On the Tightness of an Optimization-Based Delay Bounding Method for Packet-Oriented Networks under Blind Multiplexing

Huan Yang and Liang Cheng

May 2015

Abstract

Communication networks supporting real-time tasks need to guarantee timely delivery of critical data and commands. Worst-case delay performance of these networks can be analyzed using deterministic network calculus. In this report, we identify the discrepancy between existing bit-oriented delay bounding methods and packet-oriented behaviors of network devices. An optimization-based bit-oriented delay bounding method is then augmented for the more realistic packet-oriented settings. An example network is used to demonstrate how to formulate a packet-oriented delay bounding problem using theorems for packet-based network devices. Finally, we show that the delay bounds obtained for the example packet-oriented network are theoretically tight under all possible network configurations.

1 Introduction

As computer networks become increasingly pervasive, mission-critical applications, such as industrial control systems [Weiner et al., 2014; Shen et al., 2014], rely on packet-oriented networks for timely delivery of data and commands. Many of these applications are delay-sensitive or delay-intolerant [Suriyachai et al., 2012], requiring varying degrees of delay performance. Analytical delay bounding methods [Schmitt et al., 2008] based on deterministic network calculus [Cruz, 1991a] are developed to find the maximum delay experienced by data bits of any traffic flow of interest. Although existing delay bounding methods [Schmitt et al., 2007, 2008] allow us to find the worst-case delay for any flow of interest in complex feed-forward networks, all these methods assume a fluid model, the minimum indivisible unit of which is an individual data bit. However, computer networks typically organize data bits into packets. Data bits in the same packet should experience the same delay. Obviously, existing methods based on the fluid model does not fully capture the characteristics of packet-oriented networks.

In this report, we first identify packet-oriented behaviors of computer networks that are not characterized by the fluid model. Then, we augment an optimization-based bit-oriented delay bounding method proposed in [Schmitt et al., 2008] to model packet-oriented networks and derive tight worst-case delay experienced by packets, instead of data bits, of any flow of interest. Finally, we analyze a representative feed-forward network using our augmented optimization-based delay bounding method and prove that the delay bounds obtained in all cases are indeed tight. Our work extends the existing method to more realistic, packet-oriented settings so that impacts of packet sizes on worst-case delay can be correctly taken into account. Our tightness proof covers all possible cases and demonstrates how to interpret delay bounding results under the set of assumptions implicitly exploited by algorithms based on deterministic network calculus.

1.1 Bit-Oriented Delay Bounding Under Blind Multiplexing

In [Cruz, 1991a,b], basic theorems of deterministic network calculus are formally proved and then applied to simple scenarios. Scheduling behaviors of network devices (i.e., the order in which packets from different incoming flows are buffered by a network node for output) are investigated in detail. In [Bouillard and Junier, 2011], fixed-priority (FP) scheduling behaviors are studied. In [Parekh and Gallager, 1993, 1994], generalized processor sharing (GPS), which allows active links to dynamically utilize the bandwidth reserved for currently inactive links, is modeled using network calculus. In [Sariowan et al., 1999], the authors devise an earliest deadline first (EDF) algorithm that uses the service curves of nodes and facilitates real-time implementations.

We note that research on scheduling behaviors assumes that these algorithms can be implemented exactly, which is not the case for energy or resource constrained applications [AlEnawy and Aydin, 2005]. In [Schmitt et al., 2008], the authors assumes blind multiplexing, which allows a network node to occasionally exhibit behaviors that are inconsistent with the scheduling strategy it implements. In particular, blind multiplexing, sometimes termed arbitrary multiplexing, includes two assumptions. One assumption, namely the per-flow first-come first-served (FCFS) assumption, requires that the ordering of data bits within each flow be preserved by every network node. This assumption is also used in previous work on exact scheduling. The other assumption, which is the characteristic of blind multiplexing problems, allows inconsistent scheduling behaviors due to resource constraints or program bugs. In the next subsection, we give basic concepts, definitions, and theorems that enable bit-oriented delay bounding under blind multiplexing.

1.2 Network Calculus Basics

Based on min-plus algebra [Baccelli et al., 1992], deterministic network calculus serves as a theoretical tool for finding upper bounds on delay and queue length. A detailed treatment of network calculus can be found in [Le Boudec and Thiran, 2001; Chang, 2000]. In this subsection, we briefly introduced fundamental concepts and theorems upon which both bit-oriented and our augmented delay bounding methods are built. It should be noted that our approach also utilizes another model to capture packet-based behaviors, the incorporation of which is one of our contributions and is introduced in detail in later sections.

To characterize network traffic flows, we define for each input/output flow a *cumulative arrival* or *departure function*. The set of wide-sense increasing, real-valued, and non-negative functions starting from the origin is defined as

$$\mathcal{F}_0 = \{f : \mathbb{R}^+ \to \mathbb{R}^+ : \forall\, t \geq s : f(t) \geq f(s), f(0) = 0\}.$$

In network calculus, a cumulative function $F(t)$ is used to count the total number of bits generated by a flow starting from $t=0$ up to time instant t. Suppose $A(t)$ and $B(t)$ denote the input and output traffic flows of a node, respectively. $A(t) \in \mathcal{F}_0$, which is the *cumulative input/arrival function*, counts the total number of bits fed to a node starting from $t = 0$. Similarly, $B(t) \in \mathcal{F}_0$ is the *cumulative output/departure function* that counts the total number of bits that have left a node starting from $t=0$. The *virtual delay* experienced by the last bit in $A(t)$ that arrives at time t is represented by $d(t)$ and defined as

$$d(t) = \inf\{\tau \geq 0 : B(t+\tau) \geq A(t)\}.$$

The *backlog* of $A(t)$ at a node at time t is denoted by $q(t)$ and defined as $q(t) = B(t) - A(t)$.

For two sequences $f(t) \in \mathcal{F}_0$ and $g(t) \in \mathcal{F}_0$, basic operations of min-plus algebra are defined as follows

$$(f \wedge g)(t) = \min\{f(t), g(t)\}$$

$$(f \otimes g)(t) = \min_{0 \leq s \leq t} \{f(t-s) + g(s)\}$$

$$(f \oslash g)(t) = \sup_{s \geq 0}\{f(t+s) - g(s)\}$$

Algebraic characteristics of these operations, namely *pointwise minimum* denoted by $\wedge$, *min-plus convolution* denoted by $\otimes$, and *deconvolution* denoted by $\oslash$, can be found in [Chang, 2000; Le Boudec and Thiran, 2001; Baccelli et al., 1992].

It should be noted that another type of functions is defined based on the notion of cumulative functions and is more frequently used in network calculus. A function $\alpha(t) \in \mathcal{F}_0$ is an *arrival curve* for input function $A(t)$ iff

$$\forall t \geq s \geq 0 : A(t) - A(s) \leq \alpha(t-s) \Leftrightarrow A(t) = A(t) \otimes \alpha(t).$$

An example of an arrival curve is the affine arrival curve $\alpha(t) = \rho t + \sigma$, which corresponds to leaky-bucket traffic regulation [Valaee, 2001]. We will use affine arrival curves to characterize cumulative input and output functions, where ρ represents the *average rate* of a flow and σ is its *burstiness* component. We note that arrival curves are used to characterize both incoming and outgoing data flows (i.e., we do not need to define "departure curves").

To describe the processing capacity of a node, service curve and strict service curve are introduced. For a given input cumulative function $A(t)$, suppose the service provided by a node results in an output cumulative function $B(t)$. The service provided by this node can then be characterized by a *service curve* $\beta(t)$ iff

$$B(t) \geq A(t) \otimes \beta(t).$$

The service curve of a node can be regarded as its system function in min-plus algebra. In contrast, a strict service curve characterizes the processing capacity of a node in terms of its minimum output capability during busy periods. Suppose that $B(u)$ denotes the cumulative output from a node within its busy period of length u. Note that the buffer of the node is not empty throughout its busy period. The output capability of this node can be characterized by a strict service curve $\beta(u)$ iff

$$\forall u > 0 : B(u) \geq \beta(u).$$

Note that strict service curve is a special type of service curve. We use strict service curve to formulate packet-oriented delay bounding problems because our knowledge of a network device can be derived from external observations.

In this report, we focus on a particular type of service curve $\beta(t) = R[t-T]^+$, which is known as the *rate-latency service curve*, where the operator $[\cdot]^+$ is defined as $[x]^+ = \max\{x, 0\}$. In network calculus, arrival and service curves are essential concepts that enable us to determine the departing traffic, which is characterized by its corresponding arrival curve. A comprehensive survey of service curves can be found in [Fidler, 2010].

Next, we introduce three important theorems that enable us to analyze bit-oriented feed-forward networks under blind multiplexing. Given the arrival curve of the input traffic flow fed to a node and the service curve it provides, the following theorem gives tight bounds on virtual delay, backlog, and the output arrival curve.

Theorem 1. *Suppose a node provides service curve $\beta(t)$ for input cumulative function $A(t)$ that is characterized by arrival curve $\alpha(t)$. Let $B(t)$ represent the output cumulative function which is characterized by arrival curve $\alpha'(t)$. We have the following performance bounds:*

Backlog: $q(t) = \max_{0 \leq s \leq t}\{\alpha(s) - \beta(s)\} \equiv v(\alpha(t), \beta(t))$.

Delay: $d(t) = \inf\{\tau \geq 0 : (\alpha \oslash \beta)(-\tau) \leq 0\} \equiv h(\alpha(t), \beta(t))$.

Output arrival curve: $\alpha'(t) = \alpha(t) \oslash \beta(t)$.

The worst-case delay corresponds to the maximum "horizontal" distance between the input arrival curve and the service curve (i.e., $h(\alpha(t), \beta(t))$), whereas the worst-case backlog is given by the

"vertical" distance between them (i.e., $v(\alpha(t), \beta(t))$). Note that Theorem 1 only allows a single input flow. In the case where $A(t)$ represents the aggregation of several input flows, this theorem implicitly enforces an FCFS policy on the serving node. As a result, we cannot directly apply this theorem to the aggregation of multiple flows when blind multiplexing is assumed.

Furthermore, we say that a node offers a *strict service curve* $\beta(t)$ to an input traffic flow $A(t)$ if the corresponding output traffic flow is lower bounded by $\beta(u)$, where u is the length of a *backlogged period*. In [Schmitt et al., 2008], strict service curves are used as they allow to bound the maximum backlogged period of a node. In this case, the delay bound is denoted as $\overline{d}$, which is the non-zero intersection point between arrival curve $\alpha(t)$ and strict service curve $\beta(t)$ (i.e., $\alpha(\overline{d}) = \beta(\overline{d})$). Note that this gives us the worst-case delay when the per-flow FCFS order is not preserved (i.e., it is applicable to the aggregation of multiple flows when blind multiplexing is assumed).

To analyze feed-forward networks, the following concatenation theorem tells us how to model nodes connected in tandem given their respective service curves.

Theorem 2. *Suppose two nodes are connected in tandem. The preceding node offers a service curve $\beta_1(t)$ for input traffic flow $A(t)$. The succeeding node offers a service curve $\beta_2(t)$ for the output traffic flow $B(t)$ of the preceding node. Equivalently, the concatenation of the two nodes offers a service curve $\beta(t)$ for $A(t)$, where $\beta(t) = \beta_1(t) \otimes \beta_2(t)$.*

As shown in [Schmitt et al., 2007, 2008], concatenating service curves of multiple nodes, though simplifying the process of computations, results in loose delay bounds when blind multiplexing is assumed. Moreover, we note that service curve provided by a node may be specific to a certain input flow [Chang, 2000]. In other words, a network device may provide different services if the input flow changes. In this report, we assume that all service curves are *universal*, i.e., every node provides consistent services regardless of the contents of the input.

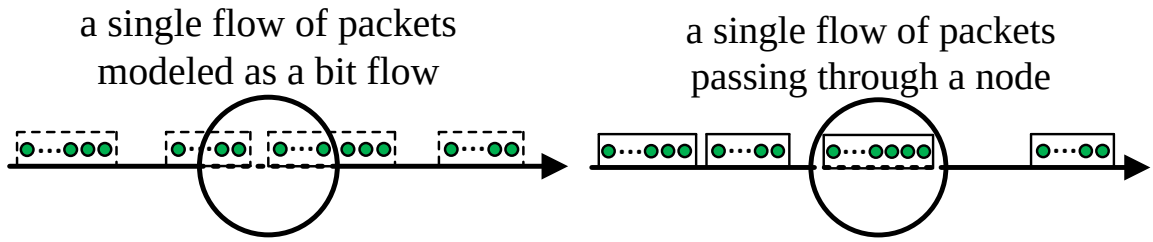
At a certain network node, the traffic flow of interest may share the available processing capacity with other traffic flows that are also fed to the same node. The following theorem introduces the concept of a *left-over service curve*, which effectively characterizes the processing power solely enjoyed by the traffic flow of interest as if the other interfering flows are absent.

Theorem 3. *Consider a node with a service curve $\beta(t)$ and two input cumulative functions respectively characterized by arrival curves $\alpha_1(t)$ and $\alpha_2(t)$. Assume per-flow FCFS ordering of data bits is preserved at this node, the left-over service curve for arrival curve $\alpha_1(t)$ is $\beta_1(t) = [\beta(t) - \alpha_2(t)]^+$.*

Note that Theorem 3 also applies to strict service curves. It has been demonstrated in [Schmitt et al., 2008] that the above definitions and theorems facilitate the derivation of delay performance bounds for wireless sensor networks. However, the use of a fluid model does not capture the essentially non-preemptive packet-oriented processing behavior of network devices. In the next subsection, we identify packet-oriented behaviors of network devices that are not modeled by Theorems 1, 2, and 3, which motivates us to investigate the packet-oriented delay bounding problem under arbitrary multiplexing.

1.3 Packet-Oriented Behaviors Of Network Devices

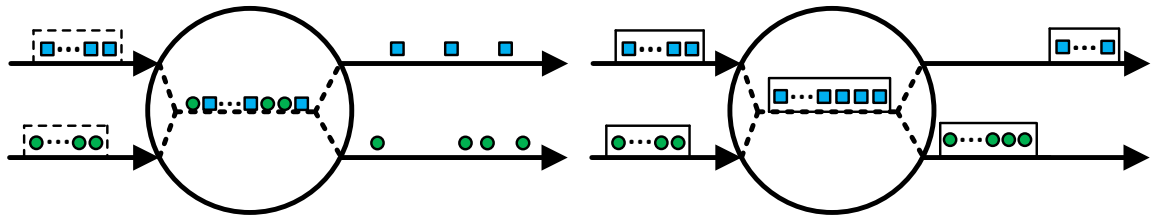
As shown in Figure 1, previously proposed bit-oriented approach does not preserve packet-level meta-structure of data flows. Specifically, a network node that assumes a fluid model may incorrectly introduce separation between data bits of the same packet. As shown in Figure 1b, packet-oriented network devices typically process packets in their entirety. In contrast, Figure 1a illustrates how a packet flow is modeled as a bit stream. We note that this approach is applicable in wireless sensor network as all packets have the same size. However, when network devices process variable-length packets, the fluid model becomes inappropriate. Regardless of the size of a packet, a packet-oriented device takes an entire packet as input.



(a) A single flow of packets is treated as a bit stream, with individual data bit as the unit of processing.

(b) Data bits are organized and processed in a packet-by-packet fashion.

Figure 1: Fluid model does not preserve packet-based meta-structure of network data flows even when there is only a single flow.



(a) When two data flows of packets merge at the same node, bit-oriented approaches allow bits from both flows to enter the node simultaneously.

(b) When data packets are processed one after another, all the bits of the same packet collectively enter and exit the node.

Figure 2: Fluid model allows data bits from different flows to enter a network device at the same time.

On the other hand, data flows interfere with each other at the packet-level granularity. As shown in Figure 2a, the fluid model does not take packet sizes into consideration and hence data bits from different flows may enter the node simultaneously, distorting the shapes of packets. Specifically, separation between bits from the same packet flow is also affected by the presence of other flows. In Figure 2b, we observe that packets enter and leave a node in their entirety, excluding the interference from any data bits of other packets. Put differently, the interference of other flows should be modeled at the packet-level instead of bit-level granularity. Suppose a packet from the flow of interest arrives at the node while it is busy processing a packet from another flow. Even if our flow of interest has the highest priority, the high-priority packet has to wait until the low-priority one passes through. In [Chang, 2000; Le Boudec and Thiran, 2001], the authors point out that packet-oriented networks that implement certain scheduling policy exactly are non-preemptive at the bit level but preemptive at the packet level.

In this report, our proposed approach can correctly model both packet-oriented behaviors that are not captured by the fluid model. For the convenience of illustration, we focus on the phenomenon shown in Figure 1, which is also justified by the set of assumptions described in later sections.

2 Modeling Packet-Oriented Behaviors Using L-Packetizers

In this section, we introduce an L-packetizer [Le Boudec, 2002] into the network-calculus-based delay bounding framework proposed in [Schmitt et al., 2008]. Additionally, we summarize important assumptions used in [Chang, 2000; Schmitt et al., 2008; Le Boudec, 2002], which helps clarify ambiguities in our augmented algorithm and its tightness proof. Both the problem formulation process as well as the tightness proof make use of these assumptions.

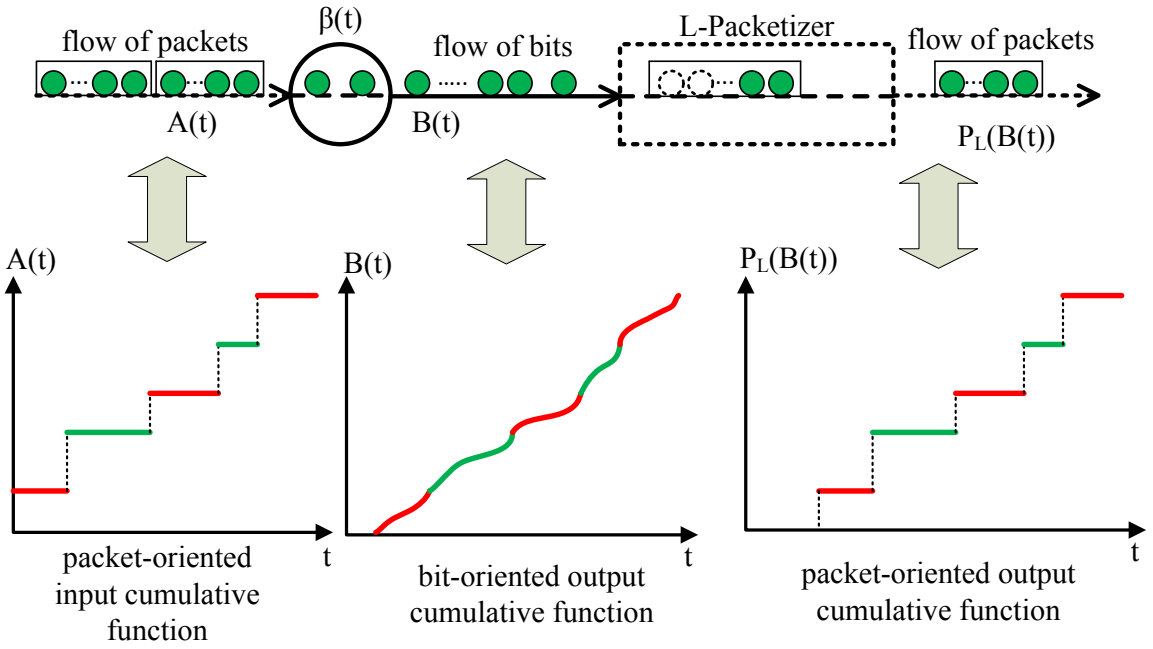


Figure 3: The insertion of an L-packetizer allows us to model a flow of packets passing through a node offering service curve $\beta(t)$ and exhibiting packet-oriented behaviors. Data bits have to wait in the L-packetizer until a packet is formed.

2.1 L-Packetizer and Packet-Oriented Theorems

In our packet-oriented model, a cumulative function $A(t)$ still counts the total number of bits that are fed to a node up to time t. In addition, we use $l(n)$ to denote the length of the n^{th} packet from $A(t)$ measured in bits. After n packets of $A(t)$ are fed to a node, we denote the total number of bits fed to the node when the n^{th} packet is present as $L(n) = \Sigma_{m=1}^{n} l(m)$. Then, we define $L^{-1}(r) = \sup\{n : L(n) \leq r\}$, where r is the cumulative number of bits. We further define the sequence $P_L(r) = L(L^{-1}(r))$, which gives the total number of bits that can be put into packets of $A(t)$. Note that $P_L(A(t)) \leq A(t)$ as certain bits may need to wait for the arrival of future bits in order to form a new packet. An *L-packetizer* [Le Boudec, 2002; Chang, 2000; Le Boudec and Thiran, 2001] can be defined as follows:

Definition 1. *A network element with input $A(t)$ and output $B(t)$ is called an L-packetizer if $B(t) = P_L(A(t))$ for all $t \geq 0$. $A(t)$ is said to be L-packetized if $A(t)$ is not affected by the L-packetizer, i.e., $A(t) = P_L(A(t))$ for all $t \geq 0$.*

Using Definition 1, Theorems 1, 2, and 3 can be modified to take packet sizes into consideration. The basic idea behind these modifications is illustrated by Figure 3: Suppose a flow of packets or a packetized input function $A(t)$ is fed to a node with service curve $\beta(t)$. The output $B(t)$ is bit-oriented. Specifically, the node preserves the per-flow FCFS ordering of data bits but $B(t)$ is no longer packetized. To restore the original packet-based meta-structure of $A(t)$, information on packet sizes is encoded into an L-packetizer. By attaching the L-packetizer to the node, we obtain $P_L(B(t))$ which is a packetized version of $B(t)$. In other words, the L-packetizer reassembles the output flow of bits into packets according to information on packet sizes of $A(t)$. It should be noted that L-packetizers are input-specific. For example, two traffic flows with variable packet lengths typically require two different L-packetizers, each of which encodes information on packet sizes of a particular traffic flow.

To include the effect of packet sizes, Theorem 1 needs to be modified as follows:

Theorem 4. *Suppose a flow of packets $A(t)$ is fed to a node with service curve $\beta(t) = Rt$ followed by an L-packetizer, and $l_{\max}$ is the maximum packet length of $A(t)$. We can obtain the following performance bounds for the packetized output $P_L(B(t))$:*

Backlog: $q_L(t) = v(\alpha(t), \beta(t)) + l_{\max}$.

Delay: $d_L(t) = h(\alpha(t), \beta(t)) + l_{\max}/R$.

The increases in worst-case backlog and delay are necessary when packets instead of data bits are fed to a node. Suppose a data bit x in $A(t)$ experiences the worst-case delay given by Theorem 1. Theorem 4 captures the corresponding worst-case scenario in the L-packetized model: Suppose the data bit immediately preceding bit x is the last bit of a packet of $A(t)$ and that bit x belongs to one of the longest packet of $A(t)$. Obviously, bit x has to wait until enough data bits of $A(t)$ have passed through the node. Similar to Theorem 1, FCFS is assumed when applying Theorem 4, which makes it inapplicable to the aggregation of multiple flows under blind multiplexing. We will use this theorem when there is only one flow left for analysis.

When an input traffic flow $A(t)$ passes through multiple nodes in tandem, we need to append an L-packetizer to each node on the path traveled by $A(t)$. This is necessary because network nodes still assume fluid model and their output are essentially bit streams. The following theorem gives us the service curve of the concatenation of a node with service curve $\beta(t)$ and an L-packetizer for $A(t)$:

Theorem 5. *Suppose an input cumulative function $A(t)$ with an arrival curve $\alpha(t)$ passes through a node with service curve $\beta(t)$ and L-packetized output $P_L(B(t))$, as illustrated by Figure 3. The service curve offered by the concatenation of the node and the L-packetizer is denoted by $\beta_L(t)$ and defined as $\beta_L(t) = [\beta(t) - l_{\max}]^+$.*

Combined with Theorem 2, Theorem 5 allows us to derive the service curve offered by two nodes connected in tandem when the nodes process data bits in a packet-by-packet fashion. Specifically, suppose the respective service curves offered by the two nodes are $\beta_1(t)$ and $\beta_2(t)$, the concatenation of the two nodes with packet-oriented behaviors offers a service curve $\beta_L(t) = [\beta_1(t) - l_{\max}]^+ \otimes [\beta_2(t) - l_{\max}]^+$. Note that we need to append an L-packetizer each time the traffic flow passes through a node because each node still uses the fluid model and exhibits behaviors described by Theorem 1. It may be tempting to simply attach an L-packetizer to the end of the concatenation of the two nodes. However, this approach will lead to bits, instead of packets, flowing between the two nodes, which breaks the packet-oriented behaviors of the whole system. In addition, we are now able to derive the output arrival curve for input $A(t)$ using Theorem 1, which gives $\alpha'_L(t) = \alpha(t) \oslash [\beta(t) - l_{\max}]^+$.

Finally, consider the scenario where multiple traffic flows are fed to a node. Theorem 3 needs to be modified so as to take packet sizes into account. The following theorem gives the left-over service curve when data flows are packet-oriented.

Theorem 6. *Suppose two traffic flows, $A_1(t)$ and $A_2(t)$, are both fed to a node with service curve $\beta(t)$. Assume the arrival curves for $A_1(t)$ and $A_2(t)$ are $\alpha_1(t)$ and $\alpha_2(t)$, respectively. Suppose the node preserves the sizes of packets of both $A_1(t)$ and $A_2(t)$, then the left-over service curve for $A_1(t)$ is $\beta_{L1}(t) = [\beta(t) - \alpha_2(t) - l_{2,\max} - l_{1,\max}]^+$, where $l_{1,\max}$ and $l_{2,\max}$ are the maximum packet lengths of $A_1(t)$ and $A_2(t)$, respectively.*

The worst-case scenario captured by Theorem 6 is when the longest packets of both traffic flows arrive at a node at the same time. Suppose the first bit x_1 of a packet with size $l_{1,\max}$ from $A_1(t)$ arrives at a node that has just started processing the first bit x_2 of a packet with size $l_{2,\max}$ from $A_2(t)$. As the node processes data packet by packet, bit x_1 has to wait until $l_{2,\max}$ bits from $A_2(t)$ have passed through the node. In addition, bit x_1 also needs to wait until all $l_{1,\max}$ bits of the containing packet have been processed. Note that Theorem 6 assumes that no priority is assigned to either traffic flow, but the worst case for the flow of interest is when all other flows have higher priorities.

We note that Theorem 6 models the scenario where multiple flows may interfere with each other: When a node starts processing a packet from a low-priority flow, packets from high-priority flows that arrive before the end of the processing must wait. An alternative approach is to attach an L-packetizer for the aggregated input (i.e., $A_1(t) + A_2(t)$). In this case, the left-over service curve for $A_1(t)$ becomes $[\beta(t) - \alpha_2(t) - l_{\max}]^+$, where $l_{\max}$ is the maximum packet length among all packets of both $A_1(t)$ and $A_2(t)$. However, it should be noted that this approach implicitly assumes that input flows can be organized into some order and packets from different flows does not interfere with each other. This assumption is further discussed in the next subsection. As mentioned in Section 1.3, we focus on the packet-oriented behavior illustrated by Figure 1, which is also justified in part by this assumptions.

2.2 Assumptions Of Our Augmented Delay Bounding Method

As we utilize theorems from both Sections 1.2 and 2.1, the algorithm we introduce inherits the assumptions and hence limitations of both fluid and packet-oriented models. To facilitate our discussion on the L-packetized optimization-based delay bounding method in later sections, we introduce and briefly discuss major assumptions that allow us to derive packet-oriented delay bounds.

- **Merging Multiple Flows.** In [Schmitt et al., 2007, 2008], multiple interfering bit flows joining the flow of interest are merged as an aggregated interfering flow, whose arrival curve is the sum of the arrival curves of individual interfering flows. In other words, we assume that bits from different flows will not collide because data bits are indivisible and stackable. In packet-oriented settings, this assumption still holds as cumulative functions are still used for traffic characterization. Moreover, the transmission of packets that may cause collisions can be prevented by appropriate MAC protocols.
- **Blind Multiplexing and Per-Flow FCFS.** When multiple flows are fed to a node, we assume that the node will merge these flows arbitrarily before processing. The ordering of bits in each flow is preserved when it passes through a node.
- **Work-Conserving Behaviors.** Any network device modeled by the aforementioned theorems is work-conserving: When a node has a non-empty buffer and is not in vacation, it will always offer its full capacity. We note that our whole knowledge of the service capacity of a node is included in its strict service curve. Hence, a node always offers at least the capacity specified by its service curve during its backlogged period.
- **Vacations and Busy/Backlogged Periods.** A node may enter a vacation with maximum length T if its backlog is empty. In other words, a node will not enter a vacation if it is currently busy (i.e., its backlog is not empty).
- **Admission Control.** For each node, we assume that the sum of the average rates of all its incoming flows is not greater than the service rate of the node.
- **Packet-Oriented Networks.** Data generated by any node is first packed into packets before transmission. The functionality of any network node has a packet-level granularity.

Note that in later sections, we only consider the impacts of variable packet sizes of the flow of interest as illustrated in Figure 1. This is because the first assumption listed above implies that multiple flows will not collide or that we are able to put together multiple flows using a certain collision-eliminating rule. As a result, an L-packetizer for the merged flow, instead of multiple per-flow L-packetizers, is used and only $l_{\max}$ is present in Theorem 6, which is the maximum packet length among all flows of packets.

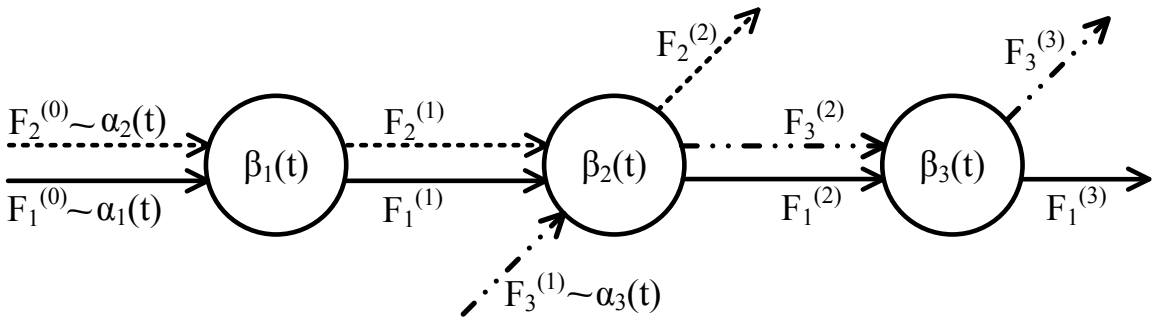


Figure 4: The sample feed-forward network shown in this figure is used in [Schmitt et al., 2008] to demonstrate a bit-oriented optimization-based delay bounding method. To facilitate comparisons between bit-oriented and our proposed packet-oriented methods, we use the same network in packet-oriented settings.

3 Optimization-Based Packet-Oriented Delay Bounding Under Blind Multiplexing

In this section, we adapt a bit-oriented optimization-based delay bounding algorithm in [Schmitt et al., 2008] for packet-oriented settings. We reiterate that our method is devised for delay performance analysis under the assumptions summarized in Section 2.2 and that our illustration in this report only addresses the impacts of variable packet sizes shown in Figure 1. It should be noted that theorems introduced in Sections 1.2 and 2.1 also enable us to model the effects depicted in Figure 2. We also note that other delay bounding methods, such as total flow analysis (TFA) and separated flow analysis (SFA) [Schmitt et al., 2007], can also be adapted using theorems introduced in Section 2.1. In this report, we focus on the derivation of theoretically tight delay bounds for packet-oriented feed-forward networks, which can only be achieved using our proposed optimization-based approach.

We introduce our augmented algorithm using the same feed-forward network in [Schmitt et al., 2008]. As shown in Figure 4, the sample feed-forward network under investigation has three nodes with strict service curves $\beta_1(t)$, $\beta_2(t)$, and $\beta_3(t)$, respectively. There are three streams of packets flowing on the network, and we denote them using cumulative functions $F_i(t)$, where $i = 1, 2, 3$. Furthermore, we use superscripts to differentiate cumulative functions for the same flow at different stages. For example, we denote one of the incoming flows for the first node as $F_1^{(0)}(t)$ and the corresponding output flow from the same node as $F_1^{(1)}(t)$. We assume that the arrival curve of an individual flow when it is first fed to the network is given. Such arrival curves may be obtained by measurements or extracted from device specifications. Specifically, the incoming flows $F_1^{(0)}(t)$, $F_2^{(0)}(t)$, and $F_3^{(1)}(t)$ have arrival curves $\alpha_1(t)$, $\alpha_2(t)$, and $\alpha_3(t)$, respectively. Our flow of interest is $F_1(t)$. The sample network depicted in Figure 4 is representative as it contains two scenarios concerning interfering flows, i.e., an interfering flow joining/leaving the path traveled by the flow of interest and an interfering flow replacing another interfering flow.

In this section, we demonstrate the formulation of an optimization problem that bounds the worst-case delay of the sample feed-forward network in packet-oriented settings. Suppose that our last observation of the network is made at time instant $t = t_3$. Furthermore, suppose that $0 \leq t_0 \leq t_1 \leq t_2$, where t_i denotes the start of the last busy period at the $(i+1)^{th}$ node immediately preceding t_{i+1}, $i = 0, 1, 2$. At node k, where $k = 1, 2, 3$, it can be derived from the definition of strict service curves that

$$F^{(k)}(t_k) - F^{(k-1)}(t_{k-1}) \geq [\beta_k(t_k - t_{k-1}) - l_{\max}^{(k-1)}]^+.$$

Note that $l_{\max}^{k-1}$ denotes the maximum packet size of the flow input to node k. As explained in

Section 1, our proposed method works for strict service curves, which allow us to characterize the processing capabilities of nodes directly. In other words, this relationship does not hold for general service curves, which characterize the processing capabilities of nodes indirectly using input cumulative functions.

Applying this relationship to the flow of interest at each node, we have

$$F_1^{(3)}(t_3) - F_1^{(2)}(t_2) \geq [\beta_3(t_3 - t_2) - l_{1,3,\max} - (F_3^{(3)}(t_3) - F_3^{(2)}(t_2))]^+. \tag{1}$$

$$F_1^{(2)}(t_2) - F_1^{(1)}(t_1) \geq [\beta_2(t_2 - t_1) - l_{1,2,3,\max} - (F_3^{(2)}(t_2) - F_3^{(1)}(t_1)) - (F^{(2)} - 2(t_2) - F_2^{(1)}(t_1))]^+. \tag{2}$$

$$F_1^{(1)}(t_1) - F_1^{(0)}(t_0) \geq [\beta_1(t_1 - t_0) - l_{1,2,\max} - (F_2^{(1)}(t_1) - F_2^{(0)}(t_0))]^+. \tag{3}$$

Note that we attach an input-specific packetizer to the end of each node and use its packet-oriented service curve. Adding Equations 1, 2, and 3, we obtain

$$\begin{aligned} F_1^{(3)}(t_3) - F_1^{(0)}(t_0) \geq & [\beta_3(t_3 - t_2) - l_{1,3,\max} - (F_3^{(3)}(t_3) - F_3^{(2)}(t_2))]^+ \\ & [\beta_2(t_2 - t_1) - l_{1,2,3,\max} - (F_3^{(2)}(t_2) - F_3^{(1)}(t_1)) - (F^{(2)} - 2(t_2) - F_2^{(1)}(t_1))]^+ \\ & [\beta_1(t_1 - t_0) - l_{1,2,\max} - (F_2^{(1)}(t_1) - F_2^{(0)}(t_0))]^+. \end{aligned} \tag{4}$$

This equation incorporates our knowledge of strict service curves for all the nodes in packet-oriented settings.

In addition, another set of constraints needs to be introduced so as to incorporate our knowledge of all the incoming flows, i.e., their respective arrival curves. In particular, we have

$$F_2^{(1)}(t_1) - F_2^{(0)}(t_0) \leq F_2^{(0)}(t_1) - F_2^{(0)}(t_0) \leq \alpha_2(t_1 - t_0). \tag{5}$$

$$F_2^{(2)}(t_2) - F_2^{(0)}(t_0) \leq F_2^{(0)}(t_2) - F_2^{(0)}(t_0) \leq \alpha_2(t_2 - t_0). \tag{6}$$

$$F_3^{(2)}(t_2) - F_3^{(1)}(t_1) \leq F_3^{(1)}(t_2) - F_3^{(1)}(t_1) \leq \alpha_3(t_2 - t_1). \tag{7}$$

$$F_3^{(3)}(t_3) - F_3^{(1)}(t_1) \leq F_3^{(1)}(t_3) - F_3^{(1)}(t_1) \leq \alpha_3(t_3 - t_1). \tag{8}$$

Note that *flow constraints* [Chang, 2000] are also applied here, which require that the traffic volume fed to a node at any time instant be greater than or equal to the traffic volume output from the same node for the same flow.

For the purpose of bounding the worst-case delay, it is essential to find out the burstiness introduced by each flow at each node, which can be achieved using slack variables. Specifically for the sample network in Figure 4, we need to introduce slack variables for Equations 5, 6, 7, and 8 as follows

$$F_2^{(1)}(t_1) - F_2^{(0)}(t_0) = \alpha_2(t_1 - t_0) - s_2^{(1)}. \tag{9}$$

$$F_2^{(2)}(t_2) - F_2^{(1)}(t_1) \leq \alpha_2(t_2 - t_0) - \alpha_2(t_1 - t_0) - s_2^{(1)}. \tag{10}$$

$$F_3^{(2)}(t_2) - F_3^{(1)}(t_1) = \alpha_3(t_2 - t_1) - s_3^{(2)}. \tag{11}$$

$$F_3^{(3)}(t_3) - F_3^{(2)}(t_2) \leq \alpha_3(t_3 - t_1) - \alpha_3(t_2 - t_1) - s_3^{(2)}. \tag{12}$$

For each interfering flow, a slack variable is introduced to represent the amount of burstiness incurred when passing its entry node into the path traveled by the flow of interest. Burstiness of an interfering flow incurred when passing through succeeding nodes can be represented using the same slack variables as shown by Equations 10 and 12. The introduction of slack variables implies the following constraint set

$$\mathcal{C}_1 = \{0 \leq s_2^{(1)} \leq \alpha_2(t_1 - t_0), 0 \leq s_3^{(2)} \leq \alpha_3(t_2 - t_1), 0 \leq t_0 \leq t_1 \leq t_2 \leq t_3\}.$$

Combining Equations 4, 9, 10, 11, and 12, we obtain

$$\begin{aligned}
F_1^{(3)}(t_3) - F_1^{(0)}(t_0) \geq & \beta_1^{(1\rightarrow 2\rightarrow 3)}(t_3 - t_0) \\
= & \inf_{\mathcal{C}_1}\{[\beta_3(t_3-t_2) - l_{1,2,\max} - (\alpha_3(t_3-t_1) - \alpha_3(t_2-t_1) + s_3^{(2)})]^+ \\
& + [\beta_2(t_2-t_1) - l_{1,2,3,\max} - (\alpha_3(t_2-t_1) - s_3^{(2)}) - (\alpha_2(t_2-t_0) - \alpha_2(t_1-t_0) + s_2^{(1)})]^+ \\
& [\beta_1(t_1-t_0) - l_{1,2,\max} - (\alpha_2(t_1-t_0) - s_2^{(1)})]^+\}. \qquad (13)
\end{aligned}$$

So far, we have incorporated flow constraints as well as our knowledge of incoming traffic flows and service capabilities into our problem formulation. However, this formulated problem is hard to solve because arrival and service curves may assume arbitrary forms. Therefore, we further instantiate arrival and service curves using leaky-bucket arrival and rate-latency service curves. In particular, let $\beta_i = R_i[t - T_i]^+$ denote the rate-latency service curve for node i, where $i = 1, 2, 3$. Suppose that arrival curve for flow F_i at the input side of its entry node is $\alpha_i(t) = r_i t + b_i$, where r_i is the average rate and b_i is the burstiness component. The constraint set $\mathcal{C}_1$ becomes

$$\mathcal{C}_2 = \{s_2^{(1)} \leq b_2 + r_2 T_1, s_3^{(2)} \leq b_3 + \frac{s_2^{(1)} + r_2 T_2}{R_2 - r_2}\}.$$

Note that the two constraint sets specify the range of both slack variables. As explained in [Schmitt et al., 2008], the slack variables introduced in Equations 9, 10, 11, and 12 can be interpreted as the flexible parts of the burstiness incurred at the respective nodes. Although the problem formulation process of our packet-oriented delay bounding method resembles that of the bit-oriented approach in [Schmitt et al., 2008], our formulation adds extra flexibility as packet sizes can now be taken into account. Furthermore, we also note that it is necessary to formulate the delay bounding problem from scratch using the theorems introduced in Section 2.1. Further explanations will be given in the next section.

After instantiation, Equation 13 becomes

$$\begin{aligned}
\beta_1^{(1\rightarrow 2\rightarrow 3)}(t_3 - t_0) = & \inf_{\mathcal{C}_1, \mathcal{C}_2} \{[R_3[(t_3-t_2) - T_3]^+ - l_{1,3,\max} - (r_3(t_3-t_2) + s_3^{(2)})]^+ \\
& [R_2[(t_2-t_1) - T_2]^+ - (b_3 + r_3(t_2-t_1) - s_3^{(2)}) - (r_2(t_2-t_1) + s_2^{(1)}) - l_{1,2,3,\max}]^+ \\
& [R_1[(t_1-t_0) - T_1]^+ - l_{1,2,\max} - (b_2 + r_2(t_1-t_0) - s_2^{(1)})]^+\} \qquad (14) \\
\geq & \inf_{\mathcal{C}_1, \mathcal{C}_2} \{(R_3 - r_3)[t_3 - t_2 - T_3 - \frac{r_3 T_3 + s_3^{(2)} + l_{1,3,\max}}{R_3 - r_3}]^+ \\
& + (R_2 - r_2 - r_3)[t_2 - t_1 - T_2 - \frac{b_3 + (r_2 + r_3)T_2 - s_3^{(2)} + s_2^{(1)} + l_{1,2,3,\max}}{R_2 - r_2 - r_3}]^+ \\
& (R_1 - r_2)[t_1 - t_0 - T_1 - \frac{b_2 + r_2 T_1 - s_2^{(1)} + l_{1,2,\max}}{R_1 - r_2}]^+\} \qquad (15) \\
= & [(R_1 - r_2) \wedge (R_2 - r_2 - r_3) \wedge (R_3 - r_3)] \cdot \inf_{\mathcal{C}_2}\{[t_3 - t_0 - (T_1 + T_2 + T_3) \\
& - \frac{r_3 T_3 + s_3^{(2)} + l_{1,3,\max}}{R_3 - r_3} - \frac{b_3 + (r_2 + r_3)T_2 - s_3^{(2)} + s_2^{(1)} + l_{1,2,3,\max}}{R_2 - r_2 - r_3} \\
& - \frac{b_2 + r_2 T_1 - s_2^{(1)} + l_{1,2,\max}}{R_1 - r2}]^+\}. \qquad (16)
\end{aligned}$$

This gives us the left-over service curve of the concatenation of the three nodes in Figure 4 for the flow of interest. Note that this left-over service curve is also a rate-latency curve and that burstiness of interfering flows incurred at each node has been factored into the latency term.

Finally, we transform the delay bounding problem into the following minimization problem under the constraints given in $\mathcal{C}_2$

$$\min \quad (\frac{1}{R_1 - r_2} - \frac{1}{R_2 - r_2 - r_3})s_2^{(1)} + (\frac{1}{R_2 - r_2 - r_3} - \frac{1}{R_3 - r_3})s_3^{(2)}.$$

This is a linear programming (LP) problem as service rates and average data rates are parameters that can either be extracted from device specifications or obtained from measurements.

4 Tightness Of Delay Bounds Under Blind Multiplexing

In this section, we solve the optimization problem formulated in Section 3 and show that the delay bounds obtained are indeed tight. First, we observe that the instantiated LP problem may have several possible cases depending on the numerical values of parameters of leaky-bucket arrival and rate-latency service curves. To simplify our notations for all the possible cases, we use the following parameters

$$A = \frac{1}{R_1 - r_2} - \frac{1}{R_2 - r_2 - r_3}, B = \frac{1}{R_2 - r_2 - r_3} - \frac{1}{R_3 - r_3}, C = \frac{r_3}{R_2 - r_2}.$$

The formulated LP problem needs to be solved for the following five cases

Case I: $A > 0, B > 0$. In this case, we have $R_3 - r_3 > R_2 - r_2 - r_3 > R_1 - r_2$.

Case II: $A \leq 0, B > 0$. In this case, we have $R_1 - r_2 \geq R_2 - r_2 - r_3$ and $R_3 - r_3 > R_2 - r_2 - r_3$.

Case III: $A \leq 0, B \leq 0$. In this case, we have $R_1 - r_2 \geq R_2 - r_2 - r_3 \geq R_3 - r_3$.

Case IV: $A > 0, B \leq 0, -B \cdot C \leq A$. In this case, we have $R_2 - r_2 - r_3 > R_1 - r_2$, $R_2 - r_2 - r_3 \geq R_3 - r_3$, and $R_3 - r_3 \geq R_1 - r_2$.

Case V: $A > 0, B \leq 0, -B \cdot C > A$. In this case, we have $R_2 - r_2 - r_3 > R_1 - r_2$, $R_2 - r_2 - r_3 \geq R_3 - r_3$, and $R_3 - r_3 < R_1 - r_2$.

Note that this LP problem is easy to solve once the relationships among parameters are completely determined so that the minimum left-over service rate for the flow of interest can be found unambiguously. Next, we show that the delay bounds obtained for all possible cases are indeed tight under the assumptions summarized in Section 2.1.

For Cases I, II, and III, the node offering the minimum left-over data rate for the flow of interest can be determined directly. In contrast, we are not able to determine which node offers the minimum left-over data rate for the flow of interest when $A > 0, B \leq 0$. As a result, we cannot solve the LP problem using Equation 16. Hence, we need to further divide the case when $A > 0, B \leq 0$ into two sub-cases by assuming a certain explicit relationship between the left-over rates offered by the first and the third nodes.

The following lemma is useful when we compute the time for a node to serve a certain volume of backlogged data in the context of continuous arrivals of new data from both the flow of interest and interfering flows.

Lemma 1. *Suppose a node with a service rate that is at least R serves both a flow of interest and serveral interfering flows. Under blind multiplexing, further assume that the flow of interest has the lowest priority and interfering flows have data backlogged by the node. If the data backlogged for the interfering flows are served either at minimum possible rates or at an infinite rate, we can denote the data volume served at minimum possible rates as $b_(m)$ and the left-over service rate for the flow of interest as r_m. It takes time t_x for the node to consume all data backlogged for interfering flows and starts to serve data from the flow of interest, where $t_x = \frac{b_m}{r_m}$.*

If all the interfering flows continue to generate new packets, the flow of interest will be served, starting from its backlogged data (as we assume per-flow FCFS), at the left-over service rate r_m.

4.1 Case I: $A>0, B>0$

The solution to the LP problem is $(s_2^{(1)}, s_3^{(2)}) = (0,0)$. The left-over service curve of the concatenation of all three nodes using packet-oriented service curves can then be simplified as

$$\beta_1^{(1\to 2\to 3)}(t) = R_{lo}[t - T_{lo}]^+, \tag{17}$$

where

$$R_{lo} = R_1 - r_2, \tag{18}$$

$$T_{lo} = (T_1+T_2+T_3) + \frac{r_3T_3+l_{1,3,\max}}{R_3-r_3} + \frac{b_3+(r_2+r_3)T_2+l_{1,2,3,\max}}{R_2-r_2-r_3} + \frac{b_2+r_2T_1+l_{1,2,\max}}{R_1-r_2}. \tag{19}$$

As explained in Sections 1.2 and 2.1, we can obtain the worst-case delay once the service curve dedicated for the flow of interest is effectively separated from the overall service curve. The L-packetized worst-case delay obtained for this case is

$$\begin{aligned} d_{1,L}^{OPT} = T_{lo} + \frac{b_1}{R_1-r_2} &= T_1 + \frac{b_1+b_2+r_2T_1+l_{1,2,\max}}{R_1-r_2} \\ &+ T_2 + \frac{b_3+(r_2+r_3)T_2+l_{1,2,3,\max}}{R_2-r_2-r_3} \\ &+ T_3 + \frac{r_3T_3+l_{1,3,\max}}{R_3-r_3} \end{aligned} \tag{20}$$

Note that per-flow FCFS is assumed as the flow of interest is the only flow enjoying the service characterized by the left-over service curve.

The tightness of this delay bound can be shown by tracking a bit of interest (b-o-i) in the flow of interest as follows:

1. **At time instant t_0.** The first node enters its vacation because it is idle (i.e., its backlog is empty). The duration of its vacation is upper bounded by T_1. At the same time, flow $F_2^{(0)}(t)$ arrives at the first node and bursts.

2. **At time $t_1 = t_0 + T_1$.** The first node starts to serve flow $F_2^{(0)}(t)$, which has accumulated $b_2 + r_2T_1$ data in the backlog of the first node during its vacation. As we assume blind multiplexing among flows, the worst case for the flow of interest is that the interfering flow $F_2^{(0)}(t)$ is served by the first node with a strictly higher priority over the flow of interest.

3. **At time $t_2 = t_1 + t_{x_{1,2}}$.** Backlogged data in the first node for $F_2^{(0)}(t)$ has been emptied, so the first node can start to serve the flow of interest. Note that flow $F_2^{(0)}(t)$ continues to generate new packets, so the flow of interest is always processed with a low priority. Using Lemma 1, we have $t_{x_{1,2}} = \frac{b_2+r_2T_1}{R_1-r_2}$. The node starts to serve the flow of interest, starting with its burstiness component. Our b-o-i is the first data bit in the packet the immediately follows the burstiness packets.

4. **At time $t_3 = t_2 + \frac{b_1+l_{1,2,\max}}{R_1-r_2}$.** The boi passes through the node after the burstiness component is processed. Then, it stays in the attached L-packetizer until all data bits from the same packet arrive at the L-packetizer. The worst case for the b-o-i occurs when it is the first bit of one of the longest packets with length $l_{1,2,\max}$. After leaving the L-packetizer attached to the first node, the b-o-i immediately arrives at the second node as we do not assume any propagation delay over the links between nodes. At the same time, the second node starts to

take a vacation, the length of which is upper bounded by T_2. Note that this is possible because the first node is the bottleneck of the tandem network of the first and the second nodes. In addition, flow $F_3^{(1)}(t)$ arrives at the second node and bursts.

5. **At time $t_4 = t_3 + T_2$.** The second node starts to serve $F_2^{(1)}(t)$ and $F_3^{(1)}(t)$. Note that Lemma 1 allows us to directly derive the time for a node to process backlogged data for interfering flows before processing the flow of interest. In other words, we do not need to explicitly assume that $F_2^{(1)}(t)$ has a strictly higher priority over $F_3^{(1)}(t)$ if our focus is the worst case for the flow of interest.

6. **At time $t_5 = t_4 + t_{x_2}$.** The b-o-i gets processed by the second node. Note that burstiness component b_1 must have been processed before the second node can enter its vacation. Similarly, the backlogged data for the two interfering flow at time t_4 is $r_2T_2 + r_3T_2 + b_3$. Using Lemma 1, we have $t_{x_2} = \frac{r_2T_2 + r_3T_2 + b_3}{R_2 - r_2 - r_3}$. After passing through the second node, the b-o-i resides in the attached L-Packetizer, waiting for succeeding bits of the same packet.

7. **At time $t_6 = t_5 + \frac{l_{1,2,3,\max}}{R_2 - r_2 - r_3}$.** Enough data bits from the flow of interest have passed through the second node, forming a packet of length $l_{1,2,3,\max}$. The b-o-i immediately arrives at the third node. Simultaneously, the third node enters its vacation, the duration of which is upper bounded by T_3.

8. **At time $t_7 = t_6 + T_3$.** The data from $F_3^{(2)}$ accumulated during T_3 gets processed by the third node with a strictly higher priority over the flow of interest. Note that this is possible because the second node is the bottleneck of the tandem network of the second and the third nodes.

9. **At time $t_8 = t_7 + t_{x_3} + \frac{l_{1,3,\max}}{R_3 - r_3}$.** The third node first processes data backlogged for $F_3^{(2)}$, then the b-o-i. The b-o-i waits in the attached L-packetizer for succeeding bits from the same packet. Using Lemma 1, it is easy to find that $t_{x_3} = \frac{r_3T_3}{R_3 - r_3}$.

The maximum delay experienced by the b-o-i is the tight delay bound in this case, which can be verified by computing $t_8 - t_0$.

4.2 Case II: $A \leq 0, B > 0$

In this case, the solution to the LP problem is $(s_2^{(1)}, s_3^{(2)}) = (b_2 + r_2T_1, 0)$. The left-over service curve offered by the concatenation of the three nodes for $F_1^{(0)}(t)$ can then be determined using the instantiated equations in Section 3:

$$\beta_1^{(1\to 2\to 3)}(t) = R_{lo}[t - T_{lo}]^+,$$

where

$$R_{lo} = R_2 - r_2 - r_3, \tag{21}$$

$$T_{lo} = (T_1 + T_2 + T_3) + \frac{r_3T_3 + l_{1,3,\max}}{R_3 - r_3} + \frac{b_3 + (r_2 + r_3)T_2 + b_2 + r_2T_1 + l_{1,2,3,\max}}{R_2 - r_2 - r_3} + \frac{l_{1,2,\max}}{R_1 - r_2}. \tag{22}$$

Note that the worst-case delay can then be derived under the per-flow FCFS assumption as the service curve dedicated to the flow of interest has been effectively separated. We also note that the $\frac{l_{1,2,\max}}{R_1 - r_2}$ term in T_{lo} can only be obtained using the formulation process demonstrated in Section 3. It may be tempting the directly augment the bit-oriented results, but the resulting delay bounds are still overly tight as the impacts of packet sizes are not fully taken into consideration. The

L-packetized worst-case delay in this case is

$$\begin{aligned} d_{1,L}^{OPT} =& T_1 + \frac{l_{1,2,\max}}{R_1 - r_2} \\ &+ T_2 + \frac{b_1 + b_2 + b_3 + r_2 T_1 + (r_2 + r_3) T_2 + l_{1,2,3,\max}}{R_2 - r_2 - r_3} \\ &+ T_3 + \frac{r_3 T_3 + l_{1,3,\max}}{R_3 - r_3}. \end{aligned} \tag{23}$$

To show the tightness of this delay bound, we again track a b-o-i in the flow of interest that experiences the worst-case delay when passing though the tandem network of three nodes shown in Figure 4:

1. **At time t_0.** The first node takes a vacation, the length of which is upper bounded by T_1. The b-o-i arrives immediately after the bursty traffic of the flow of interest. The first node enters its vacation after the burstiness component is processed.

2. **At time $t_1 = t_0 + T_1$.** The first node starts to serve the b-o-i. It should be noted that the case in which the burstiness component of the flow of interest gets backlogged at the first node does not incur the maximum end-to-end delay. This is because the second node is the bottleneck of the concatenation of the first and second nodes.

3. **At time $t_2 = t_1 + \frac{l_{1,2,\max}}{R_1 - r_2}$.** The b-o-i needs to wait until all the data bits in the same packet reach the attached L-packetizer. The worst case at this step corresponds to the scenario where the interfering flow $F_2^{(0)}(t)$ continues to generate new packets. After gathering enough bits, the L-packetizer produces a new packet, which includes the b-o-i. As we assume no propagation delay on the link between nodes, the b-o-i now arrives at the second node. At the same time, the second node enters its vacation and the flow $F_3^{(1)}(t)$ bursts.

4. **At time $t_3 = t_2 + T_2$.** The second node terminates its vacation and starts to serve traffic. As the second node is the bottleneck of the concatenation of the first and the second node, the worst case occurs when the maximum amount of data from interfering flows accumulates at this node. The maximum backlog preceding the b-o-i is $b_1 + b_2 + b_3 + r_2 T_1 + (r_2 + r_3) T_2$. Note that this does not violates our node specifications, which only specify the minimum serving capability.

5. **At time $t_4 = t_3 + \frac{b_1 + b_2 + b_3 + r_2 T_1 + (r_2 + r_3) T_2}{R_2 - r_2 - r_3}$.** The b-o-i passes through the second node. Note that we can apply Lemma 1 as all the backlogged data preceding the b-o-i is served according to the strict (minimum) service curve.

6. **At time $t_5 = t_4 + \frac{l_{1,2,3,\max}}{R_2 - r_2 - r_3}$.** The b-o-i waits until enough data bits in the same packet arrive the attached L-packetizer. The b-o-i arrives at the third node immediately after its departure from the second node's L-packetizer. Note that the second node is also the bottleneck of the concatenation of the second and the third nodes. At the same time, the third node takes a vacation, the length of which is upper bounded by T_3.

7. **At time $t_6 = t_5 + T_3$.** Data backlogged at the third node for $F_3^{(2)}(t)$ during its vacation amounts to $r_3 T_3$. Note that any larger amount of backlogged data for this interfering flow is infeasible because it is in contradiction with the fact that the second node is the bottleneck.

8. **At time $t_7 = t_6 + \frac{r_3 T_3}{R_3 - r_3}$.** Data backlogged for $F_3^{(2)}(t)$ during the third node's vacation are processed and the b-o-i passes through the third node. Then, the b-o-i needs to wait in the attached L-packetizer for succeeding bits to arrive. Note that we again apply Lemma 1 here.

9. At time $t_8 = t_7 + \frac{l_{1,3,\max}}{R_3 - r_3}$**.** Enough data bits from the flow of interest have passed through the third node. The b-o-i leaves the L-packetizer attached to the third node. Note that this is the worst case because the interfering flow continues to generate new packets, which must be served with a higher priority.

The worst-case delay experienced by the b-o-i is the delay bound in this case, which can be verified by computing $t_8 - t_0$.

4.3 Case III: $A \leq 0, B \leq 0$

In this case, the solution to the LP problem is $(s_2^{(1)}, s_3^{(2)}) = (b_2 + r_2 T_1, b_3 + r_3(T_2 + \frac{b_2 + r_2 T_1 + r_2 T_2}{R_2 - r_2}))$. The left-over service curve offered by the tandem network of the three nodes shown in Figure 4 for the flow of interest can be determined using the instantiated equations in Section 3:

$$\beta_1^{(1 \to 2 \to 3)}(t) = R_{lo}[t - T_{lo}]^+,$$

where

$$R_{lo} = R_3 - r_3, \tag{24}$$

$$\begin{aligned} T_{lo} = & (T_1 + T_2 + T_3) + \frac{r_3 T_3 + b_3 + r_3(T_2 + \frac{b_2 + r_2 T_1 + r_2 T_2}{R_2 - r_2}) + l_{1,3,\max}}{R_3 - r_3} \\ & + \frac{b_2 + r_2 T_2 + r_2 T_1 - r_3(\frac{b_2 + r_2 T_1 + r_2 T_2}{R_2 - r_2}) + l_{1,2,3,\max}}{R_2 - r_2 - r_3} \\ & + \frac{l_{1,2,\max}}{R_1 - r_2} \end{aligned} \tag{25}$$

Note that the term $\frac{l_{1,2,\max}}{R_1 - r_2}$ in T_{lo} can only be obtained if we closely follow the formulation process demonstrated in Section 3. Delay bound obtained for this scenario by directly augmenting the corresponding bit-oriented result is still overly tight because not all the impacts of variable packet lengths are incorporated. Once the left-over service curve for the flow of interest is obtained, we can find the delay bound for in this case under the per-flow FCFS assumption. The L-packetized delay bounds for this case is

$$\begin{aligned} d_{1,L}^{OPT} = & T_1 + \frac{l_{1,2,\max}}{R_1 - r_2} \\ & + T_2 + \frac{b_2 + r_2 T_1 + r_2 T_2}{R_2 - r_2} + \frac{l_{1,2,3,\max}}{R_2 - r_2 - r_3} \\ & + T_3 + \frac{b_1 + b_3 + r_3 T_2 + r_3 T_3 + r_3(\frac{b_2 + r_2 T_1 + r_2 T_2}{R_2 - r_2}) + l_{1,3,\max}}{R_3 - r_3} \end{aligned}$$

To show that the obtained delay bound is indeed tight in this case, we track the b-o-i of the flow of interest experiencing the maximum possible delay:

1. At time t_0**.** The first node enters its vacation, the length of which is upper bounded by T_1. Meanwhile, $F_1^{(0)}(t)$ and $F_2^{(0)}(t)$ burst. The b-o-i arrives immediately after the burstiness component of the flow of interest, $F_1^{(0)}(t)$.

2. At time $t_1 = t_0 + T_1$**.** The first node terminates its vacation and starts to work. The maximum possible backlog accumulated during the first node's vacation is $b_1 + b_2 + r_2 T_1$, which is served at an infinitely fast rate. Note that this is possible because our knowledge of the processing capability of a node in the form of a strict service curve does not constrain the maximum rate at which a node may serve.

3. **At time** $t_2 = t_1 + \frac{l_{1,2,\max}}{R_1 - r_2}$**.** As data backlogged during the first node's vacation is served in an instant, the b-o-i immediately passes through the first node and stays in the attached L-packetizer. It must wait until all data bits in the same packet arrive at the L-packetizer. The worst case for b-o-i corresponds to the scenario when both $F_1^{(0)}(t)$ and $F_2^{(0)}(t)$ generate packets at their respective average rate. In addition, the b-o-i and succeeding data bits from both flows are served at the rate specified by the strict service curve (i.e., a node can only serve at an infinitely fast rate at a certain time instant). After enough data bits arrive at the L-packetizer, a packet can now be formed and the b-o-i departs for the second node. Simultaneously, the second node takes a vacation, the length of which is upper bounded by T_2. The other interfering flow, $F_3^{(1)}(t)$, bursts immediately after the second node's entry into a vacation.

4. **At time** $t_3 = t_2 + T_2$**.** The second node terminates its vacation and starts to work. Note that the maximum backlog accumulated during the second node's vacation is $b_1 + b_2 + r_2(T_1 + T_2) + b_3 + r_3 T_2$. This is possible because the second node is the bottleneck of the concatenation of the first and the second nodes in this case. Note that $F_2^{(2)}(t)$ leaves the path traveled by the flow of interest. The worst case for the b-o-i hence includes the delay incurred by processing the backlogged data for $F_2^{(1)}(t)$ at the second node. The other part of the backlogged data, namely $b_1 + b_3 + r_3 T_2$, is again processed at an infinitely fast rate by the second node. This is possible and necessary because the third node is the bottleneck of the tandem network consisting the second and the third nodes.

5. **At time** $t_4 = t_3 + \frac{b_2 + r_2 T_1 + r_2 T_2}{R_2 - r_2}$**.** The backlogged data for $F_2^{(1)}(t)$ is processed at the rate specified by the service curve of the second node. Note that we apply Lemma 1 here assuming that the aggregation of $F_1^{(1)}(t)$ and $F_3^{(1)}(t)$ is the flow under investigation. It should also be noted that the interfering flow $F_3^{(1)}(t)$, which should be assumed to have a priority that is higher than $F_1^{(1)}(t)$ but lower than $F_2^{(1)}(t)$, accumulates more data during the period when $F_2^{(1)}(t)$ fully engages the second node. The newly accumulated data $r_3(\frac{b_2 + r_2 T_1 + r_2 T_2}{R_2 - r_2})$ is also processed at an infinitely fast rate. This is because the third node is the bottleneck of the concatenation of the second and the third nodes. Hence, the b-o-i passes through the second node at t_4 and temporarily resides in the attached L-packetizer.

6. **At time** $t_5 = t_4 + \frac{l_{1,2,3,\max}}{R_2 - r_2 - r_3}$**.** At this point, enough data bits arrive at the L-packetizer, forming a new packet containing the b-o-i. For this worst case, both interfering flows continue to generate new packets with higher priorities. The b-o-i arrives at the third node immediately after it departs from the L-packetizer of the second node. At the same moment, the third node starts to take a vacation, the length of which is upper bounded by T_3.

7. **At time** $t_6 = t_5 + T_3$**.** The third node exits its vacation and starts to work. The maximum possible amount of data backlogged at the third node that has impact on the delay experienced by the b-o-i is $b_1 + b_3 + r_3 T_2 + r_3 T3 + r_3(\frac{b_2 + r_2 T_1 + r_2 T_2}{R_2 - r_2})$.

8. **At time** $t_7 + \frac{b_1 + b_3 + r_3 T_2 + r_3 T3 + r_3(\frac{b_2 + r_2 T_1 + r_2 T_2}{R_2 - r_2})}{R_3 - r_3}$**.** Note that we apply Lemma 1 here to obtain the time required to empty the third node's backlog. Then, the b-o-i gets processed.

9. **At time** $t_8 = t_7 + \frac{l_{1,3,\max}}{R_3 - r_3}$**.** The b-o-i has to wait in the attached L-packetizer until all data bits in the same packet passes through the third node. Note that $F_1^{(2)}(t)$ has the lowest priority and the interfering flow $F_3^{(2)}(t)$ continues to generate new packets.

In this case, we observe that we need to allow backlogged data of both the interfering flows and the flow of interest to incur more significant delay at bottlenecks. This is also one of the reasons why the delay bounds obtained using our packet-oriented optimization-based method are theoretically the maximum possible ones. We can compute $t_8 - t_0$ to verify the tightness of the delay bound obtained for this case.

4.4 Case IV: $A>0, B\leq 0, -B\cdot C\leq A$

The solution to the LP problem in this case is $(s_2^{(1)}, s_3^{(2)}) = (0, b_3+r_3(T_2+\frac{r_2T_2}{R_2-r_2}))$. The left-over service curve for the flow of interest offered by the tandem network of all three nodes shown in Figure 4 can be obtained using the instantiated equations in Section 3:

$$\beta_1^{(1\to 2\to 3)}(t) = R_{lo}[t-T_{lo}]^+,$$

where

$$R_{lo} = R_1 - r_2, \tag{26}$$

$$\begin{aligned} T_{lo} =& (T_1+T_2+T_3)+\frac{b_2+r_2T_1+l_{1,2,\max}}{R_1-r_2} \\ &+\frac{b_3+r_3T_3+r_3T_2+r_3(\frac{r_2T_2}{R_2-r_2})+l_{1,3,\max}}{R_3-r_3} \\ &+\frac{r_2T_2-r_3(\frac{r_2T_2}{R_2-r_2})+l_{1,2,3,\max}}{R_2-r_2-r_3}. \end{aligned} \tag{27}$$

The worst-case delay for this case can then be determined under the per-flow FCFS assumption because the left-over service curve effectively characterize the processing capacity solely enjoyed by the flow of interest. The L-packetized worst-case delay experienced by the flow of interest is

$$\begin{aligned} d_{1,L}^{OPT} =& T_1+\frac{b_1+b_2+r_2T_1+l_{1,2,\max}}{R_1-r_2} \\ &+T_2+\frac{r_2T_2}{R_2-r_2}+\frac{l_{1,2,3,\max}}{R_2-r_2-r_3} \\ &+T_3+\frac{b_3+r_3T_3+r_3T_2+r_3(\frac{r_2T_2}{R_2-r_2})+l_{1,3,\max}}{R_3-r_3}. \end{aligned}$$

To show that the delay bound we obtain for this case is indeed tight, we can track the b-o-i the experiences the maximum possible delay as follows:

1. **At time t_0.** The first node takes a vacation, the length of which is upper bounded by T_1. At the same moment, the b-o-i arrives immediately after the burstiness component of the flow of interest. The interfering flow $F_2^{(0)}(t)$ also bursts immediately after the first node's entry into its vacation.

2. **At time $t_1=t_0+T_1$.** The first node terminates its vacation and starts to work. During this vacation, the maximum possible amount of data backlogged in the first node that can impact the delay experienced by the b-o-i is $b_1+b_2+r_2T_1$. We note that the first node is the bottleneck of the tandem network of all the three nodes shown in Figure 4. Hence, allowing the burstiness components of $F_1^{(0)}(t)$ and $F_2^{(0)}(t)$ to incur delay at the first node introduces the most significant impacts on the end-to-end delay experienced by the b-o-i. The first node starts to process its backlogged data at the rate specified by its service curve, and the b-o-i has to wait until it empties its backlog.

3. **At time $t_2=t_1+\frac{b_1+b_2+r_2T_1}{R_1-r_2}$.** At this moment, the first node finishes processing all backlogged data preceding the b-o-i. Note that we apply Lemma 1 to derive the time the b-o-i needs to wait before getting processed. The b-o-i now passes through the first node and stays in the attached L-packetizer.

4. **At time $t_3=t_2+\frac{l_{1,2,\max}}{R_1-r_2}$.** The b-o-i waits in the L-packetizer until all the bits from the same packet passes through the first node. Then, the L-packetizer forms a new packet for the

flow of interest, and the b-o-i immediately arrives at the second node. At the same instant, the second node enters its vacation, the length of which is upper bounded by T_2. Note that backlogged data $b_1+b_2+r_2T_1$ preceding the b-o-i in the buffer of the first node will not be present in the buffer of the second node because the first node is the bottleneck.

5. **At time $t_4=t_3+T_2$.** The second node terminates its vacation and starts to work. During this vacation, the maximum possible amount of backlogged data that can increase the delay experienced by the b-o-i is $r_2T_2+b_3+r_3T_2$. As $F_2^{(2)}(t)$ leaves the path traveled the flow of interest, allowing its backlogged data r_2T_2 to incur delay for the b-o-i should be included in this worst case.

6. **At time $t_5=t_4+\frac{r_2T_2}{R_2-r_2}$.** At this moment, the backlogged data r_2T_2 for $F_2^{(1)}(t)$ has been processed by the second node. We obtain this result by applying Lemma 1 and assuming the aggregation of $F_1^{(1)}(t)$ and $F_3^{(1)}(t)$ to be the flow under investigation. The other part of the backlogged data, i.e., $b_3+r_3T_2$, is processed at an infinitely fast rate. As explained in previous cases, this does not violates our node specifications. Furthermore, the worst case we investigate incurs maximum end-to-end delay for the b-o-i. As the third node is the bottleneck of the tandem network of the second and the third nodes, the worst case should allow this part of the backlog to incur larger delay at the third node. The b-o-i passes through the second node immediately after the departure of $r_2T_2+b_3+r_3T_2$.

7. **At time $t_6=t_5+\frac{l_{1,2,3,\max}}{R_2-r_2-r_3}$.** At this moment, enough data bits reach the L-packetizer attached to the second node. A new packet containing the b-o-i is formed. Note that this is accomplished by the second node under the influence of the continuous arrivals from $F_2^{(1)}(t)$ and $F_3^{(1)}(t)$. Simultaneously, the third node begins to take a vacation, the length of which is upper bounded by T_3.

8. **At time $t_7=t_6+T_3$.** The third node terminates its vacation and starts to work. The maximum possible amount of backlogged data that can increase the end-to-end delay experienced by the b-o-i is $b_3+r_3(T_2+T_3)+r_3(\frac{r_2T_2}{R_2-r_2})$. Note that $F_2^{(1)}(t)$ continues to generate packets while the second node processes its backlogged data with the highest priority.

9. **At time $t_8=t_7+\frac{b_3+r_3(T_2+T_3)+r_3(\frac{r_2T_2}{R_2-r_2})}{R_3-r_3}$.** It is now the b-o-i's turn to get processed. Note that we obtain this result by applying Lemma 1. The b-o-i waits in the attached L-packetizer after passing through the third node.

10. **At time $t_9=t_8+\frac{l_{1,3,\max}}{R_3-r_3}$.** Enough data bits have reached the L-packetizer and a new packet containing the b-o-i is formed.

In this case, we observe that Lemma 4 can be flexibly applied to obtain the delay incurred by backlogged data that are processed at rates specified by service curves. However, it is not applicable to backlogged data processed at any higher rate. We can compute t_9-t_0 to verify the tightness of the delay bound for this case.

4.5 Case V: $A>0, B\leq 0, -B\cdot C>A$

The solution to the LP problem in this case is $(s_2^{(1)}, s_3^{(2)})=(b_2+r_2T_1, b_3+r_3(T_2+\frac{b_2+r_2T_1+r_2T_2}{R_2-r_2}))$. The left-over service curve offered by the tandem network of the three nodes shown in Figure 4 for the flow of interest can be determined using the instantiated equations in Section 3:

$$\beta_1^{(1\to 2\to 3)}(t)=R_{lo}[t-T_{lo}]^+,$$

where

$$R_{lo}=R_3-r_3, \tag{28}$$

$$T_{lo}=(T_1+T_2+T_3)+\frac{r_3T_3+b_3+r_3(T_2+\frac{b_2+r_2T_1+r_2T_2}{R_2-r_2})+l_{1,3,\max}}{R_3-r_3}$$
$$+\frac{b_2+r_2T_2+r_2T_1-r_3(\frac{b_2+r_2T_1+r_2T_2}{R_2-r_2})+l_{1,2,3,\max}}{R_2-r_2-r_3}$$
$$+\frac{l_{1,2,\max}}{R_1-r_2} \tag{29}$$

Note that the term $\frac{l_{1,2,\max}}{R_1-r_2}$ in T_{lo} can only be obtained if we closely follow the formulation process demonstrated in Section 3. Delay bound obtained for this scenario by directly augmenting the corresponding bit-oriented result is still overly tight because not all the impacts of variable packet lengths are incorporated. Once the left-over service curve for the flow of interest is obtained, we can find the delay bound for in this case under the per-flow FCFS assumption. The L-packetized delay bounds for this case is

$$d_{1,L}^{OPT}=T_1+\frac{l_{1,2,\max}}{R_1-r_2}$$
$$+T_2+\frac{b_2+r_2T_1+r_2T_2}{R_2-r_2}+\frac{l_{1,2,3,\max}}{R_2-r_2-r_3}$$
$$+T_3+\frac{b_1+b_3+r_3T_2+r_3T_3+r_3(\frac{b_2+r_2T_1+r_2T_2}{R_2-r_2})+l_{1,3,\max}}{R_3-r_3}$$

We note that the solution to the LP problem and the delay bound obtained for this case bare the same forms as Case III. However, the specifications of the nodes are different from those in Case III. In particular, the first node is now the bottleneck of the tandem network of the first and the second nodes. To show that the obtained delay bound is indeed tight in this case, we track the b-o-i of the flow of interest experiencing the maximum possible delay:

1. **At time t_0.** The first node enters its vacation, the length of which is upper bounded by T_1. Note that the first node is the bottleneck of the concatenation of the first and the second nodes. The b-o-i arrives immediately after the burstiness component of the flow of interest.

2. **At time $t_1=t_0+T_1$.** The first node terminates its vacation and starts the work. During the vacation, the maximum possible amount of backlogged data that can increase the delay experienced by the b-o-i is $b_1+b_2+r_2T_1$. However, the first node processes all the backlogged data at an infinitely fast rate. On the one hand, the burstiness component b_1 can either be processed at the first node or the third node in this case. The third node is the bottleneck of the tandem network shown in Figure 4. Evidently, the worst case for the b-o-i should allow this component to cause larger delay at the third node. On the other hand, allowing $b_2+r_2T_1$ to be processed with the highest priority at the second node also results in larger delay as the other interfering flow will then accumulate more data in the second node's buffer. The b-o-i passes through the first node immediately after the backlogged data.

3. **At time $t_2=t_1+\frac{l_{1,2,\max}}{R_1-r_2}$.** By this moment, enough data bits have arrived at the L-packetizer attached to the first node, so the b-o-i can now depart for the second node. Meanwhile, the second node takes a vacation, the length of which is upper bounded by T_2.

4. **At time $t_3=t_2+T_2$.** The second node returns to work from its vacation. During this vacation, the maximum possible amount of backlogged data that can impact the delay experienced by the b-o-i is $b_1+b_2+b_3+r_2(T_1+T_2)+r_3T_2$. To construct the worst case for the b-o-i, it is obvious that we should allow $F_2^{(1)}(t)$ to get processed with the highest priority because this interfering flow will leave the path traveled by the flow of interest first. As a result, $F_3^{(1)}(t)$ can accumulate more data. Note that this is possible because we assume blind multiplexing among multiple flows. Furthermore, as the third node is the bottleneck, we should allow $b_1+b_3+r_3T_2$

to pass through the second node at an infinitely fast rate. This also applies to the amount of data accumulated during the processing of backlogged data for $F_2^{(1)}(t)$.

5. **At time** $t_4 = t_3 + \frac{b_2 + r_2 T_1 + r_2 T_2}{R_2 - r_2}$**.** Again, we apply Lemma 1 assuming that the aggregation of $F_1^{(1)}(t)$ and $F_3^{(1)}(t)$ is the flow under investigation. It should also be noted that the amount of data accumulated during the processing of data backlogged for $F_2^{(1)}(t)$ is $r_3(\frac{b_2 + r_2 T_1 + r_2 T_2}{R_2 - r_2})$. At this moment, the b-o-i gets processed and then stays in the attached L-packetizer.

6. **At time** $t_5 = t_4 + \frac{l_{1,2,3,\max}}{R_2 - r_2 - r_3}$**.** The L-packetizer has collected enough bits from the flow of interest to form a new packet containing the b-o-i. Note that the second node process $F_1^{(1)}(t)$ under the influence of the continuous arrivals from $F_2^{(1)}(t)$ and $F_3^{(1)}(t)$. At the same time, the third node takes a vacation, the length of which is upper bounded by T_3.

7. **At time** $t_6 = t_5 + T_3$**.** The third node returns to work from its vacation. During this vacation, the maximum possible amount of backlogged data that can impact the end-to-end delay experienced by the b-o-i is $b_1 + b_3 + r_3(T_2 + T_3 + \frac{b_2 + r_2 T_1 + r_2 T_2}{R_2 - r_2})$.

8. **At time** $t_7 = t_6 + \frac{b_1 + b_3 + r_3(T_2 + T_3 + \frac{b_2 + r_2 T_1 + r_2 T_2}{R_2 - r_2})}{R_3 - r_3}$**.** The b-o-i gets processed by the third node. Note that we apply Lemma 1 here as all the backlogged data are processed at the rate specified by the service curve of the third node.

9. **At time** $t_8 = t_7 + \frac{l_{1,3,\max}}{R_3 - r_3}$**.** At this point, the L-packetizer attached to the third node has collected enough data bits to generate a new packet containing the b-o-i.

We can compute $t_8 - t_0$ to verify the tightness of the delay bound for this case.

5 Conclusion

In this report, we demonstrate how to integrate L-packetizers [Le Boudec, 2002] into a bit-oriented optimization-based delay bounding method [Schmitt et al., 2008]. We identify the impacts of variable packet lengths on end-to-end delay that cannot be characterized by the fluid model. Furthermore, we show that our augmented approach does generate tight delay bounds under the set of assumptions identified in Section 2.2. We also observe that formulating packet-oriented delay bounding problem from scratch as demonstrated in Section 3 is necessary because direct augmentation of bit-oriented results cannot recover terms that evaluate to 0 in the bit-oriented approach.

We summarize the assumptions of our theoretical framework in Section 3, which also reveal the limitations of our method. Better node specifications may allow us to further tighten the delay bounds. However, we expect that our approach can still reach the tight bounds once new information is made available in certain forms that can be incorporated into our network-calculus-based framework.

Acknowledgement

This work is supported by ABB Ltd, the U.S. Department of Energy, and the Pennsylvania Infrastructure Technology Alliance (PITA), a program sponsored by the Pennsylvania Department of Community and Economic Development.

References

AlEnawy, T. and Aydin, H. (2005). Energy-Constrained Scheduling for Weakly-Hard Real-Time Systems. In *Real-Time Systems Symposium, 2005. RTSS 2005. 26th IEEE International*, pages 10 pp.–385.

Baccelli, F., Cohen, G., Olsder, G. J., and Quadrat, J.-P. (1992). *Synchronization and Linearity: An Algebra for Discrete Event Systems*. John Wiley & Sons Ltd.

Bouillard, A. and Junier, A. (2011). Worst-Case Delay Bounds with Fixed Priorities Using Network Calculus. In *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS '11)*, pages 381–390, ICST, Brussels, Belgium, Belgium.

Chang, C.-S. (2000). *Performance Guarantees in Communication Networks*. Telecommunication Networks and Computer Systems. Springer-Verlag.

Cruz, R. (1991a). A Calculus for Network Delay. I. Network Elements in Isolation. *Information Theory, IEEE Transactions on*, 37(1):114–131.

Cruz, R. (1991b). A Calculus for Network Delay. II. Network Analysis. *Information Theory, IEEE Transactions on*, 37(1):132–141.

Fidler, M. (2010). Survey of Deterministic and Stochastic Service Curve Models in the Network Calculus. *Communications Surveys Tutorials, IEEE*, 12(1):59–86.

Le Boudec, J.-Y. (2002). Some Properties of Variable Length Packet Shapers. *Networking, IEEE/ACM Transactions on*, 10(3):329–337.

Le Boudec, J.-Y. and Thiran, P. (2001). *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, volume 2050 of *Lecture Notes in Computer Science*. Springer Science & Business Media.

Parekh, A. and Gallager, R. (1993). A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case. *Networking, IEEE/ACM Transactions on*, 1(3):344–357.

Parekh, A. and Gallager, R. (1994). A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple-Node Case. *Networking, IEEE/ACM Transactions on*, 2(2):137–150.

Sariowan, H., Cruz, R., and Polyzos, G. (1999). SCED: A Generalized Scheduling Policy for Guaranteeing Quality-of-Service. *Networking, IEEE/ACM Transactions on*, 7(5):669–684.

Schmitt, J., Zdarsky, F., and Fidler, M. (2008). Delay Bounds under Arbitrary Multiplexing: When Network Calculus Leaves You in the Lurch... In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 1669–1677.

Schmitt, J., Zdarsky, F., and Thiele, L. (2007). A Comprehensive Worst-Case Calculus for Wireless Sensor Networks with In-Network Processing. In *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, pages 193–202.

Shen, W., Zhang, T., Barac, F., and Gidlund, M. (2014). PriorityMAC: A Priority-Enhanced MAC Protocol for Critical Traffic in Industrial Wireless Sensor and Actuator Networks. *Industrial Informatics, IEEE Transactions on*, 10(1):824–835.

Suriyachai, P., Roedig, U., and Scott, A. (2012). A Survey of MAC Protocols for Mission-Critical Applications in Wireless Sensor Networks. *Communications Surveys Tutorials, IEEE*, 14(2):240–264.

Valaee, S. (2001). A Recursive Estimator of Worst-Case Burstiness. *Networking, IEEE/ACM Transactions on*, 9(2):211–222.

Weiner, M., Jorgovanovic, M., Sahai, A., and Nikolie, B. (2014). Design of a Low-Latency, High-Reliability Wireless Communication System for Control Applications. In *Communications (ICC), 2014 IEEE International Conference on*, pages 3829–3835.