A Leftover Service Curve Approach to Analyze Demultiplexing in Queueing Networks

Hao Wang
University of Kaiserslautern
Email: wang@informatik.uni-kl.de

Florin Ciucu
T-Labs / TU Berlin
Email: florin@net.t-labs.tu-berlin.de

Jens Schmitt
University of Kaiserslautern
Email: jschmitt@informatik.uni-kl.de

***Abstract*—Queueing networks are typically subject to demultiplexing operations, whereby network nodes split flows into multiple sub-flows. The demultiplexing operation captures relevant network aspects such as packet loss or multi-path routing. In this paper we propose a novel approach to analyze queueing networks with demultiplexing. The basic idea is to represent a network node implementing a demultiplexing operation on an output flow as an equivalent system for which the corresponding input flow is logically demultiplexed according to the demultiplexing operation at the output. In this way, the service given to one of the demultiplexed sub-flows at the output can be expressed in terms of a leftover service curve, and consequently performance bounds can be derived using the network calculus methodology. Using numerical illustrations, we show that the obtained bounds improve upon existing bounds, especially in the case of the rather small sub-flows.**

***Index Terms*—Demultiplexing, network calculus, leftover service, scaling element.**

I. Introduction

A. Motivation

Over the last decade, the stochastic network calculus has established itself as a versatile alternative methodology to the classical queueing theory for the performance analysis of networks and distributed systems. The network calculus was pioneered by Cruz in the early 1990s [5] in a deterministic framework, and soon after by Chang [2] in a probabilistic framework. Subsequently, a significant number of researchers have contributed to both the deterministic and stochastic formulations of the network calculus (see the books of Chang [3], Le Boudec and Thiran [12], and Jiang and Liu [9]). The high modeling power of the network calculus has been transposed into several important applications for network engineering problems: traditionally in the Internet's Quality of Service proposals IntServ and DiffServ, and more recently in diverse environments such as wireless sensor networks [11], [14], switched Ethernets [16], Systems-on-Chip (SoC) [1], or even to speed-up simulations [10], to name a few.

From a methodological point of view, the main prospect of stochastic network calculus is that it can deal with problems that are fundamentally hard for queueing theory based on the fact that it works with (probabilistic) bounds rather than striving for exact solutions. A great challenge of any methodology for queueing analysis is to deal with queueing networks subject to flow transformations, which occur when the flows' data is altered inside the network. Flow transformations are in fact characteristic to many modern networked and distributed systems, e.g., a wireless sensor network processes the transported data, while delivering it to a sink node, for energy-efficiency purposes.

On an abstract level, one very obvious and yet highly important case of flow transformation results from the demultiplexing of flows inside the network. By demultiplexing we mean here the separation of a flow into multiple sub-flows, one of which is subject to the analysis (e.g., with respect to delay). Such an abstract demultiplexing allows to capture many concrete real-world effects like losing part of a data flow in, e.g., a wireless transmission, or distributing a data flow to a set of servers for load balancing with or without knowing the loads of the subsequent servers, or simply a (randomized) multi-path routing. Thus, the (stochastic) modeling of demultiplexing processes can be generally regarded as an important component in queueing network analysis, and in particular it opens up the modeling scope of network calculus widely.

While we have addressed demultiplexing in network calculus in previous work of ours [4], [15], we come up with a new way to solve this analytically hard problem in this paper. Our approach is based on finding an equivalent formulation (to [4]) for the demultiplexing as a leftover service curve computation problem. Interestingly, with respect to the achievable delay bounds the two methods perform quite differently and none completely dominates the other (see Section IV), though the new method shows a clear advantage in scenarios where the sub-flow of interest is rather small and only rarely outperformed by [4].

B. Related Work

So far, demultiplexing has not seen that much treatment in network calculus yet. However, interestingly, despite this observation, Cruz in his pioneering papers [5] originally introduced a demultiplexer as a member of his set of basic network calculus modeling elements. For the demultiplexer's operation, it is assumed that data units are "marked" with information about their output path. This assumption essentially means that demultiplexing decisions are statically configured (e.g., at the connection set-up in a virtual circuit setting) and cannot be made dynamically as, for example, for load-balancing purposes.

In [3], Chang introduces a network calculus modeling element called a router. The router has one data input and

output and one control input. The control input provides a functional relation between input and output data. This is very similar to a previous work of ours in which we introduced a wider framework of such scaling behavior (as it is called therein) in network calculus [7]. As we will also discuss in the course of this paper, such a router or scaling element may constitute the basis for the modeling of more flexible demultiplexing which is not statically decided. The key to real flexibility with respect to modeling dynamic demultiplexing is a stochastic bounding of the scaling behavior. This is only briefly touched on in [3] and not provided in [7].

In [15], we started to head into this direction by stochastically generalizing the scaling element from [7] and showing its utility in a load-balancing application. Though useful, the stochastic scaling from [15] has its limitations, most clearly documented by the fact that the scaling process cannot be an ergodic one. In [4], we therefore took a different approach to define a stochastic scaling element based on moment-generating functions which is much more versatile than the one from [15]. Now, in the paper at hand we continue the work started in [4] concentrating on demultiplexing as a flow transformation and provide a new analysis method based on leftover service computation. In fact, in many cases this allows to considerably improve the bounds over the ones calculated in [4], which are based on a direct analysis of the stochastic scaling.

II. A Novel Model for Flow Demultiplexing

A (network) flow is an abstraction of the data carried between two nodes in a network, the source and the destination. At each traversed node, the flow's data can be represented by a point process, i.e., a bi-dimensional stochastic process $(t_n, s_n)_{n\geq 1}$, where t_n's denote the arrival times and s_n's denote the sizes of the flow's instantaneous arrivals (or the number of data units), respectively.

In practice, flows are typically subject to *transformations* at some of the traversed nodes. We are particularly interested in the operation of *demultiplexing*, whereby a flow is split into *multiple* sub-flows. For instance, if at some node the flow is subject to random routing (say with two alternative paths), then the flow is split into two sub-flows, one for each path. As another example, if at some node the flow is subject to loss, then the flow is again split into two sub-flows: one consisting of the unaffected data, yet to be carried through, and another consisting of the lost data. If the original flow is represented by $(t_n, s_n)_{n\geq 1}$, then the sub-flows are represented by $(t_n, s_n^{(1)})_{n\geq 1}$ and $(t_n, s_n^{(2)})_{n\geq 1}$, respectively, with

$$s_n = s_n^{(1)} + s_n^{(2)} \ \forall n \geq 1 \ .$$

In this section, we introduce a *novel model* for the operation of demultiplexing a flow into two flows. This model is motivated by the need to analyze end-to-end performance metrics (e.g., delay) for *general classes* of arrival processes (i.e., of flows) subject to demultiplexing. We recall that, with the classical queueing networks theory, the demultiplexing analysis is typically possible for the class of Poisson arrivals and Bernoulli demultiplexing processes. Our goal herein is to extend these classes in a greater generality, e.g., by considering Markov modulated demultiplexing processes. Before presenting the novel demultiplexing model, let us briefly introduce some notations in the framework of the network calculus.

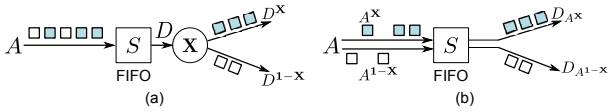


Fig. 1. Two equivalent systems for the demultiplexing operation. In (a), the output process $D(t)$ is demultiplexed according to a scaling process $\mathbf{X}$. In (b), the input process $A(t)$ is virtually demultiplexed according to the *same* $\mathbf{X}$. In both (a) and (b), the node S runs FIFO scheduling.

The time model is discrete with events (e.g., flows' arrivals) occurring at time instants $t = 0, 1, 2, \ldots$ In the framework of the network calculus, a flow $(t_n, s_n)_{n\geq 1}$ is described as a cumulative arrival process $A(s,t)$ counting the number of data units (packets) arrived in the time interval $(s,t]$. The univariate representation is $A(t) = A(0,t)$, and the instantaneous arrival process is $a(t) = A(t-1,t)$. Note the duality between the two representations of a flow, i.e., either as a point process or as a cumulative arrival process.

We next recall the *scaling element* model which was recently introduced by us in [4].

Definition 1: (Scaling Element) A scaling element transforms some arrival process into another. It consists of a random process $\mathbf{X} = (X_i)_{i\geq 1}$ taking non-negative integer values, an input (arrival) process $A(t)$, and an output (scaled) process $A^{\mathbf{X}}(t)$ for $\mathbf{X}$, which is a transformed version of the input process, and it is defined as

$$A^{\mathbf{X}}(t) = \sum_{i=1}^{A(t)} X_i \ \forall t \geq 0 \ .$$

The demultiplexing operation can be directly modelled with a scaling element. For brevity we focus on the particular case of two sub-flows demultiplexing, which can be modelled with a scaling process $\mathbf{X}$ with $X_i \in \{0,1\} \ \forall i \geq 1$. For an arrival flow $A(t)$ and a (demultiplexing) scaling process $\mathbf{X}$, the demultiplexed sub-flows are $A^{\mathbf{X}}(t)$ and $A^{\mathbf{1}-\mathbf{X}}(t)$, where $\mathbf{1} = (1, 1, \ldots)$. Note that, in general, the demultiplexing operation into n sub-flows can be modelled with multiple scaling processes $\mathbf{X}^1, \ldots, \mathbf{X}^n$, such that $\sum_{j=1}^{n} \mathbf{X}^j = \mathbf{1}$.

Let us now consider a work-conserving network node, denoted by S, with arrival and departure processes $A(t)$ and $D(t)$, respectively. We assume that a demultiplexing process $\mathbf{X}$, which is independent from the data flows, splits $D(t)$ into two sub-processes: $D^{\mathbf{X}}(t)$ and $D^{\mathbf{1}-\mathbf{X}}(t)$ (see Figure 1.(a)). Based on the demultiplexing process $\mathbf{X}$, we can virtually split the arrival process $A(t)$ into two sub-processes, $A^{\mathbf{X}}(t)$ and $A^{\mathbf{1}-\mathbf{X}}(t)$, and we denote the corresponding (virtual) output processes by $D_{A^{\mathbf{X}}}(t)$ and $D_{A^{\mathbf{1}-\mathbf{X}}}(t)$ (see Figure 1.(b)). The next lemma establishes the equivalence between the two systems from Figures 1.(a,b), and it is instrumental for our proposed approach to analyze queueing systems with general demultiplexing processes.

Lemma 1: (EQUIVALENT SYSTEMS FOR THE DEMULTIPLEXING OPERATION) Consider the systems (a) and (b) depicted in Figure 1 and described above. If the node is locally FIFO then the two systems are equivalent in the sense that for all $t \geq 0$

$$\begin{aligned} D^{\mathbf{X}}(t) &= D_{A^{\mathbf{X}}}(t) \text{ and} \\ D^{\mathbf{1}-\mathbf{X}}(t) &= D_{A^{\mathbf{1}-\mathbf{X}}}(t) \ . \end{aligned} \tag{1}$$

The proof is straightforward and relies on the assumption of *locally* FIFO scheduling, i.e., $A(t)$'s packets are served in the order of their arrivals.

PROOF. Let $t \geq 0$. The departure process $D(t)$ from system (a) can be represented as a sequence $d_1 d_2 \cdots d_{D(t)}$, where $d_i = i$, i.e., d_i's stand for the sequence numbers of $D(t)$'s packets. Then, because the scaling operation does not alter the order of the packets' arrivals, with some abuse of notation, the $\mathbf{X}$-scaled departure processes $D^{\mathbf{X}}(t)$ and $D^{\mathbf{1}-\mathbf{X}}(t)$ can be represented as the subsequences

$$\begin{aligned} D^{\mathbf{X}}(t) &= d_{x_1} d_{x_2} \cdots d_{x_i} \cdots d_{x_m} \text{ and} \\ D^{\mathbf{1}-\mathbf{X}}(t) &= d_{y_1} d_{y_2} \cdots d_{y_j} \cdots d_{y_n} \ , \end{aligned}$$

where $m, n \in [0, D(t)], m+n = D(t), x_i, y_j \in [0, D(t)], x_i < x_{i+1}, y_j < y_{j+1}$, and $x_i \neq y_j$, $\forall i \in [0, m]$ and $j \in [0, n]$. That means, those packets in $D(t)$ are mutually demultiplexed into $D^{\mathbf{X}}(t)$ and $D^{\mathbf{1}-\mathbf{X}}(t)$. In turn, because the node S runs FIFO scheduling, the arrival process $A(t)$ must be represented (again, with abuse of notation) as the following sequence

$$A(t) = d_1 d_2 \cdots d_{D(t)} d_{D(t)+1} \cdots d_{A(t)} \ ,$$

and furthermore after using the same scaling process $\mathbf{X}$ onto this sequence of arrivals we will get the virtual sub-processes $A^{\mathbf{X}}(t)$ and $A^{\mathbf{1}-\mathbf{X}}(t)$. So the starting part must stay the same as for the $\mathbf{X}$-scaled departures, and the remainder arrival packets are $\mathbf{X}$-scaled in the similar ways (with abuse of notation). Thus, we have

$$\begin{aligned} A^{\mathbf{X}}(t) &:= d_{x_1} d_{x_2} \cdots d_{x_i} \cdots d_{x_m} d_{x_{m+1}} \cdots d_{x_{m+a}} \cdots d_{x_{m+u}} \ , \\ A^{\mathbf{1}-\mathbf{X}}(t) &:= d_{y_1} d_{y_2} \cdots d_{y_j} \cdots d_{y_n} d_{y_{n+1}} \cdots d_{y_{n+b}} \cdots d_{y_{n+v}} \ , \end{aligned}$$

where $u, v \in [0, A(t) - D(t)]$, $u + v = A(t) - D(t)$, $x_{m+a}, y_{n+b} \in [0, A(t)]$, $x_{m+a} < x_{m+a+1}$, $y_{n+b} < y_{n+b+1}$, and $x_{m+a} \neq y_{n+b}$, $\forall a \in [0, u]$ and $b \in [0, v]$. So far, we only changed the notation and the departed sequences up to time t are still those two sequences $d_{x_1} d_{x_2} \cdots d_{x_i} \cdots d_{x_m}$ and $d_{y_1} d_{y_2} \cdots d_{y_j} \cdots d_{y_n}$. We can find a matched sequence of $d_{x_1} d_{x_2} \cdots d_{x_i} \cdots d_{x_m}$ in $A^{\mathbf{X}}(t)$, which proves that

$$D_{A^{\mathbf{X}}}(t) = d_{x_1} d_{x_2} \cdots d_{x_i} \cdots d_{x_m} \ ,$$

and thus $D^{\mathbf{X}}(t) = D_{A^{\mathbf{X}}}(t)$. Similarly, one can show that $D^{\mathbf{1}-\mathbf{X}}(t) = D_{A^{\mathbf{1}-\mathbf{X}}}(t)$, which completes the proof. □

Using the equivalence of systems (a) and (b) from Figure 1, we will next compute statistical end-to-end delay bounds in queueing systems subject to the demultiplexing operation. The basic idea is to use the representation from system (b) in order to construct the so-called *leftover service curves* for the demultiplexed processes. Using these service curves, the desired performance bounds follow by applying conventional techniques from stochastic network calculus.

III. STATISTICAL END-TO-END DELAY BOUNDS

In this section, we compute end-to-end delay bounds in a network with flow demultiplexing. For illustrative purposes, we focus on the single and two nodes cases, and later comment on the generalization to an arbitrary number of nodes. For numerical comparisons, we also reproduce an existing parallel result from [4].

First we briefly introduce the service curve concept and then give the leftover service curve for the flow of interest, i.e., $A^{\mathbf{X}}$ from Figure 1.(b). Service curves (in [3] called dynamic servers) provide lower bounds on the service received by an arrival flow at a network node, and are formally defined as below.

Definition 2: (SERVICE CURVE) Assume $A(t)$ and $D(t)$ are the arrival respectively departure process of a node. A bivariate random process $S(s,t)$ for $0 \leq s \leq t$ is called a service curve provided by this node, if for all $t \geq 0$ it holds that

$$D(t) \geq A \otimes S(t),$$

where '$\otimes$' denotes the $(min, +)$-convolution defined as $A \otimes S(t) = \inf_{0 \leq s \leq t} \{A(s) + S(s,t)\}$.

Note, through this work we use "service curve" instead of "service curve process" for the simplicity of naming. Further, if the inequality is replaced by equality then the service curve is called *exact*. The service curve of two concatenated nodes with service curves S_1 and S_2 is $S_1 \otimes S_2$ [3], which is achieved using the associativity property of the $(min, +)$-convolution. The leftover service curve under a FIFO scheduling model ([6], [12]) is derived using the following proposition.

Proposition 1: Consider two flows with arrival processes $A_1(s,t)$ and $A_2(s,t)$ which are FIFO-scheduled at a service curve $S(s,t)$. Assume that $S(s,t)$ is nonnegative, increasing in t, and $S(s,s) = 0$. Flow 1 sees a service curve

$$S_1(s,t) = [S(s,t) - A_2(s,t-x)]_+ \, 1_{\{t-x>s\}},$$

where parameter $x \geq 0$, $[y]_+ := max(0, y)$ and $1_{cond} = 1$ if $cond = true$, $1_{cond} = 0$ otherwise.

The proof follows directly using the bivariate arrival processes and service curves instead of arrival curve and service curve functions (see the proof of Proposition 6.2.1 in [12]). Note, $S(s,t)$ is not necessarily strict. We also point out that the leftover service curve here is a bivariate process presentation of the service, which is different from a probabilistic sample path bound presentation as given in [13]. Knowing additional information of the leftover service curve, the transformation between both presentations might be possible. But such discuss is not the scope of this paper.

So far, we know that the leftover service curve for $A^{\mathbf{X}}$, obtained in the particular case of FIFO scheduling, is

$$S_{LO}(s,t) = \left[S(s,t) - A^{\mathbf{1}-\mathbf{X}}(s,t-x)\right]_+ 1_{\{t-x>s\}} \ ,$$

where $x \geq 0$ is an optimization parameter.

A. Single Node: Main Idea

Referring to Figure 1.(a), we are interested in the virtual delay of the (sub-)flow of interest $D^{\mathbf{X}}$. The corresponding arrival process is $A^{\mathbf{X}}$ (see the interpretation from system (b)), and the leftover service curve is $S_{LO}(s,t)$ shown above. Then, a probabilistic delay bound can be obtained as below

$$\begin{aligned}
&\mathbb{P}(W(t) \geq d) \\
&\quad = \mathbb{P}\left(A^{\mathbf{X}}(t-d) \geq D_{A^{\mathbf{X}}}(t)\right) \\
&\quad = \mathbb{P}\left(A^{\mathbf{X}}(t-d) \geq D^{\mathbf{X}}(t)\right) \\
&\quad \leq \mathbb{P}\Bigg(\sup_{0\leq s<t-d}\Big\{A^{\mathbf{X}}(t-d) - A^{\mathbf{X}}(s) - \\
&\qquad \left[S(s,t) - A^{\mathbf{1-X}}(s,t-x)\right]_+ 1_{\{t-x>s\}}\Big\} \geq 0\Bigg),
\end{aligned}$$

where in the third line we used Lemma 1, in the fourth line we used the definition of a service curve and moved the infimum to the left side of the inequality. The derivation can be continued depending on two cases for x:

(i) $x > d$, i.e., $t - x < t - d$:

$$\begin{aligned}
&= \mathbb{P}\Bigg(\max\Bigg(\sup_{0\leq s<t-x}\Big\{A^{\mathbf{X}}(s,t-d) - \\
&\qquad \left[S(s,t) - A^{\mathbf{1-X}}(s,t-x)\right]_+\Big\}, \\
&\qquad \sup_{t-x\leq s<t-d}\left\{A^{\mathbf{X}}(s,t-d)\right\}\Bigg) \geq 0\Bigg) \\
&= \mathbb{P}\Bigg(\max\Bigg(\sup_{0\leq s<t-x}\Big\{A^{\mathbf{X}}(s,t-d) - \\
&\qquad \left[S(s,t) - A^{\mathbf{1-X}}(s,t-x)\right]_+\Big\}, \\
&\qquad A^{\mathbf{X}}(t-x,t-d)\Bigg) \geq 0\Bigg) \\
&= 1
\end{aligned}$$

(ii) $0 \leq x \leq d$

$$\begin{aligned}
&= \mathbb{P}\Bigg(\sup_{0\leq s<t-d}\Big\{A^{\mathbf{X}}(s,t-d) - \\
&\qquad \left[S(s,t) - A^{\mathbf{1-X}}(s,t-x)\right]_+\Big\} \geq 0\Bigg) \\
&\leq \mathbb{P}\Bigg(\sup_{0\leq s<t-d}\Big\{A(s,t-d) - \\
&\qquad S(s,t) + A^{\mathbf{1-X}}(t-d,t-x)\Big\} \geq 0\Bigg)
\end{aligned}$$

The optimal value is for $x = d$, and the last probability becomes

$$= \mathbb{P}\left(\sup_{0\leq s<t-d}\left\{A(s,t-d) - S(s,t)\right\} \geq 0\right).$$

Interestingly, the same bound can be obtained for the delay of the aggregate departure flow D. In other words, the per-flow delay bound is equal to the aggregate delay bound; the equality between the per-flow and aggregate delays is known to hold, on average, in the case of Poisson arrivals and Bernoulli demultiplexing, as a consequence of the PASTA property.

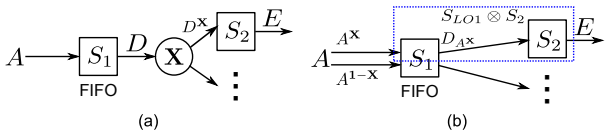


Fig. 2. A network with two nodes, S_1 and S_2, and a demultiplexer element in between.

B. Two Nodes

In this section, we derive statistical end-to-end delay bounds in a network consisting of two service nodes and one scaling element implementing a demultiplexing operation in between (see Figure 2). We shall use the idea presented above, of first computing the leftover service curve for one demultiplexed flow of interest.

Before carrying out the derivations, it is useful to compute the moment generating functions (MGFs) as well as MGF bounds of the scaled processes in terms of the MGFs of the arrival processes and scaling processes. First we recall an example of a scaling process from [4], called Markov-modulated scaling process (MMSP). Let the scaling process $\mathbf{X} = (X_i)_{i\geq 1}$ be modulated by a discrete and homogeneous Markov process $S(i)$ with states $1, 2, ..., M$ and transition probabilities $\lambda_{i,j}$ for all $1 \leq i, j \leq M$, as in Figure 3. Every state $i, 1 \leq i \leq M$, has an *i.i.d.* random process $L_i(n)_{n\geq 1}$. Then the scaling process is defined as $X_i = L_{S(i)}(i)$. In the cases of a single state, or two states with $\lambda_{1,2} + \lambda_{2,1} = 1$, the scaling process $\mathbf{X}$ is *i.i.d.*

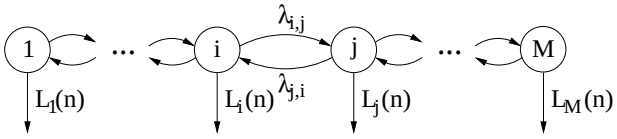


Fig. 3. A Markov chain $S(i)$ with M states and transition probabilities $\lambda_{i,j}$, modulating the scaling process $\mathbf{X}$ as $X_i = L_{S(i)}(i)$.

Now denoting $M_X(\theta) := E\left[e^{\theta X}\right]$ for some r.v. X and $\theta > 0$ we have the following lemma [4].

Lemma 2: (MOMENT GENERATING FUNCTION OF SCALED PROCESSES) Let an arrival process $A(t)$ and an MMSP $\mathbf{X} = (X_i)_{i\geq 1}$ defined as above. Then we have the MGFs for some $\theta > 0$:

1) (General case) If the matrix $\lambda = (\lambda_{i,j})_{i,j}$ is irreducible and aperiodic then

$$M_{A^{\mathbf{X}}(t)}(\theta) \leq M_{A(t)}\left(\log \mathrm{sp}\left(\phi(\theta)\lambda\right)\right), \qquad (2)$$

where $\phi(\theta) := \mathrm{diag}\left(M_{L_1(1)}(\theta), \ldots, M_{L_M(1)}(\theta)\right)$ and $\mathrm{sp}\left(\phi(\theta)\lambda\right)$ denotes the spectral radius of $\phi(\theta)\lambda$.

2) (*I.i.d.* case) If X_i's are *i.i.d.* then

$$M_{A^{\mathbf{X}}(t)}(\theta) = M_{A(t)}\left(\log M_X(\theta)\right). \qquad (3)$$

We now consider the network scenario of interest from Figure 2(a), of which the first node and the demultiplexing element can be equivalently transformed as detailed in the previous section. The result is shown in Figure 2(b). The next theorem provides the end-to-end delay bounds under different assumptions regarding the increments of the arrival process $A(t)$.

Theorem 1: (STATISTICAL END-TO-END DELAY BOUNDS IN A TWO NODES NETWORK WITH DEMULTIPLEXING) Consider the network scenario from Figure 2(a). A stationary arrival process $A(t)$ crosses two FIFO nodes offering the stationary service curves $S_1(s,t)$ and $S_2(s,t)$. Assume the MGF bounds of arrivals and services: $M_{A(s,t)}(\theta) \leq e^{\theta r(\theta)(t-s)}$ and $M_{S_k(s,t)}(-\theta) \leq e^{-\theta C_k(t-s)}$, where $k = 1, 2$ and for some free parameter $\theta > 0$. The demultiplexing process $\mathbf{X}$ is such that X_i's are either 0 or 1, and denote $\delta(\theta) = \frac{1}{\theta}\log M_{\mathbf{X}}(\theta)$ and $\bar{\delta}(\theta) = \frac{1}{\theta}\log M_{\mathbf{1-X}}(\theta)$. All the processes are assumed to be statistically independent, and we focus on the end-to-end sub-flow with departure process $E(t)$.

(1) If $A(t)$ has statistically independent increments then under some stability conditions, to be explicitly given in the proof, we have the following delay bounds for all $d \geq 0$

$$\mathbb{P}(W(t) \geq d) \leq K_1 e^{-\theta C_2(d-x)} + K_2 e^{-\theta C_1 d} e^{\theta\bar{\delta}(\theta) r(\theta\bar{\delta}(\theta))(d-x)} + K_3 e^{-\theta C_1 d} e^{\theta(C_1-C_2)(d-x)} ,$$

where K_1, K_2 and K_3 are constants to be given in the proof. x is an optimization parameter introduced by FIFO leftover service.

(2) If the increments of $A(t)$ are not necessarily independent then under some stability conditions, to be explicitly given in the proof, we have the following delay bounds for all $d \geq 0$

$$\mathbb{P}(W(t) \geq d) \leq K e^{-\theta C_2 d} ,$$

where the constant K will be given in the proof.

Note, since θ is a free parameter for each MGF function, we use the same θ for the simplicity of denotation. A direct representation of the theorem using different θ's follows by applying MGF bound respectively with different private θ's.

PROOF. As discussed in Section II, we can equivalently transform the system (a) from Figure 2 into the system (b) from the same figure. Then we have for the end-to-end sub-flow of interest with input $A^{\mathbf{X}}(t)$ and output $E(t)$:

$$\begin{aligned}
&\mathbb{P}(W(t) \geq d) \\
&= \mathbb{P}\left(A^{\mathbf{X}}(t-d) \geq E(t)\right) \\
&\leq \mathbb{P}\Big(\sup_{0\leq s<t-d} \left\{A^{\mathbf{X}}(t-d) - A^{\mathbf{X}}(s) - S_{LO} \otimes S_2(s,t)\right\} \geq 0\Big) \\
&\leq \mathbb{P}\Big(\sup_{0\leq s<t-d} \sup_{s\leq u\leq t} \left\{A^{\mathbf{X}}(t-d) - A^{\mathbf{X}}(s) - S_{LO}(s,u) - S_2(u,t)\right\} \geq 0\Big) \\
&= \mathbb{P}\Big(\sup_{0\leq s<t-d} \sup_{s\leq u\leq t} \left\{A^{\mathbf{X}}(t-d) - A^{\mathbf{X}}(s) - \left[S_1(s,u) - A^{\mathbf{1-X}}(s,u-x)\right]_+ 1_{\{u-x>s\}} - S_2(u,t)\right\} \geq 0\Big) \qquad (4)
\end{aligned}$$

In the second line we used the definition of the virtual delay process for the arrival process $A^{\mathbf{X}}(t)$ and the departure process $E(t)$, and also the equivalence $W(t) \geq d \Leftrightarrow A^{\mathbf{X}}(t-d) \geq E(t)$. In the third line we used the definition and the convolution property of service curves and then expanded the convolution and the expression of $S_{LO}(s,u)$ in the rest lines.

The rest of the proof follows the derivations from Subsection III-A, depending on the value of the parameter x:

(i) $x > d$, or, $u - x < t - d$. Let us consider only a part of the domain for the supremum as below

$$\begin{aligned}
&\sup_{t-x\leq s<t-d} \sup_{s\leq u\leq t} \left\{A^{\mathbf{X}}(t-d) - A^{\mathbf{X}}(s) - \left[S_1(s,u) - A^{\mathbf{1-X}}(s,u-x)\right]_+ 1_{\{u-x>s\}} - S_2(u,t)\right\} \\
&= \sup_{t-x\leq s<t-d} \sup_{s\leq u\leq t} \left\{A^{\mathbf{X}}(t-d) - A^{\mathbf{X}}(s) - S_2(u,t)\right\} \\
&\geq 0 ,
\end{aligned}$$

and thus the optimal value of x does not fall in this interval.

(ii) $0 \leq x \leq d$, and we continue Eq. (4) as follows:

$$\begin{aligned}
&= \mathbb{P}\Bigg(\sup_{0\leq s<t-d} \Bigg\{ max\Bigg(\sup_{s\leq u\leq s+x} \Big\{A^{\mathbf{X}}(s,t-d) - \left[S_1(s,u) - A^{\mathbf{1-X}}(s,u-x)\right]_+ 1_{\{u-x>s\}} - S_2(u,t)\Big\}, \\
&\qquad \sup_{s+x<u\leq t} \Big\{A^{\mathbf{X}}(s,t-d) - \left[S_1(s,u) - A^{\mathbf{1-X}}(s,u-x)\right]_+ 1_{\{u-x>s\}} - S_2(u,t)\Big\}\Bigg)\Bigg\} \geq 0\Bigg) \\
&\leq \sum_{0\leq s<t-d} \mathbb{P}\left(A^{\mathbf{X}}(s,t-d) - S_2(s+x,t) \geq 0\right) + \sum_{0\leq s<t-d} \sum_{s+x<u\leq t} \mathbb{P}\Big(A^{\mathbf{X}}(s,t-d) - \left[S_1(s,u) - A^{\mathbf{1-X}}(s,u-x)\right]_+ - S_2(u,t) \geq 0\Big) \\
&\leq \sum_{0\leq s<t-d} \mathbb{P}\left(A^{\mathbf{X}}(s,t-d) - S_2(s+x,t) \geq 0\right) + \sum_{0\leq s<t-d} \sum_{s+x<u\leq t} \mathbb{P}\Big(A^{\mathbf{X}}(s,t-d) + A^{\mathbf{1-X}}(s,u-x) - S_1(s,u) - S_2(u,t) \geq 0\Big) \qquad (5)
\end{aligned}$$

The first line is just a separation of the interval of u in order to eliminate the indicator function "1". Besides that we applied the union bound in the second line. In the third line, we eliminated "$[\cdot]_+$". Next we use the properties of the arrival process $A(t)$ in order to prove (1) and (2), respectively.

(1) $A(t)$ *has independent increments*. Since the two intervals $[s, t-d]$ and $[s, u-x]$ are overlapping, we can merge the overlapping parts of $A^{\mathbf{X}}(s,t-d)$ and $A^{\mathbf{1-X}}(s,u-x)$. Yet we do not know which value, $t-d$ or $u-x$, is larger. So from

Eq. (5) we separate the interval of u as below

$$
\begin{aligned}
&\leq \sum_{0\leq s<t-d} \mathbb{P}\left(A^{\mathbf{X}}(s,t-d) - S_2(s+x,t) \geq 0\right) + \\
&\quad \sum_{0\leq s<t-d} \sum_{s+x<u\leq t-d+x} \mathbb{P}\Big(A^{\mathbf{X}}(u-x,t-d) + \\
&\qquad\qquad A(s,u-x) - S_1(s,u) - S_2(u,t) \geq 0\Big) + \\
&\quad \sum_{0\leq s<t-d} \sum_{t-d+x<u\leq t} \mathbb{P}\Big(A^{\mathbf{1}-\mathbf{X}}(t-d,u-x) + \\
&\qquad\qquad A(s,t-d) - S_1(s,u) - S_2(u,t) \geq 0\Big) \\
&\leq \sum_{0\leq s<t-d} e^{-\theta C_2(t-s-x)} e^{\theta\delta(\theta) r(\theta\delta(\theta))(t-d-s)} + \\
&\quad \sum_{0\leq s<t-d} \sum_{s+x<u\leq t-d+x} \\
&\quad e^{-\theta C_1(u-s)} e^{-\theta C_2(t-u)} e^{\theta r(\theta)(u-x-s)} e^{\theta\delta(\theta) r(\theta\delta(\theta))(t-d-u+x)} \\
&\quad + \sum_{0\leq s<t-d} \sum_{t-d+x<u\leq t} \\
&\quad e^{-\theta C_1(u-s)} e^{-\theta C_2(t-u)} e^{\theta r(\theta)(t-d-s)} e^{\theta\bar{\delta}(\theta) r\left(\theta\bar{\delta}(\theta)\right)(u-x-t+d)} \\
&\leq \frac{e^{-\theta C_2(d-x)}}{e^{\theta(C_2-\delta(\theta)r(\theta\delta(\theta)))}-1} + \\
&\quad \frac{\frac{1}{e^{\theta(C_1-r(\theta))}-1} - \frac{1}{e^{\theta(C_2-\delta(\theta)r(\theta\delta(\theta)))}-1}}{1-e^{\theta(C_1-C_2-r(\theta)+\delta(\theta)r(\theta\delta(\theta)))}} e^{-\theta C_1 d} e^{\theta(C_1-C_2)(d-x)} \\
&\quad + \frac{e^{-\theta C_1 d}\left(e^{\theta\bar{\delta}(\theta) r\left(\theta\bar{\delta}(\theta)\right)(d-x)} - e^{\theta(C_1-C_2)(d-x)}\right)}{\left(1-e^{\theta\left(C_1-C_2-\bar{\delta}(\theta)r\left(\theta\bar{\delta}(\theta)\right)\right)}\right)\left(\theta C_1 - \theta r(\theta)\right)} \ . \qquad (6)
\end{aligned}
$$

In the second line we used the Chernoff bound for some $\theta > 0$ together with Lemma 2 for *i.i.d.* $X_i's$ (Eq. (3)). Because A has independent increments and is independent of $\mathbf{X}$, it follows that $A^{\mathbf{X}}(u-x,t-d)$ is independent of $A(s,u-x)$ and $A^{\mathbf{1}-\mathbf{X}}(t-d,u-x)$ is independent of $A(s,t-d)$. In the case when $\mathbf{X}$ is MMSP, we have that $\delta(\theta) = \frac{1}{\theta}\log \mathrm{sp}\left(\phi_{\mathbf{X}}(\theta)\lambda_{\mathbf{X}}\right)$ and $\bar{\delta}(\theta) = \log \mathrm{sp}\left(\phi_{\mathbf{1}-\mathbf{X}}(\theta)\lambda_{\mathbf{1}-\mathbf{X}}\right)$, according to Eq. (2). In the third line, imposing the following two stability conditions

$$\theta(C_1 - r(\theta)) > 0 \text{ and } \theta(C_2 - \delta(\theta)r(\theta\delta(\theta)) > 0$$

for the first term by letting $t\to\infty$, we get an infinite geometric series; for the second term we computed the sum of geometric series over u and got the exact sum of infinite geometric series by letting $t\to\infty$; the computation for the third term is similar. By letting

$$
\begin{aligned}
K_1 &= \frac{1}{e^{\theta(C_2-\delta(\theta)r(\theta\delta(\theta)))}-1} \\
K_2 &= \frac{1}{\left(e^{\theta(C_1-r(\theta))}-1\right)\left(1-e^{\theta\left(C_1-C_2-\bar{\delta}(\theta)r\left(\theta\bar{\delta}(\theta)\right)\right)}\right)} \\
K_3 &= K_1\cdot K_2\cdot\left(1-e^{\theta\left(C_1-\delta(\theta)r(\theta\delta(\theta))-\bar{\delta}(\theta)r\left(\theta\bar{\delta}(\theta)\right)\right)}\right)
\end{aligned}
$$

in Eq. (6), we get

$$
\begin{aligned}
&\mathbb{P}(W(t)\geq d) \leq K_1 e^{-\theta C_2(d-x)} + \\
&\quad K_2 e^{-\theta C_1 d} e^{\theta\bar{\delta}(\theta) r(\theta\bar{\delta}(\theta))(d-x)} + K_3 e^{-\theta C_1 d} e^{\theta(C_1-C_2)(d-x)} \ ,
\end{aligned}
$$

and the proof of (1) is complete.

(2) The increments of $A(t)$ *are not necessarily independent*. We can continue Eq. (5) as follows

$$
\begin{aligned}
&\leq \sum_{0\leq s<t-d} e^{-\theta C_2(t-s-x)} e^{\theta\delta(\theta) r(\theta\delta(\theta))(t-d-s)} + \\
&\quad \sum_{0\leq s<t-d} \sum_{s+x<u\leq t} e^{-\theta C_1(u-s)} e^{-\theta C_2(t-u)} \cdot \\
&\qquad\qquad E\left[e^{\theta\left(A^{\mathbf{X}}(s,t-d)+A^{\mathbf{1}-\mathbf{X}}(s,u-x)\right)}\right],
\end{aligned}
$$

after using the Chernoff bound. To compute the bound of the expectation we use Hölder's inequality because of the possible dependence between $A^{\mathbf{X}}$ and $A^{\mathbf{1}-\mathbf{X}}$. Let $g,\ h \geq 1$ such that $\frac{1}{g}+\frac{1}{h}=1$. For $t\to\infty$, we continue the above inequality with

$$
\begin{aligned}
&\leq \frac{e^{-\theta C_2(d-x)}}{\theta(C_2-\delta(\theta)r(\theta\delta(\theta)))} + \\
&\qquad \frac{e^{-\theta C_1 d} e^{\theta(C_1-C_2)(d-x)}}{\theta(C_1-C_2-\bar{\delta}(h\theta)r(h\theta\bar{\delta}(h\theta)))\cdot\theta(C_2-\delta(g\theta)r(g\theta\delta(g\theta)))} .
\end{aligned}
$$

Here, without losing tightness, we used an upper bound on the infinite geometric series for simplicity. Let us now assume the stability conditions

$$
\begin{aligned}
&\theta\left(C_2-\delta(\theta)r\left(\theta\delta(\theta)\right)\right) > 0 \\
&\theta\left(C_2-\delta(g\theta)r\left(g\theta\delta(g\theta)\right)\right) > 0 \\
&\theta\left(C_1-C_2-\bar{\delta}(h\theta)r\left(h\theta\bar{\delta}(h\theta)\right)\right) > 0 \ .
\end{aligned}
$$

To continue the previous derivation, we use the next convex optimization result

$$\inf_{x>0}\left\{\alpha e^{-\beta x} + e^{\gamma x}\right\} = \left(\frac{\alpha\beta}{\gamma}\right)^{\frac{\gamma}{\beta+\gamma}} \frac{\beta+\gamma}{\beta} \ .$$

We finally obtain that

$$\mathbb{P}(W(t)\geq d) \leq K e^{-\theta C_2 d},$$

where

$$
\begin{aligned}
K &= \frac{C_1}{C_2}\left(\frac{C_1}{(C_1-C_2)\theta(C_2-\delta(\theta)r(\theta\delta(\theta)))}\right)^{1-\frac{C_2}{C_1}} \\
&\quad \cdot\Big(\theta(C_1-C_2-\bar{\delta}(h\theta)r(h\theta\bar{\delta}(h\theta))) \\
&\qquad \cdot\theta(C_2-\delta(g\theta)r(g\theta\delta(g\theta)))\Big)^{-\frac{C_2}{C_1}} .
\end{aligned}
$$

The proof is now complete. □

C. Statistical End-to-End Delay Bounds by Commuting Scaling Element and Service

Here we briefly reproduce the computation of the end-to-end delay bound in a demultiplexing network by commuting scaling and service elements, as proposed in [4]. Starting from Figure 4.(a), we can find another system, as shown in Figure 4.(b), where $A(t)$ goes first through $\mathbf{X}$ and then through the exact service curve $T(s,t) := \sum_{i=A(s)+1}^{A(s)+S(s,t)} X_i$, such that the departures for both systems in (a) and (b) satisfy $E(t) \leq D^{\mathbf{X}}(t)$ for all $t\geq 0$; a tacit assumption is that A, $\mathbf{X}$, and S are independent.

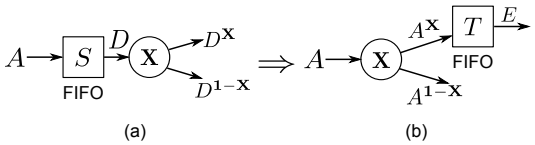


Fig. 4. Commuting service and scaling elements.

The following theorem (also from [4]) provides the statistical end-to-end delay bound.

Theorem 2: (END-TO-END DELAYS IN A FLOW TRANSFORMATION NETWORK) Consider the network scenario from Figure 4.(a) where a stationary arrival process $A(t)$ crosses a series of alternate stationary and (mutually) independent service and scaling elements denoted by $S_1, S_2, \ldots, S_n$ and *i.i.d.* $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{n-1}$, respectively. Assume the MGF bounds $M_{A(s,t)}(\theta) \leq e^{\theta r(\theta)(t-s)}$ and $M_{S_k(t)}(-\theta) \leq e^{-\theta C_k t}$, for $k = 1, \ldots, n$, and some $\theta > 0$. Under a stability condition

$$a_{n-1} r(a_{n-1}) + \log b < 0,$$

where

$$a_0 = \theta,\ a_{k+1} = \log E\left[e^{a_k X_{k,1}}\right],\ b = \sup_{k=0,\ldots,n-1} e^{-a_k C_{n-k}},$$

we have the following end-to-end steady-state delay bounds for all $d \geq 0$

$$\mathbb{P}(W > d) \leq K^n b^d,$$

where the constant $K = \frac{\left(1+\frac{d}{n}\right)^{1+\frac{d}{n}}}{\left(\frac{d}{n}\right)^{\frac{d}{n}}}$.

IV. COMPARISON

So far, we presented the dual analytical methods to calculate statistical end-to-end delay bounds in a demultiplexing scenario. Next, we compare them quantitatively. Before we can do so, we need to provide some results regarding MGF bounds, because for both methods, we assume to have the MGF bounds for the arrivals and the service, and also need to use the MGF bounds of the scaling processes and the scaled processes to compute the delay bounds.

A. MGF Bounds

In this section, we present necessary prerequisites regarding MGF bounds for three examples of arrival processes $A(t)$: Bernoulli(-modulated) arrivals with rate R together with Bernoulli parameter p_0, Poisson arrivals with rate λ and MMOO arrivals with peak rate P. We also present results for the MGF bounds of scaled processes with two examples of scaling processes $\mathbf{X}$: Bernoulli scaling and MMOO scaling.

For Bernoulli arrivals with rate R, we consider arrivals with constant rate R passing through a Bernoulli scaling process with Bernoulli parameter p_0. We know that the MGF of a Bernoulli r.v. X_B with parameter $p_B \in [0,1]$ is $M_{X_B}(\theta) = 1 - p_B + p_B e^{\theta}$. So Lemma 2 yields the MGF bound for the *i.i.d.* Bernoulli arrivals with rate R and probability $p_0 \in [0,1]$ as

$$\begin{aligned} M_{A(s,t)}(\theta) &= M_{R(t-s)}\left(\log\left(1 - p_0 + p_0 e^{\theta}\right)\right) \\ &= E\left[e^{\log\left(1-p_0+p_0e^{\theta}\right)R(t-s)}\right] \\ &= \left(1 - p_0 + p_0 e^{\theta}\right)^{R(t-s)}. \end{aligned}$$

We rewrite this as

$$M_{A(s,t)}(\theta) = e^{\theta \cdot \frac{1}{\theta}\log\left(1-p_0+p_0e^{\theta}\right)R\cdot(t-s)} = e^{\theta \cdot r(\theta)\cdot(t-s)},$$

where we denote $\frac{1}{\theta}\log\left(1 - p_0 + p_0 e^{\theta}\right) R$ as $r(\theta)$ according to the form of MGF bound for the arrivals in Theorem 1. Next, we derive $\delta(\theta)$ for the scaling process $\mathbf{X}$ as a Bernoulli process with parameter $p \in [0,1]$ and $\bar{\delta}(\theta)$ for its conjugate scaling process $\mathbf{1} - \mathbf{X}$. Again, knowing the MGF of the *i.i.d.* Bernoulli process $\mathbf{X}$ we have

$$\begin{aligned} \delta(\theta) &= \frac{1}{\theta}\log M_X(\theta) = \frac{1}{\theta}\log\left(1 - p + pe^{\theta}\right) \\ \bar{\delta}(\theta) &= \frac{1}{\theta}\log M_{1-X}(\theta) = \frac{1}{\theta}\log\left(p + (1-p)e^{\theta}\right). \end{aligned}$$

Now we have $r(\theta), \delta(\theta)$ as well as $\bar{\delta}(\theta)$. As all other parameters in Theorem 1 are already given, we can compute the delay bounds for the two nodes network. On the other hand, noting that a_1 is $\theta\delta(\theta)$, we can use Theorem 2 to compute alternative delay bounds.

In the following, we consider the following scenario: Poisson process as arrivals and Markov-Modulated On-Off (MMOO) process as scaling. The MMOO scaling processes are presented in Figure 5. μ_1 and μ_2 are transition probabili-

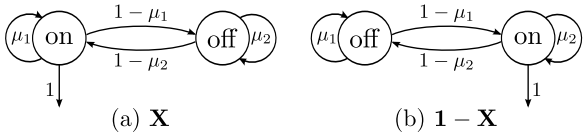


Fig. 5. Markov-Modulated On-Off (MMOO) scaling process.

ties. The processes let data pass along this sub-flow while in state 'on' (thus the rate is 1) and block while in state 'off'. If the scaling processes are MMOO, we derive $\delta(\theta)$ and $\bar{\delta}(\theta)$ from Theorem 1 as

$$\begin{aligned} \delta(\theta) &= \frac{1}{\theta}\log \mathrm{sp}\left(\phi_{\mathbf{X}}(\theta)\lambda_{\mathbf{X}}\right) \\ &= \frac{1}{\theta}\log \frac{\mu_2 + \mu_1 e^{\theta} + \sqrt{\left(\mu_2 + \mu_1 e^{\theta}\right)^2 - 4(\mu_1 + \mu_2 - 1)e^{\theta}}}{2} \\ \bar{\delta}(\theta) &= \frac{1}{\theta}\log \mathrm{sp}\left(\phi_{\mathbf{1}-\mathbf{X}}(\theta)\lambda_{\mathbf{1}-\mathbf{X}}\right) \\ &= \frac{1}{\theta}\log \frac{\mu_1 + \mu_2 e^{\theta} + \sqrt{\left(\mu_1 + \mu_2 e^{\theta}\right)^2 - 4(\mu_2 + \mu_1 - 1)e^{\theta}}}{2}, \end{aligned}$$

where μ_1 and μ_2 have the following relation for Figure 5(a), given the average 'on' probability p for scaling process $\mathbf{X}$,

$$p = \frac{1 - \mu_2}{1 - \mu_1 + 1 - \mu_2}.$$

Then, of course, the scaling process $\mathbf{1} - \mathbf{X}$ has the average 'on' probability $1 - p$. Hence, knowing p, given μ_1 we can

compute μ_2, and vice versa. Further knowing the MGF bound of the Poisson arrival process, which is

$$M_{A(t)}(\theta) = e^{\theta r(\theta) t}, \text{ where } r(\theta) = \frac{1}{\theta}\lambda(e^{\theta} - 1),$$

we can again use Theorem 1 to compute the delay bounds for the two nodes network.

Similarly, considering an MMOO process as arrival process, we use P as the peak rate instead of 1 and λ_1, λ_2 as transition probabilities instead of μ_1, μ_2. We compute the MGF bound of the arrivals as

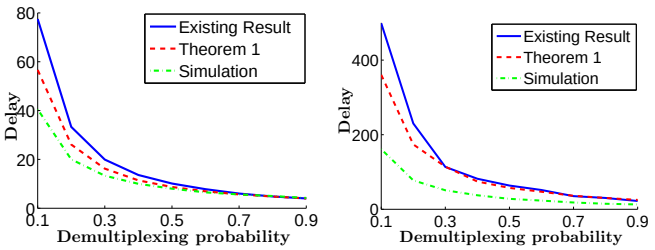
$$E\left[e^{\theta A(t)}\right] \leq e^{\theta r(\theta) t},$$

where $r(\theta) = \frac{1}{\theta}\log\frac{\lambda_1 e^{\theta P} + \lambda_2 + \sqrt{(\lambda_1 e^{\theta P} + \lambda_2)^2 - 4(\lambda_1 + \lambda_2 - 1)e^{\theta P}}}{2}$. For other combinations of different arrival and scaling cases, we have similar computations.

B. Delay Bounds

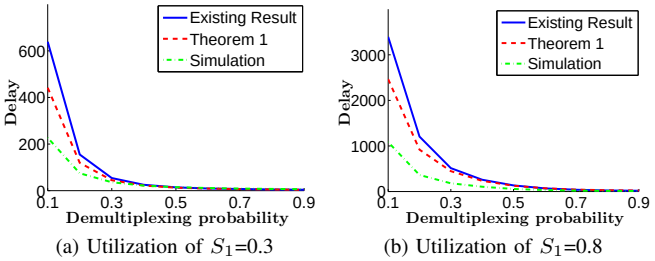
Next, we numerically compare the statistical delay bounds using both methods in the above mentioned three examples of arrivals: Bernoulli with probability p_0 and rate R, Poisson with rate λ, and MMOO with peak rate P and transition probabilities λ_1, λ_2. We consider the network senario with two nodes shown in Figure 2(a). For the scaling process we consider for two examples: Bernoulli process with probability p and MMOO process with average probability p and transition probabilities μ_1, μ_2. Here, we only show the results for MMOO arrivals with MMOO scaling, so we have five combinations and for all the combinations we compare the statistical delay bounds of the two nodes network using the dual analytical methods presented in Section III . Moreover, for the five combinations we compare the delay bounds in different utilizations for the first node - low and high utilization (30% and 80%). As a reference, we also provide the results of discrete-event simulations. We use OMNeT++ [17] version 4.2 to simulate the queueing network. To compute the empirical 10^{-3}-quantiles, we observe 10^6 packets and use the P^2 algorithm [8] for calculating the quantiles without storing so many observations. Random number generating during the simulations is done using the class cLCG32.

In the following, we give the numerical settings. First, the service rate of the first node C_1 is normalized to one packet per time unit. Correspondingly, we set the utilization of the first node as either 0.3 or 0.8, i.e., the average rates of arrivals are set as 0.3 and 0.8, respectively. We let $R = 2$, thus in the case of Bernoulli arrivals p_0 equals to 0.15 and 0.4, respectively. For the case of MMOO arrivals, we also set $P = 2$ and $\lambda_1 = 0.75$. With the average passing probability p of the scaling process $\mathbf{X}$ the capacity at the second node C_2 is set to $C_2 = p \cdot C_1$. These settings guarantee the stability conditions. We vary p from 0.1 to 0.9 in steps of 0.1. In the case of MMOO scaling, we set μ_1 as 0.75. We plot the ε-quantiles (in time units) of the end-to-end delay bounds with $\varepsilon = 10^{-3}$. In all figures, we can perceive, that the delays decrease for higher values of the pass probabilities p. This behavior is due to setting $C_2 = p \cdot C_1$ for a constant C_1.



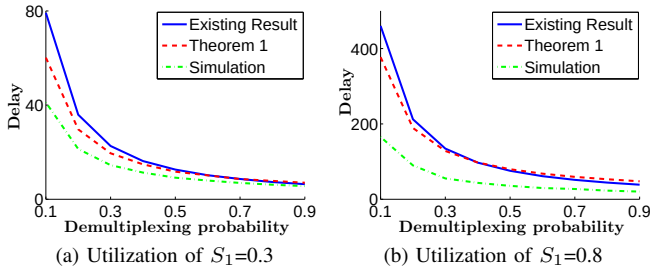
(a) Utilization of S_1=0.3 (b) Utilization of S_1=0.8

Fig. 6. Bernoulli arrivals, Bernoulli scaling.



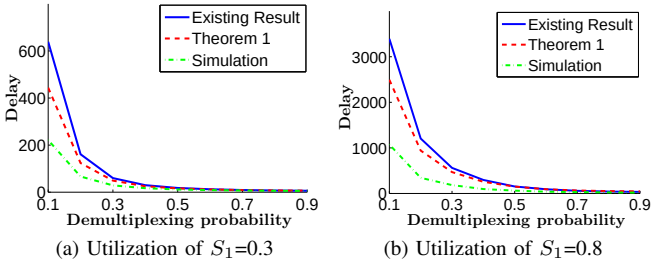
(a) Utilization of S_1=0.3 (b) Utilization of S_1=0.8

Fig. 7. Bernoulli arrivals, MMOO scaling.



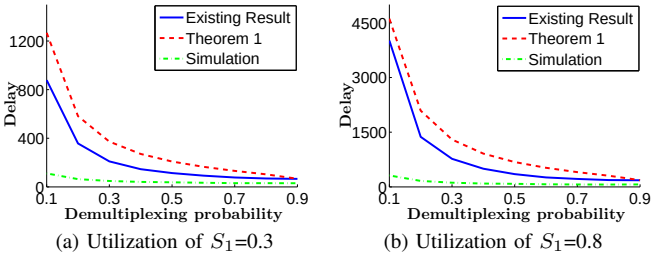
(a) Utilization of S_1=0.3 (b) Utilization of S_1=0.8

Fig. 8. Poisson arrivals, Bernoulli scaling.



(a) Utilization of S_1=0.3 (b) Utilization of S_1=0.8

Fig. 9. Poisson arrivals, MMOO scaling.



(a) Utilization of S_1=0.3 (b) Utilization of S_1=0.8

Fig. 10. MMOO arrivals, MMOO scaling.

Figure 6 shows the statistical delay bounds obtained with Theorem 1 and Theorem 2 (existing result) as well as the simulation results in the case of Bernoulli arrivals and Bernoulli scaling. For demultiplexing probabilities $p \leq 0.7$ the figures clearly illustrate the order of the results, with the new leftover service computation method as a superior option especially for smaller pass probabilities p. For $p > 0.7$, the results of Theorem 1 (with independent increments) and Theorem 2 are nearly the same. Going to the higher utilization of 0.8 in Figure 6.b of course results in higher delays, but also seems to loosen the bounds somewhat as the gap to the simulation results grows.

Figure 7 shows the statistical delay bounds in the case of Bernoulli arrivals and MMOO scaling. Basically the same observations as in the previous paragraph, for Bernoulli arrivals and Bernoulli scaling, can be made. It is interesting to note, however, how the MMOO scaling leads to much higher delay bounds compared to the Bernoulli scaling case.

Next, Figure 8-9 display the same as Figure 6-7, but now for Poison arrivals. For most of it, the conclusions for the Poisson arrivals are the same as for the Bernoulli arrivals, yet one interesting observation is that for Poisson arrivals and Bernoulli scaling for the high utilization case, the leftover service compution results in slightly, but clearly visible worse delay bounds than the method from Theorem 2. This is not the case for the MMOO scaling which indicates that things are not so simple here ...

Figure 10 shows the statistical delay bounds in the case of MMOO arrivals and MMOO scaling. Note that in this scenario we do not have independent increments of arrivals any more and and thus have to resort to the respective case in Theorem 1. We can see that the method by commuting service curve and scaling element now clearly dominates the leftover service method. As a general remark, we can observe that the bounds are quite pessimistic for low values of the pass probability p when they are compared to the simulation results. It is simply harder to cope with the correlations in arrivals and scaling processes.

C. Discussion

In the previous subsection we compared the results obtained by the dual methods from Theorem 1 and Theorem 2 for some numerical examples. We can see that, on one hand, Theorem 1 provides the opportunity to utilize some additional information on arrivals to show an advantage, especially in scenarios where the sub-flow of interest is small. On the other hand, a disadvantage of using the leftover service curve method is that the delay bound computation is not easily extensible to the n nodes case. This is because we would have to introduce different "x" for the leftover service curve element at each node, which makes the analytical solution very complicated. In contrast, Theorem 2 is more easily applicable to the n nodes case.

However, the "door is still open" for the method using the leftover service curve. In the following, we sketch the iterative computation steps to be taken, but keep the detailed computation as future work.

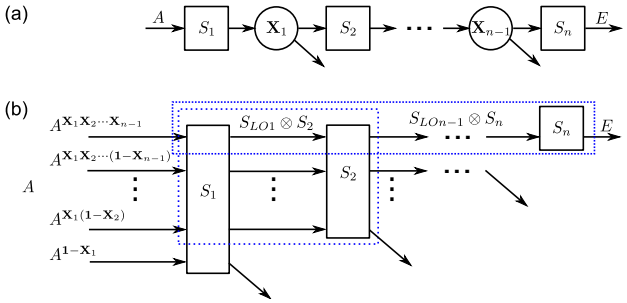


Fig. 11. A flow demultiplexing network consisting of services and scaling elements.

Consider a network scenario from Figure 11(a) which is the general extension of Figure 2(a). Because the service curves of the nodes are not necessarily strict, we can firstly apply all the scaling effects to A and after iteratively computing the leftover service curves we then obtain the transformed system in Figure 11(b). The iteration can be stated as the following equations with $n = 2, 3, \ldots$

$$S_{LO1}(s,t) = \left[S_1(s,t) - A^{\mathbf{1}-\mathbf{X}}(s,t-x_1)\right]_+ 1_{\{t-x_1>s\}}$$

$$S_{LOn-1}(s,t) = \Big[(S_{LOn-2} \otimes S_{n-1})(s,t) - A^{\mathbf{X}_1\mathbf{X}_2\cdots(\mathbf{1}-\mathbf{X}_{n-1})}(s,t-x_{n-1})\Big]_+ 1_{\{t-x_{n-1}>s\}}.$$

V. Conclusions

We have presented a novel approach to analyze queueing networks with flow demultiplexing, e.g., due to loss or dynamic routing. First we used a scaling element to model the demultiplexing and showed how to transform a network node with a demultiplexing operation on the output flow into an equivalent system with a logically demultiplexed input flow. Then we applied network calculus' leftover service computation for the logically demultiplexed input sub-flow of interest to compute a statistical delay bound. We compared this result with an alternative result from previous work of ours [4]. Although the new computation cannot completely dominate the other approach, it tends to be a more intuitive way of dealing with demultiplexing in the framework of stochastic network calculus. Furthermore, it has the advantage that specific information on the arrivals can be easier incorporated than for [4]. On the downside, the generalization to the n nodes for the leftover service method still remains open whereas the other method is already more advanced with respect to this.

References

[1] S. Chakraborty, S. Kuenzli, L. Thiele, A. Herkersdorf, and P. Sagmeister. Performance evaluation of network processor architectures: Combining simulation with analytical estimation. *Computer Networks*, 42(5):641–665, April 2003.

[2] C. S. Chang. Stability, queue length and delay, Part II: Stochastic queueing networks. In *Proceedings of the 31st IEEE Conference on Decision and Control*, pages 1005–1010, December 1992.

[3] C.-S. Chang. *Performance Guarantees in Communication Networks*. Telecommunication Networks and Computer Systems. Springer-Verlag, 2000.

[4] F. Ciucu, J. Schmitt, and H. Wang. On expressing networks with flow transformation in convolution-form. In *Proceedings of IEEE INFOCOM*, pages 1979–1987, April 2011.

[5] R. L. Cruz. A calculus for network delay, parts I and II. *IEEE Transactions on Information Theory*, 37(1):114–141, January 1991.

[6] R. L. Cruz. SCED+: Efficient management of quality of service guarantees. In *Proc. IEEE INFOCOM*, volume 2, pages 625–634, March 1998.

[7] M. Fidler and J. Schmitt. On the way to a distributed systems calculus: An end-to-end network calculus with data scaling. In *Proceedings of ACM SIGMETRICS/Performance*, pages 287–298, 2006.

[8] R. Jain and I. Chlamtac. The P-Square algorithm for dynamic calculation of percentiles and histograms without storing observations. *Communications of the ACM*, pages 1076–1085, October 1985.

[9] Y. M. Jiang and Y. Liu. *Stochastic Network Calculus*. Springer, 2008.

[10] H. Kim and J.C. Hou. Network calculus based simulation: theorems, implementation, and evaluation. In *Proceedings of IEEE INFOCOM*, March 2004.

[11] A. Koubaa, M. Alves, and E. Tovar. Modeling and worst-case dimensioning of cluster-tree wireless sensor networks. In *Proceedings of the 27th IEEE International Real-Time Systems Symposium*, pages 412–421, December 2006.

[12] J.-Y. Le Boudec and P. Thiran. *Network Calculus A Theory of Deterministic Queuing Systems for the Internet*. Number 2050 in Lecture Notes in Computer Science. Springer-Verlag, 2001.

[13] J. Liebeherr, Y. Ghiassi-Farrokhfal, and A. Burchard. On the impact of link scheduling on end-to-end delays in large networks. *IEEE Journal on Selected Areas in Communications*, 29(5):1009–1020, May 2011.

[14] J. Schmitt and U. Roedig. Sensor network calculus - a framework for worst case analysis. In *Proceedings of Distributed Computing on Sensor Systems*, pages 141–154, June 2005.

[15] J. B. Schmitt, H. Wang, and I. Martinovic. Dynamic Demultiplexing in Network Calculus - Theory and Application. *Performance Evaluation*, 68(2):211–218, February 2011.

[16] T. Skeie, S. Johannessen, and O. Holmeide. Timeliness of real-time IP communication in switched industrial ethernet networks. *IEEE Transactions on Industrial Informatics*, 2(1):25–39, February 2006.

[17] A. Varga. OMNeT++. http://www.omnetpp.org/.