, and N. Ye. Tolerance of scale-free networks against attack-induced cascades. Physical Review Letters, submitted.

Onset of traffic congestion in complex networks

Liang Zhao,[1,2] Ying-Cheng Lai,[1,3] Kwangho Park,[1] and Nong Ye[4]
[1]*Department of Mathematics and Statistics, Arizona State University, Tempe, Arizona 85287, USA*
[2]*Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, Brazil*
[3]*Department of Electrical Engineering and Department of Physics, Arizona State University, Tempe, Arizona 85287, USA*
[4]*Department of Industrial Engineering and Department of Computer Science and Engineering, Arizona State University, Tempe, Arizona 85287, USA*
(Received 30 August 2004; revised manuscript received 11 November 2004; published 24 February 2005)

Free traffic flow on a complex network is key to its normal and efficient functioning. Recent works indicate that many realistic networks possess connecting topologies with a scale-free feature: the probability distribution of the number of links at nodes, or the degree distribution, contains a power-law component. A natural question is then how the topology influences the dynamics of traffic flow on a complex network. Here we present two models to address this question, taking into account the network topology, the information-generating rate, and the information-processing capacity of individual nodes. For each model, we study four kinds of networks: scale-free, random, and regular networks and Cayley trees. In the first model, the capacity of packet delivery of each node is proportional to its number of links, while in the second model, it is proportional to the number of shortest paths passing through the node. We find, in both models, that there is a critical rate of information generation, below which the network traffic is free but above which traffic congestion occurs. Theoretical estimates are given for the critical point. For the first model, scale-free networks and random networks are found to be more tolerant to congestion. For the second model, the congestion condition is independent of network size and topology, suggesting that this model may be practically useful for designing communication protocols.

DOI: 10.1103/PhysRevE.71.026125 PACS number(s): 89.75.Hc, 89.20.Hh, 05.10.−a

I. INTRODUCTION

Free, uncongested traffic flows on networks are critical for a modern society as its normal and efficient functioning relies on such networks as the internet, the power grid, and transportation networks, etc. To ensure free traffic flows on a complex network is naturally of great interest. The aim of this paper is to address this problem via modeling. Our particular interest is to understand under what conditions traffic congestion can occur on a complex network and to explore possible ways of control to alleviate the congestion. The models we have constructed are based on the setting of information transmission and exchange on the internet. There have been many previous works in this direction [1–14]. A basic assumption used in these studies is that the network possesses a regular and homogeneous structure. Recent works reveal, however, that many realistic networks including the internet are complex with scale-free and small-world features [15,16]. It is thus of paramount interest to study the effect of network topology on traffic flow, which is the key feature that distinguishes our work from the existing ones. While our model is for computer networks, we expect it to be relevant to other practical networks in general, such as the postal service network or the airline transportation network. Our studies may be useful for designing communication protocols for complex networks.

The structure and dynamics on complex networks have attracted a tremendous amount of recent interest [16–18] since the seminal works on scale-free networks by Barabási and Albert [15] and on the small-world phenomenon by Watts and Strogatz [19]. Large networks in nature are always evolving in that nodes and links are continuously added to and/or deleted from the network. Networks are growing if, on average, the numbers of nodes and links increase with time. Most large networks are sparse, that is, the average number of links per node is much smaller than the total number of nodes in the network. Growing and complex networks may be classified according to whether there exists a hierarchy of organized structures. In particular, in a scale-free network, the number of links of various nodes follows a power-law (or algebraic) probability distribution, indicating that nodes in the network are organized into a hierarchy of connected clusters in terms of their numbers of links. In a random network [20], nodes are connected to each other in a completely random fashion and, as such, there is no organized hierarchy of structures in links. Regular networks possess only a few types of linking structures, in contrast to scale-free networks that have an infinite number of possibilities of linking. In this sense, the class of small-world networks studied by Watts and Strogatz [19] is constructed by randomly rewiring only a small fraction of links in a regular network and, hence, they are only a perturbed version of the "backbone" regular network.

Mathematically, a way to characterize a complex network is to examine the degree distribution $P(k)$, where k is the realization of a random variable K measuring the number of links at a node. Scale-free networks are characterized by

$$P(k) \sim k^{-\gamma}, \tag{1}$$

where $\gamma > 0$ is the algebraic scaling exponent. For random networks, the degree distributions are exponential,

$$P(k) \sim \exp(-ak), \tag{2}$$

where $a>0$ is a constant. The specific class of small-world networks proposed by Watts and Strogatz [19] also assumes the exponential distribution. It should be noticed that strictly scale-free networks are idealized. Realistic networks always contain both scale-free and random components. This "mixed" characteristic is the case for many networks in nature such as the scientific-collaboration network [18,21], the movie-actor network [22,23], and the conceptual network of languages [24].

Models of traffic flow on computer networks have been studied extensively [1–14]. In this context, the information processors are routers which have the same function as, say, workers in the postal service. Routers route the data packets to their destinations. In a computer network, a node may be a host or a router. A host can create packets with addresses of destination and receive packets from other hosts. A router finds, for each packet, the shortest path between the host and the destination and forward the packet along this path in each time step. Here, by "shortest" we mean the path with the smallest number of links. Previous studies focus on two different classes of computer network models. The first class treats all nodes as both hosts and routers [7,10–12], and for the second class [5,8,13,14], some nodes are hosts and others are routers. However, all existing models assume regular network topology, such as two-dimensional lattices [5,7,8,13] or Cayley trees [9–12]. In view of the recent evidence that the internet and many other realistic networks are complex to a significant extent [15,16,18], there is a need to investigate the dynamics of traffic flow on these networks.

In this paper, we construct two dynamical models, each with two parameters: the information creation rate λ and a control parameter β that measures the capacity of nodes to process information. In the first model, the capacity of packet delivery of each node is proportional to its degree, while in the second model, it is proportional to the number of shortest paths passing through the node (betweenness [21]). The quantity of interest is the critical rate λ_c of information generation (as measured by the number of packets created within the network in unit time) at which a phase transition occurs from free to congested traffic flow. In particular, for $\lambda<\lambda_c$, the numbers of created and delivered packets are balanced, resulting in a steady state, or free flow of traffic. For $\lambda>\lambda_c$, congestions occur in the sense that the number of accumulated packets increases with time, due to the fact that the capacities of nodes for delivering packets are limited. We are interested in determining the phase-transition point λ_c, given a network topology, in order to address which kind of network is more susceptible to phase transition and therefore traffic congestion. For this purpose, we study four kinds of networks: Cayley trees, regular, random, and scale-free networks. Our main result is that, in model I, λ_c is larger for networks that have a larger connectivity to betweenness ratio for the small set of nodes with the largest betweenness. Specifically, congestion is easier to occur in Cayley trees, then regular networks, then scale-free networks, and random networks are most tolerant to congestion. We give a theoretical argument to explain this phenomenon, based on identifying the existence of a subset of relatively heavily linked nodes in a network as the key. This is further supported by examining the effect of enhancing the capacities of these nodes to process information. From another standpoint, this result suggests a way to alleviate traffic congestions for scale-free networks: making heavily linked nodes [12] as powerful and efficient as possible for processing information. In the second model, we find that the congestion condition is independent of network size and topology and it thus represents a more useful protocol for alleviating traffic congestion on networks, especially for trees and regular networks.

One recent work that is particularly relevant to our study is the one addressing optimal network topologies for local search on networks [12]. This paper addressed the problem of searchability in complex networks with or without congestion. The focus was on optimal network configurations in terms of search cost, with the conclusion that there are only two classes of optimal networks: starlike or homogeneous-isotropic configurations, depending on the number of parallel searches. Our interest here is in the phase transition from free traffic to congestion and how it occurs with respect to the most representative types of complex networks found in realistic applications: regular, random, and scale-free networks. Despite the difference in the objective, the idea about the definition and analysis of congestion in Ref. [12] is very useful, which we have adopted here.

In Sec. II, we describe our traffic flow models. In Sec. III, we present a theoretical analysis for estimating the critical point for phase transition. Simulation results are given in Sec. IV and a discussion is offered in Sec. V.

II. TRAFFIC-FLOW MODELS

Our traffic-flow model is based on the routing algorithm in computer networks. To account for the network topology, we assume that the capacities for processing information are different for different nodes, depending on the numbers of links (model I) or the number of shortest paths (model II) passing through them. Our routing algorithm consists of the following steps.

(1) At each time step, the probability for node i to generate a packet is λ.

(2) At each time step, a node i delivers C_i packets one step toward their destinations, where $C_i=(1+\mathrm{int}[\beta k_i])$ in model I and $C_i=(1+\mathrm{int}[\beta B_i/N])$ in model II, $0<\beta<1$ is a control parameter, k_i is the degree of node i, and B_i is its betweenness. A packet, once reaching its destination, is removed from the traffic.

(3) Once a packet is created, it is placed at the end of the queue if this node already has several packets waiting to be delivered to their destinations. The existing packets may be created at some previous time steps or they are transmitted from other nodes. At the same time, a destination node, different from the original one, is chosen at random in the network. The router finds a shortest path between the node with the newly created packet and its destination and, the packet is forwarded along this path during the following time steps. If there are several shortest paths for one packet, the one is chosen whose next station (selected node) has the smallest

number of waiting packets or the shortest queueing length.

(4) At each time step, the first C_i packets at the top of the queue of node i, if it has more than C_i packets in its queue, are forwarded one step toward their destinations and placed at the end of the queues of the selected nodes. Otherwise, all packets in the queue are forwarded one step. This procedure applies to every node at the same time. As a result, the delivering time that a packet needs to reach its destination is related not only to the distance (number of time steps) between the source and the destination, but also to the number of existing packets along its path. Note that, here, the quantity C_i measures the forwarding capacity of node i.

Since N is the total number of nodes in the network, the total number of created packets at each time step is λN, and the total number of delivered packets at each time step is approximately $\Sigma_{i=1}^{N} C_i$ if every node has a sufficient number of packets, which is greater than the total number of created packets provided that $\lambda < 1$. Due to the network complexity, packets are more likely to be routed to the nodes with higher betweenness on their way to the final destinations. As a result, packets are more likely to be accumulated at these nodes, resulting in traffic congestion.

Qualitatively, the dynamics of traffic flow on a network is then as follows. For small values of the creation rate λ, the number of packets on the network is small so that every packet can be processed and delivered in time. Typically, after a short transient time, a steady state for the traffic flow is reached in which the instantaneous number $\langle n(t) \rangle$ of packets, averaged over all nodes in the network, fluctuates about a constant. That is, on average, the total numbers of packets created and delivered are equal, resulting in a free-flow state. This is in fact the well-known Little's law in queueing theory [25]. For larger values of λ, the number of packets created is more likely to exceed that which can be processed in time. In this case, $\langle n(t) \rangle$ grows in time and traffic congestion becomes possible. As λ is increased from zero, we thus expect to observe two phases: free flow for small λ and a congested phase for large λ, with a phase transition from the former to the latter at λ_c. To observe the phase transition and to determine λ_c, given a network structure, are main goals of this paper.

III. THEORETICAL ESTIMATION OF CRITICAL POINT

Here we give a heuristic theory for determining the phase-transition point λ_c, given a particular network structure. Because the node with the largest betweenness can be easily congested and the congestion can quickly spread to the entire network, it is necessary to consider only the traffic balance of this node. Since the packets are transmitted along the shortest paths from the source to the destination, the probability that a created packet will pass through the node with the largest betweenness i is $B_i / \Sigma_{j=1}^{N} B_j$. At each time step, on average, λ packets are generated. Thus, the average number of packets that the node with the largest betweenness receives at each time step is

$$Q_{in} = \lambda N D \frac{B_{L_{max}}}{\sum_{j=1} B_j}, \tag{3}$$

where D is the average shortest path length of the network and L_{max} is the index of the node with the largest betweenness. On the other hand, the total number of packets that the node with the largest betweenness can deliver at each time step is

$$Q_{out} = C_{L_{max}}. \tag{4}$$

Congestion occurs when the number of incoming packets is equal to or larger than the outgoing packets at the node with the largest betweenness, i.e.,

$$Q_{in} \geq Q_{out}. \tag{5}$$

Then,

$$\lambda_c N D \frac{B_{L_{max}}}{\sum_{j=1} B_j} = C_{L_{max}}. \tag{6}$$

Since $\Sigma_{j=1}^{N} B_j = N(N-1)D$, Eq. (6) can be simplified to

$$\lambda_c = \frac{C_{L_{max}}(N-1)}{B_{L_{max}}}. \tag{7}$$

Equation (7) can be applied to general networks, which is the same result as in Ref. [12].

For model I, Eq. (7) turns out to be

$$\lambda_c = \frac{(1 + \mathrm{int}[\beta k_{L_{max}}])(N-1)}{B_{L_{max}}}. \tag{8}$$

To gain insight, we consider two special cases, regular networks and Cayley trees. First, for regular networks, all nodes have the same structure and the same number of links. We thus have

$$B_{L_{max}} = (N-1)D. \tag{9}$$

The congestion condition can then be estimated by the following equation:

$$\lambda_c \approx \frac{1 + \beta k}{D}. \tag{10}$$

For regular networks $D = N/2k$, we have

$$\lambda_{c,reg} \approx \frac{2k(1 + \beta k)}{N}. \tag{11}$$

Now consider a special kind of regular network, square lattices with periodic boundary condition. For such a lattice with $L \times L$ nodes, $D = L/2$, we have

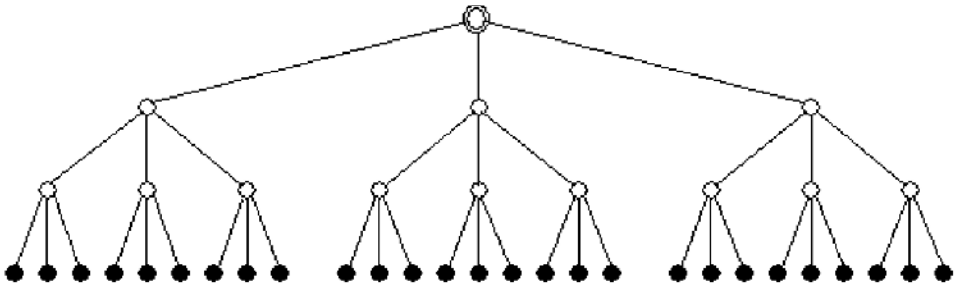


FIG. 1. Illustration of a Cayley tree with the branching factor $z=3$ and the level of the leaves $l=3$. The root is on level 0 and the depth of the tree is thus $l+1=4$. Solid circles, circles, and double circle represent leaves, intermediate nodes, and root, respectively.

$$\lambda_{c,lattice} \approx \frac{2(1+\beta k)}{L}. \tag{12}$$

If $\beta=0$, Eq. (12) recovers the estimate of Ref. [13] and the estimate from the mean field model of Fũks and Lawniczak [7].

A schematic illustration of a Cayley-tree network is shown in Fig. 1. Since the root has the highest betweenness and a small number of links, it is easy for congestion to occur at the root, which has a major impact on the whole tree. For these reasons, λ_c can be conveniently estimated by only considering the traffic flow through the root.

The total number of nodes in the tree is

$$N = z^0 + z^1 + z^2 + \cdots + z^l = \frac{z^{l+1}-1}{z-1}. \tag{13}$$

The betweenness B_r of the root can be calculated by counting the routes from any node in the tree to different first-order subtrees, which must pass through the root. We obtain

$$B_r = \frac{1}{2}\frac{(N-1)}{z}\frac{(z-1)(N-1)}{z}z \tag{14}$$

$$= \frac{z(z^l-1)^2}{2(z-1)}. \tag{15}$$

In Eq. (14), the factor $(N-1)/z$ is the number of nodes in one chosen first-order subtree and the factor $(N-1)(z-1)/z$ is the number of nodes in all other $z-1$ first-order subtrees. The number of shortest paths from the chosen first-order subtree to any other first-order subtree is the multiplication of these two factors. The factor z means that we have precisely z ways to choose a first-order subtree. The factor $1/2$ is included because each shortest path has been counted twice.

Since the number of links of the root is z, the number of packets that the root can deliver per unit time is

$$Q_{out} - C_r \approx 1 + \beta z. \tag{16}$$

Putting B_r and C_r in Eq. (7), $\lambda_{c,Cayley}$ is estimated to be

$$\lambda_{c,Cayley} \approx \frac{2(1+\beta z)}{z^l - 1}. \tag{17}$$

For model II, the delivery capacity of each node is proportional to its betweenness, i.e., $C_i = 1 + \mathrm{int}[\beta B_i/N]$. In this case, the critical generating rate for general networks becomes

$$\lambda_c = \frac{(1+\mathrm{int}[\beta B_{L_{max}}/N])(N-1)}{B_{L_{max}}} \tag{18}$$

$$\approx \frac{(N-1)}{B_{L_{max}}} + \beta \tag{19}$$

$$\approx \beta, \tag{20}$$

where, because $B_{L_{max}} \gg N$ in all networks considered in this work, the second term in Eq. (19) dominates. Equation (20) shows that the critical generating rates are roughly independent of the network size and topology.

By comparing Eq. (20) to Eqs. (11) and (17), we see that, although model II makes no significant improvement on random and scale-free networks, it can increase λ_c for regular networks and Cayley trees. A practical significance is that protocols designed based on model II can generally be robust against traffic congestion, regardless of the network topology.

IV. SIMULATION RESULTS

The primary goal of our simulation is to understand the behavior of the phase transition, which leads to traffic congestion, with respect to the network topology. Thus we focus on examining the value of the critical point λ_c for Cayley trees, regular, random, and scale-free networks. Another goal is to explore the effect of adjusting the capacity parameter β. In particular, we are interested in the possibility of increasing the capacities of a small subset of nodes with higher betweenness to improve the network's tolerance to traffic congestions. In order to characterize the transition, we use the order parameter introduced in Ref. [9]:

$$\eta = \lim_{t\to\infty} \frac{\langle \Delta\Theta \rangle}{\lambda \Delta t}, \tag{21}$$

where $\Delta\Theta = \Theta(t+\Delta t) - \Theta(t)$, $\Theta(t)$ is the total number of packets in the network at time t, and $\langle \cdots \rangle$ indicates the average over time windows of Δt. When $\lambda < \lambda_c$, the network is in the free-flow state; then $\Delta\Theta \approx 0$ and $\eta \approx 0$. For $\lambda > \lambda_c$, $\Delta\Theta$ increases with Δt.

In our simulations, the networks are generated as follows. For all kinds of networks, each node points to a linked list, which contains its nearest neighbors. For a Cayley tree with depth $l+1$ and branching factor z, there are $N = \mathrm{int}[(z^{l+1}-1)/(z-1)]$ nodes, which are labeled as

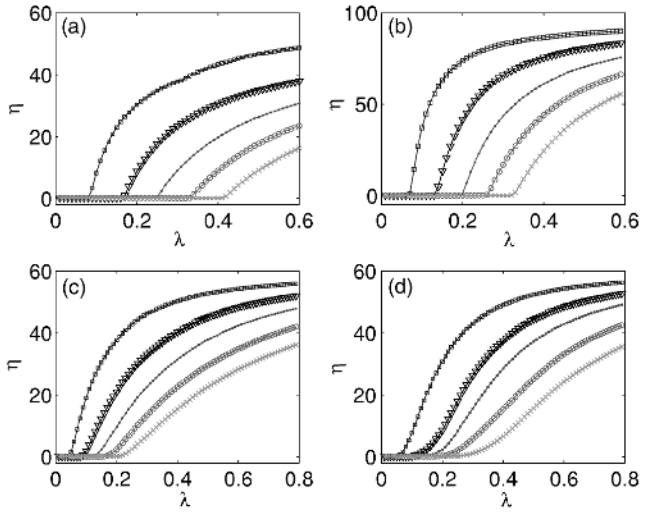


FIG. 2. (Color online) For model I, the order parameter η versus the packet-generating rate λ for the following. (a) Cayley tree, $z=3$, $l=6$, $\langle k\rangle=2$, and thus $N=1093$. Square, triangle, dot, circle, and cross curves correspond to the simulations of $\beta=10,20,30,40,50$, respectively. (b) Regular network, $N=1000$, $\langle k\rangle=4$, square, triangle, dot, circle, and cross curves correspond to the simulations of $\beta=2,4,6,8,10$, respectively (c) Scale-free network and (d) random network, $N=1000$, $\langle k\rangle=4$, where square, triangle, dot, circle, and cross curves correspond to the simulations of $\beta=0.1,0.2,0.3,0.4,0.5$, respectively. In all simulations, 50 realizations are averaged.

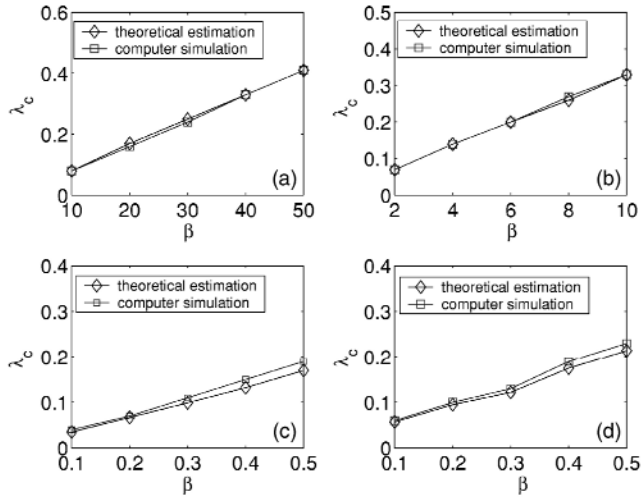


FIG. 3. (Color online) For model I, comparison of theoretical prediction (diamond curves) and simulation (square curves) of λ_c values for (a) Cayley trees, $z=3$, $l=6$, $\langle k\rangle=2$; for (b) regular networks, (c) scale-free networks, and (d) random networks, $N=1000$, $\langle k\rangle$ 4.

$\{0,1,2,\ldots,N-1\}$. Thus the list pointed to by node i at level $j\in\{0,1,\ldots,l-1\}$ contains the following node labels as children: $\{i\times Z+1, i\times Z+2,\ldots,i\times Z+Z\}$. At the same time, node i is inserted in the lists pointed to by each of its children. This process begins from the root with node label 0 and ends at the last node at level $l-1$. To generate a regular network with degree k and the label set $\{1,2,\ldots,N\}$, each node i points to a list containing the node labels $\{i-k/2,\ldots,i-2,i-1,i+1,i+2,\ldots,i+k/2\}$. However, if $i+j>N$, the label is replaced by $i+j-N$, and if $i-j<1$, it is replaced by $i-j+N$. Scale-free and random networks are generated by using the general network model proposed in Ref. [26].

First, we present simulation results with model I. Figure 2 shows the order parameter η versus λ for different capacity parameters β for the (a) Cayley tree, (b) regular network, (c) scale-free network, and (d) random network. We see that, for all cases considered, η is approximately zero when λ is small; it suddenly increases when λ is larger than a critical value λ_c. We also observe that λ_c increases with β, which means that enhanced capacity for processing packets can help alleviate possible congestions that are most likely to occur at the heavily linked nodes. As a result, phase transition can be delayed in the sense that the network can be more tolerant to traffic congestions for larger values of λ. Figure 2 also indicates that, in order to get the same order of λ_c, a large value of the capacity parameter β is required for Cayley trees and regular networks; however, small β is needed for random and scale-free networks. This means that Cayley trees and regular networks are significantly more susceptible to traffic congestion. This is because, in Cayley trees and regular networks, the most congested nodes have large betweenness, but very small number of links, i.e., the ratio $k_{L_{max}}/B_{L_{max}}$ is much smaller than those in random and scale-free networks. Equation (8) then indicates that the λ_c for Cayley trees and regular networks is much smaller than that for random and scale-free networks.

Figure 3 shows the critical generating rate λ_c from theoretical predictions and from simulations. The theoretical results are obtained by Eqs. (17), (11), and (8) for Cayley trees, regular, random, and scale-free networks, respectively. In all cases, a good agreement is observed. From Fig. 2, we see that the critical packet generation rates λ_c of scale-free and random networks are of the same order. However, a direct comparison of simulation results (Fig. 4) shows that λ_c for random networks is actually larger than that for scale-free networks. As mentioned, scale-free networks are heterogeneous in links, which causes heterogeneity in betweenness.

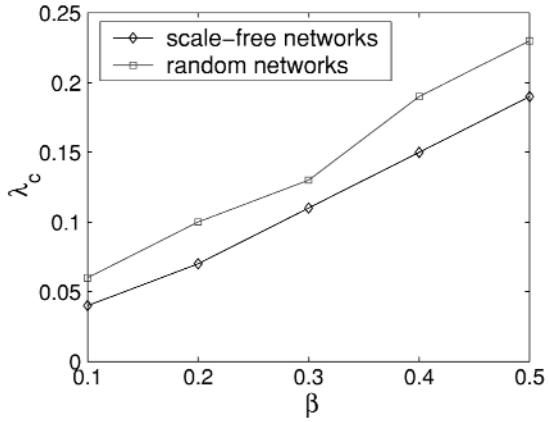


FIG. 4. (Color online) For model I with $N=1000$, $\langle k\rangle=4$, simulation results of λ_c versus β for scale-free and random networks.

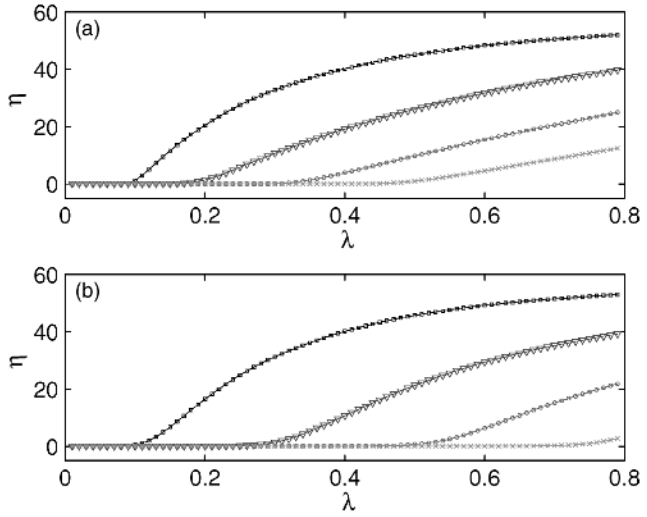


FIG. 5. (Color online) For model I with $N=1000$, $\beta=0.2$, order parameter η versus the packet-generating rate λ for (a) scale-free networks, (b) random networks. Square, triangle, circle, and cross curves correspond to the simulations of $\langle k\rangle=4,6,8,10$, respectively. In all simulations, 50 realizations are averaged.

This means that there is a small group of nodes which have large betweenness but majority of nodes in the network have small betweenness. Thus, most generated packets will have a high probability to pass through this small number of high betweenness nodes, making them vulnerable to congestion. Qualitatively, we may think that the packet transmission routes are relatively better distributed for random networks than for scale-free networks.

How does the congestion condition change with the network's average degree $\langle k\rangle$? Figure 5 shows that in both scale-free and random networks, λ_c increases as the average degree increases. This is because increasing the average degree makes nodes in the networks more connected and hence the shortest paths are less dependent on the heavily linked nodes. Consequently, congestion on the heavily linked nodes can be delayed. As mentioned, betweenness homogeneity is an important factor for traffic congestion. In order to characterize this feature, we calculate the standard deviation of betweenness defined as

$$\delta_B=\frac{1}{N}\sqrt{\sum_{i=1}^{N}(B_i-\langle B\rangle)^2}, \tag{22}$$

where $\langle B\rangle$ is the average betweenness of the network in consideration. Figure 6 shows the decreasing of the standard deviation of betweenness for both the scale-free and random networks as the average degree increases, indicating that the distribution of betweenness is more homogeneous with increasing $\langle k\rangle$. Thus, packet loads of the nodes with the largest betweenness are reduced and congestion triggered by these nodes is delayed. From the same figure, we see that, except for $\langle k\rangle=2$, the betweenness deviation in random networks is smaller than that in scale-free networks. That is, the betweenness distribution in random networks is in general more homogeneous. This is another supporting factor for the

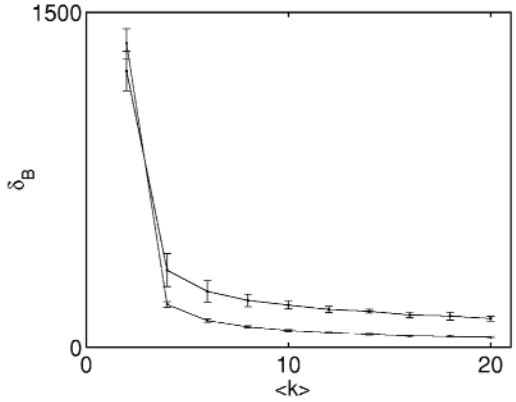


FIG. 6. (Color online) For $N=1000$ and 50 realizations, the standard deviation of betweenness δ_B versus the average degree $\langle k\rangle$ for scale-free (upper trace) and random (lower trace) networks.

explanation as to why random networks are more tolerant to congestion than scale-free networks.

We now present simulation results with model II. Here, the delivery capacity of each node is proportional to its betweenness, i.e., $C_i=1+\text{int}[B_i/N]$. Figure 7 shows the order parameter η versus λ for different capacity parameters β for (a) Cayley tree, (b) regular, (c) scale-free, and (d) random network. We see that values of λ_c are roughly the same for all kinds of networks considered here, confirming our prediction by Eq. (20).

Figure 8 shows the critical generating rate λ_c from theoretical predictions and simulations for the four kinds of networks. In all cases, good agreement is observed. These simu-

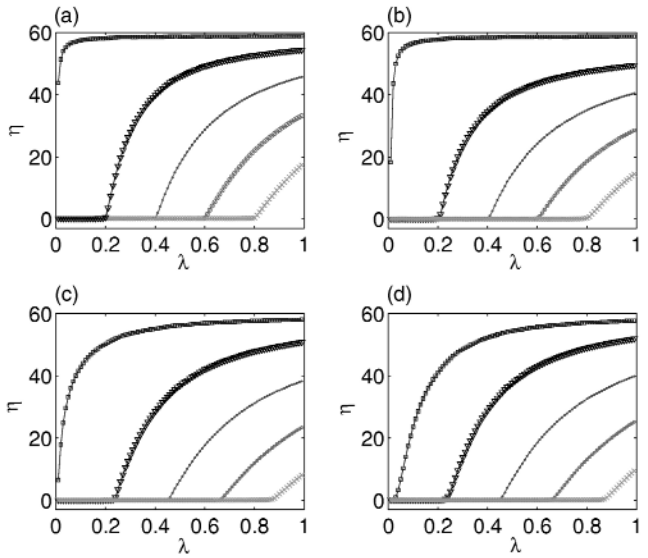


FIG. 7. (Color online) For model II, order parameter η versus the packet-generating rate λ for (a) Cayley tree, $z=3$, $l=6$, $\langle k\rangle=2$, thus $N=1093$; for (b) regular network, (c) scale-free network, and (d) random networks, $N=1000$, $\langle k\rangle=4$. In all of the four cases, square, triangle, dot, circle, and cross curves correspond to the simulations of $\beta=0.0,0.2,0.4,0.6,0.8$, respectively. The capacity of delivery of each node is proportional to its betweenness. In all simulations, 50 realizations are averaged.

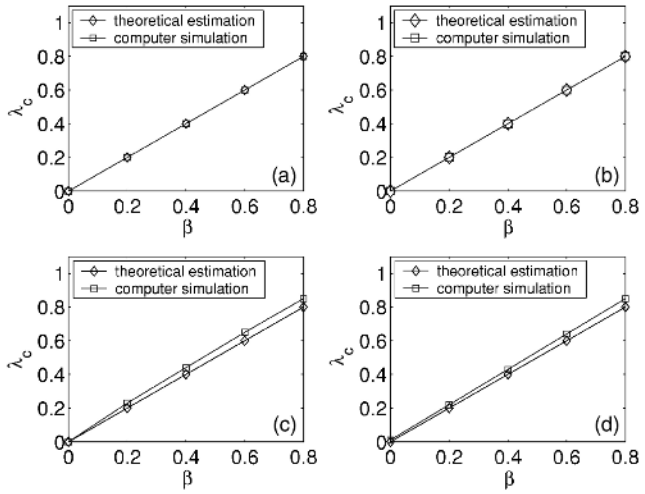


FIG. 8. (Color online) For model II, comparison of theoretical prediction (diamond curves) and computer simulation (square curves) of λ_c versus the capacity parameter β for (a) Cayley trees, $z=3$, $l=6$, $\langle k\rangle=2$; for (b) regular, (c) scale-free, and (d) random networks, $N=1000$, $\langle k\rangle=4$.

lation results thus show that protocols based on our model II are more tolerant to congestion for all kinds of networks studied here, especially for Cayley trees and regular networks.

V. DISCUSSION

We live in a modern world supported by large, complex networks. Examples range from financial markets to internet, communication, and transportation systems. Recently there has been a tremendous effort to study the general structure of these networks [16–18]. Universal features such as the small-world [19] and scale-free [15] properties, which can be characterized at a quantitative level, have been discovered in almost all realistic networks. The discoveries suggest that, to understand the dynamics on complex networks, their structures have to be taken into account.

This paper addresses the dynamics of traffic flow on complex networks. Our motivation comes from the desire to understand the influence of topological structure on the traffic dynamics on a network, as existing works in this direction often assume regularity and homogeneity for the underlying network [1–14]. We consider general network structures to couple with simple traffic-flow models determined by the rate of information generation and a parameter to describe the average capacity of nodes to process information. Our study indicates that phase transition can generally occur in the sense that free traffic flow can be guaranteed for low rates of information generation but large rates above a critical value can result in traffic congestions. Our models enable the critical value for the phase transition to be estimated theoretically and computed, given a particular network topology. We present computational results and analysis, which indicate that, in case the delivery capacity of each node is proportional to its degree, the critical value is smaller for networks of smaller ratio of degree to betweenness for the set of most easily congested nodes (the set of nodes with the largest betweenness). In this case, random and scale-free networks are more tolerant to congestion than trees and regular networks. This is further supported by examining the effect of enhancing the capacities of these nodes to process information on the global traffic flow. These results suggest a way to alleviate traffic congestions for protocol based on model I for networks with a significant heterogeneous component: making nodes with large betweenness as powerful and efficient as possible for processing and transmitting information. For protocol based on model II, the capacity of delivery of each node is proportional to its betweenness. In this case, the critical value λ_c is independent of the network topology. Compared with model I, while model II can improve a little the performance for scale-free and random networks, it can improve significantly the performance for trees and regular networks against congestion.

ACKNOWLEDGMENTS

Z. Liu provided assistance in the initial phase of this project. The work was supported by AFOSR under Grant No. F49620-01-1-0317 and by NSF under Grant No. ITR-0312131.

[1] H. Li and M. Maresca, IEEE Trans. Comput. **38**, 1345 (1989).
[2] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, Comput. Commun. Rev. **23**, 183 (1993).
[3] M. S. Taqqu, W. Willinger, and R. Sherman, Comput. Commun. Rev. **27**, 5 (1997).
[4] M. E. Crovella and A. Bestavros, IEEE/ACM Trans. Netw. **5**, 835 (1997).
[5] T. Ohira and R. Sawatari, Phys. Rev. E **58**, 193 (1998).
[6] M. Faloutsos, P. Faloutsos, and C. Faloutsos, Comput. Commun. Rev. **29**, 251 (1999).
[7] H. Fŭks and A. T. Lawniczak, Math. Comput. Simul. **51**, 101 (1999).
[8] R. V. Solé and S. Valverde, Physica A **289**, 595 (2001).
[9] A. Arenas, A. Díaz-Guilera, and R. Guimerà, Phys. Rev. Lett. **86**, 3196 (2001).
[10] R. Guimerà, A. Arenas, and A. Díaz-Guilera, Physica A **299**, 247 (2001).
[11] R. Guimerà, A. Arenas, A. Díaz-Guilera, and F. Giralt, Phys. Rev. E **66**, 026704 (2002).
[12] R. Guimerà, A. Díaz-Guilera, F. Vega-Redondo, A. Cabrales, and A. Arenas, Phys. Rev. Lett. **89**, 248701 (2002).
[13] M. Woolf, D. K. Arrowsmith, R. J. Mondragón-C, and J. M. Pitts, Phys. Rev. E **66**, 046106 (2002).
[14] S. Valverde and R. V. Solé, Physica A **312**, 636 (2002).
[15] A.-L. Barabási and R. Albert, Science **286**, 509 (1999); A.-L. Barabási, R. Albert, and H. Jeong, Physica A **272**, 173 (1999);

281, 69 (2000).
[16] R. Albert and A.-L. Barabási, Rev. Mod. Phys. **74**, 47 (2002).
[17] S. H. Strogatz, Nature (London) **410**, 268 (2001).
[18] M. E. J. Newman, SIAM Rev. **45**, 167 (2003).
[19] D. J. Watts and S. H. Strogatz, Nature (London) **393**, 440 (1998).
[20] P. Erdös and A. Rényi, Publ. Math., Inst. Hungarian Acad. Sci. **5**, 17 (1960); B. Bollobaás, *Random Graphs* (Academic, London, 1985).
[21] M. E. J. Newman, Phys. Rev. E **64**, 016131 (2001).
[22] R. Albert and A.-L. Barabási, Phys. Rev. Lett. **85**, 5234 (2000).
[23] J.-W. Kim, B. Hunt, and E. Ott, Phys. Rev. E **66**, 046115 (2002).
[24] A. E. Motter, A. P. S. de Moura, Y.-C. Lai, and P. Dasgupta, Phys. Rev. E **65**, 065102(R) (2002).
[25] See, for example, O. Allen, *Probability, Statistics and Queueing Theory with Computer Science Application*, 2nd ed. (Academic, New York, 1990).
[26] Z. Liu, Y.-C. Lai, N. Ye, and P. Dasgupta, Phys. Lett. A **303**, 337 (2002).

Anomaly and Misuse Detection in Network Traffic Streams — Checking and Machine Learning Approaches

Sampath Kannan*, Insup Lee Wenke Lee Oleg Sokolsky Diana Spears
William Spears

December 7, 2005

Abstract

The prevalence of security holes in programs and protocols, the increasing size and complexity of the Internet, and the sensitivity of the information stored throughout have made the problem of network security of paramount importance and attracted newspaper headlines[17]. Viruses and worms can spread rapidly wreaking havoc on the hosts they infect. Other types of *malware* (the generic name we use to describe any kind of program that engages in unauthorised behavior) can consume network resources, acquire disallowed privileges, and breach the confidentiality and integrity of sensitive data. The diversity of possible attacks entails a variety of techniques to counter them. One the most important defenses is a network intrusion detection system (NIDS). Such a system is deployed at routers or other intermediate points on the network to monitor traffic and detect attacks or anomalies in traffic patterns and to activate alarms when they are detected. Host-based defenses monitor and/or patch vulnerabilities that can be exploited by malware, perform formal analyses of software to detect potential problems, and observe the behavior of running software to raise alarms if warranted. In this paper we explore a variety of approaches to improve defenses against malware, including both network-based and host-based approaches.

Keywords: Network Intrusion Detection, Host-based Intrusion Detection, Anomaly Detection, Malware, Botnets, Case-based reasoning.

Categories: Network security, Intrusion Detection Systems, Run-time monitoring, Streaming algorithms, Semantic analogy-based reasoning.

1 Preliminaries

To make networks secure against such attacks we need a combination of techniques such as firewalls, strongly typed languages, secure protocols such as IP-sec and https, intrusion and anomaly detection systems, and decoys and honey pots. Furthermore, each of these protection mechanisms has to be continually adapted to combat new and increasingly sophisticated attacks.

At the network level, defense mechanisms try to detect *intrusions* and *anomalies*, terms that are defined below. "Intrusion" refers to a hacker — who may be a legitimate user or an outsider — getting privileges s/he is not entitled to, and using them to steal confidential data, misuse system resources, deface webpages, corrupt files, set up programs with backdoors for future access, or attack other systems. Intrusions usually follow a "script" or scenario, wherein the hacker first probes a system for vulnerabilities such as web pages with CGI scripts, programs with buffer overflow problems, etc. Buffer overflow occurs when a program stores a data item that is too big to fit within the space allocated for it. Without adequate protection, the

*Point of Contact: Sampath Kannan, email: kannan@cis.upenn.edu, tel: (215) 898-9514, fax: (215) 898-0587

data item simply overflows into adjacent space which may contain information that is critical to the correct behavior of a program, such as the address to which the program should jump to execute the next instruction. Malware can exploit this vulnerability by providing such oversized data to important programs in the system, and using the resulting buffer overflow to hijack the execution of the program to locations containing part of the malware code. They might also ping all machines to detect which machines are up and running, do port scans to find port numbers for various protocols, and check for the services offered by a machine. This exploration might reveal holes which they then exploit to gain access to the system, after which they try to gain further privileges. Because of this pattern of behavior, there is a "signature" or sequence of events associated with each type of intrusion and databases of such signatures are used to detect intrusions.

In anomaly detection, we analyze statistical information about network traffic and raise alarms when we see "aberrant" statistics. This definition is general and can include a variety of different tests and techniques.

Both intrusion detection and anomaly detection can be performed at routers and other intermediate points on the network (systems that do this are called Network Intrusion Detection Systems (NIDS)) or at hosts (such systems are called Host-Based Intrusion Detection Systems).

2 Research Summary

In order to produce systematic and comprehensive Network and Host-Based Intrusion Systems that are capable of detecting and adapting to new attacks as they occur major improvements are needed in the algorithmic and systems infrastructure for processing traffic. While we have been working on this larger question, we are simultaneously tackling specific problems that will be important "plug-ins" into this infrastructure. Listed below are our solutions to two important problems that arise in Network Intrusion Detection.

- We learn new signatures based on known attacks and apply them to the traffic. This will greatly reduce the false negative rate that results if we simply check for intrusions against a static database.

- We detect attacks by "botnets" (which are networks of automated and coordinated attackers) by extracting anomalies in traffic patterns.

Viruses and worms are specific types of malware that have gained notoreity in recent years. These programs are spread by a variety of means – email attachments, IRC protocols, software downloads, etc. While attempts have been made to detect viruses and worms at the network level by observing anomalous traffic patterns[18, 10], and by learning signatures[14], the more sophisticated and novel attacks will need a host-based detection component as well. While fundamentally novel viruses are difficult to create, a common approach used by hackers is to mutate the code for a virus so as to retain its functionality while evading detection. Thus, one can group viruses into families with common origins which are referred to as polymorphic families of viruses. Further complicating the situation is the possibility that the virus code exists in encrypted form and is only decrypted at the time of execution.

- We define and compute measures for the similarity of two programs viewed as control-flow graphs. We have initiated an experimental study of such an approach for detecting polymorphic families of viruses.

In addition to working on the specific problems above, we seek to improve the infrastructure for intrusion and anomaly detection. The Monitoring and Checking (MaC) framework is a framework for run-time monitoring of software systems. We have designed and implemented a prototype of this framework called JavaMaC that monitors running Java programs. In JavaMaC, the user provides information associating low-level program variables with more abstract events. The user also specifies properties to be satisfied by the running system using a formal framework such as temporal logic or automata. The MaC system instruments the Java program to extract information about the low-level variables of interest at run-time, feeds the extracted information to an event recognizer that detects the occurrence of abstract events and then sends this

sequence of abstract events to an automaton that detects violations of the desired properties. This infrastructure is natural and convenient to use in the context of intrusion and anomaly detection. The instrumentation here will extract relevant features from network traffic. The choice of the features themselves is an interesting research question. Each anomaly or intrusion detection module can then use this stream of extracted features to determine if events of interest to that module have occurred. The sequence of events is then used by the module to raise an alarm when an intrusion or anomaly is detected. Putting a number of such modules on the same platform also allows us to combine the inferences made by the individual modules to reduce false positive and false negative rates. Using statistical techniques to find the best way to combine the output of modules is another important research question.

Another infrastructure improvement necessary for intrusion detection systems is in the area of streaming algorithms. Such algorithms process massive amounts of data generated at a rapid rate. They are incapable of storing any more than a negligible amount of the observed data but must nevertheless compute various functions on the data stream. Perforce these computations will be approximate for all but the most trivial functions. Such a data stream model applies at the routers in a network. To implement signature-based intrusion detection systems at a high volume router at the backbone of a network requires being able to compute sophisticated functions on the traffic stream. This problem is difficult enough that current systems deal with very limited signatures, usually matched with one packet in the traffic stream. To deal with signatures that span multiple packets one needs algorithms that can use memory very efficiently without having to remember state for each of the currently active traffic flows. To deal with problems of this nature, basic research is needed on algorithmic questions that can be solved in the streaming model.

3 A Closer Look at Three Results

3.1 Detecting Botnets

Increasingly, attackers are using the resources of their victims. Whereas previous generations of malware merely caused harm to individual computers, attackers are now pooling their victims into large networks, and using these for malicious purposes. The resulting victim clouds constitute a more significant, growing threat on the Internet. Common examples include botnets (which perform arbitrary tasks), proxynets (which anonymize attacker communications), and zombie distributed denial-of-service (DDoS) armies (used to consume network resources).

Unlike individual infections, victim networks are difficult to effectively remediate. Further, the networks of infected individuals facilitate attacks (such as DDoS, distributed network scanning) that are more effective using large numbers of victims. More ominously, a large pool of victims creates an effective malware launching platform, so that new vulnerabilities are exploited instantaneously by numerous attackers [21]. This significantly shortens the response time and patch window that network administrators need to perform basic maintenance.

3.1.1 Motivation and Goals

Attacking networks (e.g., botnets) constitute a growing, urgent problem for information security researchers. Botmasters are creating complex, large, and mature networks of victims. Warning signs include the following:

- Botnet Complexity We are witnessing a growing list of victim networks structured with bewildering complexity. Some recent examples include: distributed phishing sites [8], which rotate the fake web page among victims in the cloud, distributed spam armies [16], which make response (e.g., patching, blacklisting) more difficult than stopping a single machine, and distributed denial of service zombies [12].

- Botnet Size The victim networks are not only complex, but also large. The scale of the botnet problem is simply staggering. Botnets can easily approach 50,000 victim members [20]. Our large-scale project at Georgia Institute of Technology [3] has captured botnets with 350,000 members. One reliable estimate is that some 170,000 new victim infections occur per day [2]. The research effort in [3] has cataloged nearly 200 individual major bot versions (or families), each with dozens of different variations and minor version.

- Botnet Maturity Botnets have undergone years of development and use, and are currently quite robust. Whereas previous botnets used IRC networks to coordinate communication between victims and the botmaster [5], current generations of botnets use non-standard protocols, or customized versions of IRC servers running on victim computers.

An important problem is the detection of the attacking network. Existing intrusion detection techniques can identify discrete "side-effects" of a botnet (e.g., detecting a scan or phishing attack), but fail to identify the *entire network* of victims performing the attacks.

Thus, in order to handle botnet attacks we must:

1. Identify the command-and-control (C&C) traffic. Without coordination, botnets are merely clusters of discrete, unorganized infections, for which there are existing orthogonal solutions. If we can identify the command-and-control traffic, we can disrupt the botnet, anticipate attacks, and perform traceBacks.

2. Identify the victims in a botnet. If all of the victims in an attacking network are enumerated, upstream providers can block their traffic, and network owners can be notified to start patching. Since most victims have multiple infections (and participate in many botnets) identifying victim clouds lets one anticipate the propagation potential of similar malware.

3.1.2 Approaches and Results

A solution to the problem posed by botnets begins with an understanding of the command and control infrastructure of botnets. This is used to design corresponding detection and response capabilities. More specifically, we define a taxonomy of botnets based on how they perform command and control over victim hosts. We have implemented and tested a next-generated technique for detecting and responding to botnets, which is described below.

Detection

We have developed several *anomaly detection* approaches to identify botnets. The main idea is to look for *abnormal* traffic behaviors caused by the command and control activities of the botnets. In this research, we focus on a class of botnets that use Dynamic DNS (DDNS) service.

The general pattern of botnet creation is detailed in Fig. 1. To start, a malware author, VX in the diagram, will purchase one or several domain names (perhaps using stolen accounts). The newly purchased domains are initially "parked" at 0.0.0.0, reserved for unknown addresses. The malware author then hard-codes the string names of their domains into a dropper, and spreads the binary, as shown in steps 1 and 2 of Fig. 1.

While the virus spreads, the malware author also creates a C&C "rallying" box for the victims. This is typically one of two types: a high-bandwidth compromised machine, or (more frequently) a high-capacity co-located box (perhaps rented with stolen funds). The C&C box is set up to run an IRC service (often a modified version), to provide a medium for the bots to communicate.

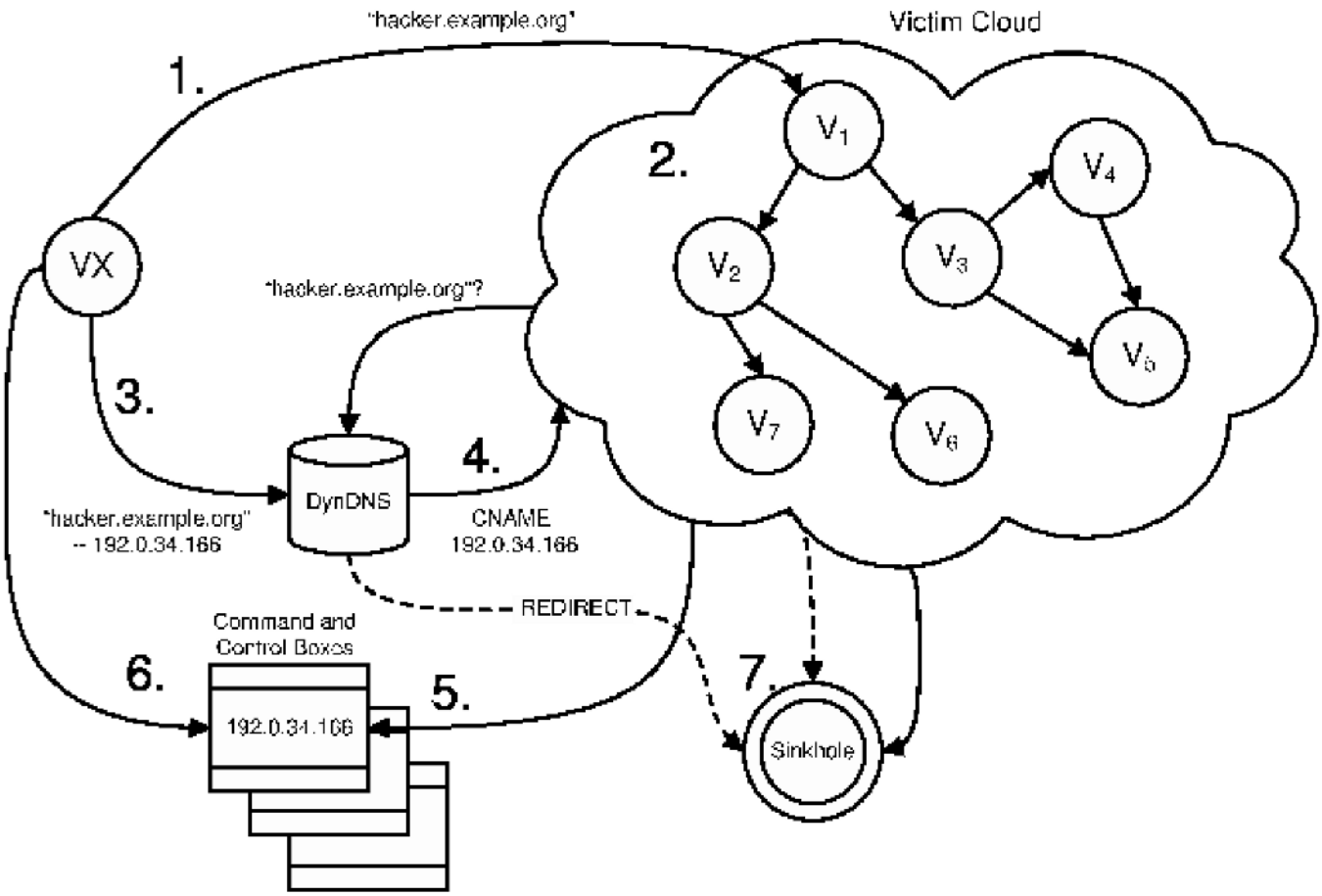


Figure 1: General spread of a botnet.

(1) A malware author writes a virus, hardcoding an address (here, "hacker.example.org") that victims should contact after infection (called "command and control" or C&C). (2) The virus spreads, reaching unknown victims. (3) The malware author purchases an inexpensive DDNS service to link the string name to the C&C box's IP address, often another compromised machine. (4) The bots lookup the IP of the C&C box from the DDNS server, and (5) rally at the C&C box. Eventually, the DDNS provider detects the abuse and revokes the account, redirecting all traffic to (6) the KarstNet sinkhole. In some fortuitous cases, prompt reverse engineering of binary samples (obtained from honeypots) let us sinkhole botnets even before they become active.

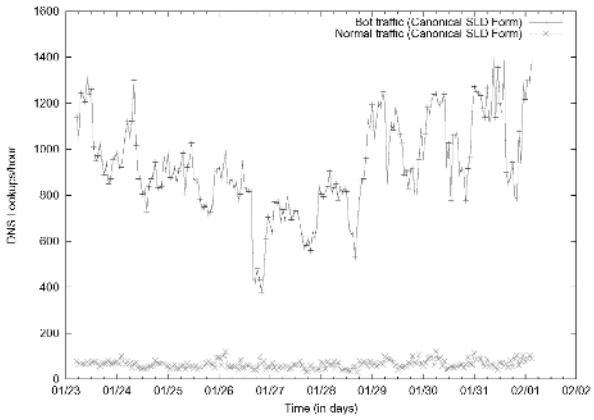


Figure 2: Comparison of Canonical DNS Request Rates

Finally, the malware author will also arrange for DNS resolution of the domain names, and register with a DDNS service, as shown at step 3 in Fig. 1. The IP address they provide is for the C&C box. As DNS propagates, more victims join the network, and within a day, the bot army swells. Overnight, the bot army can reach into the thousands, even for a novice Visual Basic virus.

There are two properties of botnet DNS behavior that help distinguish their traffic: use of subdomains, and exponential request rates.

We can classify DNS requests as either second-level domain (SLD) requests, such a "example.com", or third-level subdomain requests ($3LD$), such as "foo.example.com". From extensive empirical observations, normal users tend to have a single domain name, while bots tend to use mostly subdomains [15]. This fact helps us design a simple detection system. We define the *canonical SLD request rate* as the total number of requests observed for all the $3LD$s present in a SLD, plus any requests to the SLD. We use the term $|SLD|$ to represent the number of $3LD$s observed in a given SLD. (Thus, if the SLD "example.com" has two subdomains, then its $|SLD| = 2$.) For a given SLD_i, with rate R_{SLD_i} we calculate its canonical rate C_{SLD_i} as:

$$C_{SLD_i} = R_{SLD_i} + \sum_{j=1}^{|SLD_i|} R_{3LD_j}$$

We ran a modified version of `dnstop` on a DDNS provider's busy network for a week, and sampled approximately 1.28 million DNS requests. We filtered out all bot traffic (by hand, with the help of the DDNS provider), and calculated a mean lookup rate for normal traffic. Fig. 2 shows the average lookup rate for normal hosts, in requests per hour.

When put in canonical form, distinguishing the normal and bot traffic is straight forward. We simply set an expected mean for the rate of normal traffic, $E(X) = \mu$. We then use Chebyshev's inequality, Eq.(1), to fix an appropriate threshold for the normal request rates and request anomalous (i.e., bot) lookups.

$$P(|X - \mu| \geq t) \leq \frac{\sigma^2}{l} \tag{1}$$

The above simple threshold based approach using canonical SLD scores is very useful, and created no false positives in our test data set. We can imagine situations, however, where the botmasters could attempt to evade such detection. We developed a secondary detection filter based on *sorted request rate densities*. This second detection layer is also useful for noisy networks where short-term normal and bot DNS rates may be very similar. To reduce the chance of false positives, the second filter can be used to examine just the hosts who have excessive canonical SLD scores.

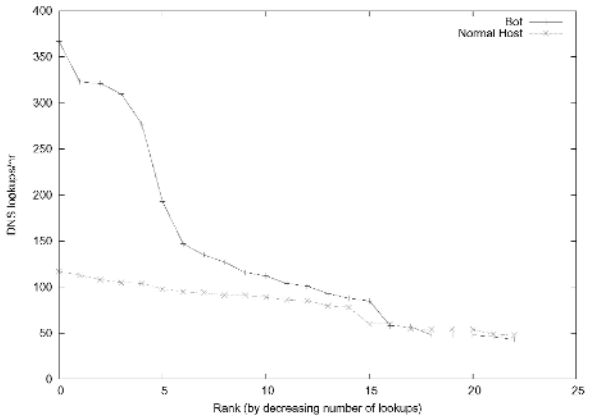


Figure 3: Comparison of Sorted DNS Rates

A key distinguishing feature for this second filter is that botnet DNS requests rates are usually exponential over a 24 hour period. The diurnal nature of bot behavior means that there are periodic spikes in bot requests. These spikes are caused by infected hosts who turn on their computers in the morning, releasing a sudden burst of DNS traffic as the bots reconnect to the C&C box. This spike is not present in normal DNS request rates, which require (usually slower and random) user interaction to generate a DNS request. In some cases, flash crowds of users visiting a popular site may behave like a botnet [9], but this is rare, and likely not sustained as seen in botnets.

We can use the sorted rates of normal DNS requests over a 24 hour period to create a distribution, or density signature for normal traffic. Figure 3 shows the sorted 24-hour average rates for normal traffic. Compared to the sorted botnet traffic, the two distributions are quite different. Because of the diurnal spikes in traffic, the botnet traffic exhibits an exponential distribution.

We can then use any standard distance metric to compare the two distributions. Mahalanobis distance is one useful measure of the distance between request rate distributions and a normal model. (Though the results obtained using the Mahalanobis distance have been encouraging, one could also imagine using one of a plethora of other notions of distance between distributions.)

The Mahalanobis distance, d, is:

$$d^2(x, \bar{y}) = (x - \bar{y})' C^{-1} (x - \bar{y}) \tag{2}$$

x and $\bar{y}$ are variable vectors (features) of the new observation and the trained (normal) profile, respectively. C is the inverse covariance matrix for each member of the training data set.

As noted in [22], the Mahalanobis distance metric considers the variance of request rates in addition to the average request rate. This detects outliers, and measures the consistency of the observed request rates with the trained (normal) samples. As in [22], we can simplify the Mahalanobis distance metric by assuming the independence of each sample in the normal traffic (and therefore removing the covariance matrix).

$$d(x, \bar{y}) = \sum_{i=0}^{n-1} \left(\frac{|x_i - \bar{y_i}|}{\bar{\sigma_i}} \right) \tag{3}$$

As with the canonical SLD request rate, we can train using the normal model, and pick an appropriate threshold. If observed traffic for a host has too great a distance score from the normal, it is deemed an outlier, and flagged as a bot.

Response

We have developed, `KarstNet`, a collection of sinkholes located at different addresses on the Internet as a response mechanism to the detected botnets. The sinkhole collection is used by third party networks that need to redirect abusive traffic. `KarstNet` coordinates exchanges between sinkhole providers and DDNS owners, letting them arrange for sinkholes, swap binary samples and logs, and monitor botnet activity. An extensive database back end lets users track infected machines, Classless Inter-Domain Routing (CIDR) [6] blocks, and even geographical locations associated with infections.

Specifically, `KarstNet` lets DDNS providers enter a canonical name (`CNAME`) or other appropriate DNS record responses (`RR`) to direct bots towards an available `KarstNet` sinkhole. Once redirected, the DDNS server no longer returns the `CNAME` for the C&C, and instead points all traffic to the sinkhole. The sinkhole then plays "TCP games" with the bots, delaying them in routing- or application-level tarpits, blackhole routing (where no replies are sent at all), and generally passively consuming victim resources. Victim join limits, sinkhole responses, and other behaviors are also controlled by the systems exchange site.

In a four month period, `KarstNet` has trapped nearly 6 million unique victims and disrupted the C&C for dozens of botnets, all in size of 100K+ victims.

3.2 Quantitative Bisimulation for Program Similarity

The state of the art in virus programming has progressed to the point when novice hackers can quickly put together new viruses from available building blocks. Several toolkits, such as DREG and NGVCK [19], are readily available on the Internet. Their widespread use leads to the proliferation of new viruses and worms that belong to several well-known families, but at the same time are different enough that signature-based methods do not detect these new viruses using existing signatures. Development of a new signature for each new variant and propagating these new signatures to virus checking software becomes prohibitively expensive and inefficient.

We describe a different approach to detection of new viruses that belong to existing virus families. The approach exploits the fact that viruses are built from standard building blocks and thus exhibit substantial similarity. In order to tell whether a new program may be a new virus, we measure its similarity to known representatives of virus families.

The similarity measure is defined by generalizing *simulation* and *bisimulation* [13], two well-known process relations. These relations are widely used in the behavioral modeling of computer systems. A system is modeled as a *labeled transition system*, which describes how the system evolves from one execution state to another by performing certain actions that appear as labels on transitions between states. These relations capture the intuition that systems that are related can perform the same actions – in other words, one can simulate the other. A state s in one system simulates a state t in another system if, whenever the system in the state t can perform an action a and move to the state t', the other system in the state s can also perform the action a and move to the state s' that simulates t'. If this relation is symmetric, that is, t also simulates s, the relation is called bisimulation.

Although simulation and bisimulation capture the right intuition for comparison of two systems, they cannot be applied for the purpose of detecting similarity between virus programs because they call for exact matches between the steps of two systems, and evaluate to a boolean value rather than to a quantitative similarity rank.

We generalize the simulation relation to a function $Q(s,t)$ that takes its values in the interval $[0,1]$. We then apply the function to the control flow graphs of the two programs. Program S is the known representative of a virus family. Program T is the suspected virus. If s_0 and t_0 are the initial nodes of the control flow graphs of S and T, respectively, the high value of $Q(s_0,t_0)$ indicates that T exhibits a high degree of similarity to S and is likely to be a member of the virus family derived from S.

To define the function Q, we assume two *local similarity* functions and a parameter $p \in [0,1]$. The first local similarity function establishes similarity of two states in the transition systems, disregarding their transitions. For example, when we are comparing control flow graphs, nodes are labeled with fragments of assembly code that are executed when the system is in that state. The local node similarity compares how similar the two assembly fragments are. The node similarity function takes values in the interval $[0,1]$. We assume that the function is symmetric, that is, $N(s,t) = N(t,s)$ and that a node is perfectly similar to itself, that is, $N(s,s) = 1$. In a similar way, we define the edge similarity function $L(a,b)$, which computes the similarity between two actions performed by the two systems. The parameter p reflects the relative importance we assign to local similarity of the two nodes compared to the similarity of the steps that can be taken from the nodes and of the nodes that can be reached by these steps.

Now, we are ready to define the quantitative similarity function with the parameter p, Q_p. It is the function that satisfies the following condition:

$$Q_p(s_1,s_2) = \begin{cases} N(s_1,s_2) & \text{if } \forall a, s_1 \overset{a}{\not\rightarrow} \\ (1-p)\cdot N(s_1,s_2) + p\cdot \frac{1}{n}\cdot \sum_{s_1 \overset{a}{\rightarrow} s_1'} \max_{s_2 \overset{b}{\rightarrow} s_2'} L(a,b)\cdot Q_p(s_1',s_2') & \text{otherwise,} \end{cases} \quad (4)$$

where n is the number of transitions leaving s_1. For each transition of s_1, the definition finds the best match provided by a transition of s_2, and computes the average of these matches. This corresponds to the expected value of the match for a randomly selected transition of s_1. Thus obtained "step" similarity value is assigned weight p, while the "local" similarity value, given by the node similarity function, is assigned weight $1-p$. If s_1 does not have any outgoing transitions, quantitative similarity is given by the local similarity function.

The values of Q_p for all state pairs in the two transition systems can be computed using linear programming. We have started experimental evaluation of the utility of Q_p for the comparison of virus control flow graphs and obtained encouraging initial results.

Quantitative simulation can be extended to quantitative bisimulation B_p by making the definition symmetric. Intuitively, bisimilarity between states s and t equals to the minimum of the similarities between s and t and, conversely, t and s. Formally, $B_p(s_1,s_2) = \min(B_p^-(s_1,s_2), B_p^-(s_2,s_1))_p$ where $B_p^-(s_1,s_2)$ is defined as

$$B_p^-(s_1,s_2) = \begin{cases} N(s_1,s_2) & \text{if } \forall a, s_1 \overset{a}{\not\rightarrow} \\ (1-p)\cdot N(s_1,s_2) + p\cdot \frac{1}{n}\cdot \sum_{s_1 \overset{a}{\rightarrow} s_1'} \max_{s_2 \overset{b}{\rightarrow} s_2'} L(a,b)\cdot B_p(s_1',s_2') & \text{otherwise.} \end{cases}$$

Computation of quantitative bisimulation is closely related to finding the value of an *infinite-horizon 2-player stochastic game with imperfect information* [1]. While no polynomial algorithms are known for this problem, there is an approximate solution that can be computed in polynomial time and guarantees the required degree of precision.

Other generalizations for bisimulation and simulation of labeled transition systems can be found in the literature. In [4] several metrics have been proposed for *quantitative transition systems*, which are labeled with tuples of numbers instead of more conventional symbolic labels. The quantity computed (representing the degree to which some state simulates another state) is *extremal* In other words, it is the worst simulated transition from state s by a transition from state t that determines the simulation score of s by t. In [7] a notion of *approximate bisimulation* is defined for linear systems. Here again, a label matches labels within an ϵ ball and we need *every* transition from s to match in this approximate sense with a transition from t in order for s and t to be assigned a high bisimulation score. In contrast, our measure is *cumulative* in that it uses a (weighted) average of the extent to which each transition is simulated. Cumulative measures seem more appropriate for detecting similarity between virus programs since it is easy to defeat extremal measures by introducing dummy transitions that do not match with any others.

3.3 Learning Malware Signatures through Case-Based Reasoning

The purpose of *intrusion detection* is to recognize malicious computer hacking, i.e., malware, based on evidential traces. Intrusion end-goals include, for example, corruption, disablement, and identity-theft. The traditional approach to intrusion detection consists of matching, as early as possible, a new script/scenario against known prior attacks, called "signatures." Signatures are typically stored in a database. If the new script is similar enough to a known former attack signature, then it is labeled "positive" (an attack) by the network intrusion detection system (NIDS); otherwise, it is labeled "negative" (not an attack). This project assumes that each script or signature has the form of a sequence of commands.

Typically, NIDS try to find an exact match between a new ongoing script and a prefix of a stored attack signature. Unfortunately in practice an exact match is rarely possible to find, even though the new script may be the beginning of an attack that is nearly identical semantically to a prefix of a previous attack. For example, the attacker might use recently invented, disguised commands that perform the same function as previous commands.

The most significant and prevalent problem for intrusion detection systems occurs when a true attack is in progress, but the NIDS fails to recognize it as an attack. This is called a *false negative*, or an *error of omission*. The primary contribution of our research is to address and substantially mitigate this problem of false negatives. In particular, a case-based reasoning (CBR) [11] approach is applied, which allows the NIDS to attempt a "flexible," rather than an exact match between a new script and a prior attack signature in the database. It is the *analogy* portion of CBR systems that provides flexibility in matching. The particular type of analogy that we use is semantic, rather than syntactic. The latter is more traditional for analogy; however, semantic analogies enable greater flexibility in matching. For example, by matching semantically, a NIDS is not deceived by deliberately camouflaged commands.

Our innovation is to convert a script or signature from syntax to semantics using an artificial intelligence (AI) technique called *plan recognition*. Our plan recognition approach infers the hacker's knowledge and goals from the hacker's ongoing sequence of commands. The output of our program is designated a *knowledge script*, which is annotated with hypothesized knowledge and is eventually stored in the database as a *knowledge signature*. It consists of a temporal sequence of inferred information tidbits assumed to be gained by the hacker while typing in commands. A knowledge script/signature is the semantic equivalent of the original script/signature. For example, after typing the command "ping," a hacker has knowledge of whether or not the pinged computer is up and running. This information is stored explicitly in a knowledge script. As expected, a semantic knowledge script/signature is substantially easier to match than the original syntactic script/signature. The matching process is flexible because it is done by analogy. If, for example, *Computer 1* and *Computer 13* play the same role in two different attacks, they are considered analogous, or synonymous, and a match between them will succeed despite the fact that their names differ. By reducing the number of false negatives, the number of *false positives* may inadvertently be increased. A false positive is defined to occur if a non-attack is incorrectly labeled "positive," i.e., as an attack, by the NIDS. This problem of false positives is also addressed using Artificial Intelligence techniques.

In summary, our semantic analogy-based approach using case-based reasoning is an appropriate solution to the problem of false negatives. We have implemented and tested our approach, and initial experimental results have demonstrated a dramatic reduction in the number of errors of omission (false negatives).

Current research on the artificial intelligence approach is focused on four major directions. First, we are running extensive, rigorous experimental comparisons between several alternative *similarity metrics* used in analogy, in order to select the one that performs the best in the context of intrusion detection. Second, we are developing novel similarity metrics that can not only label a script as a "positive" attack, but can also designate a label regarding the hypothesized *type* of attack. Third, we are working toward a formalization of similarity that is well adapted to intrusion detection, but is also "universal," in the sense that it will not depend on the representation of attacks (e.g., one might wish to compare the similarity between a script and

a stored signature that are in different computer languages). Theoretical results will be developed regarding the universality of this similarity. Finally, if our system concludes that a new script is indeed an ongoing attack (based on similarity to a stored signature), then the next step will be to use Artificial Intelligence techniques to predict the attacker's next move(s). Predictions could be extremely valuable for designing countermeasures to attacks, such as decoys or honeypots.

4 Conclusions and Future Work

We have described three results in the areas of network intrusion detection and malware detection. We are also refining the infrastructural framework that can serve as a platform on which different intrusion and anomaly detection modules reside.

Further research on the highlighted problems is focused on (1) the deployment and refinement of Karst-Net, (2) Variations and extensions of the notion of quantitative simulation to allow for imperfect synchrony between the two systems and to deal with an input consisting of more than 2 systems, and (3) The search for a notion of similarity for analogy-based reasoning that best fits the intrusion detection context.

References

[1] K. Chatterjee, M. Jurdziński, and T.A. Henzinger. Quantitative stochastic parity games. In *SODA '04: Proceedings of the 15th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 121–130, 2004.

[2] CipherTrust. Ciphertrust's zombiemeter. `http://www.ciphertrust.com/resources/statistics/zombie.php`, 2005.

[3] David Dagon, Cliff Zou, Sanjeev Dwivedi, Julian Grizzard, and Wenke Lee. Karstnet: Countering the attacking networks. Technical report, Georgia Institute of Technology, May 2005.

[4] L. de Alfaro, M. Faella, and M. Stoelinga. Linear and branching metrics for quantitative transition systems. In *ICALP '04: 31st International Colloquium on Automata, Languages, and Programming*, volume 3142 of *LNCS*, pages 97–109, 2004.

[5] Felix C. Freiling, Thorsten Holz, and Georg Wicherski. Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks. Technical Report ISSN-0935-3232, RWTH Aachen, April 2005.

[6] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless inter-domain routing (cidr): an address assignment and aggregation strategy. http://www.faqs.org/rfcs/rfc1519.html, 1993.

[7] A. Girard and G.J. Pappas. Approximate bisimulation relations for constrained linear systems. Technical Report MS-CIS-05-19, Dept. of CIS, University of Pennsylvania, September 2005.

[8] Cody Hatch. Distributed phishing. `http://marc.theaimsgroup.com/?l=dailydave&m=111505651308364`, 2005.

[9] Srikanth Kandula, Dina Katabi, Matthias Jacob, and Arthur W. Berger. Botz-4-sale: Surviving organized ddos attacks that mimic flash crowds. In *2nd Symposium on Networked Systems Design and Implementation (NSDI)*, May 2005.

[10] H.A. Kim and B.Karp. Autograph: toward automated, distributed worm signature detection. In *Proc. 13th USENIX Security Symposium*, 2004.

[11] David B. Leake, editor. *Case-Based Reasoning: Experiences, Lessons, and Future Directions.* AAAI/MIT Press, 2000.

[12] Jun Li, J. Mirkovic, Mengqiu Wang, P. Reiher, and Lixia Zhang. Save: source address validity enforcement protocol. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 1557–1566, 2002.

[13] Robin Milner. *Communication and Concurrency.* Prentice Hall Intl., 1989.

[14] J. Newsome, B. Karp, and D. Song. Polygraph: Automatically generating signatures for polymorphic worms. In *Proc. IEEE Symp. Security and Privacy (Oakland 2005)*, 2005.

[15] Prof. Randy Vaughn (Baylor Univ). Personal correspondence. April 5 2005.

[16] S.E. Schechter and M.D. Smith. Access for sale. In *2003 ACM Workshop on Rapid Malcode (WORM'03)*. ACM SIGSAC, October 2003.

[17] John Schwartz. Computer vandals clog antivandalism web site. *New York Times*, 2001.

[18] S.Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In *Proc. 6th ACM/USENIX Symp. Operating Systems Design and Implementation*, 2004.

[19] Peter Szor. *The Art of Computer Virus Research and Defense.* Addison Wesley Professional, 2005.

[20] The Honeynet Project and Research Alliance. Know your enemy: Tracking botnets. `http://www.honeynet.org/papers/bots/`, 2005.

[21] United States Government Accounting Office. Emerging cybersecurity issues threaten federal information systems. `http://www.gao.gov/new.items/d05231.pdf`, May 2005.

[22] Ke Wang and Sal Stolfo. Anomalous payload-based network intrusion detection. In *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID 2004)*, 2004.

An Ensemble of Anomaly Classifiers for Identifying Cyber Attacks*

Carlos Kelly† Diana Spears‡ Christer Karlsson§ Peter Polyakov¶

Abstract

A novel approach is presented that bridges the gap between anomaly and misuse detection for identifying cyber attacks. The approach consists of an ensemble of classifiers that, together, produce a more informative output regarding the class of attack than any of the classifiers alone. Each classifier classifies based on a limited subset of possible features of network packets. The ensemble classifies based on the union of the subsets of features. Thus it can detect a wider range of attacks. In addition, the ensemble can determine the probability of the type of attack based on the results of the classifiers. Experimental results demonstrate an increase in the rate of detecting attacks as well as accurately determining their type.

Keywords: intrusion detection, classifier ensemble

1 Problem Description.

In our current information age, and with the timely issue of national security, network security is an especially pertinent topic. One important aspect of computer network security is *network intrusion detection*, i.e., the detection of malicious traffic on a computer network.

There are two main approaches to designing Network Intrusion Detection Systems (NIDS): *anomaly detection* and *misuse detection*. Both are essentially classifiers, i.e., they label incoming network packets as "attack," "non-attack," and if an attack perhaps what type of attack. Anomaly and misuse detection are complementary approaches to intrusion detection. Anomaly detection consists of building a model of normal computer usage, and tagging outliers as "anomalies." Such systems are typically computationally efficient, but only yield a binary classification "attack" or "non-attack" [5, 12]. Misuse detection systems match potential attacks (e.g., network packets) against a database of known attacks (called *signatures*). If there is a match, then the data (packet) is labeled an "attack," and the class of attack is considered to be the same as that of the matching signature. Unfortunately, misuse detection systems tend to be slow, especially if their database of signatures is large [10].

The main thrust of our research is to bridge the gap between anomaly and misuse detection. Anomaly NIDS classify packets quickly in comparison to misuse based NIDS, but without as much information about the attack. We have created an ensemble of anomaly-based NIDS that refines the binary classification of each ensemble member and yields more detailed classification information than each member alone. Therefore, if anomaly detection precedes misuse detection, then our system will partially refine the output of anomaly detection, thereby accelerating the processing of the misuse detection system. The pipeline can be summarized as: (1) anomaly detection, (2) refinement by ensemble, and (3) misuse detection. If the computational cost of the second step, refinement by ensemble, is lower than the computational benefit that it yields by shortening the run-time of the misuse detection system, then our approach will be beneficial overall. Whether this is the case, depends on the size of the database of signatures that one maintains. Over time, as people (and systems) increase their knowledge base of attacks, we expect our approach to become increasingly more useful.

This paper describes only steps (1) and (2) of the pipeline above. Step (3) will be addressed as part of future work. In the remainder of this paper, we describe our ensemble approach, as well as experimental evaluation results that show its effectiveness for intrusion detection. Here, it is assumed that the data consists of Transmission Control Protocol (TCP) packets, sent over the network. The data we used is from the DARPA/MIT Lincoln Laboratory database (see http://www.ll.mit.edu/IST/ideval/index.html).

*Supported by the ONR URI grant "Anomaly and misuse detection in network traffic streams," subcontract PENUNV48725.

†Mathematics Department, University of Wyoming.

‡Computer Science Department, University of Wyoming.

§Computer Science Department, University of Wyoming.

¶Mathematics Department, University of Wyoming.

Presented at the International Workshop on Feature Selection for Data Mining: Interfacing Machine Learning and Statistics, April 22, 2006 [held in conjunction with the 2006 SIAM Conference on Data Mining (SDM)] <http://www.siam.org/meetings/sdm06> Reprinted by permission.

2 A Novel Ensemble Approach.

An *ensemble* of classifiers is a collection of classifiers that are combined into a single classifier. Most of the research conducted on classifier ensembles assumes homogeneous ensembles of binary classifiers, and assumes that the ensemble also outputs a binary classification. The purpose of such ensembles is to increase classification accuracy, e.g., with voting, *bagging*, or *boosting* [1]. One notable exception is the *stacked generalization* approach of Wolpert [13]. Stacked generalization assumes a heterogeneous ensemble of different classifiers, each with its own "area of expertise." Nevertheless, the purpose of stacked generalization is also to increase classification accuracy, without changing the set of classes.

To the best of our knowledge, *our approach to ensembles is the first to utilize a heterogeneous set of classifiers for the purpose of increasing information (refined classification), rather than classification accuracy.* Specifically, our approach takes a suite of classifiers (currently two), each of which outputs a binary classification, and combines them to output a probability distribution over *seven* classes. We expect the ensemble output to be increasingly more informative as the number of classifier components is increased beyond two. Furthermore, if the classifiers run on the data in parallel, adding more classifiers to the ensemble would not increase the overall time to apply the method, i.e., it is highly scalable. However, this is our first investigation into such an ensemble, so we begin with two classifiers.

The essence of our approach is to empirically build an *ensemble probability classification matrix*, abbreviated as *EPCM*, based on system performance on test data. In other words, in machine learning one typically trains the system on training data, and then tests its performance on test data. We instead partition the data into three sets: the training data, the testing data, and the validation data. Each individual classifier is trained separately on the training data. Note that the training data is attack-free – because anomaly detection systems learn models of normal (friendly) user data, and then use these models to detect anomalies, which are labeled "attacks." After training, each system has a hypothesis regarding the nature of "non-attack" data. These hypotheses are applied to the testing data, to make predictions regarding whether each system thinks each packet is an "attack" or a "non-attack." We also use the known information (from the DARPA website) on the test data regarding whether each packet is an attack or not, and if it is an attack then what class of attack (from the seven known classes). All of this information is automatically combined into an EPCM, which predicts a probability distribution over the seven classes, based on the outputs of the systems in the ensemble and the true classes of the packets (as defined by DARPA/MIT). The last step is to test the performance of the ensemble on a set of validation data, for which there is no advance knowledge given to the system regarding the (true) data classification.

Why do we expect our novel approach to work? The key to our success is the notion of *inductive bias*, or simply *bias*. Mitchell defines *bias* as "any basis for choosing one generalization over another, other than strict consistency with the observed training instances" [9]. The hypotheses output by our classifiers are special instances of what Mitchell calls "generalizations." An example of a bias is the choice of what attributes the classifier system considers. For instance, one system might only look at the header information in a packet when classifying the packet as a type of attack, whereas another system might only look at the packet payload. Certainly the choice of attributes will affect the types of attacks that the system is able to identify. One system might be able to detect some classes of attacks; another system might be able to detect other classes. In general, the classes of attacks detectable by two systems could be disjoint or overlap. By combining two systems with very different biases, we increase the set of detectable attacks. Furthermore, by exploiting known differences in system biases, we can further refine our classification knowledge. For example, if one system says "attack" and the other says "non-attack," then this combined information can tell us (with high probability) what *kind* of attack it is most likely to be. To better understand the synergistic effects of combining the information, we formalized the biases of each of the two systems. From this formalization, one can understand the classes of attacks for which each system is best suited to identify. This is the essential rationale behind our ensemble approach.

3 Ensemble Components.

Our ensemble is composed of two anomaly NIDS, LERAD [5] and PAYL [12]. LERAD's hypotheses are rule sets of expected (normal) user behavior, and PAYL's hypotheses are byte distributions derived from normal packets. Each of these systems is described in greater detail, below.

Some preprocessing of the raw network dump data was necessary for LERAD and PAYL to be able to process packets. A tool (te.cpp) provided on Mahoney's web site http://www.cs.fit.edu/mmahoney/dist preprocessed the raw network data into streams for LERAD. A Perl script (a21.pl), also provided by Mahoney, transformed

the streams into a LERAD-readable database format. The preprocessing tool te.cpp was altered so that it also produced a file of packets readable by PAYL.

Also, some postprocessing was required. LERAD and PAYL produce a list of packets that the systems consider to be "attacks" (anomalies). We created a tool that produces alarm statistics by comparing the output of LERAD and PAYL to the DARPA/MIT labeled attacks. For further details on this postprocessing stage, see Section 5 below.

Finally, before we describe each system, note that we used LERAD unmodified as found on Mahoney's web site, listed above. However, the source code for PAYL is not currently available, and therefore we re-implemented the algorithms based on [12].

3.1 LERAD and Its Biases. As mentioned above, LERAD learns a set of classification rules. Rules take the following general form: $(a_i = v_j \wedge \cdots \wedge a_n = v_m) \rightarrow (a_k = v_p \vee \cdots \vee a_q = v_s)$ where the a's are attributes and the v's are values of these attributes. Only conjunction is allowed in the rule antecedent and only disjunction is allowed in the rule consequent. An example rule might be:
If the destination port number is 80 and the source port number is 80, then the first word in the payload is GET or the first word in the payload is SET or the number of bytes in the payload is greater than 60.
Recall that each of these rules describes normal (benign) system use.

LERAD's classification algorithm is the following. Each new example (packet) receives a score, which is the sum of rule violations. If the score exceeds a threshold, T_L, defined below, then the packet is classified as an "attack."

LERAD's training algorithm inputs a set of attack-free training examples, and outputs a rule set, R. The algorithm begins with rule creation, then does rule sorting and, finally, rule pruning.

LERAD has many implicit inductive biases embedded within the system. We selected those that are most relevant to the construction of our ensemble and formalized them. By doing this, we were better able to understand and predict the types of attacks for which LERAD is best suited to detect.

What are these relevant biases of LERAD? One is the set of attributes considered by LERAD, called S_L. We know that $|S_L| = 23$, and the specific attributes are the packet date, time, last two bytes of the destination IP address, last four bytes of the source IP address, the source and destination port numbers, the TCP flags for the first, next to last and last TCP packets, the duration in seconds, the number of payload bytes, and the first eight words in the payload.

A second bias is the threshold, T_L, used by LERAD during classification. Before formalizing this threshold, we first repeat the formula for the anomaly score for each new example (packet), which we consider a bias. From [5] this is: $score_{anomaly} = \sum_{i=1}^{m} \frac{n_i \cdot t_i}{e_i} F_{r_i}$ where F_{r_i} is 0 if rule r_i is satisfied and 1 if it is not satisfied, m is $|R|$ (i.e., the number of rules), n_i is the rule support for rule r_i (defined above), e_i is the number of expressions in the antecedent of rule r_i, and t_i is the time that has elapsed since the rule was last violated. Then the threshold, T_L, is: $ln(score_{anomaly})/ln(10) > 4.5$. Finally, the last bias that we considered relevant in LERAD is the fact that its hypotheses take the form of rules, which we already formalized syntactically above.

3.2 PAYL and Its Biases. The classification hypotheses of PAYL are byte distributions, derived from the training data (see Figure 1).distribution is an empirically-derived approximation of a probability distribution $P(b_0, b_2, \ldots, b_{255})$, i.e., the probability of seeing each ASCII byte in a packet of a certain type. The types of packets are those that have a particular destination port number or a particular payload length. In other words, for each unique port number and payload length, PAYL associates (and learns) a probability distribution over the individual bytes in the payload. In fact, the full hypothesis of PAYL consists of a set of *profiles*. Each profile consists of a pair of 256-valued vectors (one for each byte). The first element of the pair is an average byte distribution, $P(b_0, b_1, \ldots b_{255})$, and the second element of the pair is a vector of standard deviations from the means, i.e., $(\sigma_0, \sigma_1, \ldots, \sigma_{255})$. Classification involves both a distance function and a threshold. If the distance between the byte distribution of a new example and the byte distribution of the hypothesis (which represents a profile of normal behavior) exceeds the threshold, then the new example is labeled an "attack."

PAYL's training algorithm consists of first classifying the training examples (packets) according to their destination port number and payload length. Then, the mean and standard deviation are calculated for each byte.

PAYL also has implicit inductive biases embedded within the system, and we selected those that are most

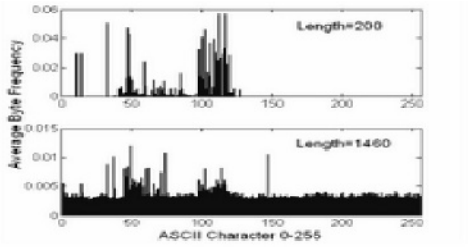


Figure 1: Sample byte distributions for different payload lengths of port 80 on the same host server.

relevant to the construction of our ensemble. The first bias is the set of attributes considered by PAYL, called S_P. Note that $|S_P| = 3$. These attributes are the destination port number, the number of bytes in the payload, and the distribution of bytes in the payload.

The second aspect of PAYL that we consider to be a bias is its distance function for computing the distance between two distributions. The function used by PAYL is (from [12]): $d(e, \overline{y}) = \sum_{i=0}^{255} \frac{|e_i - \overline{y_i}|}{\sigma_i + \alpha}$ where $\overline{y}$ and σ are the average and standard deviation, i.e., elements of the profile that constitutes PAYL's hypothesis, e is an example, and α is a conditioning variable needed to prevent divide-by-zero errors.

The threshold, T_P, was not formalized. It was derived empirically, as described in Section 5, below.

3.3 Example Application of the System Biases. By making LERAD and PAYL's biases explicit, we have been able to analyze and understand them better. From this process, we have drawn the following conclusions about the suitability of these two systems to detecting different attack characteristics:

- LERAD is sensitive to only a subset of the information in a packet, namely, the packet header and the first eight words.
- Two packets that differ by even a single byte (among the attribute fields that LERAD examines) are likely to be classified differently by LERAD. A packet that satisfies both the antecedent and consequent of a rule can be made to violate the rule by changing a single byte in one of the fields (attributes) of the consequent.
- PAYL is sensitive to only a subset of the information in a packet, namely, the packet payload and the destination port number.
- Two packets that differ by a single byte are unlikely to be classified by PAYL as being different – because the byte distributions of the two packets will probably be very similar.

The following example illustrates how these biases of LERAD and PAYL translate into specialized detection capabilities:

EXAMPLE 1. Consider the *Denial of Service* attack called "Back." This attack is a malformed web request to an HTTP port with the payload, "Get //// ..." followed by 6000-7000 slashes. One of the biases of PAYL is that it examines the byte distributions. For this particular attack, the byte distribution of the payload is almost exclusively centered on the "/" character in the ASCII table. This implies that PAYL will almost surely detect the attack. One of the biases of LERAD is that it examines the relationships of the first eight words in the payload of a message. Since "GET" followed by 6000-7000 slashes is an unusual relationship, we would also expect LERAD to detect this attack. Therefore, the "Back" attack is a specific type of attack that would be detected by both systems.

4 The Network Data.

As mentioned above, we are working with the DARPA/MIT Lincoln Laboratory database of packets. We decided to only work with the 1999 data, since the 1998 data does not include a key to differentiate normal packets from attack packets. The data from 1999 is five weeks long. The first and third weeks are attack free, whereas the

second, fourth and fifth weeks contain attacks. Each TCP packet is a binary sequence not exceeding 64,000 bytes in length. The DARPA web site classifies attacks into five categories: 1.**Denial of Service (DoS)**, 2.**User to Root (U2R)**, 3.**Remote to User (R2U)**, 4.**Probes.**, and 5. **Data.**

This classification is standard, comprehensive, and still modern [11]. Nevertheless, this DARPA classification does not result in mutually exclusive classes. Therefore, we have expanded the classification to include overlaps as two additional classes, thus resulting in a total of seven attack classes. The two additional attack classes are: 6. **Data and User to Root** and 7. **Data and Root to Local**. The 1999 DARPA data was divided into three sets. The training set consisted of all the data that was attack free. Training data: 03/08-03/12 (week one) and 03/22-03/26 (week three). Of the remaining data, the test and validation data sets were divided as follows: test data: 03/29-03/31, 04/02, 04/05 and 04/09; validation data: 04/01, 04/06-04/08. Both the test and validation data sets contain many attacks, though some attacks occur only in the test set and other attacks occur only in the validation set. There are slightly more attacks in the test set than in the validation set.

5 Ensemble Implementation.

In this section we describe in detail step (2) of the pipeline, introduced in Section 1, and called the "refinement by ensemble" step. The purpose of the ensemble is to produce a classification vector associated with a new packet, which could be an attack. For the ensemble, we concentrate only on test data that consists of attacks, i.e., non-attack packets are ignored.[1] There are a couple of reasons that we did this. For one, these are the only packets whose classification needs to be refined. Second, segmenting the data to determine the temporal boundaries of non-attacks proved to be very difficult – it proved challenging to determine the exact duration of non-attack packets. Therefore, our test data and validation data focused on attack packets only, and how to refine their classification.

When combining LERAD and PAYL, there are four possibilities for the combined outputs of the two systems: L-yes and P-yes, L-yes and P-no, L-no and P-yes, and L-no and P-no, where "yes" means the packet is labeled an "attack" and "no" means the packet is labeled a "non-attack." One of these pairs becomes the input to the ensemble system, for each new example/packet. The output of the ensemble for one pair is a probability distribution, called the *probability classification vector*, which gives the probabilities that the new example falls into each of the seven possible attack classes, described in Section 4 above. In summary, given a new packet, which we also call a *sample* to be consistent with statistical definitions, there are four possible events corresponding to the four possible class labels given by the pair of classifier systems. These input events need to be converted to an output probability distribution over the seven attack classes. Recall that this process is performed on the test data set. In other words, the training data is used for LERAD and PAYL to learn their hypotheses, and then these hypotheses make predictions over the test data to discover anomalies that are different from the hypotheses about normal user behavior. We combine LERAD and PAYL's predictions on the test data with the true (based on the DARPA web site) classifications of the test data packets. Then, we convert this information into a probability classification vector for each pair of outcomes from LERAD and PAYL. These vectors are joined together in an ensemble probability classification matrix (EPCM), and output by the ensemble (see Table 2).

To accomplish this, the first step is to formalize, in probability terminology, what precisely we are trying to find during step (2), i.e., what is the formal representation of an ensemble probability classification matrix (EPCM)? The answer is that we want to find $P(C|E)$, where C is a classification vector, i.e., a probability distribution over the seven classes, and E is an input event, i.e., the pair of labels given by LERAD and PAYL, such as L-yes and P-no. This probability value cannot be approximated directly from the results of the test data. We need to use a conditional probability rule to calculate this conditional probability. The conditional probability rule that we use is: $P(Y|X) = (P(Y,X)/P(X))$.

For example, suppose we have the results from the test data in Table 1. Each table entry that is not listed under "Sum" represents $P(Y,X)$, i.e., the frequency (which is an estimate of the probability based on the test data) of a packet giving a certain pair of binary outputs by LERAD and PAYL *and* being in a particular attack class (based on the DARPA/MIT web site classification of the test data). Furthermore, the "Sum" entry at the bottom of each column represents $P(X)$, i.e., the frequency of a certain pair of binary outputs by LERAD and PAYL. Using the conditional probability rule, given above, we calculate $P(Y|X)$, which is the output of the ensemble. Continuing our example from Table 1, by applying the conditional probability rule we get Table 2.

[1] We use the same criteria as DARPA did to label packets as "attacks."

Attack	y/y	y/n	n/y	n/n	Sum
Class 1	4	8	5	6	23
Class 2	6	4	2	1	13
Class 3	1	16	1	10	28
Class 4	0	0	2	1	3
Class 5	0	6	0	7	13
Class 6	2	1	1	0	4
Class 7	0	0	1	0	1
Sum	13	35	12	25	85

Table 1: Matrix of frequencies of attack events and classifier labels for LERAD/PAYL.

Attack	y/y	y/n	n/y	n/n
Class 1	0.3077	0.2286	0.4167	0.2400
Class 2	0.4615	0.1143	0.1667	0.0400
Class 3	0.0770	0.4571	0.0833	0.4000
Class 4	0	0	0.1667	0.0400
Class 5	0	0.1714	0	0.2800
Class 6	0.1538	0.0286	0.0833	0
Class 7	0	0	0.0833	0

Table 2: An ensemble probability classification matrix (EPCM), which is output by the ensemble. Each column is a probability classification vector.

Note that this is a matrix consisting of vectors (the columns) – one for each input event/sample, giving the vector output that is a probability distribution over the seven possible attack classes. This is what we call the "ensemble probabilistic classification vector."

For each new packet in the final validation data we can now use these vectors for classification. In particular, we run LERAD and PAYL on this new packet. If we get L-yes and P-no, then the ensemble predicts (using Table 2) the probability that the attack is of type Denial of Service is approximately 0.2268. The probability that the attack is of type User to Root is approximately 0.1143, and similarly for the remaining classes of attacks.

Given this ensemble output information, a misuse detection system could restrict its search and computations to a small subset of possible attack signatures when trying to find the most similar previous attack. The reasons for continuing with a misuse detection system are that our ensemble outputs probabilities – however a match with a signature could give further confirmation of the attack class, and also a stored signature could be used for predicting the attacker's next move.

We conclude this section by noting the role that the system inductive biases played in determining the probability classification vectors. Note that if the ensemble input is L-yes and P-yes, then the ensemble will conclude that the highest probability is that we either have an attack of Class 1 (Denial of Service) or an attack of Class 2 (User to Root). Having a high probability of being an attack of Class 1 can be explained in terms of the system biases. Recall Example 1 from Section 3.3, which was an example of a Denial of Service attack. In that case, the large number of slashes indicated that such an attack would be manifested as an unusual byte distribution and would therefore be likely to be detected by PAYL. Furthermore, the usual relationship between the slashes and one of the keywords indicated that such an attack would also probably be detected by LERAD. Based on the system biases, we therefore predicted that Denial of Service attacks would frequently result in L-yes and P-yes. Table 2 indeed confirms our prediction.

In summary, our analysis of system biases was quite helpful for both predicting and understanding the output of our ensemble. Future versions of our ensemble approach will investigate building an ensemble from first principles, based on bias analyses, rather than using a purely empirical approach.

5.1 Parameter Tuning. LERAD's process of learning a rule set involves a random element (see Section 3.1). Nevertheless, our experimental investigations revealed that there are not significant differences in performance

Attack	y/y	y/n	n/y	n/n	Sum
Class 1	5	9	8	1	23
Class 2	6	4	2	1	13
Class 3	1	13	9	5	28
Class 4	1	1	0	1	3
Class 5	2	8	2	1	13
Class 6	1	2	1	0	4
Class 7	0	0	1	0	1
Sum	16	37	23	9	85

Table 3: A random frequency matrix with the same row sums as in Table 1.

arising as a result of alternating the random seed. Therefore, we fixed LERAD's random seed to be 0, and all results described in this paper assume this same seed.

We ran extensive empirical experiments to find optimal settings for the parameters of PAYL: $T_P = 256$ and $\alpha = 0.1$. These are the values that are used in all of the empirical experiments, described below.

6 A Matrix-Matrix Comparison.

6.1 Another Matrix for Comparison. To evaluate the quality of our ensemble output, we require a comparison against a reasonable standard. For this purpose, we decided to use a *random frequency matrix*. Such a matrix is created using randomly-chosen matrix entries that are weighted based on the relative frequency of each class of attack in the test data. In other words, it is not purely random, but contains useful information about attack frequencies, and it is constructed from the test data – just like our ensemble is.

The particular methodology for creating the random frequency matrix was to use the test data to determine both the attack frequencies and to ensure that the random frequency matrix has the same row sums as the actual frequency matrix created from the test data (which was shown in Table 1 and was used directly for building the ensemble). In other words, *both* the ensemble probability classification matrix (EPCM) and the random frequency matrix are constructed based on information from the actual frequency matrix derived from the test data set. The difference between them is that the EPCM has probability entries that directly reflect the test data, whereas the random frequency matrix has characteristics that reflect those of the test data, but includes some randomness. An example of a transformation of an actual frequency matrix to a random frequency matrix is shown in Table 3. Then, we convert the random frequency matrix into entries that are probabilities, just like we did for the EPCM in Section 5. We call this final matrix a *weighted random probability classification matrix*, abbreviated *WRPCM*.

Finally, observe that the WRPCM is randomly created. Therefore comparing the EPCM with one WRPCM is statistically meaningless. To resolve this issue, we created 10,000 WRPCMs to compare with one EPCM, and took the mean and standard deviation of the differences as our evaluation.

6.2 Evaluation Metric. We created a *validation probability matrix (VPM)* over the validation data set – for the validation data set this is "ground truth" and is used as the performance standard. To measure the distance between the EPCM or a WRPCM and the VPM, we used the standard *Euclidean metric*, which sums distances between pairs of matrix entries.

We applied the Euclidean evaluation metric to compare the EPCM-VPM distance versus WRPCM-VPM distance, on the validation data set. The following section describes the results of these comparisons.

6.3 Experimental Results. The average distance from the EPCM-VPM distance value to the 10,000 WRPCM-VPM distance values is 0.7264, and the standard deviation is 0.1516. Using the Euclidean metric, we find that the distance between the EPCM-VPM value and the mean of the WRPCM-VPM values is 0.3463, and the EPCM-VPM value is 2.507 standard deviations from the mean of the WRPCM-VPM values.

6.4 Interpretation of Results. The EPCM is more than 2.5 standard deviations closer (which is better) to the VPM (considered "ground truth") on the validation set than the average of the 10,000 WRPCMs. In other words, a weighted random guess has a very low chance of being more accurate than the EPCM. In particular,

the probability that a random accuracy variable X is less than the ensemble accuracy is $P(X \leq 0.3463) = P(((X-\mu)/\sigma) \leq ((0.3463-\mu)/\sigma) = P(z \leq -2.5075) = F(-2.5073) = 0.0062$, assuming distances are normally distributed. From the experimental results, we found that only 24 of the 10,000 WRPCM accuracies were better than those of the ensemble, which is quite low.

7 Summary and Future Work.

In summary, we have introduced a novel approach to an ensemble of classifiers that is designed for classification refinement, rather than for improving classification accuracy. Our experimental results indicate that our approach is very promising, and applicable to intrusion detection. In particular, our ensemble increased the number of attack classes from one to seven. Furthermorecloser (which is better) to the correct VPM on the validation data than the average of its competitors (the WRPCMs).

Our ensemble has an important role to play in refining the binary classifications output by the anomaly detection systems, prior to running a misuse detection system. The final step of the pipeline process described in Section 1, that of feeding the ensemble output into a misuse detection system, needs to be accomplished as part of future work. For example, we might use SNORT [10], which is the most widely available commercial misuse detection system. SNORT has a rule associated with each attack, so we might consider using our ensemble to partition the rule set according to attack class, and then check a potential attack packet with the rules from the class to which there is the greatest probability (according to the ensemble) that the attack belongs. This would increase SNORT's classification speed. It is interesting to note that for this paradigm, valuable information would be produced by the ensemble even if all classifiers (members) of the ensemble individually classified the candidate packet as a "non-attack." This is because even if its component classifiers label a packet as a "non-attack," the ensemble still predicts a class of attack for the packet. Therefore, if the packet does indeed turn out to be an attack, the ensemble will be especially helpful.

Finally, recall that we mentioned earlier that this ensemble approach is not only scalable, but is likely to benefit in performance from the incorporation of additional classifiers. We intend to explore this fruitful future direction for our research.

References

[1] E. Alpaydin, *Introduction to Machine Learning*, MIT Press: Cambridge, MA, 2004.
[2] Defense Advanced Research Projects Agency, *DoD Standard Transmission Control Protocol*, Information Processing Techniques Office, Arlington, VA (1990).
[3] R. Durst and T. Champion and B. Witten and E. Miller and L. Spagnuolo, *Testing and evaluating computer intrusion detection systems*, Comm. of the ACM, 42(7), (1999), pp. 53–61.
[4] J. W. Haines and R. P. Lippmann and D. J. Fried and E. Tran and S. Boswell and M. A. Zissman, *1999 DARPA Intrusion detection system evaluation: Design and procedures*, MIT Lincoln Laboratory Tech. Report.
[5] M. V. Mahoney, *A machine learning approach to detecting attacks by identifying anomalies in network traffic*, Ph.D. dissertation, Florida Tech., 2003.
[6] M. V. Mahoney and P. K. Chan, *An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection*, Proc. RAID, (2003), pp. 220–237.
[7] ———, *Learning rules for anomaly detection of hostile network traffic*, Proc. Third International Conference on Data Mining (ICDM), (1987).
[8] J. S. Milton and J. C. Arnold, *Introduction to Probability and Statistics Principles and Applications for Engineering and the Computer Sciences*, Third Ed., McGraw-Hill: NY, 1995.
[9] T. M. Mitchell, *Machine Learning*, McGraw-Hill: Boston, MA, 1997.
[10] L. Schaelicke and K. Wheeler and C. Freeland, *SPANIDS: A scalable network intrusion detection loadbalancer*, Proc. Comp. Frontiers, (2005), pp. 315–332.
[11] J. Wang, *Loss-sensitive rules for intrusion detection and response*, Ph.D. dissertation, Univ. of Pennsylvania, 2004.
[12] K. Wang and S. J. Stolfo, *Anomalous payload-based network intrusion detection*, Proc. RAID, (2004), pp. 1–12.
[13] D. H. Wolpert, *Stacked generalization*, Neural networks 5, (1992), pp. 241–259.

Distributed System Security via Logical Frameworks

Lujo Bauer, Frank Pfenning, and Michael K. Reiter
Carnegie Mellon University

August 2005

Abstract

We describe a project to advance security in distributed systems via the application of logical frameworks. At the heart of the effort lies an *authorization logic* which plays a triple role: (1) to specify an access-control policy as a logical theory, (2) to enforce the policy by mechanically verifying proofs in the logic, and (3) to reason about the policy by characterizing the space of all possible proofs. We are deploying a security infrastructure based on these ideas using mobile phones as a universal access-control device at Carnegie Mellon University.

ACM subject classifiers: C.2.0 General—*Security and protection*; D.4.6 Security and Protection—*Access controls*; F.4.1 Mathematical Logic—*Computational Logic*; K.6.5 Security and Protection—*Authentication*

Keywords: Security, logical frameworks, authorization, access control.

1 Introduction

Our goal is to advance security in distributed systems via the application of logical frameworks. Our research targets multiple facets of the life-cycle of a distributed system, ranging from design through execution, and from sound mechanism design through sound policy enforcement. It consists of three major interconnected thrusts.

First, we use logical frameworks for encoding and enforcing access-control policies in a practical distributed system. Access-control mechanisms today, whether it be physical keys for doors or password protection for computer accounts, reflect access-control policies that are explicit only in the manual procedures of the organization that manages these resources. As such, any change in policy, e.g., creating a new computer account, or permitting a person to unlock a door, is effected through a manual process. We utilize logical frameworks to encode organizational policies within computer systems, thereby harnessing the power of these frameworks to support the management and enforcement of access-control policy, and gaining security and flexibility by doing so. We have demonstrated this capability in a ubiquitous computing test-bed that we are developing at Carnegie Mellon called Grey [6]. In this test-bed we use "smart" mobile phones as a universal access control device, for example, to open an office door or logging into a machine when approaching it.

Second, we exploit existing technologies to mechanically reason about security policies as specified in a logical framework. This closes an important security gap, helping users and managers understand the consequences of their policies. We are particularly interested in verifying non-interference properties of policies which guarantee that some principals' assertions can have no bearing on access to a a certain resource.

Third, we are developing and implementing a framework for the specification of distributed and concurrent systems and their implementations, specifically targeting the architecture outlined in the remainder of this paper. This work extends a collaboration between NRL

and Carnegie Mellon that resulted in the design of CLF, an innovative logical language for the specification of concurrent systems. CLF incorporates ideas from logical frameworks, linear logic, and monads into an expressive meta-language.[1] CLF is now fully specified and has been successfully validated on mainstream concurrency formalisms (e.g., Petri nets, the π-calculus), advanced concurrent programming languages (Concurrent ML), and security protocol specification languages (MSR). The goal of our current research is to facilitate the transition of CLF from a foundational language into an implemented tool that can be applied to the specification of complex distributed and concurrent systems. The current prototype is called LolliMon [21].

2 A Logic-Based Approach

Distributed systems are notoriously error-prone in virtually all phases of their life-cycle, including design, implementation and management. They are particularly susceptible to security breaches; since distributed systems are more complex than their centralized counterparts, modes of attack may be difficult to foresee, and defenses are often subtle and fragile. As such, distributed system security is an area that demands rigor, and there is a well-documented history of security failures resulting from informal design, implementation and management.

The last several years have witnessed substantial progress in verification methodologies based on formal logic. Nevertheless, there remain significant gaps between verifying specifications on the one hand, and translating these specifications into verified implementations and policy enforcement on the other. We have made progress toward closing this gap, through three related but complementary research thrusts. First, we are developing and deploying a system for day-to-day use in which access-control policy is expressed in and enforced via a logical framework, permitting us to utilize the sound footing of the framework to reason about the correctness of the access-control decisions it renders. Second, we are applying tools based on LF to reasoning about formally specified security policies. Third, we are developing an extended logical framework in which we can in addition formalize our distributed enforcement mechanism.

2.1 Logical Frameworks

A *logical framework* is a meta-language for specification and implementation of logical systems. Logical frameworks have a rich history and numerous applications in programming languages, logic, and automated reasoning [29, 27]. The particular logical framework most relevant to this application is LF [18] and its implementation in Twelf [31]. One of its central characteristics is that *formal proofs* with respect to a specified set of inference rules are *first-class objects*, and that checking if a given proof is well-formed is efficiently decidable.

The use of a logical framework in such applications as proof-carrying code [26, 2] or proof-carrying authorization [5] can be sketched broadly as followed.

1. Principal A would like to convince principal B of a certain claim C, such as his right to access a resource.

2. Principal B has announced what he is willing to accept as evidence of such a claim by publishing rules of proof for establishing claims. These rules of proof embody a certain *security policy*.

[1]This prior work was supported by ONR Grants N00014-01-1-0432 and N00173-00-C-2086 – *Efficient Logics for Reasoning about Security Protocols and Their Implementations.*

3. Principal A constructs a formal proof of C, represents this proof as an object in LF, and transmits it to B.

4. Principal B checks the proof of claim C with respect to his rules (and therefore with respect to his security policy). If the proof is correct, he accepts the claim C, otherwise he rejects it.

We will explain the particular variation of this general scenario adopted for our work in the next section, but it should already be clear that it is critical that *formal proofs* are *objects*.

This methodology is well established and there are several practically efficient implementations. However, how do we know that a given set of proof rules correctly implements an intended security policy? One of the central items of our work has been to exploit the recently developed meta-reasoning facilities of the Twelf implementation [32, 33] in order to formally reason about the policies that are specified and enforced by the access-control architecture sketched in the next section. Successes in reasoning about standard classical [28] and modal logics [25] provide some hints, but access-control logics are more complex along certain dimensions and remain a significant challenge for automated tools. Initial results in this direction have been reported in [30].

Another use for logical frameworks is explained in Section 2.3.

2.2 Implementation of Access Control via Logical Frameworks

Distributed authorization systems (e.g., [9, 15, 19]) provide a way to implement and use complex security policies that are distributed across multiple hosts. The methods for distributing and assembling pieces of these security policies can be described using formal logics [20, 17]—such formalization can dramatically increase confidence in the systems' correctness. Distributed authorization systems have been built by first designing an appropriate logic and then implementing the system around it [1, 4].

Most distributed authorization systems try to provide support for notions such as the ability to delegate privileges and aggregate principals into groups. While these systems strive to be as expressive as possible—that is, to be able to represent as many security policies as possible they are constrained by how they choose to implement these ideas. The SPKI/SDSI [15] notion of delegation, for example, includes a Boolean flag that describes whether the delegated privilege may be redelegated by the recipient. PolicyMaker [9], on the other hand, requires any redelegation to be explicitly approved by the security policy. Each choice may be the best one for a particular situation, but no one particular choice can be the ideal one for *all* situations. The necessity of making these design choices limits the generality and expressivity of each such system. A system may be well suited for a particular environment but cannot scale to all plausible distributed-authorization scenarios, and systems developed for use in different environments may not be able to interoperate.

Proof-carrying authorization (PCA), a particularly promising, recent approach to distributed authorization, follows a different strategy to achieving generality [3, 8]. Unlike other systems, in which axioms that define ideas like delegation are part of the logic which describes the system, PCA is based on a standard, and completely general, higher-order logic (HOL) [13]. Higher-order logic is undecidable—there is no algorithm which will always be able to prove the truth of every true statement—which raises the question: how can such a logic can be used in an authorization system? A server is typically presented with a list of credentials and has to decide whether they are sufficient for access to be granted. If the logic that models this is undecidable, the server might not be able to come to the correct conclusion.

PCA solves this problem by making it the client's responsibility to prove that access should be granted. The server's task then becomes to verify that the client's proof is valid, which

can be done efficiently even if the proof is expressed in an undecidable logic. Transferring the burden of proof from the server to the client ensures that the server's task is tractable, but doesn't explain how the client is able to construct a proof. What makes the client's task possible is that any particular client doesn't need the full expressivity of the undecidable logic. Instead, a particular client is probably content to construct proofs in some decidable subset of higher-order logic—an application-specific subset that corresponds to a particular notion of delegation, a particular way of defining groups or roles, etc. This application-specific subset can be exposed to a client as a regular authorization logic, for example, a logic that models SPKI/SDSI. The client can manipulate this logic and construct proofs without knowledge of the underlying, more general (and more confusing) framework. The server verifying the proof, on the other hand, doesn't care which particular application-specific logic the client uses. As long as the application-specific logic is presented as a subset of higher-order logic, the server sees the client's proof as just another higher-order logic proof, which it knows how to verify.

This approach gives PCA great flexibility. Unlike traditional distributed authorization systems, a PCA-based system can be customized to describe many different sets of authorization scenarios. At the same time, because the different scenarios are expressed in the same underlying framework, all of these components can be made to inter-operate.

Logical frameworks, such as LF and CLF, are ideal tools for describing and reasoning about security logics including PCA and the higher-order logic on which it is based, and thus for providing assurance that certain security properties are achieved. However, inconsistencies between the model of an authorization system and its implementation can negate some of the conclusions of such reasoning. It is thus worthwhile to consider integrating logical frameworks directly into the implementation of a system, i.e., so that the implementation of a distributed system explicitly uses the data structures and proof-generation and proof-checking techniques of logical frameworks. An earlier proof-of-concept experiment [5, 8] suggested this to be feasible, and we are in the process of extending the use of logical frameworks in the implementation of access-control mechanisms in real systems.

Technology has evolved to the point where it is no longer necessary for access control in the physical world and access control on computers to be separate domains. The development and proliferation of high-powered and relatively inexpensive mobile computing devices (PDAs and "smart" mobile phones) and abundance of local communications options (Bluetooth and WiFi) have extended the reach of computers into everyday life. At the same time, recent advances in logic-based access-control suggest that it is possible to build practical access-control systems with nearly unlimited flexibility. The convergence of these developments makes it possible to introduce powerful new paradigms in which mobile devices take the place of both physical keys and computer passwords, eliminating the need for many separate access-control systems and introducing into the world of physical access-control drastic advances in flexibility, convenience, and security.

In particular, we envision an environment in which a Bluetooth-enabled, "smart" mobile phone will be the sole access-control token that a person will need to carry, replacing keys, smart cards, and passwords. The mobile phone will enable its bearer to enter his car, unlock his office door, and log onto his computer. To make this possible the mobile phone will generate PCA proofs that demonstrate that the bearer of the phone is authorized to access his office, computer, etc. The office door and computer logon program will contain a proof checker that will verify proofs prior to allowing access. The policy itself, which must be reflected in the proof of access, may remain distributed until the moment of proof construction, with each piece housed on a different device or provided by a different entity. In addition to the convenience of having to carry only one device, such a system greatly increases security by permitting the use of flexible, distributed, and precise policies. In practice, for example, the

policy authorizing an employee access to his office often culminates in handing to the employee a key, which he can then use as he sees fit; in our system, on the other hand, the policy could be evaluated at the time of access, perhaps permitting accesses that a physical key could not and denying others, in response to pieces of the policy that have been dynamically changed.

Such uses of mobile devices will vastly increase the importance of preventing their misuse, as is particularly of concern if the device is physically captured. Ultimately the utility of a PCA proof rests on the protection of cryptographic keys from unintended disclosure, and those stored on a mobile device are especially vulnerable to an attacker who physically reverse-engineers this device. An aspect of the system currently under development are "capture protection" services to render devices far less susceptible to misuse if captured, by utilizing a remote "capture protection server" to confirm that a device remains in its proper owner's possession before permitting its cryptographic key to be used. This can be accomplished without disclosing the device's key to the servers; while permitting the device to move the capture protection server it is utilizing as convenient; and in the face of arbitrary efforts to obtain the key from the device by reverse-engineering [23, 22].

We have deployed this access-control system for our own daily use, with a limited set of resources (e.g., our own office doors, computer logins) and users. As the system matures, we intend to broaden its use to a larger user base and a range of resources and activities on the CMU campus (e.g., purchases). Deploying such a system for everyday use involved solving or making progress on a range of issues that had not been fully addressed in such applications before, some specific to our logic-based approach and others merely exacerbated by it:

- Utilizing such a general approach for access control requires that the unintended consequences of this generality can be constrained. For example, in such a logic-based approach, authority is proved in a formal framework to which a variety of parties (users, computers) contribute "statements" (credentials). This raises the question of what unintended consequences result from participants who behave unexpectedly, e.g., by uttering contradictory statements. This issue is fundamental to the viability of this approach, but remained largely unexplored until recently [30], as sketched in Section 2.1.

- Logic-based approaches have previously been demonstrated in narrow contexts, where proof-generation strategies were neatly laid out with human assistance. This does not scale, and we further conjecture that different proof strategies may be appropriate for different environments. We have developed an approach in which the task of generating a proof is automatically distributed among the set of entities best qualified to construct it. Additionally, we are designing mechanisms that permit the proof-generation infrastructure to learn from experience, i.e., in which proof generation strategies are discovered and refined automatically, over time, and made available for use by other devices as needed. Initial results are reported in [7].

- Previous work on implementing distributed authorization systems has typically been in the context of networks of well-connected machines with plentiful computational abilities. A practical system such as the one we are deploying must explicitly account for the limitations of severely resource-constrained devices (e.g., smartphones) that communicate via a range of protocols with highly variable capacity, latency, and availability (e.g., GPRS, SMS). We have developed several strategies for coping with these difficulties, including designing a modular and lightweight communications framework well suited to such a heterogeneous environment, and intuitive and streamlined user interfaces that facilitate interaction between users and smartphones. We report on the design and describe the initial implementation of our system in [6].

- Keeping authorization credentials within a device obviously raises concerns surrounding the device's capture. We are experimenting with the aforementioned techniques for "capture resilience" in our setting, i.e., techniques that render devices largely invulnerable to capture and misuse, despite their not being physically tamper-resistant [23, 22].

2.3 Distributed Architecture Specification with a Framework Extension

With the techniques sketched in the previous section we can formally specify security policies, reason about their correctness, and enforce them in a distributed implementation. However, we cannot reason about the distributed implementation itself.

We are currently taking the first critical step towards formally reasoning about the distributed implementation by designing and implementing an extension of the logic framework LF that directly supports the specification of distributed and concurrent systems. This builds on prior research on the Concurrent Logical Framework (CLF) [34, 10]. CLF allows specifying a distributed system at a high level of abstraction. It is based on a novel combination of linear logic [16], type theory [14], and monads [24]. Our current work follows two related threads in pursuing this goal:

- We have resolved some implementation challenges of CLF, resulting in a prototype called LolliMon [21], but more remain. Specifically, the efficiency of concurrent simulation does not yet allow larger-scale experiments. Nonetheless, LolliMon has been very useful for describing implementations in CLF, as the resulting programs are too large for visual inspection, while type-checking and simulation would provide at least partial assurance.

- While the LolliMon implementation allows simulation of a distributed system, it is not suited to testing reachability. This requires a theorem prover or, in some fragments, a model checker. The initial design and implementation of a theorem prover for linear logic is described in [11, 12]; current research is aimed at making it more easily applicable to CLF.

We are in the process of formally specifying the evolving PCA architecture described in the previous section in CLF. A future direction of research would be to also formalize the meta-reasoning about the architecture itself (and not just specific policies as proposed in Section 2.1). The above items of current research are promising pre-requisites for this planned future work.

3 Conclusion

We have described a distributed security architecture based on a logical framework. The logical framework plays several roles: it serves to specify the security policy as a logical theory in an authorization logic, enforce it via checking of formal proofs, and reason about it by proof-theoretic analysis. We have realized a prototype for our framework at Carnegie Mellon University, exploiting the computational power and communication capabilities of "smart" cell phones for flexible distributed access control. Our experience so far has been positive: while the logical machinery provides a sound, uniform, and inherently extensible foundation, typical situations do not require to understand this underlying machinery for day-to-day tasks.

Acknowledgments. We gratefully acknowledge contributions by Kevin Bowers, Kaustuv Chaudhuri, Deepak Garg, Scott Garriss, Jonathan M. McCune, Jason Rouse, Peter Rutenbar, and Kevin Watkins to both the theory and implementation of the system.

This ongoing research is supported by the Office of Naval Research under grant N00014-04-1-0724: *Distributed System Security via Logical Frameworks*, the National Science Foundation grant number CNS-0433540, and the U.S. Army Research Office contract number DAAD19-02-1-0389.

References

[1] M. Abadi, E. Wobber, M. Burrows, and B. Lampson. Authentication in the Taos Operating System. In *Proceedings of the 14th ACM Symposium on Operating System Principles*, pages 256–269. ACM Press, Dec. 1993.

[2] A. Appel. Foundational proof-carrying code. In J. Halpern, editor, *Proceedings of the 16th Annual Symposium on Logic in Computer Science (LICS'01)*, pages 247–256. IEEE Computer Society Press, June 2001. Invited Talk.

[3] A. W. Appel and E. W. Felten. Proof-carrying authentication. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, Singapore, Nov. 1999.

[4] D. Balfanz, D. Dean, and M. Spreitzer. A security infrastructure for distributed Java applications. In *21th IEEE Computer Society Symposium on Research in Security and Privacy*, Oakland, CA, May 2000.

[5] L. Bauer. *Access Control for the Web via Proof-carrying Authorization*. PhD thesis, Princeton University, Nov. 2003.

[6] L. Bauer, S. Garriss, J. M. McCune, M. K. Reiter, J. Rouse, and P. Rutenbar. Device-enabled authorization in the Grey system. In *Proceedings of the 8th Information Security Conference (ISC'05)*, Sept. 2005. An extended version of this paper appears as CMU Computer Science Department Tech Report 05-111.

[7] L. Bauer, S. Garriss, and M. K. Reiter. Distributed proving in access-control systems. In *Proceedings of the 2005 IEEE Symposium on Security & Privacy*, May 2005.

[8] L. Bauer, M. A. Schneider, and E. W. Felten. A general and flexible access-control system for the web. In *Proceedings of the 11th USENIX Security Symposium*, San Francisco, CA, Aug. 2002.

[9] M. Blaze, J. Feigenbaum, and M. Strauss. Compliance checking in the PolicyMaker trust-management system. In *Proceedings of the 2nd Financial Crypto Conference*, volume 1465 of *Lecture Notes in Computer Science*, Berlin, 1998. Springer.

[10] I. Cervesato, F. Pfenning, D. Walker, and K. Watkins. A concurrent logical framework II: Examples and applications. Technical Report CMU-CS-02-102, Department of Computer Science, Carnegie Mellon University, 2002. Revised May 2003.

[11] K. Chaudhuri and F. Pfenning. A focusing inverse method prover for first-order linear logic. In R.Nieuwenhuis, editor, *Proceedings of the 20th International Conference on Automated Deduction (CADE-20)*, pages 69–83, Tallinn, Estonia, July 2005. Springer Verlag LNCS 3632.

[12] K. Chaudhuri and F. Pfenning. Focusing the inverse method for linear logic. In L.Ong, editor, *Proceedings of the 14th Annual Conference on Computer Science Logic (CSL'05)*, pages 200–215, Oxford, England, Aug. 2005. Springer Verlag LNCS 3634.

[13] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56 68, 1940.

[14] R. L. Constable et al. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[15] C. M. Ellison, B. Frantz, B. Lampson, R. L. Rivest, B. M. Thomas, and T. Ylonen. *SPKI Certificate Theory*, Sept. 1999. RFC2693.

[16] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[17] J. Y. Halpern and R. van der Meyden. A logic for SDSI's linked local name spaces. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pages 111–122, Mordano, Italy, June 1999.

[18] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, Jan. 1993.

[19] R. Housley, W. Polk, W. Ford, and D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, Apr. 2002. RFC3280.

[20] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: Theory and practice. *ACM Trans. Comp. Sys.*, 10(4):265 310, Nov. 1992.

[21] P. López, F. Pfenning, J. Polakow, and K. Watkins. Monadic concurrent linear logic programming. In A.Felty, editor, *Proceedings of the 7th International Symposium on Principles and Practice of Declarative Programming (PPDP'05)*, pages 35–46, Lisbon, Portugal, July 2005. ACM Press.

[22] P. MacKenzie and M. K. Reiter. Delegation of cryptographic servers for capture-resilient devices. *Distributed Computing*, 16(4):307 327, December 2003.

[23] P. MacKenzie and M. K. Reiter. Networked cryptographic devices resilient to capture. *International Journal of Information Security*, 2(1):1–20, November 2003.

[24] E. Moggi. Notions of computation and monads. *Information and Computation*, 93(1):55–92, 1991.

[25] T. Murphy VII, K. Crary, R. Harper, and F. Pfenning. A symmetric modal lambda calculus for distributed computing. Technical Report CMU-CS-04-105, Carnegie Mellon University, Feb. 2004.

[26] G. C. Necula. Proof-carrying code. In N. D. Jones, editor, *Conference Record of the 24th Symposium on Principles of Programming Languages (POPL'97)*, pages 106–119, Paris, France, Jan. 1997. ACM Press.

[27] F. Pfenning. The practice of logical frameworks. In H. Kirchner, editor, *Proceedings of the Colloquium on Trees in Algebra and Programming*, pages 119 134, Linköping, Sweden, Apr. 1996. Springer-Verlag LNCS 1059. Invited talk.

[28] F. Pfenning. Structural cut elimination I. intuitionistic and classical logic. *Information and Computation*, 157(1/2):84–141, Mar. 2000.

[29] F. Pfenning. Logical frameworks. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, chapter 17, pages 1063–1147. Elsevier Science and MIT Press, 2001.

[30] F. Pfenning. Constructive authorization logics. 4th Workshop on Foundations of Computer Security (FCS'05), Chicago, Illinois, July 2005. Invited Talk.

[31] F. Pfenning and C. Schürmann. System description: Twelf — a meta-logical framework for deductive systems. In H. Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction (CADE-16)*, pages 202–206, Trento, Italy, July 1999. Springer-Verlag LNAI 1632.

[32] C. Schürmann. *Automating the Meta Theory of Deductive Systems*. PhD thesis, Department of Computer Science, Carnegie Mellon University, Aug. 2000. Available as Technical Report CMU-CS-00-146.

[33] C. Schürmann and F. Pfenning. A coverage checking algorithm for LF. In D. Basin and B. Wolff, editors, *Proceedings of the 16th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2003)*, pages 120–135, Rome, Italy, Sept. 2003. Springer-Verlag LNCS 2758.

[34] K. Watkins, I. Cervesato, F. Pfenning, and D. Walker. A concurrent logical framework I: Judgments and properties. Technical Report CMU-CS-02-101, Department of Computer Science, Carnegie Mellon University, 2002. Revised May 2003.

Device-Enabled Authorization in the Grey System*

Lujo Bauer, Scott Garriss, Jonathan M. McCune,
Michael K. Reiter, Jason Rouse, and Peter Rutenbar

Carnegie Mellon University, Pittsburgh, Pennsylvania, USA

Abstract. We describe the design of Grey, a set of software extensions that convert an off-the-shelf smartphone-class device into a tool by which its owner exercises and delegates her authority to both physical and virtual resources. We focus on the software components and user interfaces of Grey, highlighting the features of each. We also discuss an initial case study for Grey, in which we are equipping over 65 doors on two floors of office space for access control using Grey-enabled devices, for a population of roughly 150 persons. Further details of Grey, and this and other applications, can be found in a companion technical report.

1 Introduction

Access control today is characterized by an expanse of mechanisms that do not interoperate and that are highly inflexible. Access to physical resources (e.g., home, office) is most commonly tied to the possession of a hardware key, and in office environments possibly a swipe card or RFID card. By contrast, access to virtual resources is typically tied to the knowledge of a password and/or possession of a physical token (e.g., SecureID) for producing time-varying passwords.

In this paper we introduce the Grey system, which utilizes converged mobile devices, or "smartphones", as the technology of choice for unifying access control to both physical and virtual resources. We focus on smartphones for two central reasons. First, their nearly ubiquitous adoption is inevitable, as in the long term they stand to inherit the vast cellular phone market, which in 2004 shipped over 648 million units [30]. Second, the hardware capabilities of smartphones and the maturity of application programming environments for them have advanced to a stage that enables applications to take full advantage of rich computation, communication, and interface capabilities (e.g., a camera).

This convergence of market trends and technological advances points to a future marked by pervasive adoption of highly capable and always-in-hand smartphones. Grey is an effort to use this platform to build a ubiquitous access-control technology spanning both physical and virtual resources. This vision is not ours alone: several groups have experimented with the use of mobile phones as digital

* We gratefully acknowledge support from the National Science Foundation grant number CNS-0433540, the Office of Naval Research grant number N00014-04-1-0724, and the U.S. Army Research Office contract number DAAD19-02-1-0389.

keys [9, 26]; NTT Docomo is conducting trials on the use of mobile phones to authorize entry to apartments*; and mobile phones can already be used to purchase items from vending machines in several countries. However, to the extent that we can infer the capabilities of these systems, we believe that Grey presents a more sound and flexible platform for building a ubiquitous access-control system and, eventually, for experimenting with advanced mobile applications.

As an example of the type of flexibility not possible in other solutions, with Grey a user will be able to easily create and lend to her friend a temporary, virtual key to her car or apartment; this will happen seamlessly regardless of whether the user and her friend are standing next to each other or thousands of miles apart. Similarly, a manager could give to her secretary temporary access to her email without revealing any information (e.g., passwords) that could be used at a later time or to access a different resource. Going further, a user could specify that his office may be accessed by any three of his colleagues acting together, but at least three would have to cooperate to gain access.

Grey is a novel integration of several technologies that results in a single tool for exercising and delegating authority that we believe is far more secure, flexible and usable than any alternative available today. At the core of Grey is a flexible and provably sound authorization framework based on *proof-carrying authorization* (PCA) [3], extended with a new distributed proving technique that offers significant efficiency advances [7]. In addition to enabling a user to exercise her authority, PCA provides a framework in which users can delegate authority in a convenient fashion. For protection of phone-resident cryptographic keys in the event of phone capture, Grey incorporates *capture resilience* [22], which renders a lost or stolen phone resistant to misuse. And, on the user-interface front, we employ a technique for conveying key material and network addresses, that is as simple as taking a picture with the phone's built-in camera [23, 29]. Phone-to-phone and phone-to-infrastructure data communication utilizes an asynchronous messaging layer that we have developed to take advantage of the myriad networking technologies available to modern smartphones, including Bluetooth, cellular data service (e.g., GPRS), and messaging protocols (e.g., SMS and MMS).

In this paper we describe the adaptation of these components into a practical access-control system called Grey. At the time of this writing, we are deploying Grey to create a platform for future research on practical smartphone-based access-control systems. Our initial deployment on two floors of a new building on our university campus will involve roughly 150 users and consist of two applications: (1) controlling access to 65 offices by Grey-enabled phones; (2) using Grey for accessing Windows XP sessions. In these applications, Grey offers a more secure, flexible and convenient basis for access control than existing solutions.

Due to space limitations, we were forced to omit the descriptions of several important aspects of Grey. For more detail, including a thorough discussion of related work, a more comprehensive description of the software architecture, more extensive performance results, and a description of the Grey Windows XP login plugin, please see our companion technical report [6].

* `http://www.i4u.com/article960.html`

2 Component Technologies

Grey is a novel integration of a number of recently-developed technologies that utilize the capabilities of modern smartphones; we summarize these component technologies here.

2.1 Graphical Identifiers

A common feature of modern smartphones is a camera. In Grey we utilize this camera as a data input device for the smartphone, e.g., by asking the user to take a picture of an item she intends to interact with. Information conveyed by photographing two-dimensional barcodes is a theme common to several ubiquitous computing efforts (e.g., [13, 28]), typically to convey service information or a URL where such information can be obtained. In Grey, there are two types of identifiers that are commonly input via the camera:

An identifier for a public key. A useful identifier for a key is the collision-resistant hash of the key (e.g., [20]). In Grey, a two-dimensional barcode is used to encode the hash of a public key and can be displayed on a sticker attached to an item (e.g., on a door) or, for a device with a display (e.g., smartphone or computer), presented on the display. A camera-equipped smartphone can then photograph this identifier and authenticate the public key obtained by other means (e.g., over a wireless link) [23]. This provides a natural and user-friendly way for obtaining an authentic public key.

A network address. A barcode can also be used to encode a network address. As above, a camera-equipped smartphone can then obtain the network address by photographing the barcode. This idea has been utilized to circumvent high-latency device discovery in Bluetooth [29], and we use it in this way in Grey. In addition, this idea offers similar usability advantages to that above, as it is an intuitive operation for a user to photograph the device with which she intends to communicate.

The pervasiveness of graphical identifiers in Grey lends itself well to graphical management interfaces for collecting identifiers and managing access. We will provide an overview of the interfaces we have developed in Section 4.

2.2 Capture-Resilient Cryptography

A user's Grey-enabled smartphone utilizes a private signature key in the course of exercising the user's authority. The capture of a smartphone thus risks permitting an attacker who reverse-engineers the smartphone to utilize this private key and, as a result, the user's authority. To defend against this threat, Grey *capture protects* the phone's private key [22]. At a high level, capture protection utilizes a remote *capture-protection server* to confirm that the device is being held by the person who initialized the device (e.g., using a PIN, face recognition via the phone's camera, or other biometric if the phone supports it), before it

permits the key on the phone to be used. This server can also disable the use of the key permanently when informed that the device has been lost, or temporarily to protect the key from an online dictionary attack on the PIN (or other authentication technique). At the same time, this capture-protection server is untrusted in that it gains no information about the user's key.

In keeping with the theme that Grey is a wholly decentralized system, the capture-protection server is not a centralized resource. That is, each user can utilize her own capture-protection server (e.g., her desktop computer), and indeed there is no management required of this server in the sense of establishing user accounts. Rather, this server need only have a public key that is made available to the user's phone when the phone's key is created—perhaps by taking a picture of it displayed on the server's screen, as described in Section 2.1—and must to be reachable when the phone needs to utilize its private key.

A concern that arises with the use of a phone for exercising personal authority is the sheer inconvenience of losing one's phone, in the sense of being unable to exercise one's own authority. While this can occur with any form of access control that utilizes a token or other hardware, we note that capture protection provides a remedy. Since the capture-protection server ensures that a key can be used only by a device in possession of the person present when the key was created, a user may back up her key with little risk of exposing it in an indefensible way.

2.3 Proof-Carrying Authorization

Prior research in distributed authorization has produced a number of systems [27, 16, 15, 10] that provide ways to implement and use complex security policies that are distributed across multiple entities. Gaining access to a resource typically involves locating and gathering credentials and verifying that a set of credentials satisfies some access-control policy. Both the gathering and the verification is typically carried out by the entity or host that is trying to decide whether to allow access.

These credentials and the algorithms for deciding whether a set of credentials satisfies some security policy can be described using formal logics (e.g., [1, 18]). In early work in this vein, the design of access-control systems starts with the specification of a security logic, after which a system is built that implements as exactly as possible the abstractions and algorithms that the logic describes [31, 5]. While this approach can dramatically increase confidence in the systems' correctness [2], at best the system *emulates* the access-control ideal as captured in the formal logic. That is, since the correspondence between the formal logic and the implementation is only informal, any guarantees derived from the formal logic might fail to extend to the implemented system.

An alternative introduced in the concept of *proof-carrying authorization* (PCA) [3, 8] is to utilize this formal logic directly in the implementation of the system. In PCA the system directly manipulates fragments of logic that represent credentials; the proofs of access are likewise constructed directly in formal logic. This integration of formal logic into the implemented system provides increased assurance that the system will behave as expected. This is the high-level

approach that we adopt in Grey. As such, each Grey component (including a smartphone) includes an automated theorem prover for generating proofs in the logic, and a checker for verifying proofs.

A fundamental tension in access control is that the more expressive a system is (that is, the greater the range of security policies that its credentials can describe), the more difficult it becomes to make access-control decisions. To ensure that the access-control decision can always be made, most systems restrict the range of security policies that can be expressed, ruling out many potentially useful policies. Since Grey is meant to be used in a highly heterogeneous environment and supports ad-hoc creation of policy components, this type of inflexibility could be very limiting. An insight behind PCA is that the access-control policy concerning any particular client is likely to be far simpler to reason about than the sum of all the policies of all clients. PCA takes advantage of this insight by making it the client's responsibility to prove that access should be granted. To gain access, a client must provide the server with a logical proof that access should be allowed; the server must only verify that the proof is valid, which is a much simpler task. The common language in which proofs are expressed is a higher-order logic [11]; when constructing proofs, each client uses only a tractable subset of the higher-order logic that fits its own needs. The mechanism for verifying proofs is lightweight, which increases confidence in its correctness [4] and also enables even computationally impoverished devices to be protected by Grey.

3 A Usage Scenario

Grey's integration of the technologies described in Section 2 (and others) enables a range of interactions that enhance access control to render it more user friendly, decentralized and flexible. To illustrate this, we describe an example scenario that utilizes several of the pieces we have introduced.

The scenario we consider begins with two researchers, Alice and Bob, who meet at a conference and begin a research collaboration. Anticipating communicating electronically when they return to their home institutions, each enters the other in his/her smartphone "address book". To populate her address book entry for Bob, Alice needs merely to snap a picture of the two-dimensional barcode displayed on Bob's phone. The barcode encodes both the Bluetooth address of Bob's phone, enabling Alice's phone to connect to it, and a hash of Bob's public key, which can be used to authenticate the full key that is transferred via Bluetooth along with Bob's contact information. After Alice returns to her home institution, her phone automatically synchronizes its address book with her PC. This could permit her, for example, to authenticate electronic mail from Bob using standard protocols (e.g., [25]).

As their submission deadline approaches, Alice and Bob decide to meet in person, and so Bob makes plans to visit Alice. On the day that Bob arrives at Alice's institution, Alice is delayed at home. Bob thus arrives to Alice's locked office door. Inside the glass next to Alice's door is a barcode sticker that encodes

the Bluetooth address of a computer that can actuate Alice's door to open, if convinced to do so. Bob photographs the barcode, prompting his smartphone to connect to the computer, which challenges Bob's phone to prove his rights to access the door—a feat which his phone cannot do alone, since Bob lacks the needed credentials. The theorem prover in his phone, however, discerns that Alice's phone could assist, and initiates a communication with it.

Upon receiving Bob's phone's request, the theorem prover in Alice's phone automatically generates several options by which Alice can permit Bob to enter the door, based on credentials that she has previously created and that are stored in the phone: she can (i) simply grant him a credential to open the door only this time; (ii) add him to a group `visitors` that she previously created and granted rights to, among other things, open her door; or (iii) give him the rights of her secretary, to whom she also granted the ability to open her door. Alice's phone presents this list to Alice, who selects (ii). The phone then signs a credential to this effect and returns it to Bob's phone, enabling it to complete the proof of access.

Fig. 1. Bob entering Alice's office. In the course of proving access, Bob's phone contacts Alice's phone for help.

It is worthwhile to reflect on the presentation of this process to each of Alice and Bob. Bob, upon photographing the door barcode, is asked to enter a PIN in order to utilize his private key to sign a request to open the door—an operation protected by capture protection; see Section 2.2—and the door opens with no further interaction (albeit with some waiting while Alice makes her decision). Alice is consulted merely with a list offering her several options by which she can permit Bob to enter her office. Upon selecting one and also typing her PIN—again to activate her capture-protected key—her task is completed.

Bob's credential indicating that he is a member of Alice's `visitors` group turns out to be handy while he awaits Alice's arrival. In addition to permitting him to open Alice's office, it could grant his laptop access to the campus 802.11 network, to the floor printer, and to a back room where there is a vending machine with snacks and sodas. All these privileges are afforded to Bob due to Alice's prior creation of credentials that grant these privileges to her `visitors`.

4 Software Architecture

At a high level of abstraction, every Grey host or device is composed of some subset of the following elements: a compact and trustworthy *verifier* that mediates access to a protected resource; an extensible *prover* that attempts to construct proofs of access; a lightweight, asynchronous *communication framework* that facilitates the distributed construction of proofs and management of certificates (for details please see our companion technical report [6]); and a collection of

graphical interfaces that allows the convenient and seamless integration of Grey into everyday life. Grey is implemented in Java, which allows it to easily extend across multiple platforms (workstations, smartphones, embedded PCs, etc.) and operating systems.

4.1 Graphical User Interfaces

An emphasis in Grey is usability. In this subsection we describe the primary user interfaces involved in Grey at the time of this writing.

In order to maximize our user population, we have targeted Grey for the widest range of smartphones possible, including those of modest size—and correspondingly modest screen size. For example, our primary development platform to date has been the Nokia 6620, a smartphone with dimensions $4.28 \times 2.29 \times 0.93$ inches and a 176×208 pixel display. Due to the limited screen size on this class of smartphones, we have divided tasks into those performed on the phone by necessity, and those that can be offloaded to a companion tool run on a personal computer, after which the necessary state can be transferred to the phone via a synchronization operation. At a high level, tasks such as the creation of groups and roles (as defined in [20]), and proactive policy creation, are offloaded to the companion tool. Because these tasks are standard in a variety of access-control settings, here we focus on the phone-resident interfaces, as these are the ones that we believe to be more innovative.

The tasks performed on the smartphone with user interaction include: collecting identifiers (of persons, keys, or addresses); making an access request to a resource; and reactive policy creation, i.e., responding to a request for a credential to permit another person to complete an access proof.

Address book The first of these tasks, building an address book of identifiers and bindings among them, is performed using the camera and the keypad of the phone. As described in Section 2.1, the identifiers that can be input via the camera include pictures of public keys (and of network addresses, but these are not involved in address-book creation). The keypad permits the input of text strings. The address-book interface enables the creation of speaks-for relationships between names and keys: a user photographs the key and then either selects an already-present identifier for which the key speaks or inputs the identifier at that time. After a user photographs the two-dimensional barcode encoding a key, the key is permanently hidden from the her. While user-friendly representations of keys using "snowflakes" [17, 21], flags [14] or random art [24] have been proposed, we believe that exposing keys in the interface is unnecessary and potentially confusing.

Requesting access to a resource A user requesting access to a resource for the first time must obtain the network address of the computer that controls access to that resource. Collecting this network address can presently be done in two ways: either with Bluetooth discovery or, as discussed in Section 2.1, using the

phone's camera to photograph a two-dimensional barcode encoding the Bluetooth address (Figure 2). The latter technique is more reliable, since Bluetooth discovery can net multiple devices, and selecting the proper device is a user choice that is vulnerable to misinterpretation or the user being misled. Once the network address for a resource is captured, it is kept in a resource menu on the phone. A single click on a resource in this menu initiates an attempt to connect to the corresponding computer and start the sequence to access the resource (see Figure 3).

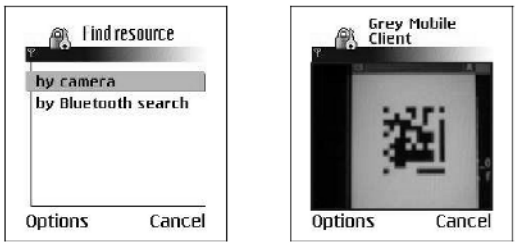


Fig. 2. Bob learns the Bluetooth address of Alice's door by taking a picture of the two-dimensional barcode visible near Alice's door.

Perhaps the most innovative aspect of this part of the user interface is its use of learned patterns of resource accesses. Most users exhibit a pattern of accesses; e.g., a typical workday begins with the user opening a building door, then a door on the floor on which she works, then her office door, and finally logging into her desktop computer. If all these resources are accessed using Grey, the user's smartphone will learn the temporal proximity and order of these accesses as a pattern, and can offer this pattern as an option when the user initiates the first access in the pattern (e.g., `Work_Garage` to `HH_D202_PC` in Figure 3 is such a pattern). If the user selects the pattern, the phone will attempt to connect to and access each of the resources in sequence, with each step contingent on the previous access in the pattern succeeding. In this way, merely two clicks and a PIN entry as the user approaches her building will enable her to reach her office and will log her into her desktop.

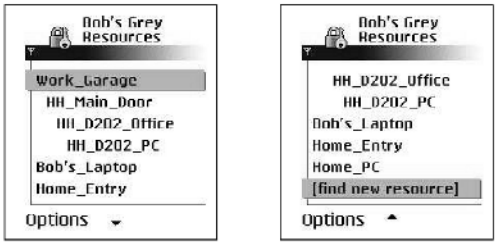


Fig. 3. Resource list on Bob's phone.

Reactive policy creation The third type of interface presented by the phone to the user permits the reactive creation of policy. This interface is launched by the prover in the user's smartphone after the prover has generated a list of credentials to which the user could consent to enable an access that is being attempted by another person. For example, in the usage scenario of Section 3, this is the interface by which Alice adds Bob to her `visitors` group by selecting this option from the menu generated by the prover (see Section 4.2).

Because this interface interrupts the user (unlike the other interfaces, which are user driven), it is important that the user can apply access control to this step and silence these interrupts at times she prefers to not be interrupted. For the former (access control), we employ the same access-control infrastructure that we use for other resources, utilizing a default, but user-configurable, policy that

permits only those in the phone's address book to request assistance. The latter, i.e., silencing all such requests, is a simple toggle, and, once activated, received requests will be silently queued for the user to handle later. The party requesting credentials from her will be informed that a response is not forthcoming, and will not be able to access the requested resource (or at least not with her help). However, if she later consents to the request, the appropriate credential will still be sent to the requester for use in the future.

4.2 Prover

As described in the example in Section 3, after arriving at Alice's office, Bob instructs his phone to unlock the door. The door's first reply contains a *challenge*—a statement, in logic, of the theorem that Bob's phone must prove before the door will unlock. The challenge that typically needs to be proved is that the door's owner believes that it is OK for access to be granted. In this case, expressed in logic, the challenge is *Alice* **says goal**(`A-111`), i.e., Bob must prove that Alice believes that it is OK to access her office, A-111.**

The straightforward way for Bob to answer the door's challenge is to scour the network for useful credentials and then attempt to form them into a proof; most distributed authorization systems use a close facsimile of this approach. There are some inherent problems, however, with this method of constructing a proof. Bob might guess, for example, that Alice has credentials that he could use, but he does not know exactly which of the credentials that she possesses will be helpful for this particular proof. It would be inefficient for Alice to send Bob *all* her credentials, since she might have hundreds. Moreover, sending all her credentials to Bob would reveal exactly the extent of Alice's authority, which is unlikely to meet with Alice's approval. Finally, there may be cases, such as in our example, when the credential that Bob needs has not yet been created; in these situations a simple search, no matter how thorough, would fail to yield sufficient credentials for Bob to access Alice's office.

An answer to these problems can be found in *distributed proving*—a scheme in which Bob's phone does not just search for individual credentials, but also solicits help in proving simpler subproofs that he can assemble into a proof of the challenge [7]. Using this approach, Bob's phone might ask Alice's phone to prove a theorem like *Bob* **says goal**(...) → *Alice* **says goal**(...). Alice's phone now has the opportunity to decide which of her credentials to use or which new credentials to create in order to prove this theorem; these credentials will be returned to Bob's phone along with the proof. This scheme of farming out subproofs to other entities spans two extremes: *eager* proving, in which a client farms out a theorem only if he is completely unable to make progress on it himself; and *lazy* proving, in which the client asks for help as soon as he isolates a theorem that

** In order to enforce the timeliness of Bob's response and to protect against replay attacks, the logical statement that must be proved also contains a nonce. This and other low-level details that are not novel are described elsewhere; we omit them from this paper in order to focus on the more abstract ideas.

someone else might be able to help with. Distributed proving can be combined with several optimizations, including caching of credentials and subproofs and deriving proof strategies based on the shape of previously encountered proofs [7].

The use of distributed proving in Grey and the details of constructing proofs in general are largely out of the view of the user. Bob's phone processes the door's challenge until it arrives at a potentially useful subtheorem; at that point, the phone consults the address book to determine how Alice can be reached (by phone or by URL, for example). Since Bob might have to pay for the communication (typically, some combination of SMS and GPRS connectivity is needed, and use of either may incur some cost) and to prevent other users from being unintentionally disturbed, Bob's phone prompts Bob to approve the help request. Alice may need reminding or convincing before she will be willing to help, and so Bob is given the option of annotating his request for a subproof with a recorded or text message.

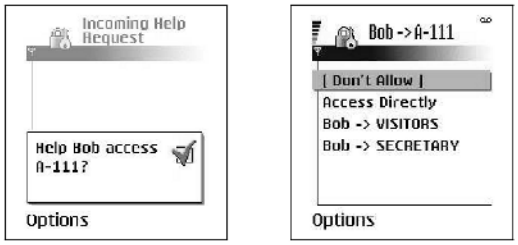


Fig. 4. Alice is given the opportunity to chose the type of credential to grant to Bob.

Upon receiving Bob's request, Alice's phone first verifies that Alice is in fact willing to help Bob (Figure 4). If Alice agrees, her phone begins to compute the subproof, which can in many cases be done without further input from Alice. Sometimes, however, construction of the subproof will require Alice to generate a new credential. In these cases, Alice is shown a list of the credentials that can be used to complete the subproof. Alice can either choose the credential she wishes to create, or decide that none of them are appropriate. When Alice makes her selection, her smartphone finishes constructing the subproof and sends it to Bob. Bob's phone incorporates Alice's subproof into the main proof and sends the proof to the door.

Although a single help request is sufficient for our example with Alice and Bob, Bob's phone may in general need to request subproofs from several other users; in addition, each of those users may in turn also need to solicit help. Through a combination of optimizations derived from observing both successful and unsuccessful past behaviors, a user's Grey smartphone can guide proof search to minimize the number of times help is requested. If multiple avenues can lead to constructing a proof, the ones most likely to be successful and quick will be the ones pursued first [7].

Figure 5 depicts the structure of the Grey application that runs on Bob's phone. The entire application is implemented in Java Micro Edition (J2ME), the restricted flavor of Java that runs on many smartphones. The process of generating proofs is managed by different components depending on whether Bob is trying to access a resource himself (ProofTalker) or help another user (HelpTalker). In addition to directing a Prolog engine to traverse the space of possible proofs, these components manage communication with the resource Bob

is trying to access and with other users via the communication framework. They also create and manage credentials using the Crypto module.

Grey makes use of a rich set of standard extensions to the core J2ME APIs to enable use of Bluetooth and other communications protocols (JSR-82 and JSR-120) and the phone's camera (JSR-135). In addition, we use the Bouncy-Castle libraries*** to implement the higher-level Grey cryptographic primitives.

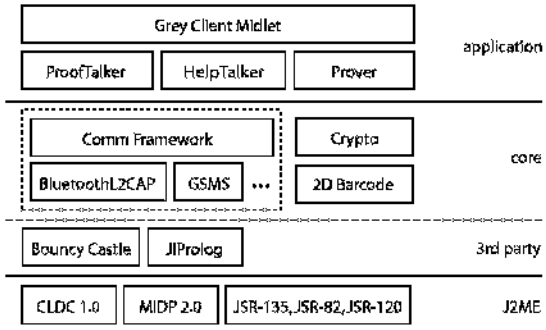


Fig. 5. The structure of the Grey application that runs on smartphones.

4.3 Verifier

One of the goals of Grey is to encompass many diverse resources that a user might wish to access. Some of these resources, such as doors and computer logins, we traditionally associate with the need for access control. Others, like thermostats, are not normally thought of the same way. However, with the ability to actuate such resources remotely, via the network or via a smartphone, also comes the need to regulate access. For example, Alice may want to adjust her office temperature before she arrives at work, but she most likely does not want passers-by to do the same.

To enable Grey to conveniently apply to a wide range of devices, it was necessary for its verification module—the component that mediates access to resources—to be simple, relatively lightweight, and device independent. At the same time, we wanted to maintain a high level of assurance that access is not granted improperly. The proof-carrying authorization paradigm fits our needs well; in PCA, access to a resource is allowed if the client presents a proof that he is authorized to use it. The verification of such proofs is a straightforward mechanical process, with none of the complexity and potential intractability of generating proofs. This distinction is fortunate, since the verifier is in the trusted computing base, while proof generation is not. Moreover, the verification process itself is independent of the security policy protecting the resource, and so also of the resource's type (e.g., door, login).

Figure 6 shows the components and control flow of the verification module, which are described in more detail in the following paragraphs. The process of gaining access to a resource is initiated by a user request. In response to the request, a *challenge* is generated. The challenge is the statement, in formal logic, of the

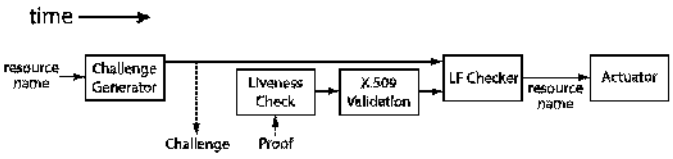


Fig. 6. Flow of the verification process.

*** http://www.bouncycastle.org

theorem whose proof a potential user must provide. As described in Section 2.3, the challenge is specified in higher-order logic; this in turn is encoded in LF, the notation of one of the most widely used frameworks for specifying logics [19].

When Bob attempts to access Alice's office, the verification module generates a challenge that includes the name of the resource, A-111, and a nonce. This challenge is sent to Bob, but also recorded for use in later stages of verification.

Bob's eventual reply to the challenge will contain a set of credentials (e.g., Bob is a member of `visitors`), and a proof, in formal logic, that the credentials satisfy the door's challenge. The first step of verifying the proof is to ensure (using the nonce) that it was created within a brief period after the door issued the challenge. Next, the credentials, which are X.509v3 certificates with customized extensions, are verified: their digital signatures and expiration times are checked. Finally, the formal proof is passed to an LF type checker, which ensures that the structure of the proof is valid (e.g., that it contains no false implications) and that the correct theorem (the one that was issued as the challenge) was proved. This algorithm is widely studied and well understood, providing high assurance that an invalid proof will never be accepted [12, 4]. If this proof is successfully verified, the LF checker signals an actuator to open the door.

Figure 7 shows the structure of the Grey application that controls access to a door. Similarly to the prover application described in Section 4.3, this application is constructed in a modular fashion: the only customization necessary was the front end (DoorTalker) that encapsulates these modules and the actuator module (StrikeController) that sends commands specific to the relay controller we use.

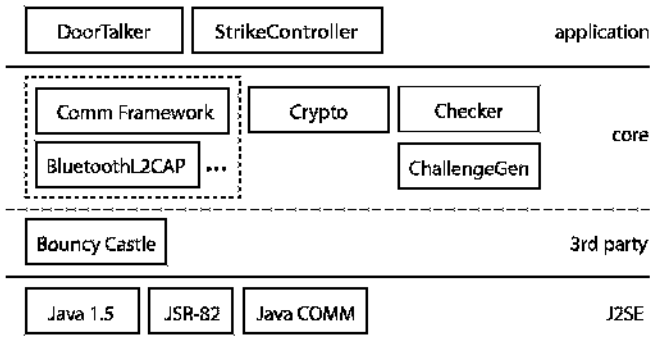


Fig. 7. The structure of the Java application that allows office doors to be Grey-enabled.

The required physical infrastructure for Grey-enabling a door is relatively minimal: a standard electric door strike actuated by an embedded PC located in the wall near each door. Our prototype embedded PC measures $4.55 \times 3.75 \times 1.70$ inches—small enough to fit *within* each door, an option we seriously considered. It is equipped with a Bluetooth adapter and an RS-485 relay controller, and to improve reliability has no moving parts (i.e., cooling is passive, and flash memory is used for non-volatile storage). The prototype embedded PC uses a commodity Pentium M on a PC-104+ mainboard; for a wide deployment of Grey a significantly more compact, custom embedded system could be designed.

Enabling a door with Grey does not preclude legacy access technologies (e.g., keys, proximity cards) from being used; Grey merely provides a parallel way to unlock the door. Of course, Grey can also be used as the sole method of controlling access.

4.4 Performance on Smartphones

In this section we provide performance measurements for certain tasks in Grey. Our primary interest is measuring delays as experienced by the user to access a resource in the common case. We report such numbers here, and additionally measure costs associated with underlying operations to shed light on the sources of these delays.

Our first macrobenchmark is the time required to open a door. The computer controlling the door lock was an embedded PC with a 1.4GHz Pentium M processor; more detail on this pilot application is given in our companion technical report [6]. Each timing was measured starting when the user selected the door from the resource list on her phone (a Nokia 6620), and ended when the door unlocked. On average, this delay was 5.36 seconds excluding any user interaction (more on this below), with an variance of 0.33 due to background work on the phone. The second macrobenchmark is the time required for a user to log into a 2GHz Windows XP workstation using Grey [6]. The methodology in this experiment was similar to that for the door. This delay averaged to 9.31 seconds, with a variance of 2.20. The bulk of the extra time was taken up by the load time for `explorer.exe` and desktop preparation.

We emphasize that these are common-case numbers in three senses. First, neither of these tests involved a remote help request. Help requests can take significantly longer (e.g., a minute), and vary depending on cellular network conditions and user responsiveness. Second, these measurements did not involve the use of a capture-resilient signing key on the phone, and as such the signing operation by the phone did not involve user input (i.e., a PIN) or interaction with a capture-protection server. In our present implementation, we have adopted a design by which the user can configure the frequency with which she is prompted for her PIN (and the capture-protection server is contacted), rather than being prompted per resource access. Her capture-resilient key is then used at these intervals to create a short-lived certificate for a non-capture-resilient public key (a step which does require PIN entry) that is used to sign access requests. As such, the common case incurs only the latency of a signature with this non-capture-resilient key. Third, the network address for each of the computers regulating access was already stored in the resource list of the phone and so, e.g., the one-time barcode-processing overhead incurred if it is first captured via the camera (roughly 1.5 sec.) is not reflected in these numbers.

Typical latencies of under six seconds to open a door and roughly nine seconds to complete a computer login are already comparable to the latencies of more traditional access control (e.g., physical keys and passwords). However, we emphasize that Grey permits these latencies to be hidden from the user more effectively than alternatives. Our current systems utilize class 2 Bluetooth devices, meaning that, e.g., a smartphone could initiate an access once it is within 10 meters of the resource (the door or computer). By the time the user reaches the resource in order to make use of it, the access typically would have completed. In our own experience with using the system, access is consequently far quicker than with the alternatives that Grey replaces for us.

5 Conclusion and Status

Smartphones offer a number of features that make them attractive as a basis for pervasive-computing applications, not the least of which is their impending ubiquity. Grey is an effort to leverage these devices beyond the games, personal information management, and basic communication (voice, email) for which they are primarily used today. We believe, in particular, that these devices can form the basis of a sound access-control infrastructure offering both usability and unparalleled flexibility in policy creation.

Grey is a collection of software extensions to commodity mobile phones that forms the basis for such an infrastructure. At the core of Grey is the novel integration of several new advances in areas ranging from device technologies (e.g., cameras) and applications thereof, to theorem proving in the context of access-control logics. This integration yields, we believe, a compelling and usable tool for performing device-enabled access control to both physical and virtual resources.

Grey is being deployed to control access to the physical space on two floors of a building recently constructed on our university campus. Construction of this building was completed in June 2005, and Grey is being phased into the building on an opt-in basis. This deployment will serve as a platform for continued research on usability, credential management, theorem proving and other technologies in the function of access control.

References

1. M. Abadi. On SDSI's linked local name spaces. *J. Computer Security*, 1998.
2. M. Abadi, M. Burrows, B. Lampson, and G. D. Plotkin. A calculus for access control in distributed systems. *ACM Trans. Prog. Lang. and Sys.*, Sept. 1993.
3. A. W. Appel and E. W. Felten. Proof-carrying authentication. In *Proc. 6th ACM Conference on Computer and Communications Security*, Nov. 1999.
4. A. W. Appel, N. Michael, A. Stump, and R. Virga. A trustworthy proof checker. *J. Automated Reasoning*, 31(3-4):231–260, 2003.
5. D. Balfanz, D. Dean, and M. Spreitzer. A security infrastructure for distributed Java applications. In *Proc. 21st IEEE Symposium on Security and Privacy*, 2002.
6. L. Bauer, S. Garriss, J. M. McCune, M. K. Reiter, J. Rouse, and P. Rutenbar. Device-enabled authorization in the Grey system. Technical Report CMU-CS-05-111, Computer Science Department, Carnegie Mellon University, Feb. 2005.
7. L. Bauer, S. Garriss, and M. K. Reiter. Distributed proving in access-control systems. In *Proc. 2005 IEEE Symposium on Security and Privacy*, May 2005.
8. L. Bauer, M. A. Schneider, and E. W. Felten. A general and flexible access-control system for the Web. In *Proc. 11th USENIX Security Symposium*, Aug. 2002.
9. A. Beaufour and P. Bonnet. Personal servers as digital keys. In *Proc. 2nd IEEE International Conference of Pervasive Computing and Communications*, Mar. 2004.
10. M. Blaze, J. Feigenbaum, and M. Strauss. Compliance checking in the PolicyMaker trust-management system. In *Proc. 2nd Financial Crypto Conference*, 1998.
11. A. Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 1940.
12. T. Coquand. An algorithm for testing conversion in type theory. In G. Huet and G. Plotkin, editors, *Logical Frameworks*, pages 255–280. 1991.

13. D. L. de Ipiña, P. Mendonça, and A. Hopper. TRIP: a low-cost vision-based location system for ubiquitous computing. *Pers. and Ubiq. Comp.*, 6(3), 2002.
14. S. Dohrmann and C. Ellison. Public key support for collaborative groups. In *Proc. First Annual PKI Research Workshop*, Apr. 2002.
15. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI certificate theory. RFC 2693, Sept. 1999.
16. C. M. Ellison, B. Frantz, B. Lampson, and R. Rivest. Simple public key certificate. Internet Engineering Task Force Draft, July 1997.
17. I. Goldberg. Visual key fingerprint code. Available at `http://www.cs.berkeley.edu/iang/visprint.c`, 1996.
18. J. Y. Halpern and R. van der Meyden. A logic for SDSI's linked local name spaces. In *Proc. 12th IEEE Computer Security Foundations Workshop*, 1999.
19. R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *J. ACM*, 40(1):143–184, Jan. 1993.
20. B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: Theory and practice. *ACM Trans. Comp. Sys.*, 10(4):265–310, Nov. 1992.
21. R. Levin. PGP snowflake. Personal communication, 1996.
22. P. MacKenzie and M. K. Reiter. Networked cryptographic devices resilient to capture. *International Journal of Information Security*, 2(1):1 20, Nov. 2003.
23. J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *Proc. 2005 IEEE Symposium on Security and Privacy*, May 2005.
24. A. Perrig and D. Song. Hash visualization: A new technique to improve real-world security. In *Proc. 1999 Intern. Work. Crypto. Techn. and E-Comm.*, July 1999.
25. B. Ramsdell. Secure/multipurpose internet mail extensions (S/MIME) version 3.1: Message specification. RFC 3850, July 2004.
26. N. Ravi, P. Stern, N. Desai, and L. Iftode. Accessing ubiquitous services using smart phones. In *Proc. 3rd Intern. Conf. Pervasive Comp. and Comm.*, 2005.
27. R. L. Rivest and B. Lampson. SDSI—A simple distributed security infrastructure. Presented at CRYPTO '96 Rumpsession, Apr. 1996.
28. M. Rohs and B. Gfeller. Using camera-equipped mobile phones for interacting with real-world objects. *Advances in Pervasive Computing*, pages 265–271, Apr. 2004.
29. D. Scott, R. Sharp, A. Madhavapeddy, and E. Upton. Using visual tags to bypass Bluetooth device discovery. *Mobile Comp. and Comm. Review*, 1(2), Jan. 2005.
30. A. Slawsby and A. Leibovitch. Worldwide mobile phone 2004–2008 forecast update and 1H04 vendor shares, Dec. 2004. `http://www.idc.com/getdoc.jsp?containerId=32336`.
31. E. Wobber, M. Abadi, M. Burrows, and B. Lampson. Authentication in the Taos operating system. *ACM Trans. Comp. Sys.*, 12(1):3–32, Feb. 1994.

Distributed Immune Systems for Wireless Networks Information Assurance

John S. Baras
Institute for Systems Research
University of Maryland College Park

Introduction

The theme of the research program summarized in this paper, is the development of innovative distributed methods and algorithms for network security and information assurance that are designed to work well in the demanding wireless mobile communications environment. When possible we have tried to take advantage of the special nature of wireless networks to improve assurance and security, while keeping the disadvantages of wireless to a minimum. Our goal is to *design 'robust' information assurance systems*, i.e. systems capable of maintaining some degree of assurance even under high levels of noise and node capture or destruction. The research program under our CIP URI was organized around three interrelated thrusts:

(1) Distributed Autonomous Immune Systems
(2) Assurance Via Distributed Physical Layer Signal Processing and Routing
(3) Distributed Computing Formalisms and Systems

Our research on methods, algorithms, modeling and analytical methods is supported by: Mobile wireless network simulation testbeds; Real experimentation with mobile wireless network testbeds

This integration is achieved by innovative ideas and schemes that focus on the following principles: Distributed automatic classification of intrusions in real-time; Automatic generation of responses for containing and nullifying an intrusion faster than it spreads; Attacking intrusions close to the 'network edge'; Utilization of synergy between physical layer and network layer assurance schemes; Hierarchical methods and schemes in both the physical and logical domain for efficiency and scalability. Furthermore we have adopted a "systems view" of security and information assurance; that is security and information assurance belongs to network management and control.

The major motivation for our methods and ideas comes from: the operational principles of biological immune systems; recent successful development of 'digital immune systems' for the protection of commercial networks from virus attacks; recent advances in complex waveform generation which can be profitably utilized to secure wireless communications in a variety of yet unexplored ways.

The research summarized here was performed by seven faculty investigators from diverse areas of engineering, computer science and mathematics (J.S. Baras, C. A. Berenstein, A. Ephremides, V. Gligor, K.J.R. Liu, H. Papadopoulos, N. Roussopoulos, M.Wu), twenty three graduate research assistants, two postdoctoral fellows and one research engineer, in the period 2001 to 2004. The project created many new interactions between the participants and cross-fertilized the group on issues of wireless security and information assurance.

1. Distributed Autonomous Immune Systems

1.1 Finite Automata Models for Anomaly Detection

A fundamental problem in intrusion detection is the fusion of dependent information sequences. We investigated the fusion of two such sequences, namely the sequences of system calls and the values of the instruction pointer. We introduced FAAD, a finite automaton representation defined for the product alphabet of the two sequences where dependencies are implicitly taken into account by a matching procedure. Our learning algorithm captures these dependencies through the application of certain parameterized functions. Through the choice of thresholds and inner product structures, we were able to produce a compact representation of the normal behavior of a program.

Intrusion detection methods can be divided into two categories: misuse detection and anomaly detection. Anomaly detection is based on an approximate representation of normal behavior of the system. A behavior which significantly deviates from this normal representation is flagged as an intrusion. We developed a new technique for program-based anomaly detection. Forrest and her collaborators have shown that the sequence of system calls can be used to represent the normal behavior of a program. Learning this behavior is accomplished by the use of the "N-grams" which are strings of length N observed during the normal use of the program. This approach is inspired by the intrinsic ability of biological immune systems to distinguish between "self" (the organism) and pathogens.

Finite Automata (FA) or Finite State Machines (FSM) have been suggested as an alternative to N-grams. Once a finite automaton is constructed, combinations of all allowed transitions can generalize it to strings not used in the learning phase but "similar" to the ones used. We developed a novel algorithm, proved its properties and evaluated its performance. The rate of false positives is non-increasing with training for FAAD. An attractive feature of FAAD is that training can continue after its use for intrusion detection begins. We performed evaluation of our algorithm with various attacks. Our new algorithm performed very well in all tests we did.

1.2 Intrusion Detection with Support Vector Machines and Generative Models

We investigated the development of algorithms for intrusion detection using Support Vector Machines (SVM). Based on the formal models we developed intrusion detection was formulated as the problem of detection and classification of symbolic strings (attack tree model), given small amounts of data. Our main objective in this research was to combine use of formal models (like attack trees) and parallel clustering algorithms of the Support Vector Machine (SVM) type, to develop fast distributed algorithms for identification of components of intrusions and abnormal behaviors. Our approach and methods are somewhat inspired from similar successful combination of formal (logic) models and SVM in genomics and proteomics, and in particular for the problem of partial identification using side logical information. These are the so-called "generative models based SVM" schemes. They have excellent learning properties for unknown patterns and generalization capabilities.

We addressed the problem of detecting intrusions in the form of malicious programs on a host computer system by inspecting the trace of system calls made by these programs. We used 'attack-tree' type generative models for such intrusions to select features that are used by a Support Vector Machine Classifier. Our approach combines the ability of an HMM generative model to handle variable-length strings, i.e. the traces, and the non asymptotic nature of Support Vector Machines that permits them to work well with small training sets. The central problem is to decide whether a computer program that runs on a single host computer system is a *normal* program that does not compromise the security of the host or a *malicious* program that is a threat to the host. To decide on the nature of a program, we examine the string (trace) of system calls that it makes and operate under the assumption that this trace contains all the information we need to make this decision. A relatively large number of training examples especially of programs that are malicious is hard to come by. Therefore we seek to use non-parametric discriminative classifiers such as Support Vector Machines that, of all learning techniques are designed to work well with small training sets.

Real attacks have a finite (and not too long) underlying *attack* sequence of system calls because they target specific vulnerabilities of the host. This and the padding are represented in a 'plan of attack' called the *Attack Tree*. The basic attack scheme encoded in the Attack Tree is not changed by modifications such as altering the padding scheme or the amount of padding (time spent in the padding nodes). Given an attack tree, it is straightforward to find the list of all traces that it can generate. Our intrusion detection executes the following steps: (1) Learn about A from the training set T; (2) Form a rule to determine the likelihood of a given trace being generated by A . These objectives can be met by a probabilistic modeling of the Attack Tree. We used parametric HMMs to derive a real valued feature vector of fixed dimension for these variable length strings that enables us to use Support Vector Machines for classification.

1.3 Detection and Classification of Network Intrusions Using Hidden Markov Models

We developed and evaluated algorithms for detection and classification of intrusions using Hidden Markov Models (HMM). We demonstrated that it is possible to model attacks with a low number of states and classify them using Hidden Markov Models with very low False Alarm rate and very few False Negatives. Hidden Markov Model training was performed on normal and anomalous sequences (traces). We tested several algorithms, applied different rules for classification and evaluated the relative performance of these. Several of the attack examples presented exploit buffer overflow vulnerabilities, due to availability of data for such attacks. We emphasize that the purpose of our algorithms is not only the detection and classification of buffer overflows, but they are targeted for detecting and classifying a broad range of attacks.

Among the attacks we studied were: *eject*, *ps*, *ffbconfig* and *fdformat* and all of them belong to the class of User to Root exploits. User to Root exploits belong to the class of attacks where the attacker gains access to a normal user account on the system and by exploiting some vulnerability obtains the root access. Certain regularities were captured in behavior of exploited programs by comparing them against normal instances of those programs, other instances of attacks detected at different periods of time and by searching

for certain events in the behavior of a program that were expected to happen by using the knowledge about the mechanism of those attacks and their goals. The studied examples show that each attack is characterized with a very simple distinguishing sequence of system calls and accompanying parameters (like size, PID, path, etc.), which can be used for recognition and identification of different attacks.

In the framework of intrusion detection the problem can be formulated as follows: *given M attack models in the form of Hidden Markov Models with known parameters, detect the one that matches the incoming sequence with the highest probability.* However, in the case of detecting an attack the incoming sequence may or may not match one of the HMM models of attack. In case it does not match one of the attack models we need to consider two cases: either the incoming sequence is not an attack or it is an unknown attack that we don't have in our database. The problem of M-ary detection was solved by calculating log-likelihoods for each of the possible models given the observed sequence and finding the maximum. One of our significant contributions is reflected in the area of classification. For known attacks our approach is based on training on anomalous sequences. Training is performed on a couple of files, each of length 100 system calls. Testing is performed on around 2-4 % of the total number of sequences of the initial data set. The strength of this approach is that it chooses only potential attacks in both the training and testing sets. Another advantage of this approach is that the attacker cannot change the behavior of normal over time by slowly adding more and more abnormal sequences since we are using anomalous sequences for training. This algorithm has been used successfully for detection of already known attacks.

1.4 On-line Adaptive IDS Scheme for Detection of Unknown Network Attacks Using Probabilistic Models and Logic

The main focus was to design a scheme that can incorporate both misuse and anomaly detection and hence be used to detect known network attacks (instances of which might not have been seen before), but more importantly, unknown network attacks. Since misuse detection introduces false negatives and anomaly detection introduces false positives, we need to be able to find a good trade-off. The idea is to set a desirable detection rate (which, in our case was 100%), and then minimize the false positive rate by filtering false positives through stages.

It is important to emphasize that this scheme's goal is to get as good results as possible with *limited* information. This means that we do not know signatures of all the attacks. If we knew that, we could just use signature detection. By incorporating probabilistic models and the administrator's knowledge about possible vulnerabilities, we can achieve very optimistic results. There are five stages in our scheme: Initialization, Parallel testing and training, Logic, Verification and Adaptive phase.

The process is as follows: Partition the probabilistic space into normal behavior, known attacks and (everything else is) unknown attacks. This is done through offsetting log-likelihoods of each model space. In the Parallel testing and training phase, we do trace detection and classification (normal, known attack, unknown attack) and if the classified sequence is not normal we go to the Logic phase (note that in this phase we also train

new HMM with the incoming sequence for possible future use – in the Verification phase). In the Logic phase, we use the administrator's knowledge databases containing possibly malicious events - sequences of (in our case) system calls. We scan the trace for those events (sequentially!). In case there are none, the decision is made that the trace is normal, so the HMM model of the trace (created in the previous phase) is forwarded to the Adaptive phase. In case there is a malicious event (or several events, depending of how many of them are needed to raise an alarm), the execution goes to the next, Verification phase. This phase does probabilistic testing (analog to the probabilistic testing in the beginning). Since all the attacks we used in our simulations belong to the same group of attacks (Buffer Overflow attacks), this represents the worst-case scenario for the scheme, since the attacks tend to look alike. With 100% detection rate, we were also able to achieve a very good false positive rate – 0.08%.

1.5 On-Line Distributed Detection of Self-Propagating Code

Worms are programs that self-propagate across a network by exploiting security flaws in widely-used services offered by vulnerable computers in the network. Worms are popular attacks because no other mechanism allows for the rapid and widespread distribution of malicious code, with virtually no way to trace the attacker. It has been stated that the spread of the theoretical flash or Warhol worms will be so fast that no human-driven communication will suffice for adequate identification of an outbreak before nearly complete infection is achieved. The appearance of such a worm was voted the greatest security threat. There is therefore great need to develop automated mechanisms for detecting worms based on their traffic patterns. In our work we completed the development and evaluation of such algorithms. In our research we focused on the fact that the self propagating code will try to use specific vulnerabilities that can be identified with certain port numbers. So we used as the traffic monitoring variable the connection attempts (probes) to a given TCP/UDP port number(s). We also assumed most of the times a probability distribution on the traffic observations. So in our framework we assume that there is a baseline of connections to the given monitored port in all sensors (computers) of the network. The observations can be made at different participating nodes enforcing policies for blocking self-propagating code once it is detected. We explored the effect of aggregation from distributed sensors. This approach is motivated by the current infrastructure of distributed Intrusion Detection Systems such as myNetwatchman, Dshield and Symantec's DeepSight Threat Management System.

We developed a novel formulation of these problems using change detection as the foundation of our approach. We developed methods that are valid without the standard i.i.d. assumption on the observations after the change, which is not true because each infected host will try in general to scan the same number of hosts in a given interval of time, and as more and more hosts become infected the observation data volume will increase fast with time. We have developed, implemented, simulated and evaluated a variety of methods using our framework. These include detection of a change in the mean, change detection in distributed sensor systems, CUSUM of aggregated traffic, exponential signal detection in noise, exponential change in the mean, nonparametric regression detection (which allows situations where the number of probes seen exhibits long range dependence and multifractal behavior), and new fully nonparametric

algorithms in order to deal with some of the more complicated problems, in particular those where no clear mean can be established. We developed algorithms based on the sequential probability ratio test (SPRT), where the goal is to optimize a hypothesis testing problem given a trade-off between the probability of errors and the observation time. We also formulated these problems as quickest change detection problems, where the trade-off is between the delay of detection and the false alarm rate. The methodologies we used to analyze these problems proceed along two main ideas: developing generalized likelihood ratio (GLR) algorithms for on-line algorithms; developing filter bank algorithms (using HMMs). We also investigated the development of robust non-parametric algorithms using cumulative sum (CUSUM) and Girshik-Rubin-Shiryaev (GRSh) statistics. In sequential versions of the problem the sequential probability ratio test (SPRT) was used.

We performed extensive analytic and experimental (based on synthetic networks and attack data) performance evaluation of the various schemes we developed. Our evaluation results seem to strongly suggest that in scale-free networks a very small set of the highly connected nodes is sufficient for detection and aggregation only improves the performance of the nonparametric statistics. If we select sensors at random or if we monitor a random network then aggregation is very important for detection. We also developed and evaluated collaborative distributed algorithms for these worm detection problems.

1.6 On-Line Distributed Detection of Distributed Denial of Service Attacks

A denial of service attack (DoS) can be defined as an attack designed to disrupt or completely deny legitimate users' access to networks, servers, services or other resources. The most common DoS attack involves sending a large number of packets to a destination causing excessive amounts of network endpoint bandwidth to be consumed and (or) cpu processing rate at the destination. In a distributed denial of service (DDoS) typically an attacker compromises a set of Internet hosts (using manual or semiautomated methods like a worm) and installs a small attack daemon on each host, producing a group of "zombies". There are various techniques and ideas for mitigation of denial of service attacks that require the identification of the routers participating (involuntarily) in the attack. Most of these techniques consume a significant amount of router resources so it is advisable to use them only when needed. A reasonable assumption for transit networks carrying a lot of traffic which cannot be analyzed at line rate, is that routers do not keep the number of packets to a specific destination, as this might be too expensive during operation. Thus we are interested only in monitoring passively the network.

We completed a novel formulation and approach to the problem of detecting when a distributed denial of service is taking place in one sub-network of a transit (core) network comprised only on routers. We assumed the transit network itself is not the target of the attack, but it is being used by the attack to reach the victim. We developed a novel formulation of the problem as sequential space-time change detection on a graph. The mathematical techniques we use for detecting an attack are thus based on change detection theory. In a distributed environment a small change in local nodes can be correlated with the state at different nodes to provide a global view and early warning

about the state of the network. We developed and applied parametric and nonparametric change detection algorithms to the problem of detecting changes in the "direction" of traffic flow. We investigated also the quickest detection problem when the attack is distributed and coordinated from several nodes against a targeted one. We developed and used a "directionality framework", which gives us a way to compute the severity and directionality of the change.

One of the main advantages in having several nodes under monitoring is that we can perform a correlation of the statistics between the different nodes in order to decrease the detection delay given a fixed false alarm rate probability. The alarm correlation can be performed by several methods. We developed and evaluated a simple algorithm that only requires the knowledge of the routing tables for the nodes being monitored. Selecting which statistics to correlate (add) is a key issue. Our algorithm not only can detect the attack (depending on the new correlation threshold), but also it can diminish the impact of the false alarm originating at some node. However another important conclusion is that without the need to extract or store header information from the packets transmitted through the network, we are able to infer (from the intersection of the two routing tables for the "winning" correlated statistic of the links) the "best" possible targets (estimated).

1.7 Detection of Attacks Against the MAC Protocol in Wireless Networks

Selfish behavior at the MAC layer can have devastating side effects on the performance of wireless networks, similar to the effects of DoS attacks. One of the most challenging detection tasks is that of detecting backoff manipulation. Due to the randomness introduced in the choice of the backoff, it is difficult to detect when a node has chosen small backoff values by chance or not. In our work we focused on prevention and detection of the manipulation of the backoff mechanism of 802.11's MAC protocol, although our approach can be extended to any probabilistic distributed MAC protocol.

Our main contributions are the introduction of two algorithms. The first one is an algorithm that prevents cheating in the backoff stage of 802.11 DCF for noncolluding nodes. The protocol is called ERA-802.11 for Ensuring Randomness in 802.11. ERA-802.11 appends a couple of bits to the RTC/CTS reservation mechanism of 802.11. In it, the sender commits to a random backoff. Once the receiver obtains this commitment it sends back an honest backoff value. Finally when the sender receives the backoff of the receiver it replies with the information required to open the commitment. Once the receiver checks the correctness of the committed value, the final backoff value, agreed among sender and receiver, is the *XOR* of the backoffs selected by both of them. This algorithm is based on Blum's flipping coins over the telephone protocol and it ensures honest backoffs when at least one, either the receiver or the sender is honest.

The second algorithm is a misbehavior detection procedure to deal with the problem of colluding selfish nodes (a pair of misbehaving sender and receivers can override ERA-802.11's prevention mechanism). We first explored the problems with previous solutions attempting to detect backoff manipulation. We showed how intelligent selfish nodes can overrride previous detection schemes while still providing high throughput by swithcing

between a large backoff and a small backoff. Finally we presented a new algorithm that detects several attempts of intelligent nodes to gain more access to the medium.

Although we have focused on the MAC layer of 802.11, our approach is general and can serve as a guideline for the design of any probabilistic distributed MAC protocol.

1.8 On-Line Detection of Routing Attacks in MANETs

Mobile -wireless- ad hoc networks (MANETS) are particularly vulnerable to attacks on their routing protocols. Unlike fixed networks, the routers usually do not reside in physically protected places and can fall under the control of an attacker more easily. Such an attacker can then send incorrect routing information. Furthermore messages can be eaves dropped and faked messages can be injected into the network without the need to compromise nodes. General attacks are misrouting, false message propagation, packet dropping, packet generation with faked source address, corruption on packet contents and denial-of-service.

One of the attacks exploiting the wireless medium is the wormhole attack. The wormhole attack can be devastating to a routing protocol. We developed a formulation and a novel approach for the detection of such attacks. Our approach builds a model capturing the dynamics of a highly mobile ad hoc network. The basic idea is that an attacker will change the routing information in such a way that our perceived mobility of the nodes will differ from our previous experience. We want to learn the allowable state transitions (which depend in our sampling interval.) We performed various simulation experiments which validated this promise. We used as the observation variable the hop count distribution at a given node. For simplicity we assumed a proactive distance vector routing protocol such as DSDV in order to have all hop counts at any time. In the change detection setup we used a CUSUM procedure applicable to the case of dependent observations.

We performed analytical and simulation evaluations of the performance of the new algorithm. Although the attacks introduced by very different and easy means, the principle of detecting an unknown attack to the routing protocol with different characteristics was demonstrated. In particular some attacks produced a change in the variance of the hop count distribution, while others produced a change in the mean of the hop count distribution. Both attacks were detected by simply testing the likelihood of our learned model.

1.9 Software Systems for Attack Detection and Defense in MANET

We have investigated a highly extensible intrusion detection system to determine its utility in solving problems of identifying previously unidentified attacks, with special interest in its application in wireless ad hoc networks. The STAT system (developed by Richard Kemmerer and his group at the University of California Santa Barbara) is a state-based detection system: each attack is mapped into a set of states called an attack scenario. Certain behaviors trigger transitions between states - these transitions represent either the progression of a possible attack or the recognition and quelling of a false alarm.

When a series of behaviors cause the final state to be reached, an attack is said to have occurred. The power of this approach lies in the identification of only the essential elements of attacks - hence if the goals of the attackers are known, it should be possible to construct attack scenarios abstract enough to capture new methods of attaining those same goals.

We extended STAT and STATL, and implemented several and tested several of our intrusion detection algorithms in STAT: buffer overflow, timing disruption, sequence falsification, wormhole, routing misbehavior and others. We are setting up a wireless testbed to analyze extensively feasibility and performance of STAT in wireless ad hoc networks by identifying energy requirements and adaptability to a dynamic attack environment.

2. Assurance Via Distributed Physical Layer Signal Processing and Routing

2.1 Physical Layer Secrecy over Wireless Channels via Chaotic CDMA

Our main objective has been to design chaotic CDMA systems that provide uncoded Pr(e) advantages to intended users in the context of multiuser communication over fading channels. The systems we have considered and optimized exploit linear modulation of a digital information-bearing signal on a chaotic sequence, *i.e.*, a sequence generated by iterating an initial condition through a chaotic mapping. The Pr(e) advantages offered to intended users are achieved by providing side information to these users in the form of the initial condition. These systems are attractive alternatives to conventional CDMA systems, *i.e.*, systems that exploit modulation on binary-valued pseudonoise (PN) spreading sequences generated by feedback shift-register structures. Indeed, chaotic CDMA systems can provide additional Pr(e) performance advantages to intended users by exploiting the inherent sensitivity to initial conditions of chaotic systems, with minimal increase in transmitter and intended receiver complexity and without the need for additional side information with respect to what is required by conventional CDMA systems.

We have designed tools for characterizing the differences in attainable performance between intended and unintended users in single-user settings (corresponding to only one transmitting user), as a function of processing gain and SNR for a large class of PC maps. In particular, we have determined the performance characteristics of DS/SS schemes with signatures generated by various families of chaotic piecewise-linear maps, in the context of signaling over AWGN and frequency nonselective fading channels and have recently started exploring the multiuser setting. As our investigative efforts have revealed, even in the single-user setting, these systems can be designed to provide secrecy benefits to intended receivers in the form of uncoded Pr(e) performance advantages. In particular, chaotic spreading can provide substantial improvement in terms of the Pr(e) advantages offered to intended users with respect to conventional DS/SS systems that make the PN sequence seed available only to intended receivers.

We developed optimized digital implementations of the underlying chaotic DS/SS as well as quantifying the extent to which these implementations preserve the important properties of the underlying chaotic DS/SS of interest. We have shown that by properly choosing the precision depth in the implementation, the pseudochaotic DS/SS systems we developed can achieve the performance characteristics of the underlying chaotic DS/SS over an arbitrarily wide (yet finite) range of channel SNR values. As a result we were able to show that 16-bit precision depths suffice to provide effectively private communication over a very wide SNR range (that includes the SNR range of practical settings) even for processing gains well below those used in practical systems. We also considered the privacy provided by chaotic DS/CDMA, i.e., the multiuser extensions of DS/SS systems.

2.2 Distributed Coding-Based Protocols for Private Computation with Intrusion Detection over Wireless Channels

We designed distributed algorithms for networks of nodes/sensors that wish to compute functions of their data with privacy, while maintaining the ability to detect intrusions with high probability. In particular, we considered multinode settings, whereby the nodes wish to effectively use resources, such as bandwidth and transmit and processing power, to compute a function of their individual data over a common wireless channel – making the desired function output, in the process, available to an arbitrary subset of the participating nodes – while achieving the following objectives:

(i) no additional information is revealed by the protocol about each participant's individual data, other than what is made available through the result of the desired computation;

(ii) intruders, actively participating in the computation in an effort to alter its end result, can be detected by means of the protocol with high probability.

We focused our efforts on a driving example involving source localization (estimation of the location of a target) by fusing noisy target range information available at spatially dispersed sensor nodes. In a typical setting, each sensor node may possess measurements which can be used to derive such information about the relative range between the target and that particular sensor. In this area, we are leveraging our recent findings of distributed algorithms that can be used to compute functions of the node data in a wireless network by using distributed locally constructed fusion rules at each node.

2.3 Communication-Friendly Encryption of Multimedia

We investigated means of protecting the confidentiality and achieving access control of multimedia information, which is one of the crucial security elements for many applications. More specifically we researched efficient and effective encryption of multimedia with a focus on communication and compression issues. We identified a set of domains along the representation and communication process of multimedia where encryption can be applied, and proposed three encryption operations through elegant combinations of multimedia signal processing and contemporary cryptography.

By moving the encryption domain from the bit stream to upper levels and therefore preserving standard compliance, more sophisticated intermediate processing can be applied directly on the encrypted data. Under such a framework, we proposed an encryption tool via a generalized index mapping, which can be applied to any scalar or vector symbols with a finite value range. The compression overhead of this scheme can be adjusted and confined to a moderate amount. The three fundamental schemes we developed can be used as building blocks and combined to form an encryption system for multimedia data. Our designs of these proposed encryption operations take into consideration the inherent structure and the underlying syntax of multimedia sources to achieve improved friendliness to communications, compression, and computation.

2.4 Topology-Aware Key Management Schemes for Wireless Multicast

Technological advancements have created the potential for many new applications that will allow users to simultaneously share content and collaborate. The most relevant enabling network technology for group communication is multicast. The problem of access control has received extensive attention in the recent literature and many solutions for the generic problem have been proposed. However, the traditional literature does not address network-specific issues.

In tree-based multicast key management schemes, most rekeying messages are only useful to a subset of users, who are always neighbors on the key management tree. This observation motivates us to design a key tree that matches the network topology in such a way that the neighbors on the key tree correspond to the topology of the wireless LAN, which consists of mobile users and access points. This key tree design proceeds in two steps:

Step 1: Design a subtree for the users connecting to each access point (AP). These subtrees are called *user subtrees.*

Step 2: Design a subtree which governs the key hierarchy between the APs and the key distribution center (KDC). This subtree shall be called the *AP subtree.*

By delivering the rekeying messages only to the users who need them, we may take advantage of the fact that the key tree matches the network topology, and localize the delivery of rekeying messages to small regions of the network.

2.5 Secure and Cost-Efficient Contributory Group Key Agreement Protocols

In contributory key agreements, every group member makes its own contribution independently when establishing group keys, and each member's personal key is not disclosed to any other entities. Compared with centralized key management schemes, the contributory key agreement schemes also have the advantages that they do not rely on centralized servers and secure communication channels. In our research we investigated methods for reducing the cost associated to key updates in contributory group key agreement protocols. We developed TCGK, a suite of cost-efficient Tree-based Contributory Group Key agreement protocols for secure group communication with dynamic membership changes. We designed a novel logical key tree structure, based on which the rekeying cost per user join or leave event can be dramatically reduced. To our

best knowledge, TCGK has the lowest cost among the existing tree-based contributory key agreement schemes, and achieves better scalability. The simulation results have also confirmed the superiority of TCGK to the existing schemes in term of cost savings.

In secure group communications, the time cost associated with key updates for member join and departure is an important aspect of quality of service, especially in large groups with dynamic membership. In time-sensitive applications, a timely key update during member join or departure assures that secure group communications can be established in a timely manner. We developed a new scheme called Join-Exit Tree (JET) Group Key Agreement. Our analytical results show that our proposed scheme achieves an average asymptotic time of O(log (log n)) for a join event, and also O(log (log n)) for a departure event when group dynamics are known a priori. We have extensively studied the performance of our scheme under different user activity scenarios, including sequential user join, the MBone (Multicast Backbone) multicast session data, and a probabilistic user behavior model. In all these scenarios, our proposed scheme has outperformed the existing schemes in terms of rekeying time complexity. In addition to the improved time efficiency, our scheme also has low communication and computation complexity.

2.6 Attacks and Protection of Dynamic Membership Information in Secure Group Communications

In secure group communications, key management is employed to prevent unauthorized access to multicast content. We discovered that the rekeying process associated with multicast key management can disclose information about the dynamics of the group membership to both insiders and outsiders. We collectively refer to group dynamics information (GDI) as the number of users in the multicast group as a function of time, and the number of users who join or leave the service during a time interval. The leakage of GDI from the rekeying process can lead to serious security and privacy problems. For centralized key management schemes, we have developed two effective strategies to steal the GDI. These strategies involve:

(1) obtaining membership dynamics from the format of rekeying messages;

(2) estimating the number of users, $N(t)$, from the size of rekeying messages.

Many popular centralized key management schemes are vulnerable to these attacks. Our simulations show that these passive-attack strategies result in accurate estimation of the GDI.

To protect the GDI, we developed an anti-attack technique utilizing batch rekeying and phantom users. The combined effects of the phantom users and the real users lead to a new rekeying process, called the *observed rekeying process*, which would be monitored by the attackers. The goal is to produce an observed rekeying process that reveals the least amount of information about the real GDI. We derived performance criteria that describe the security level of the proposed scheme using mutual information. The proposed anti-attack scheme is evaluated based on the data obtained from real MBone sessions. We also developed the analysis of the vulnerability of various contributory key management schemes and investigated techniques that can be used to protect dynamic group membership information in distributed environments.

2.7 Key Management Schemes for Distributed Sensor Networks

Distributed sensor networks (DSN) are of central importance to military operations. Our interest in this work is very large distributed sensor networks using inexpensive sensors. We have developed innovative key management schemes for such networks. This addresses an important information assurance problem for such wireless sensor networks. These very large sensor networks have significant differences from more conventional sensor networks. First, in scale, we are interested in size of 10,000 nodes as opposed to 100. Second, they have dynamic topology. Third, due to the method of deployment, like deployment by scattering no prior knowledge of sensor-node location can be assumed. Fourth they should be able to accommodate incremental addition / deletion of nodes after deployment. Fifth and most significant, they face hostile environments of operation, where they must operate unattended, and are subject to sensor nodes monitoring, capture and manipulation. Physical capture and tampering by adversary is possible, which requires tamper-detection technology, disable sensor and erase keys, detection of data inputs alteration, detection of input manipulation via data correlation.

From the perspective of key management these constraints imply that key exchange/distribution via third party is not possible: unknown network topology, intermittent operations, network scale and dynamics. Key pre-distribution is the only viable solution (to date). We have developed and analyzed a new scheme based on a *probabilistic key sharing* approach. Each node has been given k keys from a pool of P keys. If two nodes share a common key then a link exists between them. These secure links provide an overlay secure network. This overlay network has to be connected. Our new basic scheme consists of the following three steps: (1) Key pre-distribution; (2) Shared-key discovery; (3)Path-key establishment. We have analyzed this scheme and developed analytically its performance evaluation.

2.8 Attacks and Defenses Utilizing Cross-Layer Interactions in MANET

Cross-layer protocol design is one of the prevailing methodologies that have recently been adopted in networking research and leads to significant performance benefits. We assessed the performance of cross-layer interaction and investigated its effects with regard to security and information assurance of mobile ad hoc wireless networks. Using attacks in realistic wireless networks as a prototype, we found that natural cross-layer interactions between physical, MAC and network layer protocols in MANET can turn out to be a weak point, causing various attacks and intrusions. However, by allowing a controlled synergy between the affected layers, we facilitate timely detection of such attacks that are otherwise difficult to detect and may have devastating effects on network functionality and operation.

We demonstrated that natural interactions between physical layer and MAC, as well as MAC and routing protocols in MANET can lead to a variety of attacks and intrusions. We showed that without purposeful collaboration between the layers affected by such attacks, they are very difficult to detect while at the same time can have catastrophic effects on the MANET functionality and operation. To illustrate the impact of MAC layer

attacks we first described the effects of a dishonest user in the MAC layer to the performance of the network and later we concentrated on malicious users. For the majority of the work we focused on attacks involving interactions between the MAC and routing protocols and described detection and defense mechanisms we have developed for such attacks. We described several DoS attacks in realistic MANET that explicitly exploit cross-layer interactions. We used the realistic scenario, where each node initially employs legal communication patterns that prevent other nodes from communicating and after some time they start misbehaving in order to maintain priority in the network.

We used IEEE 802.11 MAC layer and by using several different scenarios we showed that attacks that originate in the MAC layer easily propagate to the routing layer causing breaking of existing routes. We also showed that attack propagation can cause not only breaking of selected routes, but can also be used to include the attackers in the new routes. We showed that the attack with colluding attackers is more powerful than the attacks using only single attacker or multiple non-colluding attackers. We proved using a game-theoretic approach that the scenario in which each attacker attempts to maximize his own gain results in minimal gain for each of the attackers.

2.9 Key and Node Revocation in Distributed Sensor Networks

Sensor network security poses a unique challenge due to the large numbers of sensor nodes involved and the limitations of sensor node hardware. A variety of techniques to bootstrap security in sensor networks have been developed using key pre-distribution techniques based on our original scheme. However, the problem of key and node revocation in sensor networks has received relatively little attention. Distributed revocation protocols pose new design challenges since these protocols need to account for the presence of active adversaries pretending to be legitimate protocol participants via compromised sensor nodes. Revocation protocols that function correctly in such environments are essential to secure sensor network operation. In the absence of such protocols, an adversary could effectively take control of the sensor network's operation by using compromised nodes which retain their network connectivity for extended periods of time. In our research, we defined a set of basic properties that distributed sensor-node revocation protocols must satisfy, and presented a protocol for distributed node revocation that satisfies these properties under general assumptions and a standard attacker model.

The low-cost, off-the-shelf hardware components in unshielded sensor-network nodes leave them vulnerable to compromise. With little effort, an adversary may capture nodes, analyze and replicate them, and surreptitiously insert these replicas at strategic locations within the network. Such attacks may have severe consequences; they may allow the adversary to corrupt network data or even disconnect significant parts of the network. Pervious node replication detection schemes depend primarily on centralized mechanisms with single points of failure, or on neighborhood voting protocols that fail to detect distributed replications. To address these fundamental limitations, we proposed two new algorithms based on emergent properties, i.e., properties that arise only through the collective action of multiple nodes. Randomized Multicast distributes node location information to randomly-selected witnesses, exploiting the birthday paradox to detect

replicated nodes, while Line-Selected Multicast uses the topology of the network to detect replication. Both algorithms provide globally-aware, distributed node-replica detection, and Line-Selected Multicast displays particularly strong performance characteristics.

2.10 Covert Channel Attacks on MANET Routing and MAC Protocols

We have demonstrated the possibility of Covert Communication imbedded at the Network Layer (through Routing) in an Ad Hoc wireless Network. We have evaluated the performance of the Covert Channel when the routing protocol is AODV; we have shown that the covert channel is almost undetectable and is capable of transmitting information at the level of a few bits per second. We have shown that such covert communication is possible for any reactive routing protocol. In addition we have developed a superior and totally undetectable covert channel that can be implemented at the MAC layer superposed on a standard collision resolution protocol and have evaluated its performance as well.

We have also investigated Anonymous Communication in Ad Hoc Networks that can protect local membership information, provide robustness against DoS Attacks, and assist in Intrusion Detection. In parallel with the above, we have launched an investigation of sensor networks that are deployed for the purpose of detection of targets or events. We have studied distributed, centralized, and hybrid processing schemes and evaluated detection performance as well as energy consumption for both RF communication and processing. We have also evaluated the robustness of these schemes with respect to loss of nodes and measurements. Also, we have considered the possibility of sequential detection and the exploitation of correlation (spatial and temporal) among the measurements. In addition we formulated the routing issue and we have developed routing link metrics that capture residual battery levels and energy consumption as well as the effect of the routing tree structure on detection performance. We are exploring several extensions of the basic model and we are formulating alternative variants that share the same cross-layer properties as our basic model.

We have extended the investigation of Covert Communication in Ad Hoc wireless networks to the MAC Layer. We have demonstrated implementation of covert channels utilizing MAC protocols based on splitting algorithms. We have developed three different covert transmission strategies; we have evaluated their performance under different variations of the MAC protocols, we have shown that when the conservative transmission strategy is used, the covert channel is totally undetectable, and that the channel is able to transmit information at the level of 0.3 bit per slot

2.11 Vertical Protocol Integration for Enhanced Security in Wireless Sensor Networks

Making upper-layer protocol choices (MAC and routing) contingent on QoS at the physical layer can increase network robustness against threats such as jamming, denial of services, etc. In our past work, we have argued that the inherent interdependencies among protocol layers dictate the joint design across multiple layers. We have focused on the lower three layers in which these interactions are strongest. We have further focused on

the resulting benefits from such integration for wireless network security and information assurance. This integration provides flexibility in designing protocols and networks.

The central thesis of our work is that flexible networking enhances significantly the capabilities of wireless networks to withstand threats. During this reporting period, we investigated the use of flexible MAC/routing protocols to enhance security of wireless sensor networks (or, more generally, any type of wireless ad hoc networks). The basic premise in this line of investigation is the exploitation of the separate degrees of freedom that MAC and routing provide for the transmission of information. In a nutshell, if the routing protocol is attacked and certain routes get congested, the MAC protocol can alleviate congestion by allocating more bandwidth to the congested nodes. Similarly, if the MAC protocol is attacked (which means some nodes are flooded with packets that block reception of desired information), the routing protocol can reroute around the congested bottlenecks. Sensor networks are especially interesting as special cases of ad hoc networks because they provide additional means of flexibility and security trade-offs.

We have focused first on the performance (i.e., the probability of correct detection) as a function of how the sensor data are processed and sent on. Specifically, we have considered the extreme case in which all the data by all sensors are sent to a single control node for processing, versus the other extreme case in which each sensor performs detection and transmits only the result of its detection. We have also considered the intermediate cases of each sensor transmitting a quantized value of its local likelihood ratio for final processing at the control node. In addition to sensing performance analysis, we are also considering the energy expenditures involved in these three options and we plan to evaluate the effectiveness of different threats on each of these alternatives.

Our method uses a novel "coloring" problem that differs from previously considered ones. Typical "coloring" problems have involved the link activation problem that minimizes the length of time needed for the transmission of given numbers of packets between pairs of nodes. Some of these problems are NP-complete and others can be solved in polynomial time. The coloring problem that results from our formulation can be viewed as a "node-group" activation problem by means of identifying sets of receivers that can be enabled simultaneously without full knowledge of the traffic demands. Our initial formulation has led to relatively straight-forward linear programs that yield time-division schedules for best "time-reuse" across the network of a given set of receiving nodes.

3. Distributed Computing Formalisms and Systems

3.1 Formal Modeling of Ad Hoc Routing Protocols for Security Analysis and Testing

Model checking routing protocols for security flaws may assist protocol designers by identifying vulnerabilities automatically. However, model checking has always suffered from the state space explosion problem as more details are added to the model. Using symbolic representations in conjunction with partial order reduction can shrink this state space in a generic fashion, however, not enough to make this approach practical. Our new

approach that may be used in conjunction with those listed above derives from careful consideration of timing. The route discovery flood, and depending on the protocol, the route reply phase, contains race conditions. Since MAC protocols are nondeterministic, it is impossible to pre-determine the results of such a race. The nondeterminism can be modeled probabilistically.

We may soften the problem of model checking to require that only a specified percentage of executions is formally established, using redundancy in implementation to cover the uncertain aspects, given that this percentage is high enough. Intuitively, this should eliminate many states because there are many unlikely race outcomes. Proceeding along these ideas leads to an interesting relationship between the probability of certain causal meshes and the volume of a corresponding class of polytopes whose half-plane contstraint coefficients obey a shortest path distance matrix. A tailored version of Lasserre's dimensional recursion has been formulated, yielding faster results than available tools. We have also investigated the integration of these ideas with human in the loop theorem provers.

3.2 Dynamic and Distributed Trust for Mobile Wireless Ad-Hoc Networks

Future battlefield networks will involve thousands of heterogeneous nodes operating under rapidly changing connectivity, and resource (bandwidth, energy, computation, etc.) constraints. Mobile Ad-hoc networks (MANET) form the basis for current and future military networks. Trust and trust establishment among communicating nodes (soldiers, vehicles, UAVs, satellites) and sensor nodes is the absolute starting point for establishing any such network. The essential and unique properties of trust management in this new paradigm of wireless networking, as opposed to traditional centralized approaches are: (1) *Uncertainty* of trust value. Trust value is represented as subject probability ranging from 0 to 1; (2) *Locality* in trust information exchange; (3) *Distributed computation.*

The main ingredients of our innovative solution of the trust management problem are: (i) An efficient, resilient, distributed scheme for distributing trust evidence documents; (ii) A distributed scheme for "spreading" trust to validated nodes; (iii) A new concept of topology control that helps trust propagation (speed) and minimizes resources (number of links and bandwidth); (iv) Fundamental analytical results, backing experimental evidence of performance, based on techniques from mathematical physics of spin glasses and phase transitions and on the mathematics of dynamic cooperative games on graphs. Our goal is to build a trust computation model based only on *local interactions*, and to investigate the global effects of these interactions. We demonstrated how phase transitions (in this case they mean node transitions from non-trusted to trusted) can appear within a MANET. We linked the existence and analysis of such phase transitions to dynamic cooperative games. The cooperative game framework we developed is useful for investigating other emergent properties of MANET: route connectivity, security, resource allocation. Agents are self-interested, and usually face a frustrated interaction. Normally outcomes without cooperation are worse than those with cooperation. Thus, it is desirable to analyze rules that force all entities to cooperate. Inspiration for our analytical methods comes from the Ising and spin glass models in physics. The Ising

model describes the interaction of magnetic moments or spins, where some spins seek to align (ferromagnetism), while others try to anti-align (antiferromagnetism). Inspired by the Ising model, we developed an interesting cooperative game, where nodes in the network correspond to spins and all nodes only interact with their neighbors, and where each player aims to maximize his payoff.

We analyze the effects of local interactions, which are realized by local policies in our scheme, on global features and dynamics of the system. One of the most important properties is the existence of trusted paths (i.e. paths where all nodes are trusted) between trusted sources and destinations. We analyzed trust dynamics within a MANET, i.e. how trust spreads and/or is revoked between nodes. We investigated and answered questions such as: Does trust spread to a *maximum* set of nodes? What parameters speed up or slow down this transition?

We investigated the effects of the physical and logical (trust) topologies on the performance of distributed trust schemes. The desired properties are: fast spreading and fast revocation of trust even with failing nodes. An important requirement is to achieve high performance efficiently, which in the framework of MANET translates to sparsity of the logical (trust) topology. In this context we showed that topologies with the so-called "*small world*" characteristic are the most efficient. This leads to simple schemes for controlling the trust graph topology so as to maintain this desirable characteristic. We provided interesting interpretations and properties of these topologies: Nodes few "trust" hops from each other; Scalable: local map is like global map.

3.3 Pathwise Trust Computation for MANET

Trust between nodes depends on their past interactions, and future interactions depend on established trust. However, when two nodes have had no direct interaction, they can base their trust estimates for each other on other nodes' experiences (second-hand evidence). In this way, one or more *trust paths* are formed. We modeled the situation as a weighted graph, in which edges represent direct trust relations. We captured and formalized two fundamental intuitive notions of trust: First, long trust paths are less reliable than short ones. Second, many trust paths are more reliable than just a few. Each trust relation takes into account not only the amount of trust that a node places on another, but also the amount of evidence that this estimate is based on.

Our formal model is based on a mathematical structure called a *semiring*. It allows us to model the trust relations, interpret the intuitive notions in a rigorous way, propose algorithms for the solution, and analyze their behavior when problem parameters change. We have developed two semirings that estimate the (indirect) trust relation between two nodes. The first is simpler, more bandwidth-efficient, faster, but less accurate than the second since it bases the estimates on the single best trust path available. The second uses all available information (weighted according to its importance), so the trust decision is perfectly accurate. However, it requires longer waiting times and more message exchanging. So, we have identified a tradeoff between accuracy and cost (which quantifies wireless network constraints such as limited bandwidth and energy).

We evaluated a solution that takes advantage of the good points of both semirings. We keep the information that influences the result the most. Therefore, we compute an accurate result, without wasting resources. We also placed great emphasis on the robustness of our solution, i.e. what happens when malicious nodes infiltrate the system. The scenario we used partitions the nodes into Good and Bad. Good nodes interact with other nodes (both Good and Bad) and gradually identify their one-hop neighbors correctly. Bad nodes always give the worst opinions for Good nodes, and the best opinions for other Bad nodes. As we increase the percentage of Bad nodes, we expect the situation to deteriorate but a graceful degradation is preferred to a catastrophe. What happens is that Good nodes only identify (Good and Bad) nodes that are close to them (in the trust graph). They reach no decision when it comes to nodes that are many hops away. Even when there are 90% Bad nodes, no Bad node is misidentified as Good, or vice-versa.

Our model is expressive enough to describe the trust computations of PGP. We believe that it can provide a platform for the design and comparison of various trust metrics that can potentially satisfy a number of different constraints.

3.4 Network Tomography for Dynamic Network Monitoring and Information Assurance

The fundamental problem addressed by Network Tomography is to obtain a spatio-temporal picture of a network from end-to-end views and measurements such as delay or packet loss. These measurements can be performed in an active fashion via probes or in a passive fashion (non-intrusive). The implementation can be either via unicast or multicast communications. An interesting such example problem involves using measured end-to-end delays, which can be thought of as representing distances in a graph. Another interesting example is to measure end-to-end packet loss. The problem is then: can we reconstruct the entire graph from a subset of these distances? This problem is an example of an *inverse problem.*

A repetitive application of these concepts leads to the problem of monitoring the status of a network by observations from the "edge". A realistic formulation of these problems must account for the fact that only partial information can be obtained by setting up monitors at a relatively small subset of the nodes. From these monitors, data can be collected and examined. The problem of discovering the detailed inner structure of the network from the collection of end to end measurements can be seen as a type of inverse problem, analogous to those arising in conventional tomography, but discrete this time.

One of the ways to try to understand what's going on, is to visualize the directed graph representing the network by laying it out in 3D hyperbolic space or even 2D hyperbolic space, since in these spaces the volume of a ball increases exponentially with the radius, as opposed to the familiar geometric increase of the volume of a ball in Euclidean 3-D space, respectively 2D Euclidean space. We have developed an innovative mathematical formulation of these problems using this representation of the network as embedded in the real hyperbolic plane. In this representation paths between nodes become the

geodesics of the hyperbolic geometry. Thus our innovative formulation and solution methodology reinforce that the *correct* tomography to use is not the Euclidean one but that in the 2-D or 3-D real hyperbolic space.

A key objective of our research is to obtain computationally efficient algorithms for solving such inverse problems. Our approach is based on our previous work, where we have studied a classical inverse problem of partial differential equations, the Inverse Conductivity Problem, also called EIT (Electrical Impedance Tomography) in the engineering literature. Our earlier work demonstrated a close relation between tomography and EIT. For the EIT problem we have obtained a very efficient computationally solution that involved Radon Transform in hyperbolic space. The EIT problems arising out of network tomography problems are more akin to discrete electrical network inverse problems as those investigated and solved by Curtis and Morrow. Our approach combines the methods of Curtis and Morrow with our earlier tomographic methods on trees and graphs, while extending these methods to probabilistic models and situations.

3.5 Dissemination and Discovery of Information Assurance Models and Data in Wireless Networks

The proliferation of wireless technologies along with the large volume of data available online are forcing us to rethink existing data dissemination techniques and in particular for aggregate data. In addition to scalability and response time, data delivery to mobile clients with wireless connectivity must also consider energy consumption. We developed a hybrid scheduling algorithm (DV-ES) for broadcast-based data delivery of aggregate data over wireless channels. Our algorithm efficiently "packs" aggregate data for broadcast delivery and utilizes view subsumption at the mobile client, which allow for faster response times and lower energy consumption.

Object location is a major part in the operation of distributed networks. We investigated and analyzed the performance of several search methods for unstructured networks. We analyzed the performance of the algorithms relative to various metrics, giving emphasis on the success rate, bandwidth-efficiency and adaptation to dynamic network conditions. Simulation results were used to empirically evaluate the behavior of nine representative schemes under a variety of different environments. We developed the Adaptive Probabilistic Search method (APS). Other proposed search methods either depend on network-disastrous flooding and its variations or utilize indices too expensive to maintain. Our scheme utilized feedback from previous searches to probabilistically guide future ones. It performs efficient object discovery while inducing zero overhead over dynamic network operations, such as new node arrivals/departures or object relocation. Extensive simulation results show that APS achieves high success rates, increased number of discovered objects, very low bandwidth consumption and good adaptation to changing topologies.

We have developed an *Adaptive Group Notification* (*AGNO*) scheme on top of APS. *AGNO* efficiently contacts large peer populations in unstructured Peer-to-Peer networks and defines a novel implicit approach towards group membership by monitoring demand

for content as this is expressed through lookup operations. Utilizing search indices, together with a small number of soft-state shortcuts, *AGNO* achieves effective and bandwidth-efficient content dissemination, without the cost and restrictions of a membership protocol. The method achieves high-success content transmission at a cost at least two times smaller than other proposed techniques for unstructured networks.

4. Publications

1. J.S. Baras and M. Rabi, "Intrusion Detection with Support vector Machines and Generative Models", *Proc. of 5th Information Security Conference*, ISC 2002, LNCS Vol. 2433, pp. 32-47.

2. J.S. Baras, A.A. Cardenas and V. Ramezani, "On-Line Detection of Distributed Attacks from Space-Time Network Flow Patterns" , *Proceedings of 23rd Army Science Conference*, Orlando, Florida, December 2-5, 2002. This paper received the **Best Paper Award** *in IT/C4ISR* (Information Technology, Information Technology/Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance) at the 23rd Army Science Conference.

3. V.R. Ramezani, Shah-An Yang and J.S. Baras, "Finite Automata Models for Anomaly Detection", *Proceedings of 37th Conference on Information Sciences and Systems*, Johns Hopkins University, Baltimore, Maryland, March 12-14, 2003.

4. A.A. Cardenas, J. S. Baras and V. Ramezani, "Distributed Change Detection for Worms, DDoS and other Network Attacks", invited paper, *Proceedings of the 2004 American Control Conference (ACC04)*, Volume 2 pages 1008-1013, Boston, MA, June 30 - July 2, 2004.

5. S. Radosavac and J. S. Baras, "Detection and Classification of Network Intrusions Using Hidden Markov Models," *Proc. of 37th Conference on Information Sciences and Systems (CISS)*, Baltimore, March 2003.

6. S. Radosavac, J. S. Baras and N. Benammar, "Cross-Layer Attacks in Wireless Ad Hoc Networks", *Proceedings of the 38th Annual Conference on Information Sciences and Systems (CISS 2004)*, pp. 1266-1271, Princeton, New Jersey, March 17-19, 2004

7. J. Baras and S. Radosavac, "Attacks and Defenses Utilizing Cross-Layer Interactions in MANET," *Workshop on Cross-Layer Issues in the Design of Tactical Mobile Ad Hoc Wireless Networks: Integration of Communication and Networking Functions to Support Optimal Information Management*, June 2–3, 2004, Naval Research Laboratory, Washington, DC.

8. A. A. Cardenas, S. Radosavac and J. S. Baras, "Detection and Prevention of MAC Layer Misbehavior for Ad Hoc Networks," *Proceedings of the 2004 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04),* pp. 17-22, Washington, DC, October 25, 2004.

9. S. Radosavac, J. S. Baras and I. Koutsopoulos, "A Framework for MAC Protocol Misbehavior Detection in Wireless Networks", *Proceedings of ACM Workshop on Wireless Security (WiSe 2005),* Cologne, Germany, September 2, 2005..

10. S. Radosavac, K. Seamon and J. S. Baras, "bufSTAT – A Tool for Early Detection and Classification of Buffer Overflow Attacks", *Proceedings of the First IEEE/Createnet SecureCom 2005,* Athens, Greece, September 5-9, 2005.

11. G. Theodorakopoulos and J. S. Baras, "On Trust Models and Trust Evaluation Metrics for Ad-Hoc Networks", accepted by *IEEE Journal of Selected Areas in Communications, special issue on Security in Wireless Ad-Hoc Networks*, June 2005.

12. G. Theodorakopoulos, J. S. Baras, "Trust Evaluation in Ad-Hoc Networks", *Proceedings of ACM Workshop on Wireless Security (WiSe 2004),* pp. 1-10, Philadelphia, Pennsylvania, October 1, 2004. **(Best Paper Award).**

13. M. Karir, J. S. Baras, "LES: Layered Encryption Security", *Proceedings of the 3rd International Conference on Networking (ICN'04),* pp. 382-388, Gosier, Guadeloupe, French Caribbean, February 29 – March 4, 2004.

14. Roy-Chowdhury, J. S. Baras, "Framework for IP Multicast Routing in Satellite ATM Networks", *Proceedings of 22nd AIAA International Communication Satellite Systems Conference & Exhibit 2004, (ICSSC),* Monterey, California, May 9-12, 2004.

15. Roy-Chowdhury, J. S. Baras, " Key Management for Secure Multicast in Hybrid Satellite Networks", *Proceedings of the 18th International Information Security Conference (IFIP/SEC 2004), Security and Protection in Information Processing Systems,* pp. 533-548, August 23-26, 2004.

16. L. Eschenauer, V. Gligor and J. S. Baras, "On Trust Establishment in Mobile Ad-Hoc Networks", *Proc. 10th International Workshop on Security Protocols*, April 2002, Cambridge, UK; in *Security Protocols*, Lecture Notes in Computer Science, Springer, 2003.

17. L. Eschenauer, J. S. Baras and V. Gligor, ``Distributed Trust Establishment in MANETs: Swarm Intelligence," *CTA Conference*, April 29 - May 1, 2003, pp. 125-129.

18. T. Jiang, J.S. Baras, "Ant-based Adaptive Trust Evidence Distribution in MANET", *Proceedings of MDC'04,* pp. 4392-4396, Tokyo, Japan, March 23-26, 2004.

19. J. S. Baras and T. Jiang, "Cooperative Games, Phase Transitions on Graphs and Distributed Trust In MANET", invited paper, in *Proc of 2004 IEEE Conference on Decision and Control,* Dec. 2004.

20. J.S. Baras and T. Jiang, "Dynamic and Distributed Trust for Mobile Ad-Hoc Networks", in *Proc. 24th Army Science Conference,* Orlando, Florida, Nov. 2004.

21. J. S. Baras and T. Jiang, "Managing Trust in Self-organized Mobile Adhoc Networks", invited paper, *Proc. Wireless and Mobile Security Workshop, Network and Distributed Systems Security Symposium*, February 2005, San Diego, USA.

22. T. Jiang and J. S. Baras, "Autonomous Trust Establishment", *Proc. 2nd International Network Optimization Conference (INOC)*, February 2005, Lisbon, Portugal.

23. J. S. Baras and T. Jiang, "Cooperation, Trust and Games in Wireless Networks", invited paper, in *Proceedings of Symposium on Systems, Control and Networks*, honoring Professor P. Varaiya, Birkhauser, June 2005.

24. A. A. Cardenas, N. Benammar, G. Papageorgiou and J.S. Baras, "Cross-Layered Security Analysis of Wireless Ad-Hoc Networks", *Proc. 24th Army Science Conference* , Orlando, Florida, Nov. 2004.

25. S. Yang and J.S. Baras, "*TORA, Verification, Proofs and Model Checking*", *Proceedings of WiOpt '03: Modeling and Optimization in Mobile, AdHoc and Wireless Networks,* Sophia-Antipolis, France, March 3-5, 2003.

26. S. Yang, J.S. Baras, "Correctness Proof for a Dynamic Adaptive Routing Algorithm for Mobile Ad-hoc Networks" *Proceedings of IFAC Workshop – Modeling and Analysis of Logic Controlled Dynamic Systems,* Irkutsk , Lake Baikal, Russia, July 30 – August 1, 2003.

27. S. Yang and J. S. Baras, "Modeling Vulnerabilities of Ad Hoc Routing Protocols," *Proceedings of the SASN 2003 Conference*, George Mason University, Fairfax, Virginia, October 31, 2003.

28. L. Eschenauer and V. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 41-47, ACM Press, 2002.

29. H. Chan, A. Perrig, V. Gligor and G. Muralidharan, "On the Distribution and Revocation of Cryptographic Keys in Sensor Networks", to appear in *IEEE Trans. On Dep. And Secure Computations*, 2005.

30. B. Parno, A. Perrig and V. Gligor, "Distributed Detection of Node Replication Attacks in Sensor Networks", to appear in *IEEE Journal on Security and Privacy*, 2005.

31. H. Chan, G. Muralidharan, V. Gligor, and A. Perrig, "On the Distribution and Revocation of Keys in Sensor Networks," invited paper, for the Inaugural Issue of the *IEEE Transactions on Dependable and Secure Computing*.

32. A. Perrig, G. Muralidharan and V. Gligor, "On the Distribution and Revocation of Hyptographic Keys in Sensor Networks," to appear in *IEEE Transaction on Dependable and Secure Computing.*

33. B. Parno, A. Perrig and V. Gligor, "Distributed Detection of Node Replication Attacks in Sensor Networks," *Proceedings of the 2005 IEEE Symposium on Security and Privacy (IEEE S&P 2005)*, pp. 49-63, Oakland, California, May 8-11, 2005.

34. I. Haitner, O. Horvitz, J. Katz, C.Y. Koo, R. Morselli, and R.Shaltiel, "Reducing Complexity Assumptions for Statistically-Hiding Commitment", *Proceedings Eurocrypt 2005.*

35. O. Horvitz and J. Katz, "Lower Bounds on the Efficiency of 'Black-Box' Commitment Schemes," *Proc. of International Colloquium on Automata, Languages, and Programming (ICALP) 2005.*

36. C. A. Berenstein and S-Y. Chung, "*w*-Harmonic Functions and Inverse Conductivity Problems on Networks," *SIAM J. Appl. Math.* 65, 2005, no. 4, 1200-1226.

37. J. Baras, C. A. Berenstein and F. Gavilánez, "Continuous and Discrete Inverse Conductivity Problems," *AMS, Contemporary Math*, Vol. 362, pp. 33-51, June 2004.

38. J. Baras, C. A. Berenstein and F. Gavilánez, "Network Tomography," *Proceedings of 2004 AMS meeting at Ryder Univ., Special Session on Tomography*, to appear in Contemporary Math.

39. Y. Sismanis, A. Deligiannakis, N. Roussopoulos, and Y. Kotidis, "Dwarf: Shrinking the PetaCube", Proc. of *ACM SIGMOD International Conference on Management of Data*, June 3-6 2002, pp.464-475.

40. Y. Sismanis, A. Deligiannakis, Y. Kotidis and N. Roussopoulos, "Hierarchical Dwarfs for the Roll-Up Cube," In Proc. of the *DOLAP Workshop* (held in conjunction with ACM CIKM'03), New Orleans, LA, USA, November 2003.

41. M. A. Sharaf, Y. Sismanis, A. Labrinidis, P. K. Chrysanthis and N. Roussopoulos, "Efficient Dissemination of Aggregate Data over the Wireless Web," In Proc. of the *Sixth International Workshop on the Web and Databases* (held in conjunction with ACM SIGMOD'03), June 12-13 2003, San Diego, CA, USA.

42. D. Tsoumakos and N. Roussopoulos, "Adaptive Probabilistic Search for Peer-to-Peer Networks," in Proc. of the 3rd *IEEE International Conference on P2P Computing*, Sept 1-3 2003, Linkoping, Sweden.

43. D. Tsoumakos and N. Roussopoulos: "A Comparison of Peer-to-Peer Search Methods," in Proc. of the *Sixth International Workshop on the Web and Databases* (held in conjunction with ACM SIGMOD'03), June 12-13 2003, San Diego, CA, USA

44. Y. Sismanis and N. Roussopoulos, "The Polynomial Complexity of Fully Materialized Coalesced Cubes," in *Proc. 30th International Conference on Very Large Databases, Toronto,* August 29th-September 3rd, 2004.

45. D. Tsoumakos and N. Roussopoulos, "A Framework for Sharing Voluminous Content in P2P Systems," in *Proc. 2004 International MultiConference in Computer Science & Computer Engineering,* Las Vegas, Nevada, June 21-24, 2004.

46. S. Kantere and D. Tsoumakos and N. Roussopoulos, "Querying Structured Data in an Unstructured P2P System," *Proceedings of the 6th ACM International Workshop on Web Information and Data Management* (WIDM 2004), November 12-13, 2004, Washington, DC, USA.

47. D. Tsoumakos and N. Roussopoulos, "AGNO: An Adaptive Group Communication Scheme for Unstructured P2P Networks," to appear in *Proc. of 2005 Euro-Par Conference.*

48. K. Bitsakos, D. Tsoumakos, N. Roussopoulos and Y. Aloimonos, "A Framework for Distributed Human Tracking," to appear in *Proc. of the 2005 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'05).*

49. L. Yu and A. Ephremides, "Detection Performance and Energy Efficiency Trade-off in a Sensor Network," *Proceedings 41st Annual Allerton Conference on Communication, Control, and Computing*, Monticello, Illinois, October 2004.

50. A. Ephremides and L. Yu, "Detection, Energy, and Robustness in Wireless Sensor Networks," invited paper, *Proceedings of Mobwiser*, Singapore, March 2004.

51. S. Li and A. Ephremides, "A Covert Channel in MAC Protocols Based on Splitting Algorithms", invited paper, in *Proceeding of 43rd IEEE Wireless Communication and Networking Conference(WCNC)*, 2005, New Orleans, LA USA.

52. S. Li and A. Ephremides, "A Network Layer Covert Channel in Ad-hoc Wireless Networks", in *Proceedings of 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Network(SECON)*, Santa Clara, CA, October 2004.

53. Y. Sun, W. Trappe, and K. J. R. Liu, "An Efficient Key Management Scheme for Secure Wireless Multicast," in *Proc. 2002 IEEE Int. Conference on Communications*, Vol 2, pp. 1236-1240, April 2002, New York City.

54. W. Trappe, Y. Wang, and K.J.R. Liu, "Establishment of Conference Keys in Heterogeneous Networks", *Proc of 2002 IEEE International Conference on Communications*, ICC 2002., Vol. 4 , pp. 2201 -2205.

55. B. Sun, W. Trappe, Y. Sun, and K.J.R. Liu, "A Time-Efficient Contributory Key Agreement Scheme for Secure Group Communications", *Proc of 2002 IEEE International Conference on Communications*, ICC 2002., Vol. 2, pp. 1159 -1163.

56. M. Wu and Y. Mao, "Communication-Friendly Encryption of Multimedia," *Proc. IEEE Multimedia Signal Processing Workshop (MMSP'02)*, St. Thomas, U.S. Virgin Islands, Dec. 2002.

57. W. Trappe, M. Wu, Z.J. Wang, K.J.R. Liu, "Anti-collusion Fingerprinting for Multimedia", *IEEE Transactions on Signal Processing*, Volume: 51 Issue: 4 , Apr 2003, Page(s): 1069 -1087

58. W. Trappe, J. Song, R. Poovendran, and K.J.R. Liu, "Key Management and Distribution for Secure Multimedia Multicast," *IEEE Trans. on Multimedia*, Vol. 5, No. 4, pp.544-557, Dec 2003.

59. Yan Sun, and K.J. Ray Liu, "Securing Dynamic Group Membership Information over Multicast: Attacks and Immunization", in Proc. *IEEE GLOBECOM,* San Francisco, CA, Dec. 2003.

60. Yan Sun, Wade Trappe, and K.J. Ray Liu, "Topology-aware Key Management Schemes for Wireless Multicast", in Proc. *IEEE GLOBECOM,* San Francisco, CA, Dec. 2003.

61. Yan Sun, and K.J. Ray Liu, "Multi-layer Management for Secure Multimedia Multicast Communications", in Proc. *IEEE International conferences on Multimedia and Expo (ICME'03)*, vol. II, pp 205-208, Baltimore, MD, July 2003.

62. Y. Sun, and K. J. Ray Liu, "Securing Dynamic Membership Information in Multicast Communications", in *Proc. IEEE INFOCOM'04*, Hong Kong, March 2004.

63. Sun, and K. J. Ray Liu, "Scalable Hierarchical Access Control in Secure Group Communications", in *Proc. IEEE INFOCOM'04*, Hong Kong, March 2004.

64. Y. Mao, Y. Sun, M. Wu and K. J. Ray Liu, "Dynamic Join and Exit Amortization and Scheduling for Time-Efficient Group Key Agreement", in *Proc. IEEE INFOCOM'04*, Hong Kong, March 2004.

65. Y. Sun, W. Trappe, and K. J. R. Liu, "A Scalable Multicast Key Management Scheme for Heterogeneous Wireless Networks," *IEEE/ACM Transactions on Networking*, Vol. 12, No. 4, pp. 653-666, August 2004.

66. W. Trappe, Y. Wang, and K.J.R. Liu, "Resource-Aware Conference Key Establishment for Heterogeneous Networks," *IEEE/ACM Trans. on Networking*, vol 13, no 1, pp.134-146, Feb 2005.

67. W. Yu and K.J.R. Liu, "Attack-Resistant Cooperation Stimulation in Autonomous Ad Hoc Networks", *accepted by IEEE JSAC special issue on Autonomic Communication Systems, June 2005.*

68. W. Yu, Y. Sun and K.J.R. Liu, "HADOF: Defense Against Routing Disruptions in Mobile Ad Hoc Networks", *Proc. IEEE INFOCOM'05*, Miami, March 2005.

69. Y. Sun, W. Yu, Z. Han and K. J.R. Liu, "Information Theoretic Framework of Trust Modeling and Evaluation for Ad Hoc Networks", *accepted by IEEE JSAC special issue on Security in Wireless Ad Hoc Networks,* June 2005.

70. Y. Mao, Y. Sun, M. Wu, and K. J.R. Liu, "Join-Exit Scheduling for Contributory Group Key Agreement", accepted by *IEEE/ACM Transactions on Networking*, May 2005.

71. Y. Hwang and H. C. Papadopoulos, "Partial-encryption analysis of a class of pseudo-chaotic spread spectrum systems," in *Proc. 40th Allerton Conf. on Comm. Control Comput*, Sep. 2002.

72. Y. Hwang and H. C. Papadopoulos, "Physical-layer secrecy with DS/SS from piecewise linear chaotic Markov maps: analysis and design," in *Proc. IEEE Wireless Commun. Net. Conf.*, pp. 642-647, March 2003.

73. Y. Hwang and H. C. Papadopoulos, "Physical-layer Secrecy in AWGN Via a Class of Chaotic {DS/SS} Systems: Analysis and Design," accepted for publication *in IEEE Trans. Signal Processing 2004.*

74. Y. Hwang and H. C. Papadopoulos, "Physical-layer Secrecy with Chaotic DS/SS: Unintended Receiver Performance Analysis and System Design," in Proc. 2004 *IEEE Int. Conf. Communications (ICC), June 2004.*

75. Y. Hwang and H. C. Papadopoulos, "Private Communication over Fading Channels with Chaotic DS/SS," in Proc. 2004 *IEEE Int. Conf. Acoust. Speech, Signal Processing (ICASSP), pp. 957-960, May 2004.*

76. D. S. Scherber and H. C. Papadopoulos, "Locally Constructed Algorithms for Distributed Computations in Ad-hoc Networks," in Proc. 2004 *Conf. Inform. Proc. Sens. Net. (IPSN).*

77. T. Pham, D. S. Scherber, and H. C. Papadopoulos, "Distributed Source Localization Algorithms for Acoustic Ad-hoc Sensor Networks," in Proc. *IEEE SAM'2004 Workshop.*

A Key Management Scheme for Distributed Sensor Networks

Laurent Eschenauer and Virgil D. Gligor

Electrical and Computer Engineering Department

University of Maryland, College Park, MD. 20742

{laurent, gligor}@eng.umd.edu

Abstract — **Distributed Sensor Networks (DSNs) are ad-hoc, mobile, networks that include sensor nodes with very limited computation, memory and communication capabilities. DSNs are dynamic in the sense that they allow addition and deletion of sensor nodes after deployment to grow the network or replace failing and unreliable nodes. DSNs may be deployed in hostile areas where communication is monitored and nodes are subject to capture and surreptitious use by an adversary. Hence DSNs require cryptographic protection of communications and sensor-capture detection, sensor disabling, and key revocation. In this paper, we present a key management scheme designed to satisfy both the operational and the security requirements of DSNs. The scheme includes selective distribution and revocation of keys to sensor nodes as well as node re-keying, and requires neither on-line, trusted authorities nor substantial communication, computation, and memory capabilities. It relies on probabilistic key sharing among the nodes of a random graph and uses simple, secure shared-key discovery and path-key establishment protocols for key distribution, revocation, re-keying, and incremental additions of nodes. The security and network connectivity characteristics supported by the key management scheme are discussed and simulation experiments presented.**

Index Terms— **distributed sensor networks, key distribution and revocation, node re-keying, connectivity of random graphs, secure shared-key discovery, path-key establishment.**

I. Introduction

Distributed Sensor Networks (DSNs) share several characteristics with the more traditional embedded wireless networks [11]. Both include: arrays of sensor nodes that are battery powered, have limited computational capabilities and memory, rely on intermittent wireless communication via radio frequency and, possibly, optical links; data-collection nodes that cache sensor data and make it available for processing to application components of the network; control nodes that monitor the status of and broadcast simple commands to sensor nodes; and some mobile nodes (e.g., data collection and control nodes placed on humans, vehicles, aircraft). However, DSNs differ from the traditional embedded wireless networks in several important areas, namely: their scale is orders of magnitude larger than that of embedded wireless networks (e.g., tens of thousands as opposed to just tens of sensor nodes); they are dynamic in the sense that they allow addition and deletion of sensor nodes after deployment to grow the network or replace failing and unreliable nodes without physical contact; and they may be deployed in hostile areas where communication is monitored and sensor nodes are subject to capture and manipulation by an adversary. These challenging operational requirements place equally challenging security constraints on DSN design. (For a detailed analysis of the operational and security constraints of DSNs, the reader is referred to the work of Carman, Kruus and Matt [3].)

Communication Security Constraints. The extreme power, computational and memory limitations of sensor nodes we are considering for large-scale DSNs preclude the use of some of the traditional cryptographic tools for securing network communication. For example, the implementation of the Smart Dust project [5,8] uses sensors that have only 8Kb of program memory and 512 bytes for the data memory, and processors with 32 8-bit general registers that run at 4 MHz and 3.0V, such as the ATMEL 90LS8535. For such DSNs, asymmetric (public-key) cryptosystems and random-number are impractical since they are computationally intensive and consume a significant amount of power. Use of symmetric-key ciphers, low-power encryption modes, and hash functions becomes the only viable means of protecting communication against DSN adversaries; e.g., on limited-capability processors, block ciphers and hash functions are three to four order of magnitude faster than asymmetric operations, such as digital signatures [6].

Although sensor nodes trust each other, physical capture of, and tampering with, a sensor node or a subset of nodes by an adversary is possible due to their ad-hoc deployment in hostile areas. This requires that tamper-detection technologies [7,13] be used to shield sensors

Reprinted by permission from *ACM CCS2002*, Washington D.C., 2002.

such that detection of physical sensor manipulation disables a sensor's operation and erases its keys. Furthermore, active manipulation of a captured sensor's data inputs by an adversary must be detectable. However, since the captured sensor nodes may not necessarily communicate in an erratic or anomalous manner, traditional anomaly detection techniques may not apply. Detecting a sensor's data input manipulation may require data correlation analysis and anomaly detection, possibly off-line, by collection and processing nodes. While such analysis can detect active insertion of bogus data by an adversary, it requires redundant sensor coverage of deployment areas and, hence, sufficient sensor-node density in the DSN.

Key Management Constraints. Traditional Internet style key exchange and key distribution protocols based on infrastructures using trusted third parties are also ruled out by sensor-node processing limitations, unknown network topology, intermittent sensor-node operation, network scale and dynamics. To date, the only practical options for the distribution of keys to sensor nodes of large-scale DSNs whose physical topology is unknown prior to deployment would have to rely on *key pre-distribution*. Keys would have to be installed in sensor nodes to accommodate secure connectivity between nodes. However, traditional key pre-distribution offers two inadequate solutions: either a single *mission key* or a set of separate *n-1 keys*, each being pair-wise privately shared with another node, must be installed in every sensor node.

The single mission-key solution is inadequate because the capture of any sensor node may compromise the entire DSN since selective key revocation is impossible upon sensor-capture detection. In contrast, the pair-wise private sharing of keys between every two sensor nodes avoids the wholesale DSN compromise upon node capture since selective key revocation becomes possible. However, this solution requires pre-distribution and storage of n-1 keys in each sensor node, and n(n-1)/2 per DSN, which renders it impractical for large-scale DSNs using, say, more than 10,000 nodes, for both intrinsic and technological reasons. First, pair-wise private key sharing between any two sensor nodes would be unusable since such connectivity is achievable only in small node neighborhoods delimited by wireless communication ranges and sensor density (e.g., of the order of tens of nodes but not tens of thousands). Second, incremental addition and deletion as well as re-keying of sensor nodes would become both expensive and complex as they would require multiple keying messages to be broadcast network-wide to all nodes during their non-sleep periods (i.e., one broadcast message for every added/deleted node or re-key operation). Third, storing *n-1* keys would push sensor-memory limits for the foreseeable future, even if only short, 64-bit, keys are used,[1] and would complicate fast key erasure upon detection of physical sensor tampering.

Our Approach. We propose a simple key pre-distribution scheme that requires memory storage for only few tens to a couple of hundred keys, and yet has similar security and superior operational properties compared to those of the pair-wise private key-sharing scheme. Our scheme relies on probabilistic key sharing among the nodes of a random graph and uses a simple secure shared-key discovery protocol for key distribution, revocation and node re-keying. We distribute a ring of keys to each sensor node, each key ring consisting of randomly chosen k keys from a very large pool of P keys, which is generated off-line, prior to DSN deployment. Because of the random choice of keys on key rings, a shared key may not exist between some pairs of nodes. Although two nodes may or may not share a key, if a path of nodes sharing pair-wise private keys exists between the two nodes at network initialization, the two nodes can use that path to exchange a key that will establish a direct link. We use random graph analysis and simulation to show that what really matters in key pre-distribution is the shared-key connectivity of the resulting network. Therefore, full shared-key connectivity offered by pair-wise private key sharing between every two nodes becomes unnecessary. For example, we show that to establish "almost certain" shared-key connectivity for a 10,000-node network, a key ring of only 250 keys have to be pre-distributed to every sensor node where the keys were drawn out of a pool of 100,000 keys, leaving a substantial number of keys available for DSN expansion (viz., Sections III and IV). We also show that the security characteristics of probabilistic key distribution and revocation based on random graphs are suitable for solving the key management problem of DSNs.

Related Work. Symmetric key pre-distribution has been used in past research, but with a focus on group and broadcast communication. For group communication [1,2], this research tries to accommodate any set of up to k users while being secure against collusion of some of

[1] Adding approximately 64K-bit *key* memories to sensors of large-scale DSNs may become feasible in a couple sensor-node generations from now.

them. Pre-distribution is used to alleviate the cost of communication between group members and to setup a common secret key; however, memory constraints are not placed on group members. Other work on broadcast encryption [4] focuses on key distribution to support broadcast communication between slave nodes and a master node – an impractical approach for DSNs.

II. Overview of the Basic Scheme

In this section, we present the basic features of our scheme, deferring its analysis for the next section.

A. Key Distribution

In our scheme, key distribution consists of three phases, namely key pre-distribution, shared-key discovery, and path-key establishment.

The *key pre-distribution* phase of our scheme consists of five off-line steps, namely generation of a very large pool of P keys and of their key identifiers; random drawing of k keys out of P without replacement to establish the key ring of a sensor; loading of the key ring into the memory of each sensor; saving of the key identifiers of a key ring and associated sensor identifier on a trusted control node; and for each node, loading the ith controller with the key shared with that node. (Note that the key shared by a node with the ith controller, K^{ci}, can be computed as $K^{ci} = E_{Kx}(ci)$, where $Kx = K_1 \oplus, \ldots, \oplus K_k$), K_i are the keys of the node's key ring, ci is the controller's identity, and E_{Kx} denotes encryption with node key Kx. Hence, the keys shared by a node with controllers, which are only infrequently used, need not take any space on the key ring.) As shown in the next section, the key pre-distribution phase ensures that only a small number of keys need to be placed on each sensor node's key ring to ensure that any two nodes share (at least) a key with a chosen probability. For example, for 0.5 probability only 75 keys drawn out of a pool of 10,000 keys need to be on any key ring.

The *shared-key discovery* phase takes place during DSN initialization in the operational environment where every node discovers its neighbors in wireless communication range with which it shares keys. In this phase, every node broadcasts a short, securely encoded message to its neighbors in wireless range and each node decodes every other node's message in a secure manner to discover the shared keys.[2] We adapt a method by Trappe et al. [9,12] for secure re-keying in broadcast groups to the secure shared-key discovery (viz., Appendix A). To discover its keys shared with its neighbors, each node X computes two B-bit random seeds μ_1 and μ_2 which, unlike in the original proposal, need not be random. Then, node X broadcasts a *hello* message that contains random seeds μ_1, μ_2 and parameter α, where

$$\alpha = \mu_2 + \prod_{i=1}^{k} f(K_i^X, \mu_1),$$

K_i^X, i= 1,…,k, are the keys present on node X's key ring, and function f is a public *parametric one-way function.* Note that the use of the parametric one-way function does not reveal any of the keys used in the *hello* message. All nodes neighboring X in wireless communication range compute

$$\mu'_2(j) = \alpha \,(\mathrm{mod}\, f(K_j, \mu_1))$$

for all K_j, j= 1,…,k, on their own key ring. If a key K_j on a neighbor's key ring matches one of node X's keys K_i, the computation of $\alpha \,(\mathrm{mod}\, f(K_j,\mu_1))$ produces a value $\mu'_2(j)$ that would match the received seed μ_2. Therefore, a neighboring node of X would know that its key K_j for which the computation of α produced the correct μ_2 is shared with the node X. The cost of shared-key discovery is k times the cost of one f computation plus one modular division and is small because the size of a key ring, k, is small (viz., analysis of Section III).

The shared-key discovery phase establishes the topology of the sensor array as seen by the routing layer of the DSN. A *link* exists between two sensor nodes only if they share a key; and if a link exists between two nodes, all communication on that link is secured by link encryption. Note that it is possible that the same key is shared by more than a pair of sensor nodes, since the key rings consist of keys drawn randomly from the same pool. This does not cause a link-security exposure because, in normal mode of operation sensor nodes trust each other and, during the revocation phase following node-capture detection, revocation of a captured node's key ring ensures that the small set of (k) keys on that ring are removed network-wide. (Potential damage caused by sensor node capture is limited to the feeding of bogus data to that sensor node's inputs since common tamper-detection shielding of each node is assumed to ensure key-ring erasure.)

The *path-key establishment* phase assigns a *path-key* to selected pairs of sensor nodes in wireless communication range that do not share a key but are connected by two or more (i.e., a path of) links at the end of the shared-key discovery phase. Path keys need

[2] The simplest way for any two nodes to discover if they share a key is that each node broadcast, in clear text, the list of identifiers of the keys on their key ring. The drawback of this simple approach is that if an attacker captures a node, the attacker knows immediately which link he can decrypt and which links he cannot.

not be generated by sensor nodes. The design of the DSN ensures that, after the shared-key discovery phase is finished, a number of keys on a key ring are left unassigned to any link. For example, both analysis (Section III) and simulations (Section IV) show that even without special provisioning a substantial number of keys are left unused on key rings. Provisioning for sufficient ring keys that are left unassigned by the determination of key-ring size (k) can also anticipate both the effects of revocation and those of incremental addition of new sensor nodes, since both may require the execution of the path key establishment phase after shared-key discovery. The analysis and simulations presented in the next sections indicates that such provisioning is especially simple.

B. Revocation

Whenever a node is detected as being compromised it is essential to be able to revoke the entire key ring of that node. To effect revocation, a controller node (which has a large communication range and may be mobile) broadcasts a single revocation message containing signed list of k key identifiers for the key ring to be revoked followed by the random seed μ and α. (We use the same secure scheme as that of Trappe *et al.* briefly presented in Appendix A.) To sign the list of key identifiers, the controller generates a key K_e and α:

$$\alpha = K_e + \prod_{i=1}^{n} f(K_i, \mu),$$

where the Ki are unique, randomly chosen, secret keys shared by the controller with each sensor node n_i during key pre-distribution phase. Each sensor node can obtain key Ke to verify the signature on the key identifier list by computing:

$$\alpha \,(\mathrm{mod}\, f(K_i,\mu)) = K_e.$$

Note that the cost of obtaining the signature key is a single network-wide broadcast of a short, $k+2$ word message[3] during the non-sleep period of each node. The computation cost is similar to that for shared-key discovery operation.

After obtaining the signature key, each node verifies the signature of the signed list of key identifiers, locates those identifiers in its key ring, and removes the corresponding keys (if any). Once the keys are removed from key rings, some links may disappear, and the affected nodes need to reconfigure those links by restarting the shared-key discovery and, possibly path-key establishment, phase for them. Because only k out of P keys are removed from the pool for every revoked node, revocation affects only a few other nodes and a small part of their key ring but it disables all connectivity of the compromised node.

C. Incremental Addition of Sensor Nodes

In our scheme, the incremental addition of sensor nodes is especially simple. That is, a new node has a key ring of k keys drawn from the same pool of P random keys loaded. Then its keys shared with the controller nodes K^{ci} are securely multicast to the controller nodes (via a similar scheme as that used by each controller to broadcast its signature key used at revocation, as discussed above) and the node is deployed. Finally, the node simply initiates shared-key discovery and, possibly path-key establishment. Network-wide broadcasts are unnecessary.

D. Re-Keying

Although it is anticipated that in most DSNs the lifetime of a key shared between two nodes exceeds that of the two nodes, it is possible that in some cases the lifetime of keys expires and re-keying must take place. Re-keying is equivalent with a self-revocation of a key by a node. As such, it does not involve any network-wide broadcast message from a controller and, hence, is especially simple. After expired-key removal, the affected nodes restart the shared-key discovery and, possibly path key establishment, phase.

E. Effect of Compromising Unshielded Sensor Nodes

Although we assume tamper-detection, sensor-node shielding that erases the keys of captured nodes, we note that our key distribution scheme is more robust than the those based on a single mission key or on pair-wise private sharing of keys even in the face of physical attacks against captured unshielded sensor nodes. In such attacks, an adversary can obtain access to a sensor's keys, and then both eavesdrop on and insert bogus messages in DSN communication. In the single mission key scheme, *all* communication links are compromised, whereas in the pair-wise private key sharing, all $n-1$ links to the captured unshielded node are compromised. In contrast, in our scheme only the $k << n$ keys of a single ring are obtained, which means that the attacker has a probability of approximately k/P to attack successfully any DSN link (viz., simulation results of Section IV). The node's shared keys with controllers could also be re-created by the adversary, but this does not affect any other sensor nodes.

[3] The alternative of broadcasting n identical lists of k identifiers encrypted in n different node keys shared with the controller would be substantially more expensive.

III. ANALYSIS

A. DSN Connectivity with Random Graphs

The limits of the wireless communication ranges of sensor nodes, not just the security considerations, preclude use of DSNs that are fully connected by shared-key links between all sensor nodes. Moreover, it is unnecessary for the shared-key discovery phase to guarantee full connectivity for a sensor node with all its neighbors in wireless communication range, as long as multi-link paths of shared keys exist among neighbors that can be used to setup a path key as needed. We use these two observations and the requirement for shared-key provisioning for new links suggested by incremental network growth, revocation and re-keying, and to establish DSN connectivity requirements based on random graph connectivity properties.

Let p be the probability that a shared key exists between two sensor nodes, n be the number of network nodes, and $d = p*(n-1)$ the degree of a node (i.e., the number of edges connecting that node with its graph neighbors). In particular, to establish the DSN shared-key connectivity, we need to answer the following two questions:

- what should the degree of a node, d, be so that a DSN of n nodes is connected? and,
- given d and the neighborhood connectivity constraints imposed by wireless communication (e.g., the number of nodes n' in a neighborhood), what values should the key ring size, k, and pool, P, have for a network of size n? In particular, if memory capacity of each sensor limits the key ring size to a given value of k, what should the size of the key pool, P, be?

Random graph theory helps answer the first question. A random graph G(n, p) is a graph of n nodes for which the probability that a link exists between two nodes is p. When p is zero the graph does not have any edge, while for p equals one, the graph is fully connected. The first question of interest to us is what is the value of p for which it is "almost certainly true" that the graph is connected.

Erdős and Rényi [10] showed that for monotone properties, there exists a value of p such that the property moves from "nonexistent" to "certainly true" in a very large random graph. The function defining p is called the *threshold function* of a property. For the property "the graph is connected," the threshold function P_c is:

$$Pc = \lim_{n \to \infty} \Pr[G(n,p) \text{ is connected}] = e^{-e^{-c}}$$

where

$$p = \frac{\ln n}{n} + \frac{c}{n}, \text{ and } c \text{ is any real constant.}$$

Therefore, given n we can find p and $d = p*(n-1)$ for which the resulting graph is connected with a desired probability P_c.

Figure 1 illustrates the plot of the degree of a node, d, as a function of the network size, n, for various values of Pc. This figure shows that, to increase the probability that a random graph is connected by one order, the degree of a node only increases by 2. Moreover, the curves of this plot are almost flat when n is large, indicating that the size of the network has insignificant impact on the degree of a node required to have a connected graph.

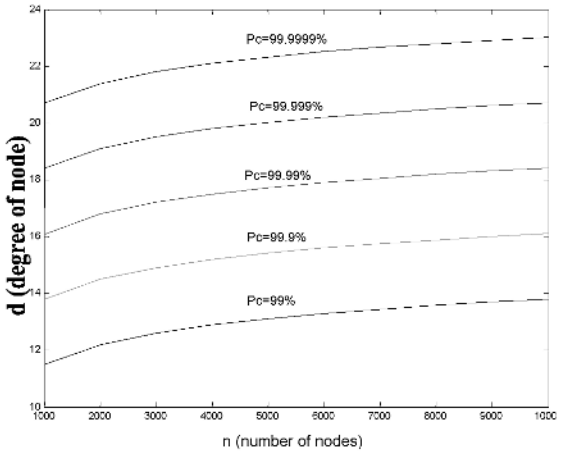


Figure 1. Degree of node vs. number of nodes where Pc=Pr[G(n,p) is connected]

To answer the second question above, we note that the wireless connectivity constraints may limit neighborhoods to $n' << n$ nodes, which implies that the probability of sharing a key between any two nodes in a neighborhood becomes $p' = d/(n'-1) >> p$. Hence, we set the probability that two nodes share at least one key in their key rings of size k chosen from a given pool of P keys to p' and then derive P as a function of k. This derivation takes into account that the size of the key pool, P, is not a sensor-design constraint. In contrast with k, which is limited by the sensor memory size, the key-pool of size P is generated and used off-line and hence can be very large. To derive the value of P, given constraint k for a p' that retains DSN connectivity with node degree d, we note that

p' = 1 – Pr[two nodes do not share any key],

and thus

$$p' = 1 - \frac{((P-k)!)^2}{(P-2k)!\,P!}$$

(viz., derivation in Appendix B). Since P is very large, we use the Stirling's approximation for n!,

$$n! \approx \sqrt{2\pi} \cdot n^{n+\frac{1}{2}} \cdot e^{-n},$$

to simplify the expression of p', and obtain:

$$p' = 1 - \frac{(1-\frac{k}{P})^{2(P\ k+\frac{1}{2})}}{(1-\frac{2k}{P})^{(P-2k+\frac{1}{2})}}.$$

Figure 2 illustrates a plot of this function for various values of P. For example, one may see that for a pool size P = 10,000 keys, only 75 keys need to be distributed to any two nodes to have the probability p = 0.5 that they share a key in their key ring. If the pool is ten times larger, namely P = 100,000, the number of keys required is 250, which is only 3.3 times the number of keys distributed in the case P = 10,000. This provides intuition for the scalability of our approach. Of course, to determine the final the size of the key ring we need to provision for additions of new nodes, revocation and re-keying. The scalablity properties of our solution indicate that such provisioning will have minimal impact on the size of key rings.

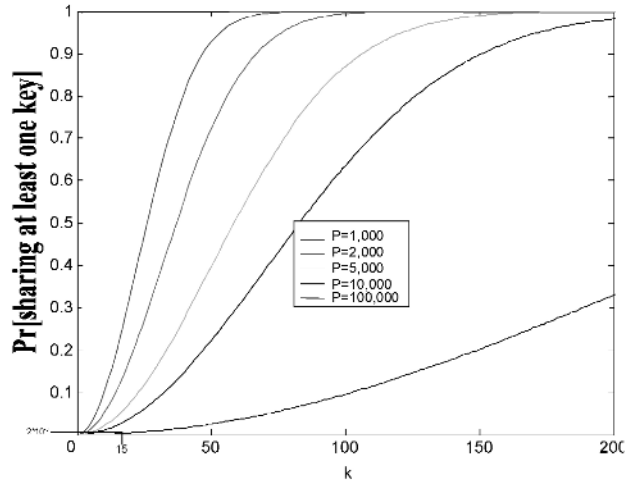


Figure 2. Probability of sharing at least one key when two nodes choose k keys from a pool of size P.

B. An example

To understand how the scheme works we present a simple numerical example. Let us assume that a DSN has n=10,000 nodes and that we want the resulting network to be connected with probability Pc = 0.99999. This means the network will "almost certainly be connected." Further, assume that that each node in the DSN has a wireless communication range that requires a neighborhood connectivity of 40 nodes.

Using the Erdős and Rényi's formula for $Pc = 0.99999 = e^{-e^{-c}}$ we find that $c = 11.5$. For this value of c we obtain $p=2*10^{-3}$ and $d = 2*10^{-3} * 9999$ other nodes. It follows that if in our network each node can communicate with 20 other nodes out of the n=10,000 nodes, the network will be (almost certainly) connected. The formula of p' above shows by setting $p' = p = 2*10^{-3}$ and selecting an especially small value of k, say k = 15, we must have a pool size P=100,000 (also viz., Figure 2). Of course, larger values of k can be accommodated by a pool size P = 100,000, as seen below.

The requirement that each neighborhood consists of n' = 40 sensor nodes implies that instead of $p = 2*10^{-3}$ we now have $p' = d/(n'-1) = 20/(40-1) \approx 0.5$. This means that either the memory size of the key ring, k, or the pool size, P, or both, must increase. For example, the formula for p' above indicates that we now need to increase the key ring size k from 15 to 250 if we intend to use the same pool size P = 100,000. Furthermore, if the neighborhood size is increased to n' = 60, then $p' = 20/(60-1) \approx 0.33$. The formula for p' above indicates that we now need only a key ring size of k = 200 for a pool size of P=100,000 keys.

IV. Simulations

We used simulations to investigate the effect of the various parameters on different DSN sizes. We were interested in understanding the efficiency and scalability of our scheme and also to characterize some parameter values that cannot be easily computed, such as the diameter of the resulting secure network.

The simulations assume a network of 1,000 nodes, having an average density of 40 sensor nodes in a neighborhood. Each simulation experiment is reproduced 10 times with different seeds for the random number generator, and the results presented represent the average values on the 10 experiments, unless otherwise noted.

A. Effect on the network topology

The fact that two nodes may not share a key during the shared-key discovery phase means that, from a network router's point of view, a link does not exist between those two nodes. This has an effect on the average path

length (number of links) between two nodes after shared-key discovery. We computed this value for various size of the key ring and show the result on Figure 3. This figure indicates that the average path length of the network depends on the size of the key ring. The smaller k is the higher the probability that a link does not have a key and, therefore, longer paths need to be found between nodes. In this example, the network gets disconnected for small k.

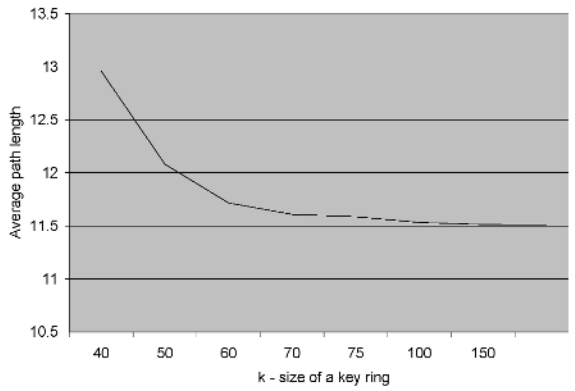


Figure 3. Average path length at the network layer

Because some links may not be keyed, a node may need to use a multi-link path to communicate with one of its wireless neighbor. Although this path would be used only once (to send the key to use for the link encryption), it should not be too long; otherwise the delay and communication cost to setup a path key with a neighbor may be high. In this example, we show how the multi-link path from a node to one of its neighbor varies with k.

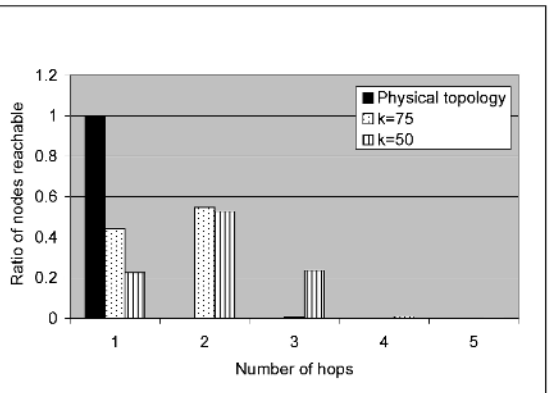


Figure 4. Path length to neighbors

Figure 4 shows us that the effect traversing multiple links to set up a path key is negligible. If a neighborhood node cannot be reached via a shared key (i.e., one link or hop), it will take at most two or three links (hops) to contact it. Since this has to be done only once to setup the path key, the effects are negligible. With $k=75$, only half of the neighbors are reachable over a single link (hop), but most of the other may be reachable over in three-link (hop) paths. While for $k=50$ only one third of the nodes are reachable over a single link, but at most four links (hops) are needed for a path to contact all of them.

B. Effect of an Attack against Unshielded Sensor Nodes

We suggested that capture of an unshielded node leads to the compromise of only k keys and that an adversary could only attack k/P links. We verified this fact by observing how many keys are used to secure links in the simulated DSN and how many links are secured with the same key.

Figure 5 shows that out of the pool of 10,000 keys, only 50% of the keys are used to secure links, only 30% are used to secure one link, 10% are used to secure two links, and only 5% are used to secure 3 links. This suggests that compromise of one key does lead to the compromise of another link with probability .3, of two other link with probability .1, and so on.

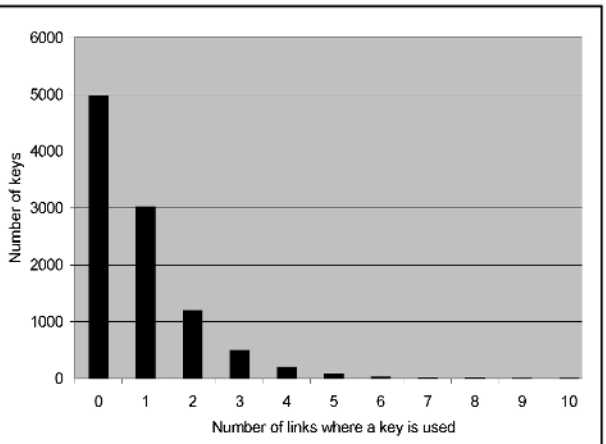


Figure 5. Usage of the key pool (size 10,000)

V. CONCLUSIONS

We presented a new key management scheme for large-scale DSNs. All such schemes must be extremely simple given the sensor-node resource limitations. Our approach is also scalable and flexible: trade-offs can be made between sensor-memory cost and connectivity, and design parameters can be adapted to fit the operational requirements of a particular environment. We illustrated the effect of the modifying design parameters using both analysis and simulations. The results indicate that our scheme is superior to the traditional key pre-distribution schemes.

ACKNOWLEDGEMENTS

REFERENCES

[1] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung, "Perfectly Secure Key Distribution for Dynamic Conferences", in Advances in Cryptology --- CRYPTO '92, Springer-Verlag, Berlin, 1993, pp. 471--486.

[2] C. Blundo, L. A. Frota Mattos and D. R. Stinson, "Tradeoffs Between Communication and Storage in Unconditionally Secure Schemes for Broadcast Encryption and Interactive Key Distribution", Lecture Notes in Computer Science 1109 (1996), pp. 387--400 (Advances in Cryptology --- CRYPTO '96).

[3] D. W. Carman, P. S. Kruus and B. J. Matt,"Constraints and Approaches for Distributed Sensor Network Security", dated September 1, 2000. NAI Labs Technical Report #00-010, available at *http://download.nai.com /products/media/nai/zip/nailabs-report-00-010-final.zip*

[4] A. Fiat and M. Naor, "Broadcast Encryption", in Advances in Cryptology --- CRYPTO '93, Springer-Verlag, Berlin, 1994, pp. 480--491.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, "System architecture directions for network sensors", Proc. of ASPLOS-IX, Cambridge, Mass. 2000.

[6] Y.-C. Hu, D. B. Johnson and A. Perrig, "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks", Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002), IEEE, Calicoon, NY, June 2002 (to appear).

[7] IBM, *IBM 4758 General Information Manual*, available at *http://www.ibm.com/security/cryptocards/*

[8] J. M. Kahn, R. H. Katz and K. S. J. Pister, "Mobile Networking for Smart Dust", ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99), Seattle, WA, August 17-19, 1999, pp. 271 – 278.

[9] J. Song, R. Poovendran, W. Trappe, and K. J. R. Liu, "A Dynamic Key Distribution Scheme Using Data Embedding for Secure Multimedia Multicast", Proceedings of SPIE Security and Watermarking for Multimedia, Vol. 4314, pg. 618-628, 2001

[10] J. Spencer, *The Strange Logic of Random Graphs, Algorithms and Combinatorics 22*, Springer-Verlag 2000, ISBN 3-540-41654-4

[11] F. Stajano, *Security for Ubiquitous Computing*, John Wiley and Sons, New York, Feb. 12, 2002, ISBN: 0-470-84493-0, 267 pp.

[12] W. Trappe, J. Song, R. Poovendran, and K. J. R. Liu, "A Dynamic Key Distribution Scheme Using Data Embedding for Secure Multimedia Multicast", Submitted to *IEEE Transactions on Multimedia*, available at http://www.eng.umd.edu/~wxt/papers/spiekey1.pdf.

[13] S.R. White and L. Comerford, ABYSS: A Trusted Architecture for Software Protection., Proc. of IEEE Security and Privacy, Oakland, CA, 1987, pp. 38-51.

APPENDIX A

In this section we briefly review the method of Trappe et al. [12] for re-keying in a broadcast environment. We adopt this method both for the discovery of shared keys and for broadcasting a controller's signature key. The security of this method relies on *parametric one-way functions* (POWFs).

Definition. *A parametric one-way function (POWF) h is a function from $K \times Y \rightarrow Z$, such that given $z = h(k, y)$ and parameter y it is computationally difficult to determine x.*

Parametric one-way functions are families of one-way functions that are parameterized by the parameter y. Such a function can be implemented by means of a symmetric encryption cipher. For example, if we let g be a suitable hash function and E_k a symmetric cipher invocation with a key k, then $h(k,y) = g(E_k(y))$ is a POWF. [Note that g need not be a cryptographic hash function.] It is easy to see that efficient POWFs that map sequences of 2B bits into a sequence of B bits exist.

We outline the construction of Trappe *et al.*'s only for signature-key broadcasting as our adaptation to shared-key discovery follows immediately.

1) Signature-key broadcasting

A POWF h that maps a sequence (k,y) of 2B bits to B bits is made available to every user. A new POWF f is defined by prepending a single 1 bit to the output of $h(k,y)$, that is $f(k,y)=1 \,||\, h(k,y)$. Assume that a controller wants to broadcast a new signature key K_e to all network nodes with which it shares keys K_i. The controller first broadcasts a B-bit random seed μ. Next, it generates the new key K_e and computes the broadcast message α as:

$$\alpha = K_e + \prod_{i=1}^{n} f(K_i, \mu).$$

Any node sharing secret key K_i with the controller may decode α by computing:

$$\alpha \,(\mathrm{mod}\, f(K_i,\mu)\,) = K_e.$$

2) Security of signature key broadcasting

We only need to show the security of the signature key distribution. Trappe *et al.* show that it is computationally difficult for a node to recover the secret key of another node from the public messages exchanged. First, they consider a variation of the controller's broadcast message:

$$\alpha = K_e + \prod_{i=1}^{n-1} K_i \,.$$

This variation requires that $K_e < min_i\{K_i\}$. This scheme

would successfully distribute the message to all nodes, however it would be possible for a node n_i to recover the another node's key K_j by computing:

$$A_i = \frac{\alpha - K_e}{K_j} = \prod_{j \neq i} K_j$$

and then factor A_i to recover the other K_j's.

The complete scheme, using function f and μ broadcast in the clear only allows an adversary to compute

$$A_i = \prod_{j \neq i} f(K_j, \mu),$$

whose factoring reveals information only about $f(K_j, \mu)$. It is computationally infeasible for an adversary to retrieve K_j given μ and $f(K_j, \mu)$.

APPENDIX B

The probability that two key rings share a key is 1 – Pr[two nodes do not share any key]. To compute the probability that two key rings do not share any key, we note that each key of a key ring is drawn out of a pool of P keys without replacement. Thus, the number of possible key rings is:

$$\frac{P!}{k!(P-k)!}$$

Pick the first key ring. The total number of possible key rings that do not share a key with this key ring is the number of key-rings that can be drawn out of the remaining P-k unused key in the pool, namely:

$$\frac{(P-k)!}{k!(P-2k)!}$$

Therefore, the probability that no key is shared between the two rings is the ratio of the number of rings without a match by the total number of rings:

$$\frac{k!(P-k)!(P-k)!}{P!k!(P-2k)!}.$$

Thus, the probability that there is at least a shared key between two key rings is:

$$p' = 1 - \frac{((P-k)!)^2}{(P-2k)!P!}.$$

Summary of the Hi-DRA Project A System for High-Speed, Wide-Area Network Detection, Response, and Analysis

Richard A. Kemmerer Giovanni Vigna
Reliable Software Group
University of California, Santa Barbara
Email: {kemm,vigna}@cs.ucsb.edu

Antonio Carzaniga Alexander L. Wolf
University of Colorado
Boulder, Colorado
Email: {carzanig,alw}@cs.colorado.edu

Abstract

The goal of the Hi-DRA project is to develop a system for network surveillance, analysis, and response for high-speed, wide-area networks. The Hi-DRA approach advocates that the protection of the network infrastructure must rely on *local surveillance* and *global coordination and control*. Local surveillance addresses the problem of securing a protection domain by means of proactive security measures, extensive monitoring, and real-time response. Global coordination and control enables the correlation of security-related information coming from different subsystems to obtain a global view of the security state of the infrastructure. To achieve its goals, the Hi-DRA approach relies on five key concepts: high-speed network monitoring and analysis; highly configurable intrusion detection sensors; comprehensive correlation; advanced network modeling; and a scalable infrastructure for global command and control. This paper presents an overview of the Hi-DRA project and provides a description of key ideas underlying the approach.

Keywords: *Intrusion Detection, Misuse Detection, Anomaly Detection, Alert Correlation, Security, Computer Security, Network Security, High-Speed Networks.*

I. Introduction

The incredible growth of networking technology has been driven by the immediate need for higher bandwidths and advanced services. This trend has delivered a number of promising results, but has also produced solutions that are only partially satisfactory. Security is one aspect that has often been overlooked in the design and implementation of new network services. The pressure created by users' needs and the reduced time-to-market has not allowed for thorough security analysis and the sound design of protocols, services, and applications.

In the last ten years new mechanisms have been designed and implemented to support privacy and integrity of network services. Examples of these solutions are firewalls [9], [10], virtual private networks (VPNs) [19], and intrusion detection systems (IDSs) [1]. Even though they represent a first step towards improved security, the solutions proposed so far suffer from a number of limitations in terms of performance, configurability, completeness, and scalability. The research carried out on the Hi-DRA project at the University of California, Santa Barbara (UCSB) and at the University of Colorado, Boulder (UCB) over the last four years addresses these limitations. Hi-DRA is a network surveillance, analysis, and response infrastructure for high-speed, wide-area networks.

a) Addressing performance issues in high-speed networks: As networks become faster, delivering bandwidths of gigabits per second to end nodes, there is a need for new techniques that can keep up with the increased network throughput. Existing network sensors can barely keep up with bandwidths of hundreds of Mbps. Furthermore, there is also a need to better handle attacks directed at the network infrastructure. More efficient monitoring solutions based on load balancing techniques have been proposed, but these solutions concentrate only on the traffic load and do not take into account the target network's characteristics and the parameters of the security analysis being performed on the network data. A more

effective approach would rely on semantic-rich descriptions of both the protected targets (e.g., what services are deployed and what the network topology is) and the intrusion patterns (e.g., what type of events may be involved in the attacks to be detected).

To be able to perform a broader range of traffic analysis in high-speed networks, we have developed an intelligent network traffic partitioning technique that enables parallel execution of the network analysis process. Our partitioning approach to network security analysis supports in-depth, stateful intrusion detection on high-speed links. The approach is centered around a *slicing* mechanism that divides the overall network traffic into subsets of manageable size. The traffic partitioning is done so that a single slice contains all the evidence necessary to detect a specific attack, making sensor-to-sensor interactions unnecessary.

b) Addressing configurability issues of surveillance sensors: The protection of the national information infrastructure requires a means to dynamically configure the infrastructure's security posture as a reaction to detected attacks. This reaction may vary from recording information, to raising the level of concern, to rerouting traffic, etc. For this reason, there is a need for the dynamic reconfiguration of surveillance sensors in terms of positioning of the sensors, types of attacks to be detected, and the types of response.

We have developed a highly configurable "web of sensors" that supports dynamic configuration. Our web of sensors research built on the State Transition Analysis Technique (STAT) [21] previously developed at UCSB. The STAT approach describes computer attacks at a high level. Attack scenarios are abstracted into *states*, which describe the security status of a system, and *transitions*, which model the evolution between states. By modeling an attack as an evolving scenario it is possible to preempt ongoing attacks and perform responses in real-time. To address the complexity of intrusion detection infrastructures, we developed the STAT framework, which overcomes the limitations of current approaches. Instead of providing yet another system tailored to some domain-specific requirements, STAT provides a software framework for the development of new intrusion detection functionality in a modular fashion. When using the STAT framework, intrusion detection sensors are built by dynamically composing domain-specific components with a domain-independent runtime. The resulting intrusion detection sensors represent a software family. Each sensor has the ability to reconfigure its behavior dynamically. The reconfiguration functionality is supported by a component model and by a control infrastructure, called MetaSTAT. The final product of the STAT framework is a highly-configurable, well-integrated intrusion detection infrastructure.

c) Addressing alert correlation issues: As more IDSs are developed, network security officers (NSOs) are faced with the task of analyzing an increasing number of alerts resulting from the analysis of different event streams. In addition, IDSs are far from perfect and may produce both false positives and non-relevant positives. Non-relevant positives are alerts that correctly identify an attack, but the attack fails to meet its objective. For instance, the attack may be exercising a vulnerability in a service that is not provided by the victim host. As an example, consider a "Code Red" worm that attacks a Linux Apache server. Although an actual attack is seen on the network, this attack will fail, because Apache is not vulnerable to the exploit utilized by the worm. Clearly, there is a need for tools and techniques that allow an NSO to aggregate and combine the outputs of multiple IDSs, filter out spurious or incorrect alerts (such as "Code Red" attacks against Apache installations), and provide a high-level view of the security state of the protected network.

We have developed an alert correlation process that produces a succinct overview of security-related activity on the network. Our process uses a set of components that focus on different aspects of the overall correlation task. We have also designed and implemented a tool, called AlertSTAT, which uses the MetaSTAT infrastructure to collect the alerts produced by the intrusion detection sensors and then performs correlation on the resulting data using a number of different components.

d) Addressing network security modeling issues: Security analysis should take advantage of a reliable knowledge base that contains semantically-rich information about a protected network. Unfortunately,

existing network security solutions rely on incomplete data models. Networks are complex systems and most approaches oversimplify their target models in an effort to limit the problem space. For example, firewall technologies often consider a limited subset of security mechanisms and do not take into account specific characteristics of the network being protected, such as the interactions among different protocols and services, or the particular architecture and operating systems installed on the protected nodes. In addition, network management protocols (e.g., SNMP) do not consider important aspects of the managed hosts such as trust relationships or protection domains. There is a need for a comprehensive model that can be the basis for coordinated security analysis and infrastructure status monitoring.

We have developed a network reference model and a tool based on the model to support sophisticated network discovery, validation, and analysis. The model is not limited to a specific network level, but integrates network information throughout the layers. The model contains information about topology, infrastructure, and deployed services. In addition, the relationships among different entities in different layers of the model are made explicit. For example, the model includes trust relationships between service clients and servers, as well as relationships between services and configuration objects (e.g., files) used to define the application behavior.

The model served as the basis for NetMap, which is a security tool for network modeling, discovery, and analysis. The modeled information is managed by using a suite of composable network tools that can determine various aspects of network configurations through scanning techniques and heuristics. Tools in the suite are responsible for a single, well-defined task. Each tool has an abstract specification of the input, the output, the type of processing, and the requirements for carrying out a task. Tool descriptions are expressed in a Network Tool Language. The tool descriptions are then stored in a database. By using the network model and the tool descriptions, NetMap is able to automatically determine which tools are needed to perform a particular complex task and how the tools should be scheduled to obtain the requested results.

e) Addressing scalability and survivability issues for global command and control: Surveillance of protected networks is the basis for securing the information infrastructure, but there is also the need to take into account infrastructure-wide security issues. This was made clear in the many distributed denial-of-service attacks, which were coordinated aggressions targeting a number of different companies. The attacks were detected singularly by each target site and the tasks of analysis and response were carried out locally. Coordination among the attacked sites, when there was any, was performed through an *ad hoc* spontaneous network composed of system administrators communicating through email (when possible) or by phone. It is clear that in order to be able to assess and respond to more serious attacks, the scope of the deployed infrastructure protection mechanisms should be national and even international.

An emergency response capability with international scope can only be realized if the protection mechanisms can take advantage of a scalable communication and coordination infrastructure. Existing solutions have evolved from local-area network scale to enterprise-wide scale by extending and replicating local solutions. This approach has already shown its limitations and cannot be pursued further for the creation of an Internet-scale protection infrastructure. New general, scalable mechanisms must be devised to support coordination and communication on a larger scale.

We developed a ubiquitous event notification service as the basis for communication and coordination among the components of the protection infrastructure. To realize this service, we adapted and extended Siena [3], [4], which is a publish/subscribe event notification service developed at UCB. Siena is designed specifically to operate in the context of wide-area networks such as the Internet. Scalability is achieved by performing efficient content-based routing of alerts and control messages. In particular, Siena takes advantage of commonalities among subscriptions to reduce the network and computational cost of propagating notifications from publishers to subscribers. In using Siena as the Internet-scale event notification infrastructure, we employed the network reference model described above as a common ontology for fusing and correlating alerts coming from different sensors in different network domains. Siena offers

an expressive and generic data model on top of which we instantiated alerts and control messages in compliance with this common ontology. In addition to employing the shared data model in this way, we also extended Siena to recognize a wider range of event patterns in order to support the kinds of alert fusion and correlation required by the data model.

This paper is intended as a summary of the research carried out as part of the Hi-DRA project. The material presented in this paper has appeared in other publication venues, all of which are referenced or cited here. The five reprinted articles that accompany this summary [8], [14], [20], [23], [24] are a good starting point to understand the details of the Hi-DRA research project.

The remainder of this paper is organized in five sections, which overview the five approaches presented above. Each section also gives a pointer to where more complete information can be obtained on the research being summarized. More specifically, Section II presents our approach to high-speed monitoring and analysis, Section III presents the web of sensors, Section IV presents our component-based approach to alert correlation, Section V overviews our advanced network modeling approach, and Section VI presents our work on information dissemination and classification. The final section of the paper summarizes the papers that we have published in each of these areas.

II. High Speed Monitoring and Analysis

To perform in-depth, stateful analysis it is necessary to divide the traffic volume into smaller portions that can be thoroughly analyzed by intrusion detection sensors. This approach has often been advocated by the high-performance research community as a way to distribute the service load across many nodes. In contrast to the case for standard load balancing, the division (or slicing) of the traffic for intrusion detection has to be performed in a way that guarantees the detection of all the threat scenarios considered. If a random division of traffic is used, sensors may not receive sufficient data to detect an intrusion, because different parts of the manifestation of an attack may have been assigned to different slices. Therefore, when an attack scenario consists of a number of steps, the slicing mechanism must assure that all of the packets that could trigger those steps are sent to the sensor configured to detect that specific attack.

The problem of intrusion detection analysis in high-speed networks can be effectively attacked only if a scalable solution is available. We specified the requirements for a scalable solution in [14]. These requirements were used as the basis for the design of our network-based intrusion detection system. The system consists of a *network tap*, a *traffic scatterer*, a set of m *traffic slicers*, a *switch*, a set of n *stream reassemblers*, and a set of p *intrusion detection sensors*.

The network tap component monitors the traffic stream on a high-speed link. Its task is to extract the sequence F of link-layer frames $\langle f_0, f_1, ..., f_l \rangle$ that are observable on the wire during a time period. This sequence of frames is passed to the scatterer which partitions F into m sub-sequences $F_j : 0 \leq j < m$. Each F_j contains a (possibly empty) subset of the frame sequence F. Every frame is an element of exactly one sub-sequence and therefore $\cup_{j=0}^{j<m} F_j = F$. The scatterer can use any algorithm to partition F. Hereafter, it is assumed that the scattering algorithm simply cycles over the m sub-sequences in a round-robin fashion. As a result, each F_j contains an m-th of the total traffic.

Each sub-sequence F_j is transmitted to a different traffic slicer S_j. The task of the traffic slicers is to route the frames they receive to the sensors that may need them to detect an attack. This task is not performed by the scatterer, because frame routing may be complex, requiring a substantial amount of time, while the scatterer has to keep up with the high traffic throughput and can only perform very limited processing per frame.

The traffic slicers are connected to a switch component, which allows a slicer to send a frame to one or more of n outgoing channels C_i. The set of frames sent to a channel is denoted by FC_i. Each channel C_i is associated with a stream reassembler component R_i and a number of intrusion detection sensors. The set of sensors associated with channel C_i is denoted by IC_i. All the sensors that are associated with a channel are able to access all the packets sent on that channel. A problem that could occur with this

approach is that the original order of two packets could be lost if the two frames took different paths over distinct slicers to the same channel. Therefore, the reassemblers associated with each channel make sure that the packets appear on the channel in the same order that they appeared on the high-speed link. That is, each reassembler R_i must make sure that for each pair of frames $f_j, f_k \in FC_i$, (f_j before f_k) $\iff j < k$.

Each sensor component I_j is associated with q different attack scenarios, and each attack scenario has an associated *event space*. The event space specifies which frames are candidates to be part of the manifestation of the attack. For example, consider an attack targeting a Web server called `dino` within the network protected by the intrusion detection system. In this case, the event space for that attack is composed of all the TCP traffic that involves port 80 on host `dino`.

Event spaces are expressed as disjunctions of *clauses*, where each clause is an expression of the type xRy. x denotes a value derived from the frame (e.g., a part of the frame header) while R specifies an arithmetic relation (e.g., =, ! =, <). y can be a constant, the value of a variable, or a value derived from the same frame. Clauses and event spaces may be derived automatically from the attack descriptions, for example from signatures written in attack languages such as Bro [16], Sutekh [17], STATL [11], or Snort [18].

The effectiveness of the scatterer/slicer/ reassembler architecture were experimentally evaluated on a system using three traffic slicers ($m = 3$) and four stream reassemblers ($n = 4$) with one intrusion detection sensor per stream with favorable results. The details of the implementation and of these experiments can be found in the accompanying reprint [14].

III. Web of Sensors

We have developed a novel approach to distributed intrusion detection. The idea is that a protected network is instrumented with a "web of sensors" composed of distributed components integrated by means of a local communication and control infrastructure. The task of the web of sensors is to provide fine-grained surveillance inside the protected network. The web of sensors implements local surveillance against both outside attacks and local misuse by insiders in a way that is complementary to the mainstream approach where a single point of access (e.g., a gateway) is monitored for possible malicious activity. The outputs of the sensors, in the form of alerts, are collected by a number of "meta-sensor" components. Each meta-sensor is responsible for a subset of the deployed sensors, and may coordinate its activities with other meta-sensors. The meta-sensors are responsible for storing the alerts, for routing alerts to other sensors and meta-sensors (e.g., to perform correlation to identify composite attack scenarios), and for exerting control over the managed sensors.

Control is the most challenging (and most overlooked) functionality of distributed surveillance. Most existing approaches simply aggregate the outputs of distributed sensors and focus mainly on the intuitive presentation of alerts to the network security officer. This is not enough. There is a need for fine-grained control of the deployed sensors in terms of scenarios to be detected, tailoring of the sensors with respect to the protected network, and dynamic control over the types of response. These are requirements that can be satisfied only if the surveillance sensors are *highly configurable* and if configuration can be performed dynamically, without stopping and restarting sensors when a reconfiguration is needed.

We have designed a suite of highly configurable surveillance sensors and a command and control meta-sensor that allows the network security officer to exert a very fine-grained control over the deployed surveillance infrastructure. Meta-sensors can be organized hierarchically to achieve scalability and can be replicated to support fault-tolerance. This web of sensors is built around the State Transition Analysis Technique (STAT) framework developed by the Reliable Software Group at UCSB. The STAT framework provides a platform for the development of highly configurable probes in different domains and environments. The resulting set of applications is a software family whose members share a number of features, including dynamic reconfigurability and a fine-grained control over a wide range of characteristics [22].

The STAT approach is centered around five key concepts: the STAT technique, the STATL language, the STAT Core, the CommSTAT communication infrastructure, and the MetaSTAT control system.

The approach provides the basic mechanisms to reconfigure, at run-time, which input event streams are analyzed by each sensor, which scenarios have to be used for the analysis, and what types of responses must be carried out for each stage of the detection process. In addition, the approach explicitly models the dependencies among the modules composing a sensor so that it is possible to automatically identify the steps that are necessary to perform a reconfiguration of the deployed sensing infrastructure. In addition, the possibility of retrieving current configurations from remote sensors allows one to determine if a reconfiguration is valid or meaningful.

The STAT framework is the only known framework-based approach to the development of intrusion detection systems. Our experience with the framework shows that by following this approach it is possible to develop intrusion detection systems with reduced development effort, with respect to an *ad hoc* approach. In addition, the approach is advantageous in terms of the increased reuse that results from using an object-oriented framework and a component-based approach. The details of the web of sensors approach can be found in the accompanying reprint [23].

IV. Alert Correlation

To address the issue of analyzing a large number of alerts, researchers and vendors have proposed *alert correlation*, which is an analysis process that takes the alerts produced by intrusion detection systems and produces compact reports on the security status of the network under surveillance. Although a number of correlation approaches have been suggested, there is no consensus on what this process is or how it should be implemented and evaluated. In particular, most existing correlation approaches operate on only a few aspects of the correlation process, such as the fusion of alerts that are generated by different intrusion detection systems in response to a single attack, or the identification of multi-step attacks that represent a sequence of actions performed by the same attacker. In addition, correlation tools that do cover multiple aspects of the correlation process are evaluated "as a whole," without an assessment of the effectiveness of each component of the analysis process. As a result, it is not clear if and how the different parts of the correlation process contribute to the overall goals of correlation.

Another problem is that many existing approaches operate under the assumption that the alert stream is composed of detections of successful attacks. Unfortunately, experience shows that this assumption is wrong. Intrusion detection systems are very noisy, and, in addition to false positives, they produce alerts with different levels of relevance. As a consequence, the effectiveness of alert correlation is negatively affected by the poor quality of the input alert stream.

The process of alert correlation consists of a collection of components that transform intrusion detection sensor[1] alerts into intrusion reports. Because alerts can refer to different kinds of attacks at different levels of granularity, the correlation process cannot treat all alerts equally. Instead, it is necessary to provide a set of components that focus on different aspects of the overall correlation task.

The AlertSTAT correlation tool that we implemented uses a pipeline architecture, consisting of ten different components. The process starts with a *normalization* component that translates every alert that is received into a standardized format that is understood by all other correlation components. This is necessary because alerts from different sensors can be encoded in different formats. Next, a *pre-processing* component augments the normalized alerts so that all required alert attributes (such as start-time, end-time, source, and target of the attack) are assigned meaningful values.

The core of the correlation process consists of components that implement specific functions, which operate on different spatial and temporal properties. That is, some of the components correlate events that occur close in time and space (e.g., alerts generated on one host within a small time window), while

[1] We use *intrusion detection system* and *intrusion detection sensor* (or just *sensor*) interchangeably.

others operate on events that represent an attack scenario that evolves over several hours and that includes alerts that are generated on different hosts (e.g., alerts that represent large-scale scanning activity). It is natural to combine events that are closely related (spatially and temporally) into composite alerts, which are in turn used to create higher-level alerts.

The next four correlation components of our process all operate on single, or closely related, events. The *fusion* component is responsible for combining alerts that represent the independent detection of the same attack instance by different intrusion detection systems. The *verification* component takes a single alert and determines the success of the attack that corresponds to this alert. The idea is that alerts that correspond to failed attacks should be appropriately tagged and their influence on the correlation process should be decreased. The *thread reconstruction* component combines a series of alerts that refer to similar attacks launched by a single attacker against a single target. The *attack session reconstruction* component associates network-based alerts with host-based alerts that are related to the same attack.

The next two components in our process operate on alerts that involve a potentially large number of different hosts. The *focus recognition* component has the task of identifying hosts that are either the source or the target of a substantial number of attacks. This is used to identify denial-of-service (DoS) attacks or port scanning attempts. The *multi-step correlation* component has the task of identifying common attack patterns, such as an island-hopping attack (i.e., an attack where an intruder breaks into a host and uses it as a launch pad for more attacks). These patterns are composed of a sequence of individual attacks, which can occur at different points in the network.

The final components of the correlation process contextualize the alerts with respect to a specific target network. The *impact analysis* component determines the impact of the detected attacks on the operation of the network being monitored and on the assets that are targeted by the malicious activity. The results of this analysis are then used by the *prioritization* component to assign an appropriate priority to every alert. This priority information is important for quickly discarding information that is irrelevant or of less importance to a particular site.

In its current configuration, alerts that are correlated by one AlertSTAT component are input to the next component. Although this process is sequential, all alerts are not required to pass through components sequentially. Some components could operate in parallel, and it is even possible that alerts output by a sequence of components could be fed back as input to a previous component of the process.

We have applied our correlation tool to a large number of diverse data sets, to analyze if and how each component contributes to the overall correlation process. These experiments demonstrate that the effectiveness of each component of the process is dependent on the data sets being analyzed. To be more precise, these experiments show that the performance of the correlation process is significantly influenced by the network topology, the characteristics of the attack, and the available meta-data. Thus, one cannot, in general, determine a ranking among components with respect to their effectiveness. The details of the AlertSTAT tool and of the experiments can be found in the accompanying reprint [20].

V. Advanced Network Modeling and Analysis

Network security is achieved by composing the functionality of a number of security applications, such as firewalls and intrusion detection systems. Deploying and configuring security applications requires an in-depth knowledge of the network to be protected. In addition, continuous monitoring of both the network and the configuration of the security applications is the basis for determining the current network security posture.

Unfortunately, knowledge about the network being protected often exists only in the "mind" of the network administrator, and this knowledge is obtained by using a number of tools, each of which can only provide a subset of the information about the protected network. For example, the information about the services active on a host could be determined by scanning the ports of the host. In addition, the results obtained from the execution of one tool are often used as the basis for additional analysis and

possibly as input for the execution of other tools. In the previous example, once the open ports have been determined, banner-grabbing tools can help to determine the type and version of the server applications. The coordination of tool executions and the composition of their results is usually a human-intensive task. This is the case even when ad-hoc scripts and procedures developed by network administrators through years of experience in integrating the results of network monitoring and analysis are available.

NetMap is a novel approach that provides support for automated network discovery and security analysis. NetMap is centered around a model of both the network to be analyzed and the tools to be used for analysis. The network model has been designed by taking into account the models used by existing network management and vulnerability scanning tools. The model is not limited to a specific network level; it integrates network information throughout the layers, and it contains information about topology, infrastructure, and deployed services. In addition, the security-relevant relationships between different entities in different layers of the model are made explicit. For example, the model includes trust relationships between clients and servers for specific services, as well as relationships between services and configuration objects (e.g., files) used to define the application behavior. The network model is implemented as a database management system, called NetDB.

In addition to the network model, we have developed a tool model that supports the abstract description of a suite of network discovery and scanning tools using a Network Tool Language (NTL). Each tool in the suite is responsible for a single, well-defined task. Each tool has a specification of the input, the output, the type of processing, and the requirements for carrying out a task. The tool descriptions are stored in a tool repository, called the Network Tool Database (NTDB).

NetMap allows a network administrator to specify high-level discovery/analysis tasks in a query language, called NetScript. Tasks range from pure network discovery, to the validation of existing information, to vulnerability scanning. Given a task description, a query processor component uses the tool descriptions to determine which tools are needed to perform a particular complex task, what their schedule should be, and how the results should be inserted into an instance of the network model that represents the protected network. The details of the NetMap tool can be found in the accompanying reprint [24].

In addition to the NetMap approach, we developed a number of tools to support the monitoring and analysis of the network routing infrastructure. These tools analyze the events associated with the reconfiguration of traffic distribution to identify both anomalous behavior and inconsistencies [13], [15].

VI. Information Dissemination and Classification

Our work on information dissemination and classification concentrated on integrating intrusion-detection sensors through a scalable, loosely-coupled communication service, called Siena, and on developing a generic, flexible, and high-performance data classifier called SFF. These are discussed in the following two subsections.

A. High-Level Integration of Intrusion-Detection Sensors

Enterprise-wide dynamic response and countermeasure requires a non-traditional communication mechanism to distribute alerts, and to control and configure sensors deployed throughout the network. The Siena content-based communication service [5] is ideally suited to this application, since it has been designed specifically to support loosely coupled communication among components distributed across a wide-area network. Moreover, it has been designed for scalability in terms of the number of participants and the number of messages they exchange.

The model of communication underlying Siena is *publish/subscribe*, whereby the service delivers messages, not through an explicit destination-addressing scheme as in a point-to-point service, but instead by matching the content of messages generated by publishers against the interests expressed by subscribers. Thus, the service does not require the sender to know the set of receivers, and in fact, that set can vary over time without any intervention by the sender.

A message carries information in the form of a set of attribute/value pairs. For example, the message displayed in Figure 1a represents a portscan alarm. A *filter*, which represents the interests of a subscriber, selects messages by specifying a conjunction of constraints on the values of some attributes. The operators available for use within filters include all the common equality and ordering relations ($=$, $\neq$, $<$, $>$, etc.), the substring ($*$), prefix ($>*$), and suffix ($*<$) operators for strings, and an operator *any* that matches any value. With these selection operators, Siena achieves the expressiveness of a significant subset of SQL. Figure 1b shows a filter that matches any security alarm for hosts "*.bar.com".

```
string     class = network/security/alarm
time        date = Mar 4 11:43:37 MST 1998
string    tohost = foo.bar.com
string  fromhost = pub32.internetcafe.com
string    attack = portscan
```

(a)

```
string   class   >*   network/security/
string   tohost   *<   .bar.com
string   attack   any
```

(b)

Fig. 1. Example of a Message (a) and a Filter (b).

Siena comes with a complete, fully-documented API, with language mappings for Java and C++. This API allows applications to publish and subscribe, and also offers some value-added services, including message buffers and support for mobile client applications [2]. In addition to this primary, functional interface, Siena implements a management interface that allows applications to remotely control some server parameters and operations.

These two interfaces are the basis for the integration of higher-level components of the Hi-DRA infrastructure. In particular, a Siena client can be used as an alert generator within the STAT framework (see Section III). This is done using a module called SienaSTAT, which feeds Siena-based IDMEF alert messages into the AlertSTAT correlator. SienaSTAT uses a Siena-XML interface API called SXML. This value-added API automates the process of publishing XML documents (such as IDMEF alerts) as Siena messages. Applications can subscribe for XML information by posing subscriptions in the form of XPath expressions, as well as publish XML documents. The XML publication mechanism and XPath subscription mechanism are completely parameterized and integrated. This means that both processes are controlled by a set of translation rules that determine how XML data are represented within a Siena message.

Siena was conceived as a federation of servers, each of which is responsible for dispatching messages passed to it either from clients or from other servers. However, its design has inspired the development of a true network architecture that we call *content-based networking* [7]. The service interface of this network eschews receiver addresses in favor of more powerful *receiver predicates* (disjunctions of what are called "filters" in Siena), with the flow of messages from senders to receivers determined by the evaluation of message content against receiver predicates. As with any network-level service, the fundamental challenge is the design of highly efficient and reliable routing and forwarding functions to support message delivery. The routing function we developed is based on the use of symmetric broadcast trees pruned by content-based predicates [6]. The forwarding function provides a means to index predicates so that the content of messages can be quickly compared to large numbers of predicates and thereby determine to which next-hop router a given message should be forwarded. Content-based networking provides an advanced infrastructure upon which to implement a publish/subscribe service, as well as related high-level services such as query/advertise [12]. The details of content-based forwarding can be found in the accompanying reprint [8].

B. High-Performance Classification

The problem of forwarding in a content-based network turns out to share many characteristics with the problem of data or packet classification found in the domain of intrusion detection. In this context, the

forwarding function can be used to enhance the capabilities of the high-performance monitor architecture discussed in Section II.

Recall that in the monitor architecture a traffic *scatterer* distributes network packets to a battery of traffic *slicers* using a simple round-robin scheduling policy. The advantage of this design is that the most performance-critical component of the architecture can be implemented using a simple algorithm and, in fact, this component can be easily implemented using specialized hardware. The disadvantage is that related packets (i.e., packets that must be taken together to match a specific attack signature) must be eventually processed by a single sensor, which in turn means that traffic flows that are initially spread out by the round-robin scatterer, must be reassembled and reordered in the correct arrival sequence at a later stage.

To overcome the need to reassemble, we can instead treat the problem as a flow problem, sieving traffic through a hierarchical network of *splitters* (rather than "scatterers"). In this architecture, each splitter makes an informed decision as to where to forward each packet, either dropping the packet or forwarding it along to one or more downstream splitters or sensors. The rules that determine the behavior of a splitter are derived from a combination of the rules defined by the IDS sensors downstream, with the root splitter using a few simple rules, and with other splitters at each level applying progressively refined and more specific rules. Intuitively, a splitter achieves its ideal effectiveness in filtering traffic when its rules partition its input stream evenly over its output (downstream) links.

The implementation of each splitter is based on the algorithms developed for the content-based forwarding function, together with network-specific input and output interfaces. The input interface presents network packets as messages, each one containing a set of attribute/value pairs, to the forwarding engine. For example, the network message interface translates packet data into attributes such as `ARP.Sender.IP`, `IP.Source`, `IP.Protocol`, `ICMP.Type`, and `TCP.SourcePort`. The output interface operates simply by forwarding the packet to one or more other splitters or sensors for further processing. Each splitter is loaded with a set of rules expressing conditions over the attributes.

The splitter algorithm, which we refer to as SFF, is a generic *multiclassifer* in that it can be used to classify a stream of input data of any kind (e.g., network packets, security alerts, log entries, and practically anything that can be expressed as a set of attributes and values) into any number of possibly overlapping categories.

Of course, genericity comes at a price. Our goal was to support gigabit and greater line speeds, so a software implementation on a commodity platform, such as FreeBSD on x86, was limiting the performance of the algorithm, especially when applying the algorithm to network intrusion detection. The time spent in kernel space, instead of running the classification algorithm, mostly consisted of managing network I/O buffers, which dominated the processing time for packets. Thus, we designed and developed a port of the SFF algorithm to the Intel IXP 1200 network processor.

VII. Papers Published

A. High Speed Monitoring and Analysis

- C. Kruegel, F. Valeur, G. Vigna, and R.A. Kemmerer, **Stateful Intrusion Detection for High-Speed Networks**, in *Proceedings of the IEEE Symposium on Research on Security and Privacy*, pp. 285-293, IEEE Press, Oakland, CA, May 2002.
- C. Kruegel and T. Toth, **Using Decision Trees to Improve Signature-based Intrusion Detection**, in *Proceedings of the 6th Symposium on Recent Advances in Intrusion Detection (RAID)*, Lecture Notes in Computer Science, Springer Verlag. Pittsburgh, PA, September 2003.

B. Web of Sensors

- G. Vigna, R.A. Kemmerer, and P. Blix, **Designing a Web of Highly-Configurable Intrusion Detection Sensors**, in *Proceedings of the 4th International Symposium on Recent Advances in*

Intrusion Detection (RAID 2001), pp. 69-84, LNCS 2212, Springer-Verlag, Davis, CA, October 2001.

- R.A. Kemmerer and G. Vigna, **Intrusion Detection: A Brief History and Overview**, in *IEEE Computer, Special Issue on Security and Privacy*, pp. 27-30, IEEE Press, April 2002.
- G. Vigna, B. Cassell, and D. Fayram, **An Intrusion Detection System for Aglets**, in *Proceedings of the International Conference on Mobile Agents (MA '02)*, pp. 64-77, LNCS 2535, Springer-Verlag, Barcelona, Spain, October 2002.
- S.T. Eckmann, G. Vigna, and R.A. Kemmerer, **STATL: An Attack Language for State-based Intrusion Detection** in *Journal of Computer Security*, vol. 10, no. 1/2, pp. 71-104, IOS Press, 2002.
- S. Soman, C. Krintz, and G. Vigna, **Detecting Malicious Java Code Using Virtual Machine Auditing**, in *Proceedings of the 12th USENIX Security Symposium*, pp. 153-167, Washington, DC, August 2003.
- G. Vigna, F. Valeur, and R.A. Kemmerer, **Designing and Implementing A Family of Intrusion Detection Systems**, in *Proceedings of the European Conference on Software Engineering (ESEC)*, Helsinki, Finland, September 2003.
- G. Vigna, W. Robertson, V. Kher, and R.A. Kemmerer, **A Stateful Intrusion Detection System for World-Wide Web Servers**, in *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pp. 34-43, Las Vegas, NV, December 2003.
- J. Zhou and G. Vigna, **Detecting Attacks That Exploit Application-Logic Errors Through Application-Level Auditing**, in *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pp. 168-178, Tucson, AZ, December 2004.
- G. Vigna, S. Gwalani, K. Srinivasan, E. Belding-Royer, and R. Kemmerer, **An Intrusion Detection Tool for AODV-based Ad Hoc Wireless Networks**, in *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pp. 16-27, Tucson, AZ, December 2004.
- R.A. Kemmerer and G. Vigna, **Sensor Families for Intrusion Detection Infrastructures**, in *Managing Cyber Threats: Issues, Approaches and Challenges*, Springer-Verlag, January 2005.

Alert Correlation

- F. Valeur, G. Vigna, C. Kruegel, and R. Kemmerer, **A Comprehensive Approach to Intrusion Detection Alert Correlation**, in *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 3, pp. 146-169, July-September 2004.
- C. Kruegel and W. Robertson, **Alert Verification - Determining the Success of Intrusion Attempts**, in *Proceedings of the Workshop on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, Germany, July 2004.
- C. Kruegel, W. Robertson, and G. Vigna, **Using Alert Verification to Identify Successful Intrusion Attempts**, in *Practice in Information Processing and Communication (PIK)*, vol. 27, no. 4, pp. 219-227, October/December, 2004.
- C. Kruegel and F. Valeur, and G. Vigna, **Intrusion Detection and Correlation: Challenges and Solutions**, Springer-Verlag, ISBN 0-387-233398-9, 2005.

Advanced Network Modeling and Analysis

- G. Vigna, F. Valeur, J. Zhou, and R.A. Kemmerer, **Composable Tools For Network Discovery and Security Analysis**, in *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pp. 14-24, IEEE Press, Las Vegas, NV, December 2002.
- V. Mittal and G. Vigna, **Sensor-Based Intrusion Detection for Intra-Domain Distance-Vector Routing**, in *Proceedings of the ACM Conference on Computer and Communication Security (CCS'02)*, pp. 127-137, ACM Press, Washington, DC, November 2002.
- G. Vigna and A. Mitchell, **Designing and Implementing Network Short-Term Memory**, in *Proceedings of ICECCS '02*, pp. 91-100, IEEE Press, Greenbelt, MD, December 2002.

- G. Vigna, **A Topological Characterization of TCP/IP Security**, in *Proceedings of the 12th International FME Symposium*, pp. 914-940, LNCS 2805, Springer-Verlag, Pisa, Italy, September 2003.
- C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, **Topology-based Detection of Anomalous BGP Messages**, in *Proceedings of the 6th Symposium on Recent Advances in Intrusion Detection (RAID)*, Lecture Notes in Computer Science, Springer Verlag. USA, September 2003.

E. Information Dissemination and Classification

- A. Carzaniga, M.J. Rutherford, and A.L. Wolf, **A Routing Scheme for Content-Based Networking**, in *Proceedings of IEEE INFOCOM 2004*, Hong Kong, China, March 2004.
- M. Caporuscio, A. Carzaniga, and A.L. Wolf, **Design and Evaluation of a Support Service for Mobile, Wireless Publish/Subscribe Applications**, in *IEEE Transactions on Software Engineering*, vol. 29, no. 12, pp. 1059–1071, December 2003.
- N. Arshad, D. Heimbigner, and A.L. Wolf, **Deployment and Dynamic Reconfiguration Planning for Distributed Software Systems**, in *15th International Conference on Tools with Artificial Intelligence (ICTAI '03)*, pp. 39–46, Sacramento, California, November 2003.
- A. Carzaniga and A.L. Wolf **Forwarding in a Content-Based Network**, in *Proceedings of ACM SIGCOMM 2003*, p. 163–174. Karlsruhe, Germany, August 2003.
- M. Castaldi, A. Carzaniga, P. Inverardi and A.L. Wolf, **A Lightweight Infrastructure for Reconfiguring Applications**, in *SCM 2001/2003*, pp. 231–244, LNCS 2649, Springer-Verlag, Portland, Oregon, May 2003.
- A. Carzaniga and A. Orso, **Continuous Remote Analysis for Improving Distributed Systems Performance**, in *RAMSS'03, 1st International Workshop on Remote Analysis and Measurement of Software Systems*, pp. 21–24, Portland, Oregon, May 2003.
- M. Caporuscio, A. Carzaniga, and A.L. Wolf, **An Experience in Evaluating Publish/Subscribe Services in a Wireless Network**, *Third International Workshop on Software and Performance*, Rome, Italy, July 2002.
- C. Wang, A. Carzaniga, D. Evans, and A.L. Wolf, **Security Issues and Requirements for Internet-scale Publish-Subscribe Systems**, In *Proceedings of the Thirty-Fifth Annual Hawaii International Conference on System Sciences (HICSS-35)*, Big Island, Hawaii, January 2002.
- D. Heimbigner, **Adapting Publish/Subscribe Middleware to Achieve Gnutella-Like Functionality**, in *Proceedings of the ACM Symposium on Applied Computing*, pp. 11–14, March 2001.

ACKNOWLEDGMENTS

This research was supported by the Army Research Laboratory and the Army Research Office, under agreement DAAD19-01-1-0484. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

REFERENCES

[1] E. Amoroso. *Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response.* Intrusion.Net Books, February 1999.

[2] Mauro Caporuscio, Antonio Carzaniga, and Alexander L. Wolf. Design and evaluation of a support service for mobile, wireless publish/subscribe applications. *IEEE Transactions on Software Engineering*, 29(12):1059–1071, December 2003.

[3] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Interfaces and algorithms for a wide-area event notification service. Technical Report CU-CS-888-99, Department of Computer Science, University of Colorado, October 1999, revised May 2000.

[4] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, pages 219–227, Portland OR, USA, July 2000.

[5] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.

[6] Antonio Carzaniga, Matthew J. Rutherford, and Alexander L. Wolf. A routing scheme for content-based networking. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 918–928, March 2004.

[7] Antonio Carzaniga and Alexander L. Wolf. Content-based networking: A new communication infrastructure. In *Proceedings of the NSF Workshop on an Infrastructure for Mobile and Wireless Systems*, number 2538 in Lecture Notes in Computer Science, pages 59–68. Springer-Verlag, 2002.

[8] Antonio Carzaniga and Alexander L. Wolf. Forwarding in a content-based network. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, pages 163–174, August 2003.

[9] B. Chapman and E. Zwicky. *Building Internet Firewalls*. O'Reilly & Associates, 1995.

[10] W. Cheswick, S. Bellovin, and A. Rubin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley, second edition, 2003.

[11] S.T. Eckmann, G. Vigna, and R.A. Kemmerer. STATL: An Attack Language for State-based Intrusion Detection. In *Proceedings of the ACM Workshop on Intrusion Detection Systems*, Athens, Greece, November 2000.

[12] Dennis M. Heimbigner. Adapting publish/subscribe middleware to achieve gnutella-like functionality. In *Proceedings of the ACM Symposium on Applied Computing*, pages 11–14. ACM Press, March 2001.

[13] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Topology-based Detection of Anomalous BGP Messages. In *Proceedings of the 6^{th} International Symposium on Recent Advances in Intrusion Detection (RAID)*, Pittsburgh, PA, September 2003.

[14] C. Kruegel, Fredrik Valeur, G. Vigna, and R.A. Kemmerer. Stateful Intrusion Detection for High-Speed Networks. In *Proceedings of the IEEE Symposium on Research on Security and Privacy*, Oakland, CA, May 2002. IEEE Press.

[15] V. Mittal and G. Vigna. Sensor-Based Intrusion Detection for Intra-Domain Distance-Vector Routing. In R. Sandhu, editor, *Proceedings of the ACM Conference on Computer and Communication Security (CCS'02)*, Washington, DC, November 2002. ACM Press.

[16] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, January 1998.

[17] J. Pouzol and M. Ducassé. From Declarative Signatures to Misuse IDS. In W. Lee, L. Mé, and A. Wespi, editors, *Proceedings of the RAID International Symposium*, volume 2212 of *LNCS*, pages 1 – 21, Davis, CA, October 2001. Springer-Verlag.

[18] M. Roesch. *Writing Snort Rules: How To write Snort rules and keep your sanity*. `http://www.snort.org`.

[19] C. Scott, P. Wolfe, M. Erwin, and A. Oram. *Virtual Private Networks*. O'Reilly, second edition, December 1998.

[20] F. Valeur, G. Vigna, C.Kruegel, and R. Kemmerer. A Comprehensive Approach to Intrusion Detection Alert Correlation. *IEEE Transactions on Dependable and Secure Computing*, 1(3):146–169, July-September 2004.

[21] G. Vigna, S. Eckmann, and R. Kemmerer. The STAT Tool Suite. In *Proceedings of DISCEX 2000*, Hilton Head, South Carolina, January 2000. IEEE Computer Society Press.

[22] G. Vigna, R.A. Kemmerer, and P. Blix. Designing a Web of Highly-Configurable Intrusion Detection Sensors. In W. Lee, L. Mè, and A. Wespi, editors, *Proceedings of the 4^{th} International Symposium on Recent Advances in Intrusion Detection (RAID 2001)*, volume 2212 of *LNCS*, pages 69–84, Davis, CA, October 2001. Springer-Verlag.

[23] G. Vigna, F. Valeur, and R.A. Kemmerer. Designing and Implementing a Family of Intrusion Detection Systems. In *Proceedings of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2003)*, Helsinki, Finland, September 2003.

[24] G. Vigna, F. Valeur, J. Zhou, and R.A. Kemmerer. Composable Tools For Network Discovery and Security Analysis. In *Proceedings of the 18^{th} Annual Computer Security Applications Conference (ACSAC '02)*, pages 14 24, Las Vegas, NV, December 2002. IEEE Press.

Stateful Intrusion Detection for High-Speed Networks

Christopher Kruegel Fredrik Valeur
Giovanni Vigna Richard Kemmerer

Reliable Software Group
University California, Santa Barbara
{kruegel,fredrik,vigna,kemm}@cs.ucsb.edu

Abstract

As networks become faster there is an emerging need for security analysis techniques that can keep up with the increased network throughput. Existing network-based intrusion detection sensors can barely keep up with bandwidths of a few hundred Mbps. Analysis tools that can deal with higher throughput are unable to maintain state between different steps of an attack or they are limited to the analysis of packet headers. We propose a partitioning approach to network security analysis that supports in-depth, stateful intrusion detection on high-speed links. The approach is centered around a slicing *mechanism that divides the overall network traffic into subsets of manageable size. The traffic partitioning is done so that a single slice contains all the evidence necessary to detect a specific attack, making sensor-to-sensor interactions unnecessary. This paper describes the approach and presents a first experimental evaluation of its effectiveness.*

Keywords: Intrusion Detection, High-Speed Networks, Security Analysis.

1 Introduction

Network-based intrusion detection systems (NIDSs) perform security analysis on packets obtained by eavesdropping on a network link. The constant increase in network speed and throughput poses new challenges to these systems. Current network-based IDSs are barely capable of real-time traffic analysis on saturated Fast Ethernet links (100 Mbps) [3]. As network technology presses forward, Gigabit Ethernet (1000 Mbps) has become the de-facto standard for large network installations. In order to protect such installations, a novel approach for network-based intrusion detection is necessary to manage the ever-increasing data volume.

Network speeds have increased faster than the speed of processors, and therefore centralized solutions have reached their limit. This is especially true if one considers in-depth, stateful intrusion detection analysis. In this case, the sensors have to maintain information about attacks in progress (e.g., in the case of multi-step attacks) or they have to perform application-level analysis of the packet contents. These tasks are resource intensive and in a single-node setup may seriously interfere with the basic task of retrieving packets from the wire.

To be able to perform in-depth, stateful analysis it is necessary to divide the traffic volume into smaller portions that can be thoroughly analyzed by intrusion detection sensors. This approach has often been advocated by the high-performance research community as a way to distribute the service load across many nodes. In contrast to the case for standard load balancing, the division (or slicing) of the traffic for intrusion detection has to be performed in a way that guarantees the detection of all the threat scenarios considered. If a random division of traffic is used, sensors may not receive sufficient data to detect an intrusion, because different parts of the manifestation of an attack may have been assigned to different slices. Therefore, when an attack scenario consists of a number of steps, the slicing mechanism must assure that all of the packets that could trigger those steps are sent to the sensor configured to detect that specific attack.

This paper presents an approach to in-depth, stateful intrusion detection analysis and a tool based on this approach. The approach allows for meaningful slicing of the network traffic into portions of manageable size. The slicing approach and a tool based on the approach are presented in Section 3, after a discussion of related work in Section 2. Section 4 presents the results of the quantitative evaluation of the first prototype of the tool. Section 5 presents some final remarks and outlines future research.

Reprinted with permission from *Proceedings of the IEEE Symposium on Security and Privacy,* May 2004, 285–294.

2 Related Work

The possibility of performing network-based intrusion detection on high-speed links (e.g., on OC-192 links) has been the focus of much debate in the intrusion detection community. A common position is to state that high-speed network-based intrusion detection is not practical because of the technical difficulties encountered in keeping pace with the increasing network speed and the more widespread use of encrypted traffic. Others advocate locating highly distributed network-based sensors at the periphery of computer networks; the idea being that the traffic load is, possibly, more manageable there.

Even though both of the advocated approaches above have good points, analysis of network traffic on high-speed links still represents a fundamental need in many practical network installations. The commercial world attempted to respond to this need and a number of vendors now claim to have sensors that can operate on high-speed ATM or Gigabit Ethernet links. For example, ISS [4] offers *NetICE Gigabit Sentry*, a system that is designed to monitor traffic on high-speed links. The company advertises the system as being capable of performing protocol reassembly and analysis for several application-level protocols (e.g. HTTP, SMTP, POP) to identify malicious activities. The tool claims to be the "first network-IDS that can handle full Gigabit speeds." However, the authors of the tool also state that "GigaSentry handles a full Gigabit in lab conditions, but real-world performance will likely be less. [...] Customers should expect at least 300 Mbps real-world performance, and probably more depending up the nature of their traffic. [...] GigaSentry can only capture slightly more than 500,000-packets/second." These comments show the actual difficulties of performing network-based intrusion detection on high-speed links. Other IDS vendors (like Cisco [1]) offer comparable products with similar features. Unfortunately, no experimental data gathered on real networks is presented. TopLayer Networks [11] presents a switch that keeps track of application-level sessions. The network traffic is split with regard to these sessions and forwarded to several intrusion detection sensors. Packets that belong to the same session are sent through the same link. This allows sensors to detect multiple steps of an attack within a single session. Unfortunately, the correlation of information between different sessions is not supported. This could result in missed attacks when attacks are performed against multiple hosts (e.g., ping sweeps), or across multiple sessions.

Very few research papers have been published that deal with the problem of intrusion detection on high-speed links. Sekar et al. [10] describe an approach to perform high-performance analysis of network data, but unfortunately they do not provide experimental data based on live traffic analysis. Their claim of being able to perform real-time intrusion detection at 500 Mbps is based on the processing of off-line traffic log files. This estimate is not indicative of the real effectiveness of the system when operating on live traffic.

3 A Slicing Approach to High-Speed Intrusion Detection

The problem of intrusion detection analysis in high-speed networks can be effectively attacked only if a scalable solution is available. Let us consider the traffic on the monitored network link as a bi-directional stream of link-layer frames (e.g., Ethernet frames). This stream contains too much data to be processed in real-time by a centralized entity and has to be divided into several smaller streams that are fed into a number of different, distributed sensors. Each sensor is only responsible for a subset of all detectable intrusion scenarios and can therefore manage to process the incoming volume in real-time. Nevertheless, the division into streams has to be done in a way that provides each sensor with enough information to detect exactly the same attacks that it would have witnessed when operating directly on the network link.

3.1 Requirements

The overall goal is to perform stateful intrusion detection analysis in high-speed networks. The approach presented in this paper can be characterized by the following requirements.

- The system implements a misuse detection approach where *signatures* representing attack scenarios are matched against a stream of network events.
- Intrusion detection is performed by a set of sensors, each of which is responsible for the detection of a subset of the signatures.
- Each sensor is autonomous and does not interact with other sensors.
- The system partitions the analyzed event stream into slices of manageable size.
- Each traffic slice is analyzed by a subset of the intrusion detection sensors.
- The system guarantees that the partitioning of traffic maintains detection of all the specified attack scenarios. This implies that sensors, signatures, and traffic slices are configured so that each sensor has access to the traffic necessary to detect the signatures that have been assigned to it.

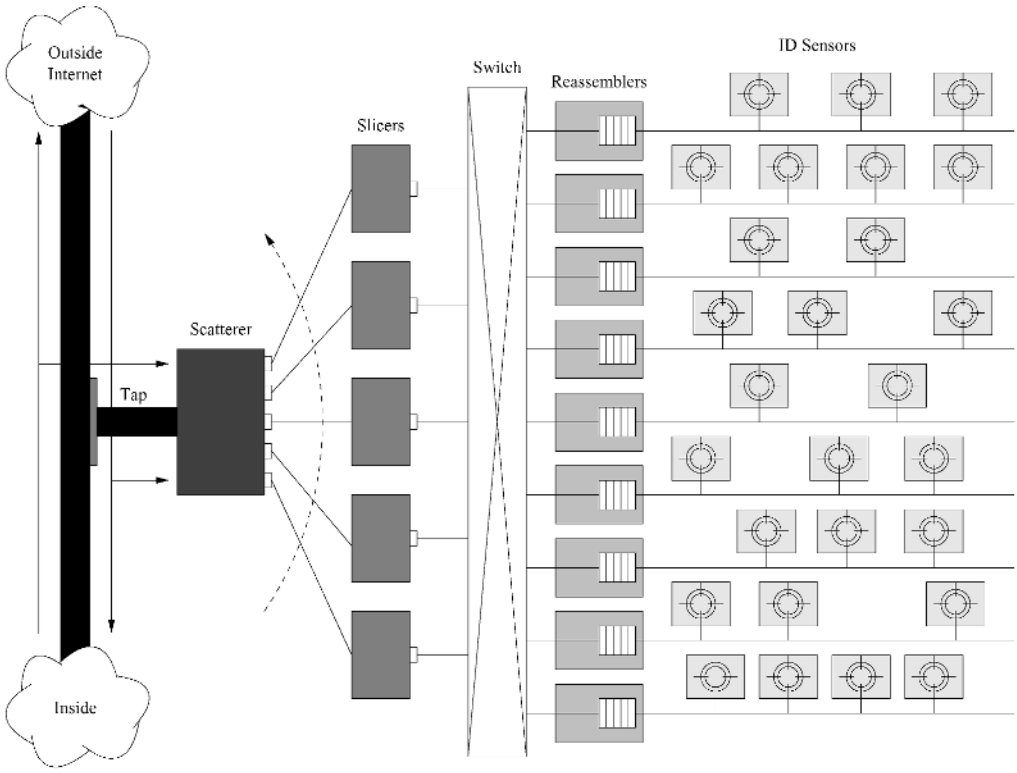


Figure 1. High-level architecture of the high-speed intrusion detection system.

- Components can be added to the system to achieve higher throughput. More precisely, the approach should result in a scalable design where one can add components as needed to match increased network throughput.

3.2 System Architecture

The requirements listed in the previous section have been used as the basis for the design of a network-based intrusion detection system. The system consists of a *network tap*, a *traffic scatterer*, a set of m *traffic slicers* $S_0, ..., S_{m-1}$, a *switch*, a set of n *stream reassemblers* $R_0, ..., R_{n-1}$, and a set of p *intrusion detection sensors* $I_0, ..., I_{p-1}$. A high-level description of the architecture is shown in Figure 1.

The network tap component monitors the traffic stream on a high-speed link. Its task is to extract the sequence F of link-layer frames $\langle f_0, f_1, ..., f_t \rangle$ that are observable on the wire during a time period Δ. This sequence of frames is passed to the scatterer which partitions F into m subsequences $F_j : 0 \leq j < m$. Each F_j contains a (possibly empty) subset of the frame sequence F. Every frame f_i is an element of exactly one sub-sequence F_j and therefore $\cup_{j=0}^{j<m} F_j = F$. The scatterer can use any algorithm to partition F. Hereafter, it is assumed that the splitting algorithm simply cycles over the m sub-sequences in a round-robin fashion, assigning f_i to $F_{i \bmod(m)}$. As a result, each F_j contains an m-th of the total traffic.

Each sub-sequence F_j is transmitted to a different traffic slicer S_j. The task of the traffic slicers is to route the frames they receive to the sensors that may need them to detect an attack. This task is not performed by the scatterer, because frame routing may be complex, requiring a substantial amount of time, while the scatterer has to keep up with the high traffic throughput and can only perform very limited processing per frame.

The traffic slicers are connected to a switch component, which allows a slicer to send a frame to one or more of n outgoing channels C_i. The set of frames sent to a channel is denoted by FC_i. Each channel C_i is associated with a stream reassembler component R_i and a number of intrusion detection sensors. The set of sensors associated with channel C_i is denoted by IC_i. All the sensors that are associated with a channel are able to access all the packets sent on that channel. The original order of two packets could be lost if the two frames took different paths over distinct slicers to the same channel. Therefore, the reassemblers associated with each channel make sure that the packets appear on the channel in the same order that they appeared on the high-speed link. That is, each reassembler R_i must make sure that for each pair of frames $f_j, f_k \in FC_i$ it holds that (f_j before f_k) $\iff j < k$.

Each sensor component I_j is associated with q different attack scenarios $A_j = \{A_{j0}, ..., A_{jq-1}\}$. Each attack scenario A_{jk} has an associated *event space* E_{jk}. The event space specifies which frames are candidates to be part of the manifestation of the attack. For example, consider an attack targeting a Web server called `spider` within the network protected by the intrusion detection system. In this case, the event space for that attack is composed of all the TCP traffic that involves port 80 on host `spider`.

Event spaces are expressed as disjunctions of *clauses*, that is, $E_{jk} = c_{jk_0} \vee c_{jk_1} \vee ... \vee c_{jk_n}$, where each clause c_{jk} is an expression of the type xRy. x denotes a value derived from the frame f_i (e.g., a part of the frame header) while R specifies an arithmetic relation (e.g., =, !=, <). y can be a constant, the value of a variable, or a value derived from the same frame. Clauses and event spaces may be derived automatically from the attack descriptions, for example from signatures written in attack languages such as Bro [6], Sutekh [7], STATL [2], or Snort [8].

3.3 Frame Routing

Event spaces are the basis for the definition of the filters used by the slicers to route frames to different channels. The filters are determined by composing the event spaces associated with all the scenarios that are "active" on a specific channel. More precisely, the set of active scenarios is $AC_i = \bigcup_{j=0}^{j<u} A_j$ where A_j is the set of scenarios of $I_j \in IC_i$. The event space EC_i for a channel C_i is the disjunction of the event spaces of all active scenarios, which corresponds to the disjunction of all the clauses of all the active scenarios. The resulting overall expression is the filter that each slicer uses to determine if a frame has to be routed to that specific channel. Note that it is possible that a certain frame will be needed by more than one scenario. Therefore, it will be sent on more than one channel.

The configuration of the slicers as described above is static; that is, it is calculated off-line before the system is started. The static approach suffers from the possibility that, depending on the type of traffic, a large percentage of the network packets could be forwarded to a single channel. This would result in the overloading of sensors attached to that channel. The static configuration also makes it impossible to predict the exact number of sensors that are necessary to deal with a Gigabit link. The load on each sensor depends on the scenarios used and the actual traffic. The minimum requirement for the slicers is that the capacity of their incoming and outgoing links must be at least equal to the bandwidth of the monitored link.

One way to prevent the overloading condition is to perform dynamic load balancing. This is done by reassigning scenarios to different channels at run-time. This variant obviously implies the need to reconfigure the filter mechanism at the traffic slicers and update the assignment of clauses to channels.

In addition to the reassignment of whole scenarios to different channels, it is also possible to split a single scenario into two or more refined scenarios. The idea is that each refined scenario catches only a subset of the attacks that the original scenario covered, but each can be deployed on a different channel. Obviously, the union of attacks detectable by all refined scenarios has to cover exactly the same set of attacks as the original scenario did.

This can be done by creating additional constraints on certain attributes of one or more basic events. Each constraint limits the number of attacks a refined scenario can detect. The constraints have to be chosen in a way such that every possible value for a certain attribute (of the original scenario) is allowed by the constraint of at least one refined scenario. Then the set of all refined scenarios, which each cover only a subset of the attacks of the original one, are capable of detecting the same attacks as the original.

A simple mechanism to partition a particular scenario is to include a constraint on the destination attribute of each basic event that represents a packet which is sent by the attacker. One has to partition the set of possible destinations such that each refined scenario only covers attacks against a certain range of hosts. When the union of these target host ranges covers all possible attack targets, the set of refined scenarios is capable of finding the same attacks as the original scenario.

Such an approach is necessary when a single scenario causes too much traffic to be forwarded to a single channel.

In addition, obviously innocent or hostile frames could be filtered out before the scenario clauses are applied, thereby eliminating traffic that needs no further processing. This could be used, for instance, to prevent the system from being flooded by packets from distributed denial-of-service slaves that produce traffic with a unique, known signature.

4 Evaluation

The initial set of experiments were primarily aimed at evaluating the effectiveness of the scatterer/slicer/ reassembler architecture. For these experiments, we deployed three traffic slicers ($m = 3$) and four stream reassemblers ($n = 4$) with one intrusion detection sensor per stream. The next section presents the details of the hardware and software used to realize the initial prototype, and the section after that gives the details of each experiment and presents the corresponding results.

4.1 Prototype Architecture

The prototype is composed of a number of hosts responsible for the analysis of the traffic carried by a Gigabit link.

The Gigabit link is realized as a direct connection (crossover cable) between two machines equipped with Intel Xeon 1.7 GHz processors, 512 MB RAM and 64-bit PCI 3Com 996-T Gigabit Ethernet cards running Linux 2.4.2 (Red Hat 7.1). One of the two machines simulates the network tap and is responsible for creating the network traffic (via `tcpreplay` [12]). The other machine acts as the traffic scatterer and is equipped with three additional 100 Mbps 3Com 905C-TX Ethernet cards.

The scatterer functionality itself is realized as a kernel module attached to the Linux kernel bridge interface. The bridge interface provides a hook that allows the kernel to inspect the incoming frames before they are forwarded to the network layer (e.g., the IP stack). The scatterer module intercepts frames coming from the Gigabit interface and immediately forwards them to one of the outgoing links through the corresponding Fast Ethernet card. The links are selected in a round-robin fashion. The scatterer also attaches a sequence number to each packet, which is later used by the reassemblers. In order to overcome the problem of splitting Ethernet frames with a length close to the maximum transferable unit (MTU), the sequence number has to be integrated into the Ethernet frame without increasing its size. To leave the data portion untouched, we decided to modify the Ethernet header. We also wanted to limit the modifications of the Ethernet frame to a minimum in order to be able to reuse existing hardware (e.g., network interface cards, network drivers). Therefore, the MTU had to remain unchanged. For this reason, we decided to use the six-byte Ethernet source address field for sequence numbers. As a result, before the traffic scatterer forwards a frame, it writes the current sequence number into the source address field and increments it.

The experimental setup demonstrates that the partitioning of traffic is possible and that it allows for the detailed analysis of higher traffic volume (including defragmentation, stream reassembly, and content analysis). Because we only use three traffic slicers (with an aggregated bandwidth of 300 Mbps), sustained incoming traffic of 1 Gbps would overload our experimental setup. However, the introduction of additional traffic slicers would allow us to handle higher traffic inputs.

The traffic slicers (Intel Pentium 4 1.5 GHz, 256 MB RAM, 3Com 905C-TX fast Ethernet cards running Linux 2.4.2 - Redhat 7.1) have the NIC of the link that connects them to the traffic scatterer set to promiscuous mode, in order to receive all incoming frames. The data portion of each incoming frame is matched against the clauses stored for each channel. Whenever a clause for a channel is satisfied, a copy of the frame is forwarded to that channel. Note that this could (and usually does) increase the total number of frames that have to be processed by the intrusion detection sensors. Nevertheless, a sufficiently large number of sensors combined with sophisticated partitioning enable one to keep the amount of traffic at each sensor low enough to handle. In our test setup, the partitioning (i.e., the clauses) was determined as follows. Similar to Snort [9], we distinguished between an inside network and an outside network, representing the range of IP addresses of the protected network and its complement, respectively. The protected network address range is divided according to the existing class C subnetworks. The network addresses are then grouped into four sets, each of which is assigned to a different channel. This partitioning allows the system to detect both attacks involving a single host and attacks spanning a subnetwork. As explained in Section 3.3 more sophisticated schemes are possible by analyzing additional information in the packet headers or even by examining the frame payload.

Once the filters have been configured, the frames have to be routed to the various channels. As in the case for the transmission between the scatterer and the traffic slicers, we want to prevent frames from being split when sent to the channels. This makes it necessary to include the destination address information of the intended channel in the Ethernet frame itself without increasing its size and without modifying the payload. To do this we use the Ethernet destination address. Therefore, the destination address is rewritten with values `00:00:00:00:00:01`, `00:00:00:00:00:02`, etc., depending on the destination channel. There were two reasons for using a generic link number instead of the actual Ethernet addresses as the target address for sensors. First, a number of sensors may be deployed on each channel, processing portions of the traffic in parallel. Since each sensor has to receive all packets on the channel where it is attached, selecting the Ethernet address of a single sensor is not beneficial. Second, whenever the NIC of a sensor has to be replaced, the new Ethernet address would have to be updated at each traffic slicer. In order to save this overhead, each traffic slicer simply writes the channel number into the target address field of outgoing frames.

The actual frame routing is performed by a switch (a Cisco Catalyst 3500XL) that connects traffic slicers with reassemblers. The MAC address-port table of the switch holds the static associations between the channel numbers (i.e., the target Ethernet addresses set by the traffic slicers) and the corresponding outgoing ports. In general backplanes of switches have very high bandwidth compared to Ethernet links, so they are not likely to be overloaded by traffic generated by the scatterer.

In our setup, the stream reassemblers are located at each sensor node (using the same equipment as the traffic slicers), and they provide the intrusion detection sensors with a temporally sorted sequence of frames by using the encapsulated sequence numbers. The reassembly procedure

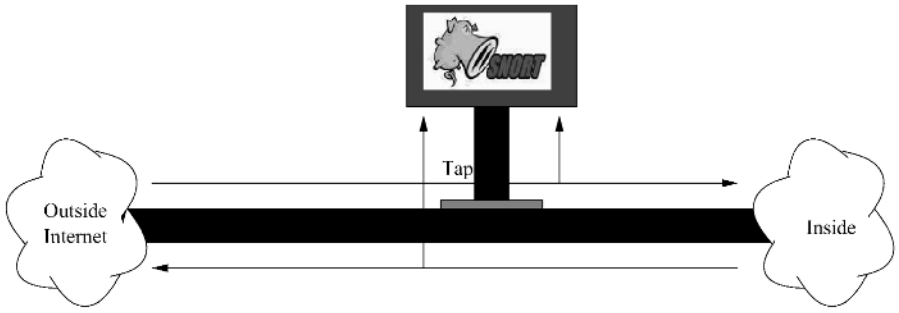


Figure 2. Single-node Snort setup.

has been integrated into `libpcap` so that every sensor that utilizes these routines to capture packets can be run unmodified. For each frame, we assume that no other frame with a smaller sequence number can arrive after a certain time span (currently 500 ms). This means that when an out-of-order packet is received, it is temporarily stored in a queue until either the missing packets are received and the correctly-ordered batch of packets is passed to the application, or the reassembler decides that some packets have been lost because a timeout expired and the packet is passed without further delay. Therefore, each received packet is passed to the sensors with a worst case delay being the timeout value. The timeout parameter has to be large enough to prevent the situation where packets with smaller sequence numbers arrive after subsequent frames have already been processed but small enough so that the reaction lag of the system is within acceptable limits. Since the processing and transmission of frames is usually very fast and no retransmission or acknowledgments are utilized, one can expect frames to arrive at each reassembler in the correct order most of the time. In principle, this allows one to safely choose a very short time span. We expect to have no problems in reducing the current timeout value, but at the moment we have no experimental evaluation of the effect of different timeout values on the effectiveness of intrusion detection.

The network cards of the nodes would normally be receiving traffic at rates close to their maximum capacity. If administrative connections, such as dynamically setting clauses, reporting alarms, or performing maintenance work were to go through the same interfaces, these connections could potentially suffer from packet loss and long delays. To overcome this problem, each machine is connected to a second dedicated network that provides a safe medium to perform the tasks mentioned above. An additional communication channel decoupled from the input path has the additional benefit of increasing the resiliency of the system against denial-of-service attacks. That is, alarms and reconfiguration commands still reach all intended receivers, since they do not have to compete against the flood of incoming packets for network access.

4.2 Experimental Results

The goal of the set of experiments described in this section is to get a preliminary evaluation of the practicality and effectiveness of our approach. The general assumption is that we are interested in in-depth, stateful, application-level analysis of high-speed traffic. For this reason, we chose Snort as our "reference" sensor and we enabled reassembling and defragmenting.

To run our experiments we used traffic produced by MIT Lincoln Labs as part of the DARPA 1999 IDS evaluation [5]. More precisely, we used the data from Tuesday of the fifth week. The traffic log was injected on the Gigabit link using `tcpreplay`. To achieve high speed traffic we had to "speed up" the traffic. We assumed that this would not affect the correctness of our experiment. We also assumed that the LL/MIT traffic is a reasonable approximation of real-world traffic. This assumption has often been debated, but we think that for the extent of the tests below this assumption is reasonable.

The first experiment was to run Snort on the tcpdump traffic log. The results of this "off-line" run are: 11,213 detections in 10 seconds with an offline throughput of 261 Mbps. The ruleset used included 961 rules.

The second experiment was to run Snort on a single-node monitor. The setup is shown in Figure 2. In practice, Snort is run on the scatterer host and it reads directly from the network card. We measured the decrease in effectiveness of the detection when the traffic rate increases[1]. The ruleset used included only the 18 rules that actually fired on the test data. Figure 3 shows the results of this experiment. The reduced performance is due to packet loss, which becomes substantial at approximately 150 Mbps. This experiment identifies the saturation point of this setup.

The third experiment was to run Snort in the simple setup of Figure 2 with a constant traffic rate of 100 Mbps and an increasing number of signatures. The experiment starts with only the eighteen signatures that are needed to achieve

[1]The limit of 200 Mbps in the graphs is the maximum amount of traffic that `tcpreplay` is able to generate.

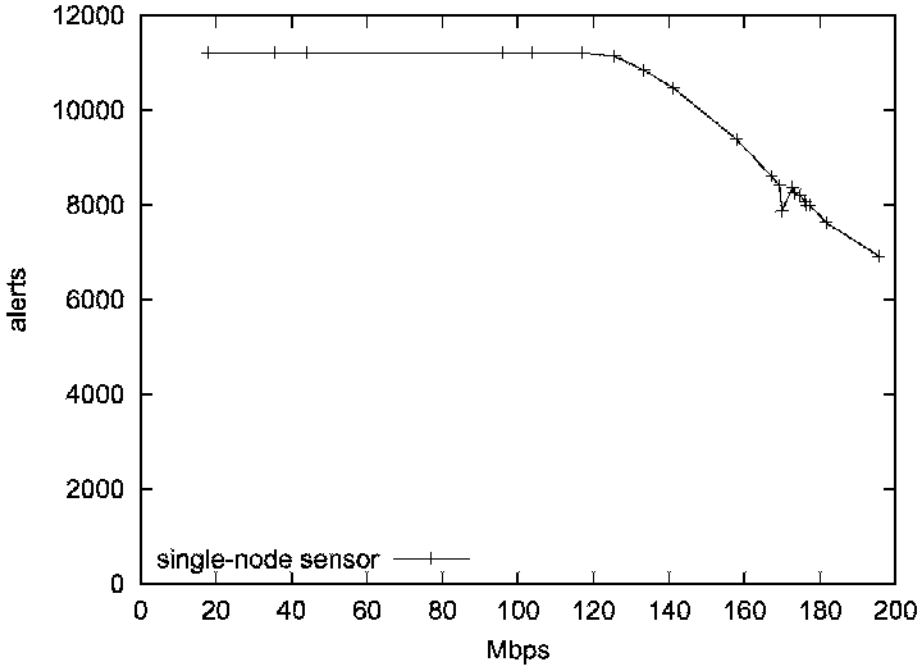


Figure 3. Single-host monitor detection rate for increasing traffic levels.

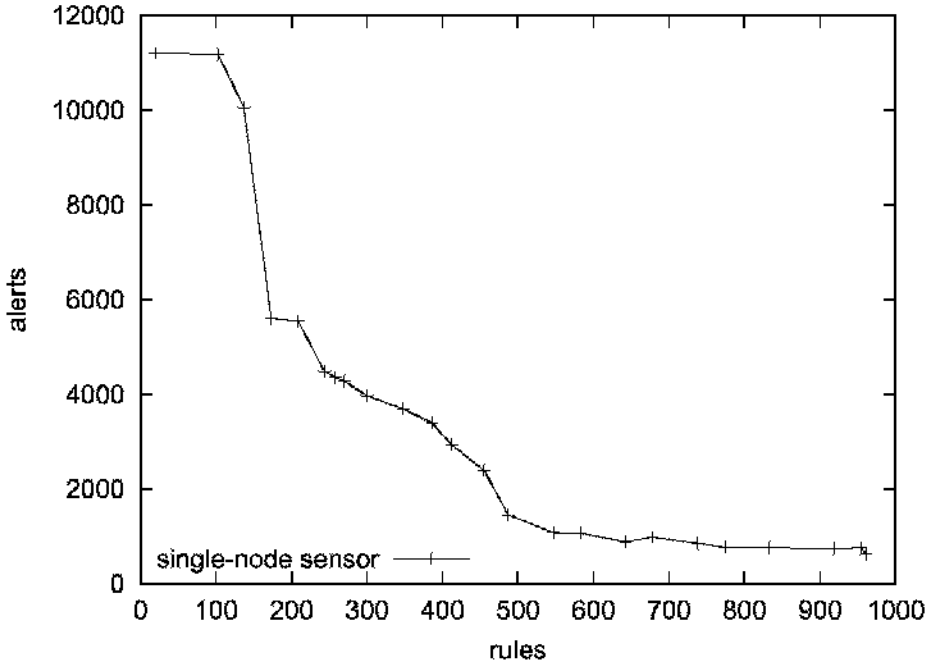


Figure 4. Single-host monitor detection rate for increasing number of signatures.

maximum detection for the given data. The plot in Figure 4 shows how the performance decreases as more signatures are added to the sensor. This experiment demonstrates that such a setup is limited by the number of signatures that can be used to analyze the traffic stream.

The fourth and fifth experiments repeated the previous two experiments using Snort sensors in the proposed architecture. Figure 5 and 6 present the results of these experiments. The performance of the single-node experiments are included for comparison. The drop in detection rate at high speeds by the distributed sensor, which can be seen in Figure 5, is caused by packet loss in the scatterer. The network cards currently used for the output traffic are not able to handle more than about 170 Mbps. The experimental results show that the proposed architecture has increased throughput and is much less sensitive to the number of signatures used.

5 Conclusion and Future Work

This paper presents the design, implementation, and experimental evaluation of a distributed network monitor. The system supports stateful, in-depth analysis of network traffic on high-speed links. The evaluation of the first prototype showed that the approach is more scalable than the single-host monitor approach. The current results are very preliminary and a thorough evaluation will require experimentation in a real-world environment.

Future work will include a more thorough evaluation of the trade-offs when configuring the system, the development of a mechanism for dynamic load balancing, and the use of hierarchically structured scatterers/slicers to achieve higher throughput levels.

Acknowledgments

This research was supported by the Army Research Office, under agreement DAAD19-01-1-0484 and by the Defense Advanced Research Projects Agency (DARPA) and Rome Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-97-1-0207. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Army Research Office, the Defense Advanced Research Projects Agency (DARPA), Rome Laboratory, or the U.S. Government.

References

[1] CISCO. CISCO Intrusion Detection System. Technical Information, Nov 2001.

[2] S.T. Eckmann, G. Vigna, and R.A. Kemmerer. STATL: An Attack Language for State-based Intrusion Detection. In *Proceedings of the ACM Workshop on Intrusion Detection Systems*, Athens, Greece, November 2000.

[3] NSS Group. Intrusion Detection and Vulnerability Assessment. Technical report, NSS, Oakwood House, Wennington, Cambridgeshire, UK, 2000.

[4] ISS. BlackICE Sentry Gigabit. http://www.networkice.com/products/sentry_gigabit, November 2001.

[5] MIT Lincoln Laboratory. DARPA Intrusion Detection Evaluation. http://www.ll.mit.edu/IST/ideval/, 1999.

[6] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, January 1998.

[7] J. Pouzol and M. Ducassé. From Declarative Signatures to Misuse IDS. In W. Lee, L. Mé, and A. Wespi, editors, *Proceedings of the RAID International Symposium*, volume 2212 of *LNCS*, pages 1 – 21, Davis, CA, October 2001. Springer-Verlag.

[8] M. Roesch. Writing Snort Rules: How To write Snort rules and keep your sanity. http://www.snort.org.

[9] M. Roesch. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of the USENIX LISA '99 Conference*, November 1999.

[10] R. Sekar, V. Guang, S. Verma, and T. Shanbhag. A High-performance Network Intrusion Detection System. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, November 1999.

[11] Toplayer networks. http://www.toplayer.com, November 2001.

[12] M. Undy. tcpreplay. Software Package, May 1999.

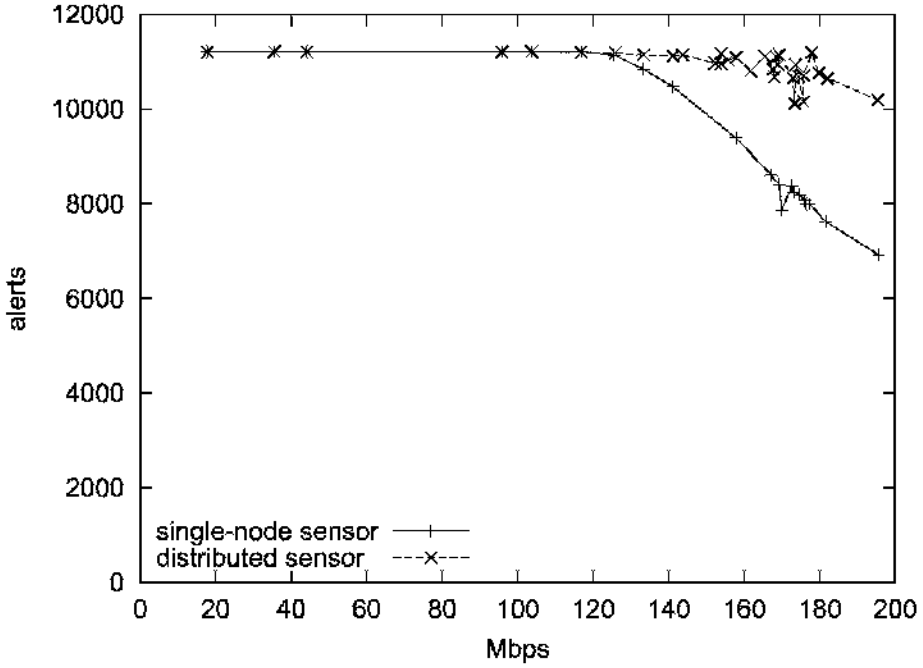


Figure 5. Distributed monitor detection rate for increasing traffic levels.

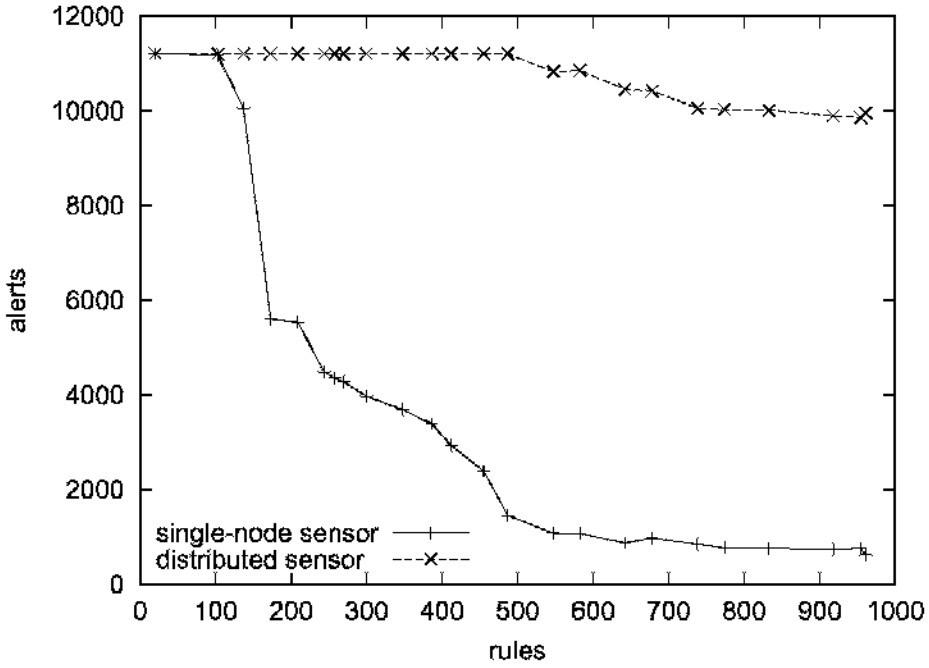


Figure 6. Distributed monitor detection rate for increasing number of signatures.