1

# Content Distribution on the Internet

In the beginning, Internet applications were based on textual information. Users were used to exchange email messages, transfer files via File Transfer Protocol (FTP) and access remote servers. Today, the Internet is a complex multimedia-information system based on content distribution. Documents, videos, audio, images, Web pages, for example, are “contents” [PLA 05]. Metadata used to find, understand and manage contents are also considered “contents”. However, in order to enable users to request and receive contents efficiently, several basic requirements must be satisfied. First, content persistence must be assured. Persistence means that content identifiers[1] should be unique and valid during the lifetime of the associated content. Recently, with the advent of Web 2.0, the number of content publishers has hugely increased. Today, even users with low technical knowledge are able to publish content on the Internet easily. Thus, it is quite hard to assure content persistence in the current Internet. The second requirement is scalability. Content-search and forwarding mechanisms should be efficient regardless of the number of users and contents offered. Both must be able to operate at Internet scale. Finally, the secure access to contents is an important requirement to provide authentication and access control mechanisms to available contents. Currently, there is no solution that satisfies all these three requirements at the same time. Several techniques try to partially satisfy them

1 In this book, *identifiers* and *names* are synonyms.

and thus make the current Internet architecture more suitable to content distribution. In this chapter, a few of these techniques are briefly presented.

## 1.1. End-to-end concept and limitations

Three characteristics of the current Internet architecture are barriers to satisfy the requirements of content distribution: there are no guarantees of (i) quality of service, (ii) end-to-end security and (ii) no scalable forwarding mechanisms.

The Internet is a packet-switched network on a global scale, in which packets are forwarded based on the best-effort service model offered by the Internet Protocol (IP). There is neither resource reservation nor service differentiation during packet forwarding. Consequently, contents are distributed with no performance guarantees. Furthermore, the current architecture is focused on communication between hosts, which means that a source host includes in packet headers the IP address of the host that it wants to communicate with. Then, packets are forwarded hop-by-hop, based solely on the destination IP address. This paradigm is well suited for the first Internet application, because its main goal is to share remote resources offered by a specific host, such as a Web server, printer server and file server. Nevertheless, such a paradigm is not able to satisfy content distribution requirements, because it compels users to know not only where a content is located, but also the name of the content they want.

Currently, content distribution on the Internet is supported by “patches”, that is a set of protocols and mechanisms that partially satisfy application requirements. The Hypertext Transfer Protocol (HTTP) redirect is an example used for searching non-persistent contents. With this mechanism, HTTP objects are requested by using resource locators, referred to as Uniform Resource Locators (URLs), which are in the headers of HTTP messages. Thus, HTTP redirect events are triggered by the server that hosted those objects originally. In this case, the server sends back to the client an HTTP redirect message containing the new URL in its header. This mechanism, however, must know where contents are placed and thus it is necessary to develop complementary mechanisms to assure persistent access to these contents, regardless of location, properties or other characteristics related to them. This example also illustrates how the client–server model works. In this case, one point-to-point communication channel is established between one client and one server. If several users simultaneously request a given content hosted by a server, multiple point-to-point channels are established and one copy of the same content is sent over each channel. Therefore, the more popular the content is, the less the efficiency of content distribution mainly in terms of bandwidth. Although not efficient, this model is widely adopted by the current content distribution applications. In summary, large-scale content distribution applications require the development of scalable forwarding mechanisms that must be quite different from the traditional client–server model.

Content distribution applications also try to provide content authentication and secure communication over the Internet. Currently, all of them employ mechanisms to provide a secure channel between the source and destination hosts instead of explicitly providing security to the content itself. Consequently, additional messages and process overheads are introduced [SME 09]. Internet Protocol Security (IPSec), for example, is a patch used to provide secure communication. Basically, IPSec allows users to establish reliable connections by introducing authentication headers (AHs), applying cryptography to data, using encapsulating security payloads (ESPs) and finally by employing key management mechanisms. In this connection-oriented approach, however, the content security depends on the trust of the host that stores this content and also on the connection established between hosts. Once again, scalability is the barrier to surpass. In this case, the same content are not available to be shared among different users, because the content is carried out within a secure channel between two hosts. The alternative is to establish multiple secure connections among content sources and different users, which are not scalable [SME 09]. Clearly, specific solutions for content distribution applications are currently mandatory.

## 1.2. Multicast communication

Multicast communication is one of the first proposals to increase the content distribution efficiency on the Internet. In practice, this technique is implemented by an IP multicast in the network layer [DEE 89]. Basically, with the IP multicast, one datagram sent by a host can reach multiple hosts that are interested in the same content. For that reason, these hosts are aggregated in a *group*, which is identified by only one IP address. Thus, if a host sends a datagram to the IP address of a given group, all the hosts that have joined this group receive a copy of this datagram. The role of the network layer, in this case, is to forward and replicate this datagram, when necessary, over the entire distribution tree that covers all the hosts interested in the group content. The advantage is to save bandwidth by not forwarding unnecessary copies of the same datagrams over one link.

IP multicast, proposed in the 1990s, is not currently adopted on a large scale on the Internet. For several authors, the main reason is the complexity to configure and manage the set of protocols needed by IP multicast. This complexity comes from the service model proposed by the IP multicast itself. In summary, a given host is able to join and leave a group at any time, it may be a member of more than one group simultaneously, and it does not need to be a member of a group to send datagrams to the group [COS 06].

## 1.3. Peer-to-peer systems

Peer-to-peer (P2P) systems aim at increasing content distribution efficiency by promoting content sharing among the users of the system. Basically, nodes interested in the same content, referred to as peers, create an overlay network at the application layer and altruistically share bandwidth, the process and storage capacity. Thus, they are able to exchange contents. The key idea is that each peer contributes to a given amount of its resources and uses the service offered by the system [PAS 12]. Consequently, the more peers there are in the system, the more is the capacity of the system to satisfy the user requirements (delivery time, content availability and among others). Thus, the scalability needed by content distribution applications is intrinsically provided by P2P systems. In addition, P2P systems do not require changes in the network core as IP multicast does.

Another key aspect is that users today are interested in receiving a given content – a file or a multimedia streaming – no matter who sends it. With BitTorrent, for instance, a new peer in the system randomly chooses its partners, that is the nodes allowed to exchange content chunks with it. These partners are selected at random from a subset of peers who are interested in the same content and no information about location or identification of peers is taken into account during the selection process. The huge success of both P2P file-sharing and P2P streaming systems – with millions of users – clearly indicates that the paradigm of the Internet application is changing. This is the basis for the development of Information-Centric Networks: users are more and more interested in the content itself and not in its sender.

Although scalable to distribute content, P2P systems suffer from security problems and the lack of incentives for peers to share their resources. P2P systems rely on the collaborative behavior of peers to work properly. Thus, the trust in data forwarded by other peers is a crucial point that must be taken into account by these systems. Another problem is the robustness of the system against peer churn, that is the capacity to deal with frequent arrivals and departures of peers. Peer churn may reduce content availability and distribution efficiency because there is no dedicated infrastructure to manage those events.

## 1.4. Content distribution networks

Content Distribution Networks (CDNs) are proposed to increase the efficiency and scalability of the client–server model employed by most of the content distribution applications today [PAS 12]. CDNs are composed of a set of distributed servers interconnected through the Internet that cooperatively work to distribute content [BUY 08]. Contents are replicated on different servers – preferably by different Internet Service Providers (ISPs) – and thus CDNs increase content availability and communication efficiency. Basically, CDNs redirect content requests to one of the replicas stored by a server closer to the requester. The main idea is to reduce the number of hops between clients and servers. Consequently, clients should experience low latency and high delivery rate because the congestion probability decreases.

Two building blocks comprise the core of a CDN: the distribution and replication service and the request redirection service [PAS 12]. Content producers use the distribution and replication service to find proper servers, to allocate storage capacity and, finally, to allocate contents to the selected servers. In addition, the request redirection service is the CDN interface with content consumers.

Basically, this service receives content requests and then forwards each request to the more suitable CDN server to satisfy it.

CDNs are typically composed of two types of servers: an origin server and a replica server. On the one hand, the origin server attributes the content identifier, stores and announces the content. Replica servers, on the other hand, forward the content to clients. In general, clients send requests to the origin server who redirects these messages to the replica server closer to the client and that stores the desired content. Figure 1.1 illustrates this process. In summary, redirection mechanisms severely impact on a CDNs performance.

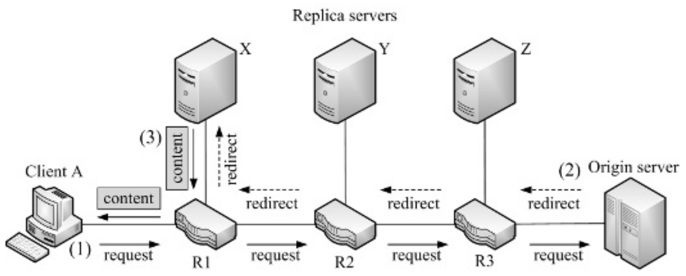

**Figure 1.1.** *A simple example of how CDNs work: (1) client A sends a content request to the origin server that (2) redirects this request to the replica server closest to A. Then, (3) X sends the content to A*

The simplest way to redirect requests in CDNs is to use the redirection mechanism originally offered by HTTP. In this case, all requests to HTTP objects are performed by Web browsers running on client hosts. When the origin server receives a request, it sends back to the requester an HTTP redirect message with the address of the best replica server. The origin server, in this case, is the bottleneck of the system and also a single point of failure because it processes all content requests. The Domain Name System (DNS) is also used by CDNs to redirect contents. Basically, the CDN DNS server receives messages requesting the address associated with the name of the origin server and then sends back to the client the address of the proper replica server. Both techniques – HTTP or DNS redirection – can select the “best” replica servers based on the number of hops or round-trip time (RTT) between clients and replica servers and/or based on the servers load. The main problem of these two techniques, however, is to guarantee content persistence. If the owner, domain or any other property of a given content changes, users may not be able to retrieve this content by using the same URL already known. In this case, for every change, users have to query centralized structures in order to obtain the new place of the content, which may increase the content delivery time [KOP 07].

The lack of interoperability between CDNs is another problem. Most of these networks are proprietary and specific for a given application and thus CDNs cannot be considered a general solution to satisfy the different content distribution applications on the Internet. In addition, server placement algorithms, capacity planning of servers and cache replacement policies have key roles on a CDNs performance [PAS 12]. For example, redirection mechanisms must select the best replica server in real time in order to have less impact on the delivery time, but it implies high computational costs. Also, the more the number of replica servers there are, the higher is the probability of finding a server close to the client. However, a CDN provider has to increase its budget to achieve that.

There are several examples of both academic, such as CoDeeN [WAN 04], and commercial CDN providers, such as Limelight[2] and Akamai[3], which are very popular. Akamai

2 http://www.limelight.com/
3 http://www.akamai.com/

has approximately 100,000 servers spread over the entire Internet, with points of presence in 72 countries and supports trillions of interactions per day [AKA 12].

## 1.5. Publish/subscribe systems

Publish/subscribe systems, or simply pub/sub, also indicates that the paradigm for current Internet applications is changing. Similar to P2P users, pub/sub users are interested in receiving the content regardless of its sender. In pub/sub systems, contents desired by users are referred to as *events* and the delivery of the content is called *notification*. The basic operation of a simple pub/sub system is the following. First, publishers create events and make them available to subscribers. Second, subscribers are able to announce their interest in events or event patterns defined by publishers. Thus, a subscriber is notified whenever an event that matches their interests is generated by any publisher.

Publishers and subscribers are decoupled in both time and space [EUG 03]. A subscriber, for example, may announce that it is interested in an event not yet published by any publisher. In addition, this interest should not be necessarily announced when the publisher is online. Decoupling, in this context, provides scalability to pub/sub systems because it allows publishers and subscribers to work independently [EUG 03]. Publishers add events to the system by calling the function `publish()`. Subscribers call the function `subscribe()` to register their interest in events. The pub/sub system, in this case, has a key role. The system itself have to store all interests announced and deliver contents to all interested subscribers, as shown in Figure 1.2. This operation mode allows pub/sub systems to distribute content between a huge number of users because publishers do not store states related to the interests of subscribers and subscribers receive content from any publisher, no matter if the sender is unknown [MAJ 09].

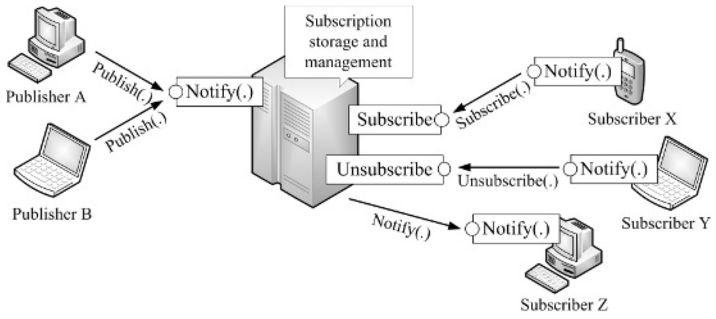

**Figure 1.2.** *Subscribe and event notification functions in a simple pub/sub system*

The first proposed pub/sub system is based on topics identified by keywords and is called a topic-based pub/sub system. Examples of this kind of system include enterprise application integration, stock-market monitoring engines, Really Simple Syndication (RSS) feeds, online gaming, among others [CHO 07]. With topic-based systems, users subscribe and publish events by using a topic, which is conceptually similar to the group defined by IP multicast, described in section 1.2. Each topic is a unique event service, identified by a unique name and provides interfaces to users that want to call publish and subscribe functions. Spidercast [CHO 07] and TERA [BAL 07] are examples of topic-based pub/sub systems.

Content-based systems are the next step in the evolutionary line of pub/sub systems. Basically, these systems allow users to subscribe to events based on properties of the events themselves and not based on previously defined and static characteristics, such as topic identification. With content-based systems, subscribers are able to specify filters to define their subscriptions by using restrictions based on attribute-value pairs (AVPs) and basic logical and comparative operators, such as $=$, $<$, $>$, $\leq$ and $\geq$. Restrictions can be logically combined by using Boolean operators, such as `AND` and `OR`, in order to define complex subscription patterns. These patterns are used in two basic functions of the system. First, patterns identify events of interest specified by a given subscriber. Second, notifications are forwarded through the system based on patterns, as detailed in section 3.1. Filters simplify the declaration of interests compared with topic-based systems. However, filters can introduce a communication overhead in the case of partially declared interests. Siena [CAR 01] and Kyra [CAO 04] are examples of content-based systems.

The different architectures employed by pub/sub systems can be classified into centralized or distributed, regardless of the way the subscribers specify their events of interest [EUG 03]. With the centralized architecture, on the one hand, event publishers send messages to a central entity that stores these messages and redirects them to subscribers on demand. With the distributed architecture, on the other hand, all system nodes must process and forward interests and notifications because there is no central entity. In general, distributed architectures rely on multicast communication and, thus, are prone to deliver content efficiently. In this case, topic-based systems benefit from this characteristic of distributed architectures. Content-based systems, however, face a huge challenge to efficiently provide multicast communication. Multicast performance is impacted by the computational cost of filtering needed during content forwarding, which varies with the amount of subscriptions in the system.