End-to-End Delay Bounds for Variable Length Packet Transmissions under Flow Transformations

Hao Wang and Jens Schmitt
DISCO Lab, University of Kaiserslautern
{wang, jschmitt}@informatik.uni-kl.de

ABSTRACT

A fundamental contribution of network calculus is the convolution-form representation of networks which enables tight end-to-end delay bounds. Recently, this has been extended to the case where the data flow is subject to transformations on its way to the destination. Yet, the extension, based on so-called scaling elements, only applies to a setting of identically sized data units, e.g., bits. In practice, of course, one often has to deal with variable-length packets. Therefore, in this paper, we address this case and propose two novel methods to derive delay bounds for variable-length packets subject to flow transformations. One is a relatively direct extension of existing work and the other one represents a more detailed treatment of packetization effects. In a numerical evaluation, we show the clear superiority of the latter one and also validate the bounds by simulation results.

Categories and Subject Descriptors

C.2.m [**Computer-Communication Networks**]: Miscellaneous; C.4 [**Performance of Systems**]: Modeling techniques

General Terms

Performance, Theory

Keywords

Variable length packet, demultiplexing, network calculus, packet scaling element.

1. INTRODUCTION

Network calculus has been established as a promising approximative approach to queueing theory. Simply speaking, by using inequalities instead of equalities, it can circumvent some long-standing fundamental problems in queueing theory, especially in networks with non-Poisson arrivals and multiple nodes. Network calculus was originally conceived by Cruz [5] in the early 1990s and soon after by Chang [2]. Subsequently, many researchers have contributed to it ([3, 11, 9]). The high modelling 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 for example wireless sensor networks [14], switched Ethernets [15], System-on-Chip (SoC) [1], or even to speed-up of simulations [10].

A key to good performance bounds is the *convolution-form* expression of multi-node networks. If we describe the service provided by a node i as a lower bound process $S_i(s,t)$ for time $0 \leq s \leq t$, a tandem of n nodes also provides a lower bound for the service process

$$S_1 \otimes S_2 \otimes \cdots \otimes S_n \ ,$$

where "$\otimes$" denotes the $(min,+)$ convolution defined as $S_1 \otimes S_2 = \inf_{0\leq s\leq t}\{S_1(0,s) + S_2(s,t)\}$. As a consequence, the end-to-end performance analysis can be obtained by applying a single-node analysis. Yet, this convolution-form has a limitation: the flows in the network are assumed to be transported unaltered. However, in many real-world applications a data flow is often transformed during its transfer; for instance, some parts are lost, routed to another destination, or even aggregated with other data. Previous work in deterministic network calculus has proposed the so-called data scaling element [7] to model such flow transformation. A subsequent work then introduced the stochastic scaling element [4], which represents the network in convolution-form and provides a flexible means of capturing actual transformations. The key idea therein is to commute the scaling element with a dynamic server element recursively in order to obtain a single-node form representation of the network. However, these models have a limitation: they scale the flow only at the granularity of identical length data units (bits or packets). This is quite restrictive as many networks use variable-length packets and information and events like sending and receiving cannot be observed at the bit-level. In this paper, we therefore propose a new scaling element which respects flows as a sequence of (variable-length) packets rather than just bits. The critical challenge in defining such a scaling element at the packet-level is that it should preserve the convolution-form expression of multi-node networks. To ease the exposure we focus on an abstract but widely applicable flow transformation operation: the demultiplexing of packets, i.e. to thin a flow of packets by selecting only some of them, e.g. due to network operations such as routing, load balancing, or simply loss of packets.

VALUETOOLS 2014, December 09-11, Bratislava, Slovakia

DOI 10.4108/icst.valuetools.2014.258194

We are, of course not the first to treat the case of variable-length packets (though, to the best of our knowledge we are the first to take this into account under flow transformations). In particular, the packetizer element [11, 3] has been introduced in network calculus to deal with flows of variable-length packets. [12, 8] have extended it to the stochastic settings. [12] models heavy-tailed arrivals with packet distributions. [8] reveals the inherent dependence brought by the packet process to the arrivals and services. We also use the packetizer but now in combination with a scaling element in order to model flow transformations at the packet level. In previous work of ours [16] we showed a novel model to understand the demultiplexing for first-in-first-out (FIFO) servers and to compute tighter end-to-end delay bounds. Yet, for the n-node network to preserve the convolution-form the computation of the performance measures turned out to be hard. Therefore, in this paper, we commute the packet-level scaling element and the dynamic server (as proposed for the bit level in [4]), in order to provide a tractable end-to-end delay bound computation.

The rest of this paper is organized as follows. In Section 2, we recall several fundamental definitions of network calculus and provide extensions of some of them under the assumption of variable length packets. In Section 3, we present two methods to compute the end-to-end delay bounds. The first is a relatively direct extension of [4], whereas the second provides a more detailed treatment of the packetization of flows. In Section 4 we compare two methods and compare them against simulation results. We conclude in Section 5.

2. MODELLING THE DEMULTIPLEXING OF VARIABLE LENGTH PACKET FLOWS

In this section we first recall the definitions of demultiplexing, the scaling element, and the packetizer. Next, we define the packet scaling element. Further, we discuss alternative system models when analyzing the end-to-end delay of a packet. Throughout the paper, the time model is discrete.

In the framework of network calculus, a data flow from a source to a destination is modelled by an arrival process $A(t)$, which counts the number of arriving data units (bits) in time interval $(0, t]$. The bivariate form is accordingly $A(s,t) := A(t) - A(s)$. We also denote $a(t) = A(t-1, t)$, i.e., the arriving data units in time slot t. We model the flow departing from a server as the process $D(t)$ with the corresponding definition. These model the flow of bits. For many networks, the flow of bits is transformed on the way to their destination. For example, a flow can be expanded by some extra data or some parts of the flow can be lost. One interesting transformation is the *demultiplexing*, whereby the flow is split into multiple sub-flows. For example, if a part of the original flow is routed to another destination at some node, then the flow is demultiplexed into two sub-flows, one for each destination. We denote these as two arrival processes $A^{(1)}(t), A^{(2)}(t)$ satisfying

$$A(t) = A^{(1)}(t) + A^{(2)}(t) \ ,$$

for all $t \geq 0$. If we describe the splitting operation on the bit-level as an indicator function $\mathbf{1}_{\{\text{“this bit goes to destination (1)”}\}}$, which equals to 1 if term true, 0 otherwise, we have that $A^{(1)}(t) = \sum_{i=1}^{A(t)} \mathbf{1}_{\{\text{“bit } i \text{ goes to destination (1)”}\}}$. Denoting this indicator function for bit i as X_i, we define the *scaling element* as a random process $\mathbf{X} = (X_i)_{i \geq 1}$ (a more general definition can be found in [4]). We denote the scaled arrivals as $A^{\mathbf{X}}(t)$ and

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

Then we can use this scaling element to model the demultiplexing operation. Clearly, with $\mathbf{1} = (1, 1, \dots)$ we get $A^{(2)}(t) = A^{\mathbf{1}-\mathbf{X}}(t)$. As we assume that the demultiplexing operation happens instantaneously, the scaling element has no queue.

However, in real-world the bits perhaps belong to different (variable length) packets, and transformations happen on the whole packet instead of each bit. For the demultiplexing example, we may simply know the routing probability of a complete packet to one destination. To model this (packet-level) demultiplexing operation, we need to extend the scope of the scaling element. Yet, before we do that, we first integrate variable length packets into the network calculus framework. We denote the packet lengths as a sequence of positive integer random variables $l_1, l_2, \dots$. A packet process $L(n), n \geq 1$ is a cumulation of these $r.v.$'s, $L(n) = l_1 + l_2 + \cdots + l_n$, and $l_n = L(n) - L(n-1)$ with $L(0) = 0$. A packet flow is modelled using the definition of packetizer ([3], [11]).

Definition 1 (Packetizer). *Given a packet process $L(n)$ and an arrival process $A(t)$, an L-packetizer is a network element expressed by a function $P^L(\cdot)$ satisfying for all $A(t), t \geq 0$*

$$P^L\left(A(t)\right) = L(N_t) \ ,$$

where

$$N_t = \max\left\{m : L(m) \leq A(t)\right\} \ . \tag{1}$$

We say that a flow $A(t)$ is L-packetized if $A(t) = P^L\left(A(t)\right)$ for any $t \geq 0$. So a packet flow is an L-packetized arrival process. Note, the function P^L is not restricted to a real network element with a queue, it can also be used to parse a bit flow (e.g., with marks) into packets and not change its timing. In the rest of this paper, we will use both meanings.

Now we consider the demultiplexing of a packet flow. We extend the definition of the scaling element.

Definition 2 (Packet Scaling Element). *A packet scaling element consists of an L-packetized arrival process $A(t) = \sum_{i=1}^{N_t} l_i$, a packet scaling process $\mathbf{X}$ taking non-negative integer values and a scaled packetized flow defined for all $t \geq 0$ as*

$$A^{\mathbf{X}}(t) = \sum_{i=1}^{N_t} l_i X_i \ .$$

We can use the packet scaling element to model the transformation of the packet flow, specifically, the demultiplexing case. $l_i X_i$ means $l_i \cdot \mathbf{1}_{\{\text{“packet } i \text{ goes to destination (1)”}\}}$, i.e., demultiplexing operates on each packet and X_i equals either 0 or 1.

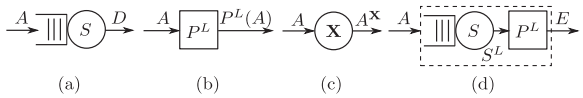

Figure 1: Network elements: (a) dynamic server, (b) packetizer, (c) packet scaling element, (d) packetized server.

A packet flow is usually processed or served by a queueing system before or after being demultiplexed. To analyze the delay of a packet through this system we distinguish two models. One is, after being served by each node the output is always packets, i.e., the bits are packetized by a packetizer P^L. The other is, there is no packetizer after service, yet we observe from the bit flow the last bit of each packet according to a packet process L. In previous work [4] we derive the end-to-end delay bound for the bit flow under flow transformation. The second case can be a critical challenge for that approach (L-modulated scaling process and sampling due to L). In this paper, we focus on the first case and assume that the packetization is not changed along the path.

In network calculus, we characterize the queueing system using a *dynamic server* ([3]). By convention, we denote it as $S(s,t)$ for $0 \leq s \leq t$. Note that it is not the server itself but only a property of the server. It defines a lower bound process on the service such that the following *convolution* inequality holds for all $t \geq 0$.

$$D(t) \geq A \otimes S(t) := \inf_{0 \leq s \leq t} \{A(s) + S(s,t)\} .$$

When the inequality holds with equality, we say the dynamic server is *exact*. Note that the convolution of two concatenated dynamic servers $S_1 \otimes S_2$ is still a dynamic server (concept of convolution-form network). We define a *packetized server* as a bit server followed by a packetizer P^L, and denote the dynamic server of it as $S^L(s,t)$. Given the dynamic server of the bit server S and the packet process L and assuming that a maximum packet size l_{max} exists, we obtain a possible S^L,

$$S^L(s,t) = [S(s,t) - l_{max}]_+ . \quad (2)$$

The proof follows using a busy time analysis.

We illustrate the network elements in Figure 1. Now we define the packet delay.

Definition 3 (Packet Delay). *A process $W(t)$ is called packet delay (process), if for all $t \geq 0$*

$$W(t) = \inf \{d \geq 0 : P^L(A(t)) \leq P^L(D(t+d))\} .$$

Here we assume the service is FIFO. The packet delay is a virtual delay that would be experienced by a packet which arrives at time t.

3. END-TO-END DELAY OF A NETWORK WITH FLOW DEMULTIPLEXING

Figure 2: A network model consisting of packetized arrivals, services and packet scaling elements.

In this section, we compute the end-to-end packet delay for networks with multiple demultiplexers. According to previous work, there are two ways to compute the end-to-end delay: (1) commute service and scaling elements [4], (2) get the leftover service for the flow of interest if the server uses FIFO scheduling [16]. In this paper, we use the first, i.e., we repeatedly move all the packet scaling elements in front of the packetized servers and obtain the convolution-form of the network. Then we calculate the end-to-end delay bounds. Here, we have two choices: one is to "normalize" the packet flow as well as the bitwise service with packet size, so that the observation is directly on each packet irrespective of its size ($\rightarrow$ Section 3.1); the other is to use Definition 3 and derive the delay bound directly through observing the original bit flow with packetizers ($\rightarrow$ Section 3.2). For the packet flow we assume that the packet lengths l_i's are *i.i.d.* with $l_{max} < \infty$. In fact, this assumption can be justified in many real-world applications with heterogeneous, large-scale, and high degree of multiplexing environment.

3.1 Observing the Packet Flow

Consider Figure 2, we lift our observation of the flow directly from the bit level to the packet level. This means we view each packet as a single data unit ignoring its size. Then we re-express the service this packet receives. After doing so we can derive the end-to-end packet delay bound directly using the calculation from [4].

Consider the arrivals consist of packets whose arrival times are defined as the arrival time of the last bit, we can model these time jumps with a counting process and together with a packet size distribution, model the arrival process as a compound process - $A(t) = \sum_{i=1}^{N(t)} l_i$, where $\{N(t), t \geq 0\}$ is the counting process, i.e., the number of arriving packets in time t, and l_i is the i-th packet size. This seems to be a slightly different description of a packetized flow, because here we do not assume a packetizer element in the network. Yet, the packetized process resulting from packetizer is also covered by this definition. Consequently, we obtain an arrival process of packets - $\{N(t), t \geq 0\}$. We call this approach "normalization" of the bit flow by the packet sizes.

Such a sequence of packets will be served by a service element described by the bitwise service capacity together with a packetizer. How much service capacity does a packet receive? To answer this question is not very hard. For example, assume that a packet with length l will be served by a server with constant service capacity C *bits/s*, so the service rate for this packet is C/l *packets/s*. This is the "normalization" on the service side. The constant capacity server is transformed into a variable capacity server. We write it as $S^{norm}(s,t) = \sum_{i=s}^{t} c(i)$ for all $0 \leq s \leq t$. Here all the $c(i)$'s are the time varying capacities of serving a packet at time i.

In [4], we derive the end-to-end delay bounds for a flow with identically sized data units. The derivation is based on moment generating function (MGF, denoted by $M_X(\theta)$ for *r.v.* X and any $\theta > 0$, $M_X(\theta) = E[e^{\theta X}]$) bounds of the arrivals and the services and expresses a network with flow transformations in a convolution-form. Therein, the servers are assumed to have constant MGF bounds. However, to use the same derivation is quite challenging, because now on one hand, the servers have variable capacities and to know their MGF bounds is hard; on the other hand, they are "normalized" by the same packet process and hence dependent of each other.

To obtain the MGF of the dynamic server we can firstly express the inter-service time. Then we use the (inverse) Laplace transform of the convolution of inter-arrival times and packet size distributions to compute the *p.d.f.* of the inter-service time. Thirdly, we use renewal theory to obtain the *p.d.f.* of the counting process of the service. At last, the MGF follows by its definition. About the dependency, Hölder inequality might be a solution, but many parameters are introduced. We may also construct or prove some negative correlations after we use the Chernoff bound in Theorem 1 of [4]. All of these approaches lay their focus on the accuracy of the expressiveness, yet, they lose the analytical tractability. In this section, we just want to provide a method to calculate the end-to-end delay for variable-length packet flows that follows closely the approach in [4] and then compare it against the more sophisticated method using the packet scaling element. Assume that the bit-wise capacity $S(t)$ is offered work-conserving with variant rates and let $S(t) \geq Ct$ for any $t \geq 0$ such that MGF bound $M_{S(t)}(-\theta) \leq e^{-\theta Ct}$ for $\theta > 0$, then we can vaguely write $c(i) \geq C/l_x$, where l_x means either some packet length or ∞. We assume the packet size has a limit, i.e., $l_x \leq l_{max}$. Clearly, $c(i) \geq C/l_{max}$. We obtain a lower bound of this normalized dynamic server $S^{norm}(s,t) \geq \frac{C}{l_{max}}(t-s)$. Now, we represent the dynamic server as a server with the normalized capacity - C/l_{max}. And this solves the above problems at the same time. Consider the same network scenario as in Theorem 1. We assume the compound process as the arrivals instead of using packetizer. We also assume $M_{S_j(t)}(-\theta) \leq e^{-\theta C_j t}, j \geq 1$ at each bit server. The end-to-end delay has the following stochastic bound

$$Pr(W > d) \leq K^n b^d \ .$$

We point out, the only difference between this result and Theorem 1 in [4] is that the MGF bound of each service is

$$M_{S_i^{norm}(s,t)}(-\theta) \leq e^{-\theta \frac{C_i}{l_{max}}(t-s)} \ . \tag{3}$$

We also point out, when we do the "normalization" to the service, whether there exists a real packetizer component or not does not change the packet delay analysis, because only after the last bit of a packet is served by the bit-wise server, the service of this packet is considered to be finished, this is just as if there was a packetizer virtually.

3.2 Observing the Original Bit Flow with Packetizers

In the previous subsection, we provided an approach to calculate end-to-end delay bounds for variable-lengths packet flows under flow demultiplexing which observes a flow on the packet-level rather than the bit level. Now we directly observe the flow on the bit level as in Figure 2. From [4] we know when deriving the end-to-end delay bound we should avoid summing up the delay bounds node-by-node, but rather use the "pay burst only once" principle. To do so, we express the network in convolution-form through moving the scaling elements between two servers to the front. The challenge now is that the scaling element is not at the bit level anymore. We provide the following lemma to commute the service and scaling element at packet-level, which is instrumental to the derivation of end-to-end delay bounds.

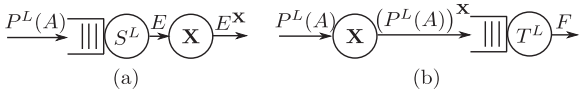

Figure 3: Commutation of packetized service and packet scaling.

Lemma 1. (Commutation of Packet Scaling Element and Dynamic Server). *Consider system (a) and (b) with packetized arrivals $A(t) = P^L(A(t))$ in Figure 3. We define $T^L(s,t) := \sum_{i=M_s+1}^{N_t} l_i X_i$ as the exact dynamic server in (b), where $A(s) = \sum_{i=1}^{M_s} l_i$, $A(s) + S^L(s,t) = \sum_{i=1}^{N_t} l_i$. If A, S, $\mathbf{X}$, and L are independent, then for all $t \geq 0$,*

$$F(t) \leq E^{\mathbf{X}}(t) \ .$$

The proof follows by using the definitions of the exact dynamic server and T^L. See details in [17]. Through this lemma, we see that there are less departures for the transformed system than in the original system, which ensures that the delays are higher. The expression of T^L looks complicated, but the meaning is clear. $\sum_{i=M_s+1}^{N_t} l_i$ are the packets served from time s to t. And because after passing through a scaling element $\mathbf{X}$, only a scaled part of these packets is sent to the next server, the service they received should also be a scaled part of the total service. After recursively using this lemma we get an expression for the network in terms of a scaled arrival process served by a dynamic server in convolution-form. The arrivals have the form

$$\left(\cdots \left(A^{\mathbf{X}_1} \right)^{\mathbf{X}_2 \cdots} \right)^{\mathbf{X}_k} (t) \ ,$$

if there are k packet scaling elements. We denote it as $A^{(k)}(t)$. The alert reader may note that, for the bit flow, the concatenation of scaling elements can be naturally formulated as $\left(A^{\mathbf{X}_1}\right)^{\mathbf{X}_2}(t) = \sum_{i=1}^{\sum_{j=1}^{A(t)} X_{1,j}} X_{2,i}$, whereas for the packet flow, this is not true any more. We point out that they are just different in appearance but the same in essence - after each round of scaling we choose a part of the packets (bits) from the input flow. Therefore, we provide the delicate expression of $A^{(k)}(t)$, which will be used in the rest of this section. Assume an L-packetized flow $A(t) = l_1 + l_2 + \cdots + l_{N_t}$, where N_t is given in Eq. (1). We first denote the packets respectively the number of packets in the arrivals until time t after each round of scaling as $l_{k,i}$ respectively $m_t^{(k)}$. Clearly for $k > 0$

$$m_t^{(0)} \quad = \quad N_t \ ,$$

$$
\begin{aligned}
m_t^{(1)} &= \sum_{i=1}^{m_t^{(0)}} 1_{\{X_{1,i}>0\}} \ , \\
&\cdots \\
m_t^{(k)} &= \sum_{i=1}^{m_t^{(k-1)}} 1_{\{X_{k,i}>0\}} \ . \qquad (4)
\end{aligned}
$$

Further we denote $A^{(k)}(t)$ as

$$
\begin{aligned}
A^{\mathbf{X}_1}(t) &= l_1 X_{1,1} + \cdots + l_{N_t} X_{1,N_t} \\
&= l_{1,1} + \cdots + l_{1,m_t^{(1)}} \ , \\
\left(A^{\mathbf{X}_1}\right)^{\mathbf{X}_2}(t) &= l_{1,1} X_{2,1} + \cdots + l_{1,m_t^{(1)}} X_{2,m_t^{(1)}} \\
&= l_{2,1} + \cdots + l_{2,m_t^{(2)}} \ , \\
&\cdots \\
A^{(k)}(t) &= l_{k,1} + \cdots + l_{k,m_t^{(k)}} \\
&= \sum_{i=1}^{m_t^{(k-1)}} l_{k-1,i} X_{k,i} \ . \qquad (5)
\end{aligned}
$$

Next, we provide two useful lemmas for deriving the end-to-end delay bounds. The proofs can also be found in [17].

LEMMA 2 (STATIONARITY BOUND). *Assume that the packets l_i's of a packet process L are i.i.d. , the X_i's of a packet scaling element $\mathbf{X}$ are also i.i.d. , A and B are two L-packetized arrival processes, then for all $s, t, x > 0$,*

$$
Pr\left(A^{\mathbf{X}}(t) - B^{\mathbf{X}}(s) \geq x\right) \leq Pr\left((A(t) - B(s))^{\mathbf{X}} \geq x\right) \ .
$$

LEMMA 3. (RECURSIVE MGF BOUND OF SCALED PROCESS). *Assume that A is an L-packetized process, S_i^L is the packetized server, l_i's are i.i.d. with maximal length $l_{max} < \infty$, and $\mathbf{X}_i$'s are Markov-Modulated On-Off (MMOO) loss processes and independent of A and S_i^L, if we denote $V_{n-1}(\theta_n)$ as*

$$
E\left[e^{\theta\left(\cdots\left(A(t-s) - S_1^L(s,u_1)\right)^{\mathbf{X}_1} - \cdots - S_{n-1}^L(u_{n-2},u_{n-1})\right)^{\mathbf{X}_{n-1}}}\right] \ ,
$$

then for all $0 \leq s \leq u_1 \leq \cdots \leq u_{n-1} \leq t$, and $n > 1$,

$$
V_{n-1}(\theta_n) \leq e^{-\theta_{n-1} S_{n-1}^L(u_{n-2},u_{n-1})} V_{n-2}(\theta_{n-1}) \ ,
$$

where $\theta_i > 0$, $1 \leq i \leq n$ is given in the proof.

We now derive the end-to-end delay bound and show that it grows in $\mathcal{O}(n)$ where n is the number of nodes.

THEOREM 1. (END-TO-END DELAY BOUNDS IN A PACKET FLOW TRANSFORMATION NETWORK). *Consider the network scenario from Figure 2 where an L-packetized arrival process $A(t) = P^L(A(t))$ traverses a series of stationary and (mutually) independent bit level service elements followed by an L-packetizer and scaling elements denoted by $S_1^L, S_2^L, \ldots, S_n^L$ and i.i.d. loss processes $X_1, X_2, \ldots, X_{n-1}$, respectively. Assume the packet lengths of L - l_i's are i.i.d. . Assume the MGF bounds $M_{A(s,t)}(\theta) \leq e^{\theta r_A(\theta)(t-s)}$ and $M_{S_k(t)}(-\theta) \leq e^{-\theta C_k t}$, for $k = 1, 2, \ldots, n$, and some $\theta > 0$. We also assume that the maximum packet length $l_{max} < \infty$. Under a stability condition, to be explicitly given in the proof, for $\theta_i > 0, i = 1, 2, \ldots, n$, we have the following end-to-end steady state delay bounds for all $d \geq 0$*

$$
Pr(W > d) \leq e^{\left(\sum_{i=1}^n \theta_i + \theta_1\right) l_{max}} K^n b^d \ , \qquad (6)
$$

where the constants K and b are also given in the proof. Moreover, the ε-quantiles scale as $\mathcal{O}(n)$, for $\varepsilon > 0$.

PROOF. First we use Lemma 1 to transform the system view. To do so, we iteratively commute the packetized server and the packet scaling element k times. See Figure 4. Since the output of the transformed system is smaller than or equal to the original system, the delay bound of the transformed one must be larger than or equal to the delay bound of the original one, hence, we compute the delay bound of this transformed system.

Figure 4: Apply Lemma 1 for k times.

Next, fix $t, d \geq 0$. For $k, s \geq 0$ we define $U_0(s, u_0) = A(s)$, for $u_0 = s$, and then recursively

$$
U_k(s, u_k) = \left(U_{k-1}(s, u_{k-1}) + S_k^L(u_{k-1}, u_k)\right)^{\mathbf{X}_k}
$$

for $k \geq 1$ and $u_{k-1} \leq u_k$. We prove the theorem at the first steps by induction. For $k \geq 1$ we assume the following two statements $(\mathcal{S}_1)$ and $(\mathcal{S}_2)$ for the induction:

$$
\begin{aligned}
(\mathcal{S}_1) \qquad & Pr(W_k(t) > d) \leq \sum_{0 \leq s \leq t} \sum_{s \leq u_1 \leq \cdots \leq u_{k-1} \leq t+d} \\
& Pr\left(A^{(k-1)}(t) > U_{k-1}(s, u_{k-1}) + S_k^L(u_{k-1}, t+d)\right) \ ,
\end{aligned}
$$

and for fixed s and u_k,

$$
\begin{aligned}
(\mathcal{S}_2) \qquad & \left(A^{(k-1)}(s) + T_{k-1}^L \otimes S_k^L(s, u_k)\right)^{\mathbf{X}_k} \\
& = \inf_{s \leq u_1 \leq \cdots \leq u_k} U_k(s, u_k) \ ,
\end{aligned}
$$

where T_k^L is defined recursively as $T_0^L(0) = 0$, $T_0^L(s) = \infty$ for all $s > 0$, and for N_s the number of packets in $A(s)$

$$
\begin{aligned}
& T_k^L(s, u_k) := \sum_{i=m_s^{(k-1)}}^{N_{u_k}} l_{k-1,i} X_{k,i} \ , \\
& \text{where} \sum_{i=1}^{m_s^{(k-1)}} l_{k-1,i} = A^{(k-1)}(s) \ , \\
& \sum_{i=1}^{N_{u_k}} l_{k-1,i} = A^{(k-1)}(s) + T_{k-1}^L \otimes S_k^L(s, u_k) \ . \qquad (7)
\end{aligned}
$$

First we prove the initial step of the induction, i.e., $k = 1$. For statement $(\mathcal{S}_1)$, we have

$$
\begin{aligned}
Pr(W_1(t) > d) &= Pr(A(t) > D(t+d)) \\
&\leq Pr\left(A(t) > A \otimes S_1^L(t+d)\right) \\
&\leq \sum_{0 \leq s \leq t} Pr\left(A(t) > A(s) + S_1^L(s, t+d)\right)
\end{aligned}
$$

$$= \sum_{0\le s\le t} Pr\left(A^{(0)}(t) > U_0(s,u_0) + S_1^L(s,t+d)\right) .$$

In the first line we used the definition of packet delay. In the second line we used the definition of dynamic server. And in the third line we expanded the convolution and used Boole's inequality. In turn for statement $(\mathcal{S}_2)$, we have

$$\begin{aligned}
&\left(A^{(0)}(s) + T_0^L \otimes S_1^L(s,u_1)\right)^{\mathbf{x}_1} \\
&= \left(A(s) + \inf_{s\le x\le u_1}\left\{T_0^L(s,x) + S_1^L(x,u_1)\right\}\right)^{\mathbf{x}_1} \\
&= \left(A(s) + S_1^L(s,u_1)\right)^{\mathbf{x}_1} \\
&= \inf_{s\le u_1}\left(A(s) + S_1^L(s,u_1)\right)^{\mathbf{x}_1} \\
&= \inf_{s\le u_1}\left(U_0(s,u_0) + S_1^L(u_0,u_1)\right)^{\mathbf{x}_1} \\
&= \inf_{s\le u_1} U_1(s,u_1) .
\end{aligned}$$

In the third line we used that $T_0^L(0) = 0, T_0^L(s) = \infty$. In the fourth line we rewrote the third line using inf, because s and u_1 are actually fixed. In the fifth line we used the definition of U_0. In the last line we used the recursive definition of U_k.

For the induction we next assume that $(\mathcal{S}_1)$ and $(\mathcal{S}_2)$ hold for $k \ge 1$. Then we prove them for $k+1$. Using the argument from the initial step of the induction we can write the end-to-end delay until the $k+1^{th}$ node

$$\begin{aligned}
&Pr(W_{k+1}(t) > d) \\
&\le Pr\left(A^{(k)}(t) \ge \inf_{0\le s\le t+d}\left\{A^{(k)}(s) + T_k^L \otimes S_{k+1}^L(s,t+d)\right\}\right) \\
&\le \sum_{0\le s\le t}\sum_{s\le u_k\le t+d} Pr\big(A^{(k)}(t) \ge A^{(k)}(s) + T_k^L(s,u_k) \\
&\quad + S_{k+1}^L(u_k,t+d)\big) \\
&= \sum_{0\le s\le t}\sum_{s\le u_k\le t+d} Pr\Bigg(A^{(k)}(t) \ge \sum_{i=1}^{m_s^{(k-1)}} l_{k-1,i}X_{k,i} + \\
&\quad \sum_{i=m_s^{(k-1)}}^{N_{u_k}} l_{k-1,i}X_{k,i} + S_{k+1}^L(u_k,t+d)\Bigg) \\
&= \sum_{0\le s\le t}\sum_{s\le u_k\le t+d} Pr\Bigg(A^{(k)}(t) \ge \sum_{i=1}^{N_{u_k}} l_{k-1,i}X_{k,i} + \\
&\quad S_{k+1}^L(u_k,t+d)\Bigg) \\
&= \sum_{0\le s\le t}\sum_{s\le u_k\le t+d} Pr\Big(A^{(k)}(t) \ge \big(A^{(k-1)}(s) + \\
&\quad T_{k-1}^L \otimes S_k^L(s,u_k)\big)^{\mathbf{x}_k} + S_{k+1}^L(u_k,t+d)\Big) \\
&= \sum_{0\le s\le t}\sum_{s\le u_k\le t+d} Pr\Big(A^{(k)}(t) \ge \inf_{s\le u_1\le\cdots\le u_k} U_k(s,u_k) \\
&\quad + S_{k+1}^L(u_k,t+d)\Big) \\
&\le \sum_{0\le s\le t}\sum_{s\le u_1\le\cdots\le u_k\le t+d} Pr\big(A^{(k)}(t) \ge U_k(s,u_k) + \\
&\quad S_{k+1}^L(u_k,t+d)\big) .
\end{aligned}$$

In the third line we expanded the convolution and used Boole's inequality. In the fourth line we used Eq. (4), (5), and (7). In the sixth line we used Eq. (7) again. Next we used the inductive hypothesis for $(\mathcal{S}_2)$ and Boole's inequality in the last two lines, which completes the induction for $(\mathcal{S}_1)$.

To prove $(\mathcal{S}_2)$ for $k+1$ we have

$$\begin{aligned}
&\left(A^{(k)}(s) + T_k^L \otimes S_{k+1}^L(s,u_{k+1})\right)^{\mathbf{x}_{k+1}} \\
&= \left(A^{(k)}(s) + \inf_{s\le u_k\le u_{k+1}}\left\{T_k^L(s,u_k) + S_{k+1}^L(u_k,u_{k+1})\right\}\right)^{\mathbf{x}_{k+1}} \\
&= \inf_{s\le u_k\le u_{k+1}}\left(A^{(k)}(s) + T_k^L(s,u_k) + S_{k+1}^L(u_k,u_{k+1})\right)^{\mathbf{x}_{k+1}} \\
&= \inf_{s\le u_k\le u_{k+1}}\Bigg(\sum_{i=1}^{m_s^{(k-1)}} l_{k-1,i}X_{k,i} + \sum_{i=m_s^{(k-1)}}^{N_{u_k}} l_{k-1,i}X_{k,i} \\
&\quad + S_{k+1}^L(u_k,u_{k+1})\Bigg)^{\mathbf{x}_{k+1}} \\
&= \inf_{s\le u_k\le u_{k+1}}\left(\sum_{i=1}^{N_{u_k}} l_{k-1,i}X_{k,i} + S_{k+1}^L(u_k,u_{k+1})\right)^{\mathbf{x}_{k+1}} \\
&= \inf_{s\le u_k\le u_{k+1}}\Bigg(\left(A^{(k-1)}(s) + T_{k-1}^L \otimes S_k^L(s,u_k)\right)^{\mathbf{x}_k} \\
&\quad + S_{k+1}^L(u_k,u_{k+1})\Bigg)^{\mathbf{x}_{k+1}} \\
&= \inf_{s\le u_k\le u_{k+1}}\left(\inf_{s\le u_1\le\cdots\le u_k} U_k(s,u_k) + S_{k+1}^L(u_k,u_{k+1})\right)^{\mathbf{x}_{k+1}} \\
&= \inf_{s\le u_1\le\cdots\le u_{k+1}} U_{k+1}(s,u_{k+1}) .
\end{aligned}$$

In the sixth line we used Eq. (7). In the seventh line we used the induction hypothesis. In the last line we used the definition of U_k.

Next, we use the statement $(\mathcal{S}_1)$ to compute the end-to-end delay bound on $W_n(t)$ for $k = n$. We have

$$\begin{aligned}
&Pr(W_n(t) > d) \\
&\le \sum_{0\le s\le t}\sum_{s\le u_1\le\cdots\le u_{n-1}\le t+d} Pr\Bigg(A^{(n-1)}(t) > \Big(\cdots \\
&\quad \big((A(s) + S_1^L(s,u_1))^{\mathbf{x}_1} + S_2^L(u_1,u_2)\big)^{\mathbf{x}_2} + \cdots \\
&\quad + S_{n-1}^L(u_{n-2},u_{n-1})\Big)^{\mathbf{x}_{n-1}} + S_n^L(u_{n-1},t+d)\Bigg) \\
&\le \sum_{0\le s\le t}\sum_{s\le u_1\le\cdots\le u_{n-1}\le t+d} Pr\Bigg(\Big(\cdots\big((A(t-s) - \\
&\quad S_1^L(s,u_1))^{\mathbf{x}_1} - S_2^L(u_1,u_2)\big)^{\mathbf{x}_2} - \cdots - \\
&\quad S_{n-1}^L(u_{n-2},u_{n-1})\Big)^{\mathbf{x}_{n-1}} > S_n^L(u_{n-1},t+d)\Bigg) \\
&\le \sum_{0\le s\le t}\sum_{s\le u_1\le\cdots\le u_{n-1}\le t+d} e^{-\theta_n S_n^L(u_{n-1},t+d)} .
\end{aligned}$$

$$
\begin{aligned}
& E\Bigg[e^{\theta_n \Big(\cdots \Big((A(t-s) - S_1^L(s,u_1))^{\mathbf{x}_1} - S_2^L(u_1,u_2) \Big)^{\mathbf{x}_2} - } \\
& \qquad \cdots - S_{n-1}^L(u_{n-2},u_{n-1}) \Big)^{\mathbf{x}_{n-1}} \Bigg] \\
\leq & \sum_{0 \leq s \leq t} \sum_{s \leq u_1 \leq \cdots \leq u_{n-1} \leq t+d} e^{-\theta_n S_n^L(u_{n-1},t+d)} \\
& e^{-\theta_{n-1} S_{n-1}^L(u_{n-2},u_{n-1})} \cdots e^{-\theta_1 S_1^L(s,u_1)} e^{\theta_1 r_A(\theta_1)(s,t)} \ .
\end{aligned}
$$

In the second line we expanded the recursion in the statement ($\mathcal{S}_1$). In the third line we repeatedly applied the stationarity bound from Lemma 2. In the fourth line we used Chernoff's Bound for some $\theta_n > 0$. In the fifth line we recursively applied Lemma 3. To do so, we let $\theta_i R_{i-1}(\theta_i) = \log M_l(\theta_{i-1})$, which is already stated in Lemma 3, $R_{i-1}(\theta_i) = \frac{1}{\theta_i} \log M_{\mathbf{X}_{i-1}}(\log M_l(\theta_i))$. Here, all S_i^L's are packetized dynamic servers in the form of Eq. (2). Note that, if we let

$$
b = \sup \left\{ e^{-\theta_n C_n}, e^{-\theta_{n-1} C_{n-1}}, \ldots, e^{-\theta_1 C_1} \right\} \ , \qquad (8)
$$

we have

$$
\begin{aligned}
& Pr(W_n(t) > d) \\
\leq & \sum_{0 \leq s \leq t} e^{d \cdot \log b} e^{\sum_{i=1}^n \theta_i l_{max}} e^{(\log b + \theta_1 r_A(\theta_1))(t-s)} \\
& \qquad \cdot \sum_{s \leq u_1 \leq \cdots \leq u_{n-1} \leq t+d} 1 \\
\leq & \ b^d e^{\sum_{i=1}^n \theta_i l_{max}} K^n \ .
\end{aligned}
$$

Here we let $K = \frac{\left(1+\frac{d}{n}\right)^{1+\frac{d}{n}}}{\left(\frac{d}{n}\right)^{\frac{d}{n}}}$ and used $\log b + \theta_1 r_A(\theta_1) < 0$ as the stability condition. Taking $t \to \infty$ proves the result. We used the same argument as in [6] for the last step of computation. Finally, the order of growth of the ε-quantiles for $0 < \varepsilon < 1$ follows directly as $\mathcal{O}(n)$. □

4. NUMERICAL EVALUATION

To evaluate the analytical results, we use the following numerical example settings. First, we let the packet sizes be discrete uniformly distributed *i.i.d. r.v.*'s, $l \sim U[a,b]$. Thus, we know $M_l(\theta) = \frac{e^{a\theta} - e^{(b+1)\theta}}{(b-a+1)\left(1-e^{\theta}\right)}$. Let $a = 1, b = 16$ for illustration. Clearly, $l_{max} = 16$. Next, we use the Bernoulli process as the scaling process - $\mathbf{X} \sim B(p)$, where p represents the data through probability, so that we know $R(\theta) = \frac{1}{\theta} \log(1 - p + pM_l(\theta))$. Further we assume that all servers are work-conserving with constant bit rate C_i. Next, we first compare the delay bounds from Section 3.1 with those from Section 3.2 ($\rightarrow$ Theorem 1) and also validate them against simulation results. Then we evaluate our main result from Theorem 1 changing the scaling parameters.

For the first comparison we assume that the arrivals are a compound process instead of being packetized by a packetizer before being served. Note, our results in Theorem 1 also imply this case, since the MGF bound of the arrival process that the theorem requires can be given directly. Without loss of applicability in real-world, we assume $A(t)$ is a compound Poisson process, so that $r_A(\theta) = \frac{1}{\theta}\lambda(M_l(\theta) - 1)$. The average rate of the Poisson process $N(t)$ is normalized to one data unit (bit) per one time unit, i.e., $\lambda = 1$. The number of the scaling elements varies from 1 to 9, which means maximal 10 servers. We assume the utilization of the first server is 0.8, so $C_1 = 1.25$. To choose $C_2, \ldots, C_{10}$, we refer to Eq. (8). Avoiding that some server becomes the bottleneck, we can let all the terms in Eq. (8) be equal, i.e., $\theta_i C_i = \theta_{i-1} C_{i-1}, 2 \leq i \leq n$, where θ_i's are implied in Lemma 3. This is actually a criterion to assign the service capacities along the path a flow traverses. It must not be so strict, or in other words, the service capacities in practice may already be set before we know the other network settings. So here, for simplicity, we just statically set the capacities as $C_2 \ldots C_{10} = [1.15, 1.05, 0.95, 0.85, 0.80, 0.75, 0.70, 0.65, 0.60]$. The quantile ε is set to 10^{-3}. We use Omnet++ to do the simulations. We measure 10^6 packet delays at the destination node and use the empirical quantile from these for the simulation results. This will increase the result accuracy so that we ignore the confidence intervals.

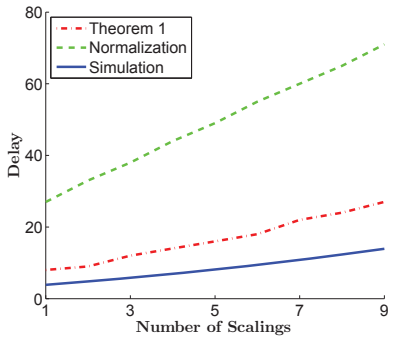

Figure 5: Delay bounds with Theorem 1, "normalized" flow, and simulation.

Figure 5 shows the bounds on the 10^{-3}-quantiles of the delay. The plot shows the $\mathcal{O}(n)$ order of growth. We observe that the results from Theorem 1 are much closer to the simulation results than the results from analyzing the normalized flow. The mathematical reason is that, although with both methods we used the maximum packet size l_{max}, in Theorem 1 we used the form of $[C_i \cdot t - l_{max}]_+$, while for the normalization we used the form of $C_i / l_{max} \cdot t$. Obviously, the loss in precision caused by the division is higher than for subtraction. The gap to the simulation results implies that the tightness still can be improved. Yet, as this work is the first attempt to model the variable length packet flow transformation, we focused on the expression of such a network scenario and provided the first insights calculate delay bounds in this setting. The key to improve on the tightness will be to make smarter usage of the packet length distribution, than just resorting to l_{max}. On the other hand, as you can also see in [11, 3], it can circumvent several technical difficulties, otherwise we would have to consider the inherent correlations among arrivals, services and packet scaling elements, which is, however, as we discussed in previous sections or in [12, 8], very difficult even in the single node case without flow transformations. Furthermore, the usage of Boole's inequality could be improved by the construction of a martingale as in [13]. Yet, again this is, so far only possible for the single node case. So, we leave this for future work.

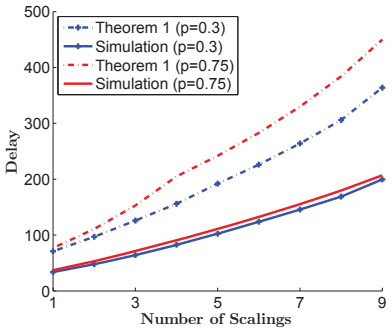

Figure 6: Delay bounds with Theorem 1 and the simulation.

For the second comparison we slightly change the arrival description. Frequently we only know the statistical properties of the bit flow and that the bits are packetized. The result from Theorem 1 can also deal with this. So we use a bit flow followed by a packetizer as the arrival for the server. Assume that the original arrival flow of bits is a Poisson process $Poi(\lambda)$. Then we know $r_A(\theta)(s,t) \leq \frac{\lambda(e^\theta-1)}{\theta}(t-s)+l_{max}$. The other numerical settings we use the same as before.

Figure 6 shows the bounds on the 10^{-3}-quantiles of the delay under varying scaling parameters. We can see that Theorem 1 increases with the through probability p. That means if more of the flow is kept during the transformation, the higher the burstiness at the next server node will become. Interestingly, the gap between those curves from the theorem is larger than that of the simulation results. The reason is that we use l_{max}/C as the extra latency for each packet after being served by the packetized server, while actually most packets have a much smaller latency increase. This treatment enlarges the sensitivity of the results, because the more the flow passes through, the more tightness we lose.

5. CONCLUSION

In this paper, we extended network calculus to model networks with variable length packet flow transformations. The main contribution is the definition of a scaling element that works on the packet level (rather than the bit level). This facilitates a commutation of the service element with the scaling element on the packet level, and thus preserves the convolution-form expression of this kind of networks. Based on this we derived the end-to-end delay bounds. We also discussed another method, which is a direct extension of a previous model by normalizing the bit flow and the bit-wise service with the packet sizes, as if the flow was treated as a flow with identical data units and the service rate was in $packets/s$. We evaluated both methods and validated them against simulations. We found that the method based on the new packet scaling element is much closer to the simulation results than the other one. However, we also point out that improving the tightness is still a challenge for future work. We hope to achieve this by finding a more precise expression for the dynamic server of the packetized service.

6. 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 of deterministic and stochastic queueing networks. *IEEE Transactions on Automatic Control*, 39(5):913–931, May 1994.

[3] C.-S. Chang. *Performance Guarantees in Communication Networks*. 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, Part I and II. *IEEE Transactions on Information Theory*, 37(1):114–141, January 1991.

[6] M. Fidler. An end-to-end probabilistic network calculus with moment generating functions. In *Proceedings of IEEE IWQoS*, pages 261–270, June 2006.

[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] Y. Jiang. Stochastic service curve and delay bound analysis: A single node case. In *Proceedings of the 25th International Teletraffic Congress (ITC 25)*, September 2013.

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

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

[11] 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.

[12] J. Liebeherr, A. Burchard, and F. Ciucu. Delay bounds in communication networks with heavy-tailed and self-similar traffic. *IEEE Transactions on Information Theory*, 58(2):1010–1024, February 2012.

[13] F. Poloczek and F. Ciucu. Scheduling analysis with martingales. *Performance Evaluation*, 79:56–72, September 2014.

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

[15] 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.

[16] H. Wang, F. Ciucu, and J. Schmitt. A leftover service curve approach to analyze demultiplexing in queueing networks. In *Proceedings of VALUETOOLS*, pages 168–177, October 2012.

[17] H. Wang and J. Schmitt. Delay bounds calculus for variable length packet transmissions under flow transformations. Technical Report 390/14, University of Kaiserslautern, Germany, November 2014.