Adding Percentiles of Erlangian Distributions

Bushra Anjum and Harry Perros, *Fellow, IEEE*

***Abstract*—In networking, enterprise computing and many other areas the issue of adding percentiles of a performance metric, such as the response time, arises regularly. Percentiles cannot be added using the arithmetic sum, and surprisingly there are no known formulae that permit us to do so correctly. In this paper, we obtain an exact analytical expression for adding percentiles of random variables which can be represented by a series of generalized exponential stages (e.g., Erlang, hypoexponential and two-stage Coxian). We demonstrate the applicability of our results by an example in which we use our expressions in the Dijkstra's algorithm to calculate the shortest 'percentile delay' path.**

***Index Terms*—Adding percentiles, Erlang, two-stage Coxian, quality of service, hypoexponential, shortest path calculation.**

I. Introduction

WE often deal with performance metrics such as the response time of a router or a web service to guarantee service. Typically, we use the average of this metric as an indicator of its performance. For instance, we will say that the average time it takes for a specific web service is 5ms. However, we all know that averages can be misleading as they are influenced by outliers and they do not represent the range of values of the response time under study. On the other hand, a percentile of the metric provides a better understanding than the mean value, since it bounds the behavior of a system statistically. The c^{th} percentile, such as the 95^{th} percentile, of a variable X is a value below which X lies $c\%$ of the time.

Let us consider the case of providing QoS in multi-domain routing. User traffic typically originates at a local area network, and then it traverses an access network before it is channeled into a WAN or a series of WANs, each operated by a different ISP, to reach its destination. For time-sensitive traffic, such as VoIP and interactive video, each ISP typically guarantees the 95^{th} percentile of the time to traverse its domain and of the jitter generated within the domain. Based on these individual percentiles, what guarantees can one provide for the end-to-end delay and jitter? A similar situation may arise in execution of a web-based service which may involve several sites, each carrying out part of the service flow. Given that each site can guarantee the 95^{th} percentile of its own response time, the question is what is the 95^{th} percentile of the total end-to-end response time?

Surprisingly, despite the many uses of the percentile of a performance metric, there is no known formula for adding percentiles, or carrying out other operations such as subtractions and partitions. In this paper, we present an exact analytic expression for adding percentiles, starting with the simple exponential components and working our way to a more generalized two-stage Coxian distribution.

The paper is organized as follows. In section 2 we give a literature review and define the problem. In section 3, we present the expressions for adding up percentiles of generalized Erlangian random variables. In section 4 we demonstrate the use of the expressions through an example in which we obtain the shortest path in a graph that minimizes a given percentile of the end-to-end delay. Finally, the conclusions are given in section 5.

B. Anjum and H. Perros are with the Computer Science Department , North Carolina State University, Raleigh, NC, USA e-mail: {banjum,hp}@csc.ncsu.edu.

II. Literature Review and Problem Definition

As mentioned above, very little work has been done on how to add percentiles. Kreifeldt [1] reported on the error of adding and subtracting percentiles using the arithmetic sum and subtraction respectively 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 finding is that the error between the added/subtracted percentile and the actual percentile depends on the percentile point, the correlation coefficients, and the standard deviation ratios of the components. The issue of adding percentiles was also addressed in a white paper on inter-provider QoS by the MIT Communications Futures Program [2]. The paper addressed the issue of how to allocate the end-to-end response time, packet loss and jitter across multiple operators.

The problem addressed in this paper can be formulated as follows. Let us consider a system consisting of n individual and independent components as shown in Fig. 1, and let us assume that each component is represented by a metric of interest, such as the response time, which follows a known Erlangian distribution. Given that we know the percentile of this metric for each component, we obtain an exact formula for combining these percentiles to obtain the end-to-end percentile of the metric over all the n components.

Fig. 1. Composition of n components

III. Calculation of the Weight Function

A. Exponential Components with identical rate parameter

We start with the simplest case where each component variable is assumed to be exponentially distributed with the same rate, i.e., $\mu_1=\mu_2=...=\mu_n=\mu$. In this case, the end-to-end distribution is an Erlang distribution with n phases. The CDF of Erlang distribution is:

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

We are looking for a weight function such that we can directly convert the c^{th} percentile of an exponentially distributed random variable (x_{exp}) to the corresponding percentile of an Erlang random variable (x_{Erl}) and vice versa. That is, we want to calculate a weight factor w such that:

$$x_{Erl} = w_{exp} \quad (2)$$

The CDF of an exponential random variable is given as $1 - e^{-\mu x_{exp}} = c$, which can be re-written as $x_{exp} = \frac{-ln(1-c)}{\mu}$. Replacing this value of x_{exp} in (2) we get $x_{Erl} = w\frac{-ln(1-c)}{\mu}$. Finally, replacing the value of x_{Erl} in (1) and re-arranging the terms, we arrive at the following expression:

$$1 - c = e^{wln(1-c)} \sum_{i=0}^{n-1} \left\{ \frac{(-wln\,(1-c))^i}{i!} \right\}$$

Taking natural logarithm of both sides,

$$ln(1-c) = wln(1-c) + ln \sum_{i=0}^{n-1} \left\{ \frac{(-wln(1-c))^i}{i!} \right\} \quad (3)$$

Equation (3) gives the relation between the weight function *w*, the number of nodes *n* and the percentile *c* in implicit form. The expression has no analytical closed-form solution, however it can be solved numerically for a given *c* and *n* (e.g. by using Bisection or Newton's method in Matlab). Thus we get *w*, the required weight function, and when multiplied by x_{exp}, it gives the required value of x_{Erl}. Note that (3) is not dependent on either x_{exp} or μ. Thus for the Erlang case, the weight *w* is constant for any given *c* and *n*.

B. Exponential Components with different rate parameter

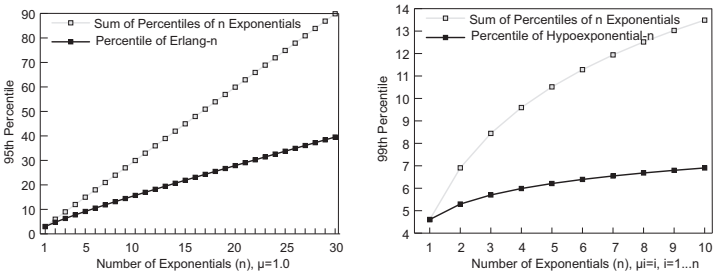
In this case, the end-to-end distribution is a hypoexponential distribution. This can be seen as a generalized Erlang distribution where each stage *i* has a different rate μ_i. The CDF of hypoexponential is:

$$c = \sum_{i=1}^{n} \left\{ \left(1 - e^{-\mu_i x}\right) \prod_{j=1, j\neq i}^{n} \frac{\mu_j}{\mu_j - \mu_i} \right\}$$

Let x_H be the c^{th} percentile of hypoexponential distribution and let x_i be the c^{th} percentile of the i^{th} exponential stage, *i*=1,...,n. Then, using the same approach as above, the weight function can be calculated numerically from (4) and (5).

$$x_H = w \sum_{i=1}^{n} x_i \quad (5)$$

In general, the arithmetic sum of the individual c^{th} percentiles is always greater than the c^{th} percentile of the end-to-end distribution. In the case, where each component is exponentially distributed, the difference increases as *n* increases. These observations are illustrated in Fig. 2(a) and 2(b).



(a) Identical exponential components (b) Different exponential components

Fig. 2. Arithmetic sum of exponential component percentiles vs. actual end-to-end percentile

C. Two-stage Coxian

The same idea can be applied to a more generalized distribution, like a two-stage Coxian (C_2) distribution. It's known that the PDF of any random variable, whose Laplace transform is a rational polynomial, can be represented by a Coxian distribution with k stages, Cox [3]. The C_2 is a popular distribution because it is simpler, and any Coxian distribution can be approximated by a C_2. Also, C_2 is a special type of Phase Type (PH) distribution. The PH distribution is any continuous distribution, *X*, on [0,∞) which can be obtained as a distribution of time until the absorption state is reached in a continuous time finite state Markov Chain with *n* transient states. The probability vector of starting in any of the *n* phases is given as α. The PDF and CDF of a PH distribution are as follows:

$$f(x) = \alpha e^{Sx} S_0, \quad F(x) = 1 - \alpha e^{Sx} \mathbf{1}$$

Where *S* is a *nxn* transition rate matrix and S_0 is defined as: S_0 **= - *S1***. ***1*** is an nx1 vector with each element 1.

The C_2 distribution is defined by three variables (μ_1,μ_2, α) and the Markov process can be represented using the following rate matrix *S* and start vector α:

$$\alpha = [1, 0], \quad S = \begin{bmatrix} -\mu_1 & \alpha\mu_1 \\ 0 & -\mu_2 \end{bmatrix}$$

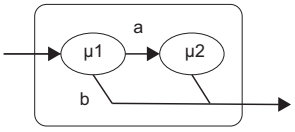


Fig. 3. C_2 distribution

Now let us consider a series of *n* C_2 tandem. The end-to-end service time distribution can be represented by a PH distribution with the following *2nx2n* rate matrix:

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

Let x_{e2e} be the c^{th} percentile of the PH distribution corresponding to the *n* component C_2 distribution. Let x_i be the

$$c = \sum_{i=1}^{n} \left\{ \left(1 - e^{-wln(1-c)\frac{\sum_{k=1}^{n} x_k}{x_i}} \right) \prod_{j=1, j\neq i}^{n} \frac{x_j ln(1-c)}{x_j ln(1-c) - x_i ln(1-c)} \right\} \quad (4)$$

c^{th} percentile of the i^{th} C_2 stage, *i*=1,...,n. We are looking for a weight function *w* such that:

$$w(x_1 + x_2 + ... + x_n) = x_{e2e}$$

Following the same steps as before, if the triplet (μ_1, μ_2, α) is known for each component, one can calculate the weight function from the expressions:

$$c = 1 - \alpha e^{Sw \sum_{i=1}^{n} x_i} \mathbf{1}, \quad x_{e2e} = w \sum_{i=1}^{n} x_i$$

IV. Single Source Shortest Path using Dijkstra's Algorithm

The expressions obtained in section III 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 (as opposed to the total average cost) of a performance metric such as, delay, energy, jitter or power attenuation of signal. Below we present an example to illustrate this.

Consider the network given in Fig. 4. We are interested in finding the shortest path from node O to all other nodes, i.e., the minimum spanning tree (MST) rooted at node O. The link cost is the delay which we assume to be exponentially distributed.

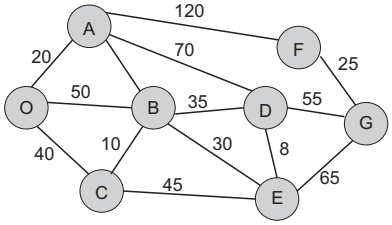


Fig. 4. The network under study

First we do a standard run of Dijstra'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. Fig. 5 gives the MST (represented by solid dark links) rooted at node O. The average delay to reach any node from node O is given above the node.

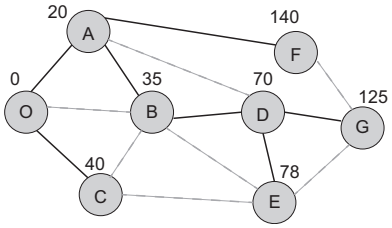


Fig. 5. The minimum spanning tree using the average delay

Now we use Dijkstra's algorithm to combine percentile delays. Again using Fig. 4, we now define the per link cost to be the 95^{th} percentile delay to traverse the link and the shortest path is defined as the path with the least end-to-end '95^{th} percentile' delay. The addition of the percentiles is done using (4) and (5). The resultant MST is shown in Fig. 6 (represented by solid dark links).

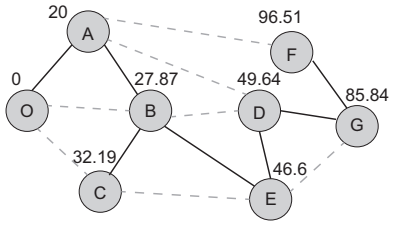


Fig. 6. The minimum spanning tree using the percentile delay

Notice how the resultant MSTs present two very different routing views of the network. For example, consider the path from O to F. If one wants to minimize the average end-to-end delay, packets from O to F should be routed through A (Fig. 5). This path guarantees that the average delay will be 120 time units or less. Whereas, if one is concerned about minimizing the 95^{th} percentile delay, the packets should be routed through A→B→E→D→G (Fig. 6). This path, though having increased 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.

V. Conclusions

It is inaccurate to add percentiles using the arithmetic mean. Surprisingly there are no known formulae that permit us to do so correctly. In this paper, we obtained a simple expression for adding percentiles of distributions with generalized exponential stages. Also we demonstrated its usefulness through the use of an example where we employed the expressions in Dijkstra's algorithm to find the shortest path minimizing end-to-end percentile delay.

References

[1] Kreifeldt, J. G., and Nah, K., *Adding and subtracting percentiles - How bad can it be?*, Human Factors and Ergonomics Society Annual Meeting Proceedings, 1995, pp. 301-305.

[2] MIT Communications Futures Program, *Inter-provider Quality of Service*, white paper draft 1.1. 2006. URL: http://cfp.mit.edu/groups/qos.shtml

[3] Cox, D.R., *A use of complex probabilities in the theory of stochastic processes*, Mathematical Proceedings of the Cambridge Philosophical Society (1995)