1

Partitioning the End-to-End QoS Budget to Domains

Quality of service (QoS) constraints for a packet flow are generally specified for the entire path, which may traverse several domains operated by different internet service providers (ISPs). In view of this, and given that each provider allocates bandwidth and provides QoS guarantees independently from the other providers, we need a way to combine this information for the entire end-to-end path. QoS constraints, such as the one-way delay (OWD), are typically specified as percentiles, and in view of this, we need an expression for adding percentiles of a number of random variables. As will be seen, we cannot simply add percentiles arithmetically, i.e. if x_1 and x_2 are the 95th percentiles of random variables X_1 and X_2, respectively, then $x_1 + x_2$ is not the 95th percentile of $X_1 + X_2$. How to achieve this is the issue addressed in this chapter.

This chapter is organized as follows. In the next section, we give examples where an expression for adding percentiles can be useful. In section 1.2, we present an expression for adding up percentiles of random variables that are exponentially distributed with either the same or different rates. The results are then extended to the more general Coxian-2 distribution. Next, the distribution of the QoS budget to individual providers is addressed in section 1.3. In section 1.4, we provide an example of calculating the shortest path using Dijkstra's algorithm that minimizes the total percentile of a

performance metric, such as delay, energy, jitter and power attenuation of a signal. Finally, the conclusions are given in section 1.5.

1.1. The need for adding percentiles

Let us consider a performance metric such as the response time of a router and a web service or of a software process. Typically, we use the average of this metric as a performance indicator. For instance, we will say that the average time it takes for a specific web service is 2 ms. However, we all know that averages can be misleading as they do not represent the range of values which the metric under study may take. A percentile of the metric provides a better understanding of a system, since the percentile statistically bounds the behavior of the system. The qth percentile, such as the 95th percentile, of a variable X is defined as a value below which X lies q% of the time.

There is a plethora of situations in practice where we have to add percentiles of different random variables in order to calculate an aggregate percentile. Below, we describe some examples where this problem arises:

– *response time in a web service:* the execution of a web-based service may involve several sites, each carrying out part of the service flow. Given that each site can guarantee the 95th percentile of its own response time, the question is what is the 95th percentile of the total time? This end-to-end percentile can then be used in the negotiation of the contract with the user;

– *testing a large suite of software:* let us consider a suite of software components that provide a service, such as the IP multimedia subsystem (IMS). This is the signaling protocol used to setup multimedia sessions over the Internet Protocol (IP) network, and it is also used in long term evolution (LTE). Testing for software bottlenecks is standard routine before the software is released. However, due to the complexity of IMS, it is impossible to have all the components present in a lab. In view of this, the components that are not available for testing are often represented by idealized

simulations that are generally built as "no-op" stubs that return results artificially fast. As a result, the end-to-end response distribution cannot be reliably obtained. An alternative solution is to test only subgroups of software components at a time and obtain the percentile of the response time for each group. The individual percentiles can then be added in order to get an estimate of the end-to-end percentile response time;

– *QoS in multidomain routing:* user traffic typically originates at a local area network, and then it traverses an access network before being channeled into a wide area network (WAN) or a series of WANs, each operated by a different ISP, to reach its destination that may be another access network. Time sensitive traffic, such as Voice over IP (VoIP) and interactive video, needs to be treated by the ISPs in such a way so that the end-to-end delay and the end-to-end jitter is minimum. Again, the same problem arises here. Each ISP typically guarantees the 95th percentile of the time to traverse its domain and the jitter generated within the domain due to congestion. Based on the individual percentiles, what guarantees can we provide for the end-to-end delay and jitter?

– *controlling the power budget:* an interesting problem arises in green data centers. Specifically, let us assume that we want to limit the total power consumption so that 95% of the time it is less than a power budget P. The question arises as to how this can be calculated if we know the percentiles of the power consumption of the individual devices or groups of devices;

– a similar problem is also encountered in cloud computing where multiple software components run in a virtual environment on the same blade, one component per virtual machine (VM). Each VM is allocated a virtual central processing unit (CPU), which is a fraction of the blade's CPU. The hypervisor automatically monitors CPU usage for each VM. The question here is how do we allocate the blade's CPU to the multiple VMs running on the same blade so that a given percentile of the response time of each VM is satisfied while at the same time the percentile of the overall power consumption is bounded?

1.2. Calculation of the weight function

Very little work has been done on how to add percentiles and also how to partition a percentile to individual components. Kreifeldt and Nah [KRE 95] reported on the error of adding and subtracting percentiles of anthropometric dimensions in order to derive other relevant dimensions. The work focuses particularly on Gaussian distributions and adding/subtracting equal percentile points. The key findings are that the error between the assumed (added/subtracted) percentile and actual percentile depends on the percentile point, the correlation coefficients and the standard deviation ratios of the components. Also, the error decreases as the correlation increases and/or the standard deviation ratio decreases. The issue of adding percentiles was also addressed in a white paper on interprovider QoS by the MIT Communications Futures Program [CFP 06]. The paper addressed the issue of how to allocate the end-to-end response time, packet loss and jitter across multiple operators. The response time was expressed as the mean, and the jitter was expressed as a percentile of the interarrival time at the destination. The authors proposed a method for adding the individual operators' jitter. As will be shown in section 1.4, their method is grossly inaccurate.

The problem studied in this chapter can be defined as follows. Let us consider a system consisting of n individual and independent components, as shown in Figure 1.1, each characterized by a random variable X_i, $i = 1, 2, \ldots, n$.

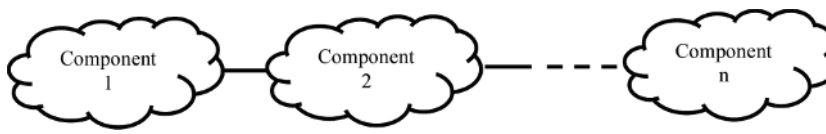

Figure 1.1. *Composition of n components*

We assume that for each component i we know x_i, the qth percentile of a given metric of interest, such as the response time, power consumption and jitter. We calculate x, the qth percentile of the end-to-end metric over all the n components, by computing a weight w with which we weigh up the arithmetic sum. We first work with

random variables that are exponentially distributed with either the same or different rate. Then, we extend the results to the more general Coxian-2 distribution.

1.2.1. *Exponential components with identical rate parameters*

We start by assuming that our metric of interest is exponentially distributed with a probability density function (PDF): $f(x) = \mu e^{-\mu x}$ and a cumulative density function (CDF): $F(X) = 1 - e^{-\mu x}$, where μ is the rate parameter of the exponential distribution. The sequence of the n components can then be represented by a sequence of n exponential distributions as shown in Figure 1.2.

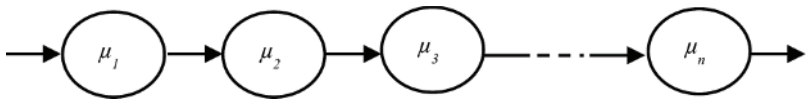

Figure 1.2. *A sequence of exponential distribution*

In the case where $\mu_1 = \mu_2 = \cdots = \mu_n = \mu$, the end-to-end distribution is the well-known Erlang-n distribution. The PDF $f(x)$, CDF $F(X)$ and Laplace transform $f_x^*(s)$ of an Erlang distribution are as follows:

$$f(x) = \mu e^{-\mu x} \frac{(\mu x)^{n-1}}{(n-1)!}$$

$$F(X) = 1 - \sum_{i=0}^{n-1} e^{-\mu x} \frac{(\mu x)^i}{i!}$$

$$f_x^*(s) = \left(\frac{\mu}{\mu + s}\right)^n$$

It is of interest to examine how the qth percentile of the individual components is related to the qth percentile of the end-to-end distribution. In Figure 1.3, we plotted the 80th percentile of an Erlang-n along with the sum of the 80th percentile of the individual exponential components, with μ = 5. The plots are given as a function

of the number of components n, which was varied from 1 to 30. Similar results are given for the 95th percentile with $\mu = 1$.

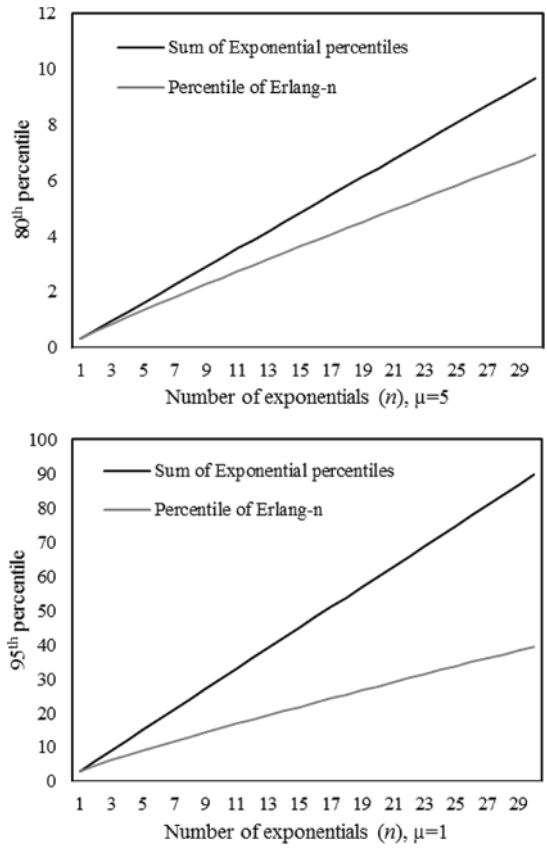

Figure 1.3. *Erlang-n versus sum of n exponentials*

In general, for a given value of μ, the arithmetic sum of the individual qth percentiles is greater than the qth percentile of the end-to-end distribution, and the difference increases as n increases. Also, the difference increases as q increases. Finally, as will be shown below, for a given value of q, the difference between the arithmetic sum of the individual qth percentiles and the qth percentile of the end-to-end distribution for each n is independent of the value of the parameter μ.

Let x_{exp} be the qth percentile of an exponential distribution with rate μ, notated as q_{exp}, and let x_{Erl} be the qth percentile of an Erlang distribution with n stages each with parameter μ, notated as q_{Erl}. Then, from their respective CDF formulas we have

$$1 - e^{-\mu x_{exp}} = q_{exp} \quad [1.1]$$

and

$$1 - \sum_{i=0}^{n-1} e^{-\mu x_{Erl}} \frac{(\mu x_{Erl})^i}{i!} = q_{Erl} \quad [1.2]$$

Given the values of q_{exp}, q_{Erl} (which may or may not be the same), μ and x_{exp}, we are interested in finding out a weight function w such that:

$$x_{Erl} = w x_{exp} \quad [1.3]$$

Equation [1.1] can be rewritten as:

$$x_{exp} = \frac{-ln(1 - q_{exp})}{\mu}$$

Hence, equation [1.3] becomes:

$$x_{Erl} = w \frac{-ln(1 - q_{exp})}{\mu}$$

Substituting x_{Erl} in equation [1.2], we have:

$$1 - \sum_{i=0}^{n-1} e^{-\mu w \frac{-ln(1-q_{exp})}{\mu}} \frac{(\mu w \frac{-ln(1 - q_{exp})}{\mu})^i}{i!} = q_{Erl}$$

or

$$1 - q_{Erl} = e^{w ln(1-q_{exp})} \sum_{i=0}^{n-1} \frac{(-w ln(1 - q_{exp}))^i}{i!}$$

or

$$ln(1 - q_{Erl}) = wln(1 - q_{exp}) + ln\sum_{i=0}^{n-1}\left\{\frac{(-wln(1-q_{exp}))^i}{i!}\right\} \quad [1.4]$$

Expression [1.4] has no analytical closed-form solution, but it can be solved for w numerically for a given q and n (e.g. using bisection or Newton's method). Thus, we get w, the required weight function, and when multiplied by x_{exp}, it gives the required value of x_{Erl}. Note that [1.4] is not dependent on either x_{exp} or μ. Thus for the Erlang case, the weight w is constant for any given q and n.

It should be noted here that $q_{\exp}$ needs not be the same as q_{Erl}, i.e. equation [1.4] calculates a weight for converting exponential percentile to any Erlang percentile (and vice versa). Also, μ is not present in equation [1.4], confirming our previous observation that the difference between the arithmetic sum of individual qth percentiles and the qth percentile of the end-to-end distribution for each n is independent of μ.

Thus, given a fixed parameter μ, equations [1.3] and [1.4] give us an exact formula to calculate a percentile of an Erlang distribution, given a specific percentile of the corresponding exponential distribution. Figure 1.4 gives results for the 95th and 99th percentiles, respectively, for μ = 1, 5, 100. In each figure, we plotted x computed using equations [1.3] and [1.4] and also using the CDF of the Erlang-n, for n = 1, 2, ..., 30. As expected, the results are identical.

1.2.2. *Exponential components with different rate parameters*

In the case where the rate parameters of the exponential components are not necessarily the same, the end-to-end distribution is a hypoexponential distribution. In general, if we have n independently distributed exponential random variables X_i, then the random variable, $X = \sum_{i=1}^{n} X_i$, is hypoexponentially distributed. We note that the PDF and CDF formulas of the hypoexponential distribution are not readily available in the literature. They can be

obtained by inverting its Laplace transform using partial fraction expansion.

$$f_x^*(s) = \frac{\mu_1}{\mu_1 + s}\frac{\mu_2}{\mu_2 + s}\ldots\frac{\mu_n}{\mu_n + s}$$

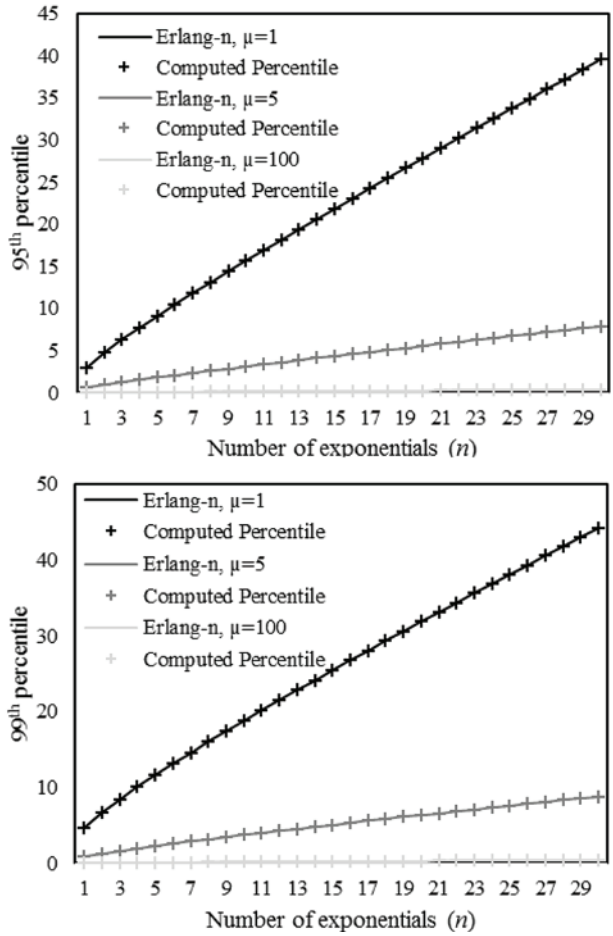

Figure 1.4. *95th and 99th percentiles using Erlang-n CDF and weight function for μ =1, 5, 100*

We have that:

$$f_x^*(s) = \frac{\mu_1\mu_2..\mu_n}{(\mu_2-\mu_1)(\mu_3-\mu_1)...(\mu_n-\mu_1)}\frac{1}{\mu_1+s} + \frac{\mu_1\mu_2..\mu_n}{(\mu_1-\mu_2)(\mu_3-\mu_2)...(\mu_n-\mu_2)}\frac{1}{\mu_2+s} + \cdots + \frac{\mu_1\mu_2..\mu_n}{(\mu_1-\mu_n)(\mu_2-\mu_n)...(\mu_{n-1}-\mu_n)}\frac{1}{\mu_n+s}$$

We observe that $\mu_i/(\mu_i + s)$ is the Laplace transform of an exponential distribution with parameter μ_i. Hence, we have,

$$f(x) = \frac{\mu_1\mu_2..\mu_n}{(\mu_2-\mu_1)(\mu_3-\mu_1)...(\mu_n-\mu_1)}e^{-\mu_1 x} + \frac{\mu_1\mu_2..\mu_n}{(\mu_1-\mu_2)(\mu_3-\mu_2)...(\mu_n-\mu_2)}e^{-\mu_2 x} + \cdots + \frac{\mu_1\mu_2..\mu_n}{(\mu_1-\mu_n)(\mu_2-\mu_n)...(\mu_{n-1}-\mu_n)}e^{-\mu_n x}$$

or

$$f(x) = \sum_{i=1}^{n}\left\{\mu_i e^{-\mu_i x}\prod_{\substack{j=1\\ j\neq i}}^{n}\frac{\mu_j}{\mu_j-\mu_i}\right\} \quad [1.5]$$

From equation [1.5], we can obtain the CDF of a hypoexponential distribution. We have

$$F(X) = \sum_{i=1}^{n}\left\{(1-e^{-\mu_i x})\prod_{\substack{j=1\\ j\neq i}}^{n}\frac{\mu_j}{\mu_j-\mu_i}\right\} \quad [1.6]$$

The observations made earlier for Erlang distribution are also valid for the hypoexponential distribution. In Figure 1.5, we plotted the sum of the 95th and 99th percentile of n exponential components, where the parameter value of each component i is $\mu_i = i$, $i = 1, 2, \ldots, n$, along with the 95th percentile of the corresponding end-to-end hypoexponential distribution.

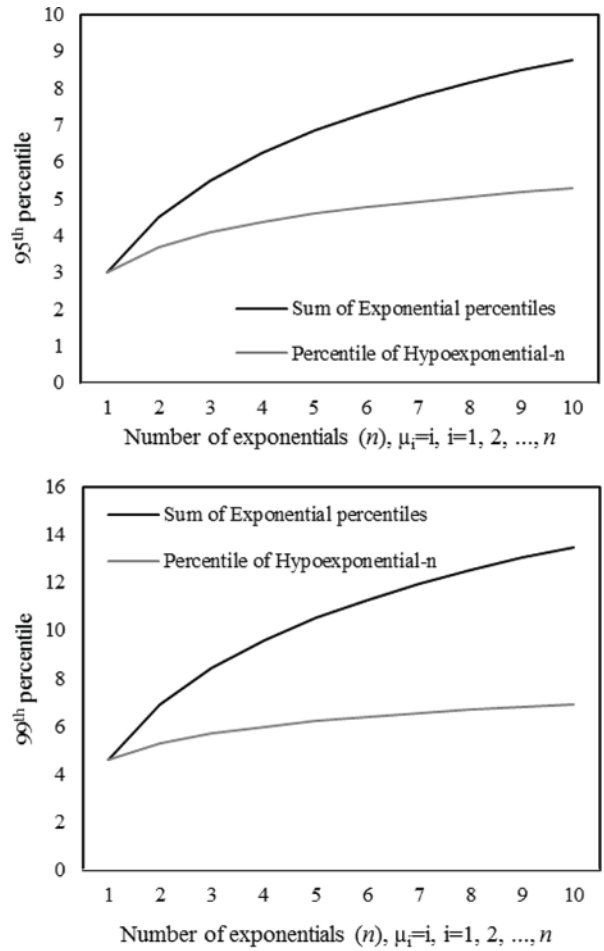

Figure 1.5. *Hypoexponential-n versus sum of n exponentials*

It should be mentioned here that the shape of the *qth* percentile of the hypoexponential distribution depends on the value of the μ parameters.

The results of equations [1.3] and [1.4] can be easily generalized to the hypoexponential case. First, let us consider a hypoexponential distribution with two stages, each with parameters μ_1 and μ_2 respectively. Let x_H be the *qth* percentile of the hypoexponential

distribution, notated as q_H, and let x_i be the *qth* percentile of the *ith* exponential stage, $i = 1, 2$, notated as q_i. The CDF is given by:

$$(1 - e^{-\mu_1 x_H})\frac{\mu_2}{\mu_2 - \mu_1} + (1 - e^{-\mu_2 x_H})\frac{\mu_1}{\mu_1 - \mu_2} = q_H \qquad [1.7]$$

Now, we can find out the weight function, w, such that:

$$x_H = w(x_1 + x_2) \qquad [1.8]$$

Again, the CDF of each exponential component i can be written as:

$$x_i = \frac{-ln(1 - q_i)}{\mu}$$

or

$$\mu = \frac{-ln(1-q_i)}{x_i} \qquad [1.9]$$

Putting the value of x_H from equation [1.8] and μ_i from equation [1.9] in equation [1.7], we get

$$q_H = \left(1 - e^{-\frac{-\ln(1-q_1)}{x_1} w(x_1 + x_2)}\right)\left(\frac{\frac{-\ln(1-q_2)}{x_2}}{\frac{-\ln(1-q_2)}{x_2} - \frac{-\ln(1-q_1)}{x_1}}\right) + \left(1 - e^{-\frac{-\ln(1-q_2)}{x_1} w(x_1 + x_2)}\right)\left(\frac{\frac{-\ln(1-q_1)}{x_1}}{\frac{-\ln(1-q_1)}{x_1} - \frac{-\ln(1-q_2)}{x_2}}\right) \qquad [1.10]$$

Given x_i and q_i, $i = 1, 2$, we can calculate the weight function, w, using the above expression for any percentile q_H of the two stage hypoexponential. This weight, when multiplied by the sum of x_i, $i = 1, 2$, gives the value of x_H.

The above expression can easily be generalized to n stages, as follows:

$$q = \sum_{i=1}^{n}\left\{\left(1 - e^{-w ln(1-q)\frac{\sum_{k=1}^{n} x_k}{x_i}}\right)\prod_{\substack{j=1 \\ j \neq i}}^{n}\frac{x_j \ln(1-q)}{x_j \ln(1-q) - x_i \ln(1-q)}\right\} \qquad [1.11]$$

and

$$x_H = w \sum_{i=1}^{n} x_i \qquad [1.12]$$

For illustration purposes, we have set $n = 6$ and $q = 0.95$, and the parameters μ_i have been varied as follows. In the left graph of Figure 1.6, we have set $\mu_i = i\mu_1$, $i = 1, \ldots, 6$, and in the right graph we have set $\mu_i = i/\mu_1$, $i = 2, \ldots, 6$. In Figure 1.7, $\mu_i = i$, for $i = 1, ..., 4$, μ_5 increases and μ_6 decreases over the same range for each successive observation. In each figure, we plotted x computed using equations [1.11] and [1.12] and also using the CDF of the hypoexponential distribution.

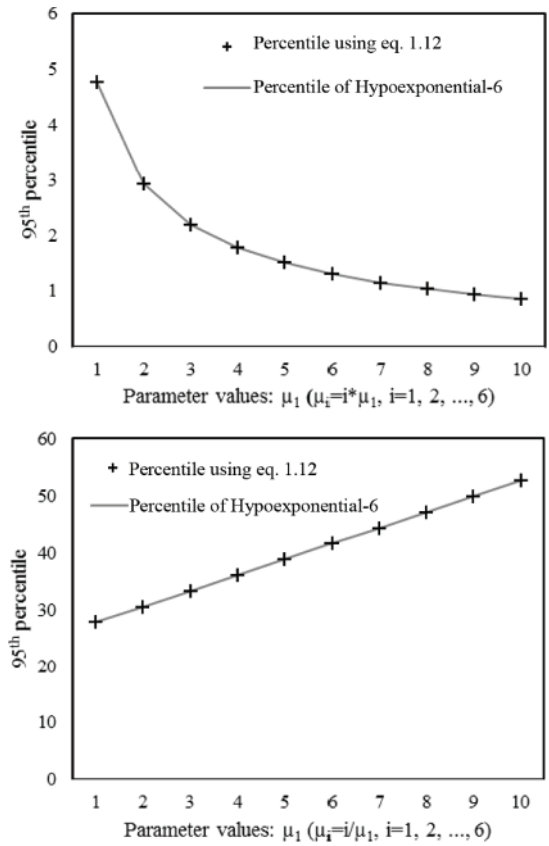

Figure 1.6. *95th percentile using hypoexponential-6 CDF and weight function*

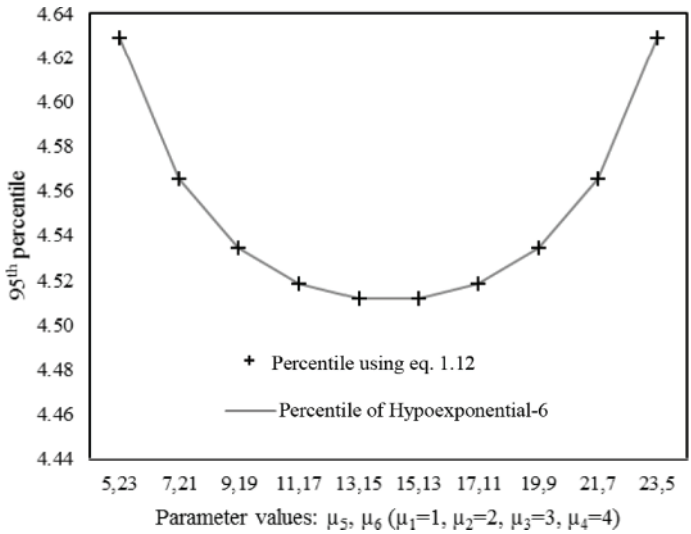

Figure 1.7. *95th percentile using hypoexponential-6 CDF and weight function*

1.2.3. *Two-stage Coxian*

The same ideas can be applied to a more generalized distribution, like a two-stage Coxian distribution which in turn is a special case of the phase type distribution (PHD). Before we proceed with the calculation of the weight *w*, we give a brief review of the PH and Coxian distributions.

Consider a continuous-time Markov process with $n + 1$ states, $n \geq 1$, such that the states 1, ..., n are transient states and state $n + 1$ is the absorbing state. Further, the process has an initial probability of starting in any of the $n + 1$ states given by the probability vector (α, α_{n+1}), where α is a 1xn vector. The PHD is a continuous distribution in $[0, \infty]$ of the time until the absorption state is reached in a continuous-time finite state Markov process. This process can be written in the form of a generator matrix as follows:

$$Q = \begin{matrix} \\ transient \\ absorbing \end{matrix} \begin{matrix} \begin{matrix} transient & absorbing \end{matrix} \\ \begin{bmatrix} S & S_0 \\ O & 0 \end{bmatrix} \end{matrix}$$

Where S is a $2n$x$2n$ transition rate matrix and S_0 is defined as: $S_0 = -Sxe$. O is a 1xn vector with each element set to 0, and e is an nx_1 vector with each element set to 1. The pair (α, S) is called the representation function of the PHD.

The PDF and CDF of a PHD are as follows:

$$f(x) = \alpha e^{Sx} S_0$$

$$F(X) = 1 - \alpha e^{Sx} e$$

where e^{Sx} represents a matrix exponential, which is defined as:

$$e^A = \sum_{k=0}^{\infty} \frac{1}{k!} A^k = 1 + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots$$

A Coxian distribution with n stages, referred to as Coxian-n, is a generalization of the Erlang distribution, and a special case of the PHD.

One of the most commonly used Coxian distributions is the Coxian-2 distribution shown in Figure 1.8. It consists of an exponentially distributed state with rate μ_1, followed by a second exponentially distributed state with rate μ_2 with probability a (= 1 - b).

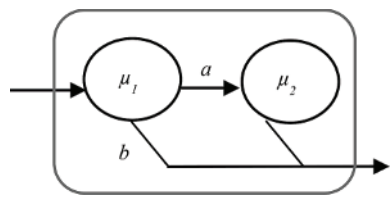

Figure 1.8. *The Coxian-2 distribution*

The Laplace transform of Coxian-2 is given by:

$$f_x^*(s) = a \frac{\mu_1}{\mu_1 + s} \frac{\mu_2}{\mu_2 + s} + b \frac{\mu_1}{\mu_1 + s}$$

The PDF and CDF of Coxian-2 are given by:

$$f(x) = b\mu_1 e^{-\mu_1 x} + a\left(\frac{\mu_1\mu_2}{\mu_2 - \mu_1}e^{-\mu_1 x} + \frac{\mu_1\mu_2}{\mu_1 - \mu_2}e^{-\mu_2 x}\right)$$

$$F(X) = b(1 - e^{-\mu_1 x}) + a\left\{\frac{\mu_2}{\mu_2 - \mu_1}(1 - e^{-\mu_1 x}) + \frac{\mu_1}{\mu_1 - \mu_2}(1 - e^{-\mu_2 x})\right\}$$

The Coxian-2 is a special type of the PHD, and it can be described with the rate matrix:

$$S = \begin{bmatrix} -\mu_1 & a\mu_1 \\ 0 & -\mu_2 \end{bmatrix}$$

Now, let us consider a series of n Coxian-2 components connected in tandem as shown in Figure 1.9.

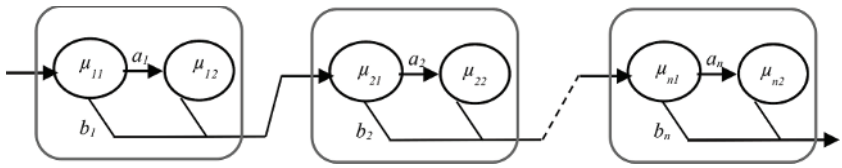

Figure 1.9. *A series of Coxian 2 distributions*

The end-to-end distribution, where each component is a Coxian-two, can be represented by a PHD with the following 2*n*x2*n* rate matrix:

$$S = \begin{bmatrix} -\mu_{11} & a_1\mu_{11} & b_1\mu_{11} & 0 & 0 & & 0 & 0 \\ 0 & -\mu_{12} & \mu_{12} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & -\mu_{21} & a_2\mu_{21} & b_2\mu_{21} & & 0 & 0 \\ 0 & 0 & 0 & -\mu_{22} & \mu_{22} & & 0 & 0 \\ & & \vdots & & \vdots & & & \vdots \\ 0 & 0 & 0 & & & & -\mu_{n1} & a_n\mu_{n1} \\ 0 & 0 & 0 & & \cdots & & 0 & -\mu_{n2} \end{bmatrix},$$

and with a starting vector $\alpha = [1, 0, \ldots, 0]$ of length n.

The CDF of the PH can be solved to get the value of x corresponding to a percentile q using an analytical tool such as Matlab and Mathematica.

Let x_{e2e} be the qth percentile of the PHD corresponding to the n component Coxian-2 distribution. Let x_i be the qth percentile of the ith Coxian-2 stage, i = 1, ..., n. We are looking for a weight function w such that:

$$w\,(x_1 + x_2 + \ldots + x_n) = x_{e2e}$$

Following the same steps as in the previous section and assuming that we know the parameters (μ_1, μ_2, a) of each Coxian-2 component, we have:

$$q = 1 - \boldsymbol{\alpha} e^{\boldsymbol{S} w \sum_{i=1}^{n} x_i} \mathbf{1} \qquad [1.13]$$

$$x_{e2e} = w \sum_{i=1}^{n} x_i \qquad [1.14]$$

For illustration purposes, we plotted in Figure 1.10 different percentiles calculated from the original CDF and expression [1.14] for n = 1, 2, …, 10, where the parameters (μ_1, μ_2, a) of the components are all equal to (1, 2, 0.5). Similar results are also given in Figure 1.11, assuming that the parameters of the ith component (μ_1, μ_2, a) = (i, 2i, 0.5).

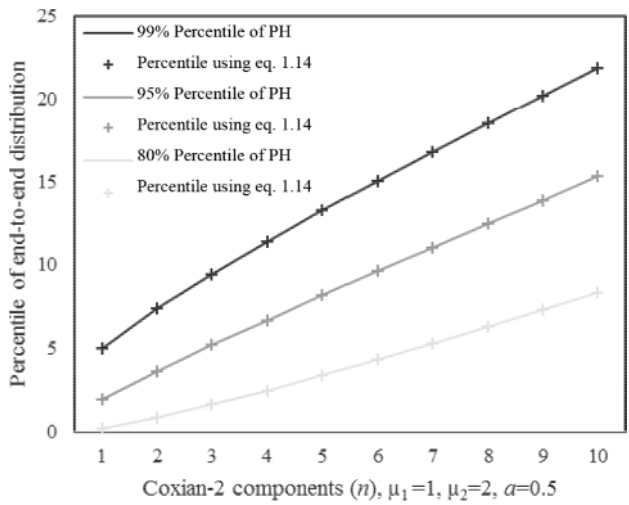

Figure 1.10. *95th percentile using phase-type CDF and weight function*

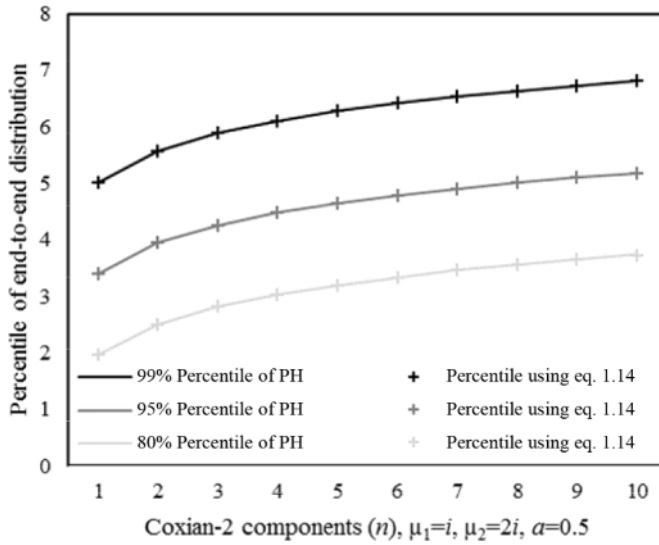

Figure 1.11. *Percentile using phase-type CDF and weight function*

1.3. Interprovider quality of service

A set of recommendations is presented in a white paper [CFP 06] to simplify deployment of interprovider QoS for services spanning multiple networks. Enabling QoS-based peering among various providers is an area of open research and debate. The document recommends standards and best practices that can help simplify the deployment of QoS for traffic that traverses the network of various providers.

The authors considered three network performance metrics: OWD, one-way IP packet loss ratio (IPLR) and one-way IP packet delay variation (IPDV) also known as jitter. Also, they defined two QoS classes, a single-low latency service class and a best effort class. The low latency class is suitable for applications like VoIP and is consistent with the service class definition of Y.1541 [ITU 05]. The parameters specified in Y.1541 are as follows. The OWD is defined as the mean one-way end-to-end delay with suggested values ranging

from 100 to 400 ms (depending on geographic distance). The IPDV is defined as a percentile of the interarrival time of successive packets at the destinations with suggested value of 50 ms being the 99th percentile, and the suggested IPLR value is 1×10^{-3}. In order to support time sensitive traffic with desired QoS in a multiprovider network, the end-to-end performance metrics must be met as specified above.

The white paper lists some best practices that, if used by a substantial number of network providers, can accelerate the planning and deployment of QoS-enabled networks supporting above mentioned performance metrics. Specifically, the authors propose a fixed allocation of the end-to-end QoS metrics to each provider. The basic idea is that a provider should not depend on *a priori* knowledge of other providers' networks for planning his/her own network and providing QoS guarantees.

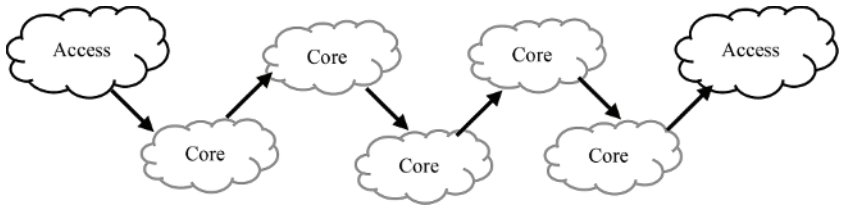

Figure 1.12. *A multidomain network*

Figure 1.12 gives the multidomain network as considered in the white paper. It consists of three concatenated core networks. In addition, one access network and zero or one metro networks per end are assumed. As the metro networks are not treated differently from the core networks, as far as budget allocation is concerned, the complete network under consideration consists of two access networks and five core networks. This is considered as a realistic topology for the end-to-end services.

We are particularly interested in the budgeting of the IPDV across various segments of the network as our mathematical modeling

suggests that the thresholds proposed in [CFP 06] have reasonable room for improvement.

The first observation made by the authors about IPDV is that a simple arithmetic division will result in more stringent requirements than actually required to meet the end-to-end goals. This observation is consistent with our finding in section 1.2. Also, the authors recognized the statistical nature of IPDV, and used probabilistic measures to allocate the metric. A major portion of the IPDV budget is allocated to the access network, where lower link speeds mandate more generous allocations. We agree with the analysis so far.

The thresholds proposed are as follows: the 99th percentile of the IPDV for a core network should be less than 2 ms and the 99th percentile of the IPDV for an access network should be less than 16 ms. It is important to mention here that these threshold values are not calculated using a mathematical method, but they were proposed so that they can be readily achievable in core and access networks. Based on these thresholds, they calculated that in a network of five core segments, the end-to-end IPDV is less than 20 ms with a probability of 0.99994, and in a network consisting of two access and five core segments it is less than 50 ms with a probability of 0.9998. Below are the calculations as presented in the paper for a five core segment network.

Prob (e2e IPDV < 20 ms) ≈ Prob (Sum of IPDV thresholds < 20 ms) ≥

Prob (all 5 intervals are “Low IPDV”) + Prob (4 out of 5 intervals are Low IPDV and one is High IPDV) + Prob (3 out of 5 intervals are Low IPDV and 2 are High IPDV)

$$= (0.99)^5 + \binom{5}{4}(0.99)^4(0.00999) + \binom{5}{3}(0.99)^3(0.00999)^2 = 0.99994$$

However, we find that the above calculations are grossly inaccurate. IPDV is expressed as a percentile of the interarrival time

of successive packets at the destination, and as stated earlier, percentiles cannot be added arithmetically, so the statement "Prob (e2e IPDV < 20 ms) ≈ Prob (Sum of IPDV thresholds < 20 ms)" is not accurate mathematically.

The document states that for a five core segment network, if the 99th percentile of each core segment is less than 2 ms, then the end-to-end IPDV is less than 20 ms with 0.99994 probability. Let us verify the above statement assuming an Erlang model with five states. That is, we assume that the PDF of the OWD through a core network is exponentially distributed. Each of the exponential stages has a 99th percentile of 2 ms. Hence, we can use results from section 1.2.1 to calculate the weight *w* such that:

$x_{Erl} = wx_{exp}$, where $x_{exp} = 2$ ms

Using equation [1.4]:

$$\ln(1 - q_{Erl}) = w\ln(1 - q_{exp}) ln\left(\sum_{i=0}^{n-1} \frac{(-w\ln(1-q_{exp}))^i}{i!}\right),$$

where $q_{Erl} = 0.99994$ and $q_{exp} = 0.99$, we calculate the weight function to be $w = 4.00125$, and hence: $x_{Erl} = 8.0025$ ms

That is, if we have a five core segment network where the 99th percentile of each component is 2 ms, then with probability 0.99994, the end-to-end IPDV will be less than or equal to 8.0025 ms. This is less than half of what has been calculated in [CFP 06], i.e. 20 ms.

Also, we consider the inverse problem of calculating the 99th percentile of the individual core component, given that the end-to-end IPDV is less than 20 ms with 0.99994 probability. Again using equation [1.3] with $x_{Erl} = 20$ and $w = 4.00125$, we get:

$x_{exp} = 5$ ms

That is, if we have a five core segment network and we want the 99.994 percentile of the end-to-end IPDV to be 20 ms, then the 99th

percentile of each of the core segment should be 5 ms (instead of 2 ms as proposed in [CFP 06]).

Next, we model a seven component network with two access and five core networks, using a hypoexponential distribution. As proposed in the paper, the 99th percentile of IPDV for core should be less than 2 ms and the 99th percentile of IPDV for access should be less than 16 ms. Using equations [1.11] and [1.12], we calculate the weight function w such that:

$$x_H = w \sum_{i=1}^{7} x_i$$

and we obtain that x_H = 40.8815 ms. That is, the end-to-end IPDV is less than equal to 40.8815 ms with probability 0.9998 (as opposed to 50 ms calculated in [CFP 06]) thus allowing 18% more of IPDV for a more generous allocation at the access and/or core segments.

1.4. Single source shortest path using Dijkstra's algorithm

The expressions obtained in section 1.2 can be directly used in a search algorithm, like Dijkstra's algorithm, to calculate the shortest path in a graph that minimizes the total percentile cost of a performance metric such as delay, energy, jitter and power attenuation of signal. Below, we present an example to illustrate this.

Consider the network given in Figure 1.13 where the link cost represents the delay to reach from one node to another. We are interested in finding the shortest/quickest path from node O to all other nodes, i.e. the minimum spanning tree (MST) rooted at node O. We assume the delay to be exponentially distributed.

First, we do a standard run of Dijkstra's algorithm where the cost of each link is the average delay to traverse the link, and the shortest path is defined as the path with the least end-to-end average delay.

Figure 1.14 gives the MST (represented by dark lines) rooted at node O. The average delay to reach any node from node O is given above the node.

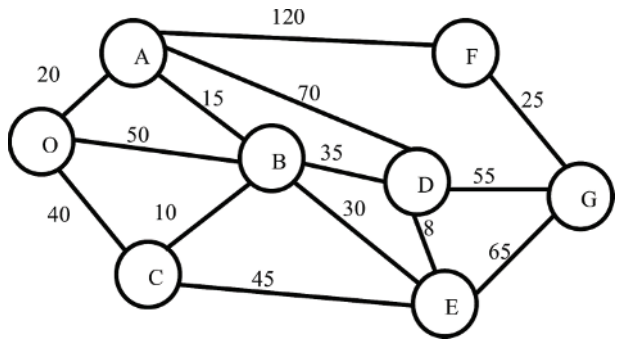

Figure 1.13. *The network under study*

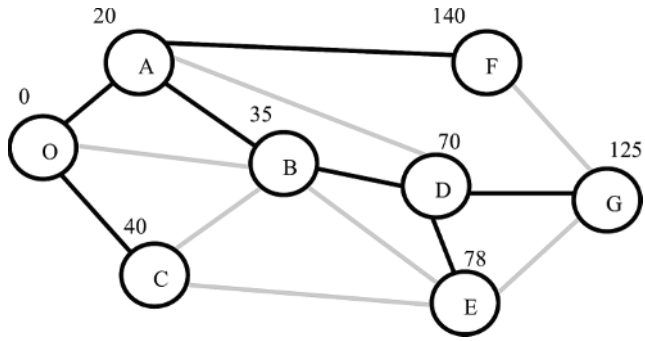

Figure 1.14. *The minimum spanning tree using the average delay*

Now, we use Dijkstra's algorithm to combine percentile delays. Again using Figure 1.13, we now define the per link cost to be the 95th percentile delay to traverse the link, and the shortest path is defined as the path with the least end-to-end 95th percentile delay. The addition of the percentiles is done using [1.11] and [1.12]. The resultant MST is shown in Figure 1.15 (represented by dark lines).

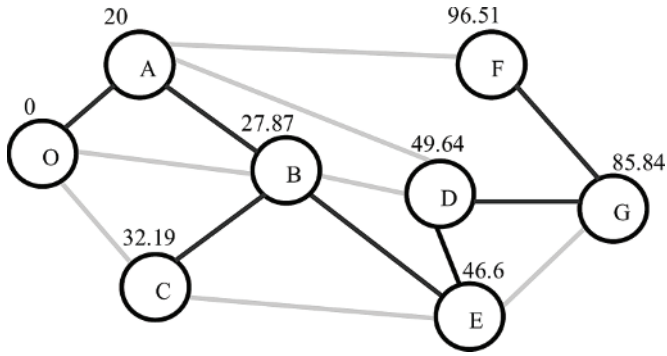

Figure 1.15. *The minimum spanning tree using the percentile delay*

Notice how the resultant MSTs present two very different routing views of the network. For example, consider the path from O to F. If we want to minimize the average end-to-end delay, packets from O to F should be routed through A. This path guarantees that the average delay will be 120 time units or less (Figure 1.14). However, if we are concerned about minimizing the 95th percentile delay, the packets should be routed through A→B→E→D→G (Figure 1.15). This path, though having a higher number of hops, guarantees that 95% of the packets will experience delay of 96.51 time unit or less.

The preference of one view over the other depends on how service level agreements (SLAs) are defined. If an SLA is defined in terms of average delays, the traditional average delay MST suffices. However, as is the case with present-day networks, for real-time communications, statistical bounding of the delay is preferred over simple averaging. In this case, routing based on delay percentiles seems to be more meaningful.

1.5. Conclusions

In this chapter, we described a method to add n percentiles of exponentially distributed random variables with or without the same mean. This method was also extended to the case where the random

variables follow a Coxian-2 distribution. There is a plethora of situations in practice where we have to add percentiles of different random variables in order to calculate an aggregate percentile. We demonstrated the usefulness of the results obtained in this chapter through two examples. In the first example, we addressed an issue that arose in a set of recommendations for interprovider QoS, and in the second example we employed Dijkstra's algorithm to find the shortest path minimizing the end-to-end percentile delay.