Comparison of Delay Bounds for Ethernet Networks Based on Simple Network Calculus Algorithms

Michael Menth*, Mark Schmidt*, Sebastian Veith*, Stephan Kehrer†, Andreas Dreher†
* University of Tuebingen, Chair of Communication Networks, Tuebingen, Germany
† Hirschmann Automation and Control GmbH, Neckartenzlingen, Germany

***Abstract*—This paper gives a simple introduction to Network Calculus (NC), a theory to calculate end-to-end delay bounds for packet-switched communication networks. It illustrates how various networking parameters affect delay bounds when flows are multiplexed. Various algorithms for the calculation of end-to-end delay bounds are revisited and their performance is compared under different conditions. The framework is applicable to feed-forward networks and useful to determine delay bounds for flows in Ethernet networks.**

I. Introduction

Critical facilities such as power generation and distribution or industrial automation networks are currently being automated using packet-switched communication technologies such as industrial Ethernet. Applications in these domains are often mission-critical and require the network to support real-time communication in the sense that packet loss and excessive delays must not occur. In particular, delay bounds need to be met. To ensure these requirements and to allow for the planning of industrial Ethernet networks based on more than just the experience of the network engineers involved, tools are required to predict the maximum delay a flow possibly experiences under worst-case conditions. This can be supported by the theory of Network Calculus (NC) which is based on the pioneering work of [1], [2]. A good overview is given in [3]–[5]. Tight approximation of delay bounds for FIFO networks is possible, but very complex [6]–[9]. While deterministic NC – as treated here – computes worst case bounds, stochastic NC relaxes some assumption for large number of flows [10] and gives statistical bounds. Some authors have proposed to apply NC to Ethernet networks [11], [12].

NC requires that all flows in a network are known as well as their paths and their traffic profiles. Furthermore, the network, its capacities, and node delays are given. These are inputs for the NC analysis. Unfortunately, NC is expressed in rather difficult mathematical equations that require some time to get familiar with. While NC for a single link is well presented in most tutorial documents, the extension to an entire network is generally harder to understand. In this paper, we provide a simple and intuitive introduction to NC. We first explain NC on a single link and discuss performance tradeoffs resulting from multiplexing different flows. Then we revisit two ways from [13] to apply NC to an entire network and propose a third one. These algorithms are relatively simple and provide different delay bounds that we illustrate under various networking conditions.

The paper is structured as follows. Section II explains NC for a single link. Performance tradeoffs are illustrated for small example scenarios in Section III. Section IV revisits several extensions for the application of NC to entire networks and again discusses performance tradeoffs. Finally, Section V summarizes this work and gives an outlook on further research.

This work was funded by Hirschmann Automation and Control GmbH. The authors alone are responsible for the content of the paper.

The authors thank Jens Schmitt for his help to get acquainted with Network Calculus.

II. Network Calculus for a Single Link

NC for a single link requires traffic descriptions for flows which are expressed by arrival curves. They may characterize, e.g., the rate and burstiness of the flows. The transmission properties such as processing and transmission delay and transmission rate of a link are described by a service curve. The arrival curve of a flow and the service curves of a link can be used to calculate performance metrics like backlog, virtual delay, and output bound, which is a traffic description of the flow after link traversal. When multiplexing multiple flows, the transmission capacity of a link is shared so that individual flows experience a reduced service curve depending on the scheduling discipline. This leads to the concepts of a minimum and maximum service curve which need to be taken into account when calculating performance metrics.

A. Mathematical Definitions

NC uses functions to describe how much traffic passes a certain network element over time. They belong to a set of real-valued, non-negative, and wide-sense increasing functions with $f(0) = 0$:

$$\mathcal{F} = \{f : \mathbb{R}_0^+ \to \mathbb{R}_0^+, \forall t \geq s : f(t) \geq f(s), f(0) = 0\}. \quad (1)$$

Addition (Equation (2)), subtraction (Equation (3)), convolution (Equation (4)), and deconvolution (Equation (5)) are defined on this set of functions:

$$(f+g)(t) = f(t) + g(t) \quad (2)$$

$$[f-g]^+(t) = \max(f(t) - g(t), 0) \quad (3)$$

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t} (f(s) + g(t-s)) \quad (4)$$

$$(f \oslash g)(t) = \sup_{s \geq 0} \{f(t+s) - g(s)\}. \quad (5)$$

An example of the convolution is illustrated in Figure 2. The neutral element $e_\otimes$ for this operation ($e_\otimes \otimes f = f = f \otimes e_\otimes$) is the function $e_\otimes(t) = \begin{cases} 0 & t = 0 \\ \infty & t > 0 \end{cases}$. An example of the deconvolution is given in Figure 1. For $t = 0$ the deconvolution $(f \oslash g)(0)$ yields the maximum distance that f is above g. Note that $f \oslash g$ may not need to belong to $\mathcal{F}$ if both g and f belong to that class of functions.

B. Arrival Curve

Let $F(t)$ be the input flow and let $F^*(t)$ be the output flow of a link. These functions count the number of bytes that arrived at time t at the head-end and the tail-end node of a link, respectively.

An arrival curve $\alpha(t)$ gives an upper bound on the cumulative amount of data that can arrive within an interval of duration t from the input flow $F(s)$. That means

$$\alpha(t) \geq (F(s+t) - F(s)) \text{ for } s \geq 0. \tag{6}$$

From this follows $\alpha(0) = 0$. For $s = 0$ we get $\alpha(t) \geq F(t)$. Furthermore, if F is given, a lower bound for α is given by $F \oslash F$. A simple arrival curve consists of an initial data burst of size b and a maximum rate r. It is illustrated in Figure 1.

C. Minimum Service Curve

The output flow at the tail-end node of the link is constrained by the input flow and the ability of the link to transmit data. The latter is described by a minimum service curve $\beta(t)$. Given an input flow $F(t)$, the minimum service curve $\beta(t)$ allows to compute a lower bound on the output flow

$$F^*(t) \geq (F \otimes \beta)(t), \tag{7}$$

i.e., $F^*(t) \geq F(t-s) + \beta(s)$ holds for at least one $0 \leq s \leq t$. Assuming that the head-end node has sufficient data to send between $0 \leq t \leq t_0$, then $\beta(t)$ describes the minimum amount of traffic arrived at the tail-end node at time t.

The transmission ability of communication links can often be described by a rate-latency curve featuring an initial delay of T time and a constant rate R as depicted in Figure 1. The initial delay T models the processing delays of the head- and tail-end nodes of a link and the propagation delay between them; thus, a byte arrived at a non-backlogged head-end node at time t shows up at the tail-end node *not later* than $t+T$. The constant rate R models a minimum transmission capacity of the link. In some papers, the minimum service curve is called just service curve.

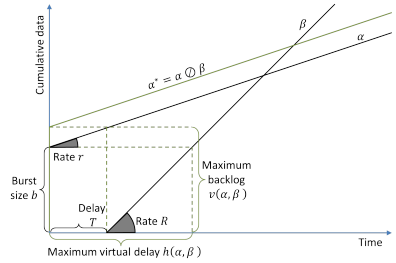

Fig. 1: The arrival curve α and a minimum service curve β allow for the computation of a maximum backlog $v(\alpha, \beta)$, maximum virtual delay $h(\alpha, \beta)$, and a maximum output bound α^*.

D. Backlog, Virtual Delay, and Output Bound

The backlog in the system is the amount of data that has already arrived at the sender but has not yet been received by the receiver. It is bounded by the vertical deviation $v(\alpha, \beta)$ between the arrival curve α and the service curve β. It is illustrated in Figure 1 and can be computed as

$$v(\alpha, \beta) = (\alpha \oslash \beta)(0). \tag{8}$$

The maximum virtual delay experienced by the data is bounded by the maximum horizontal deviation $h(\alpha, \beta)$ between the arrival curve α and the service curve β. It is illustrated in Figure 1 and can be computed as

$$h(\alpha, \beta) = \inf_{t \geq 0} \{(\alpha \oslash \beta)(-t) \leq 0\}. \tag{9}$$

In the following we refer to this formula by the algorithm DELAYBOUND(α, β).

The output flow $F^*(t)$ may be used as input flow for another link. Therefore, it is useful to give a bound α^* for that flow so that $\alpha^*(t) \geq F^*(t+s) - F^*(s)$ holds for all $s \geq 0$. Such an output bound can be computed as

$$\alpha^*(t) = \alpha \oslash \beta(t). \tag{10}$$

It is quite coarse because only a minimum service curve is given, i.e., the transmission capability of the link is not limited. In theory, the maximum backlog could be transmitted at once immediately followed by new traffic arriving at the head-end node of the link. This presents a worst-case that is exactly bounded by α^* which is also illustrated in Figure 1.

E. Maximum Service Curve

Given an input flow $F(t)$, the maximum service curve $\overline{\beta}(t)$ allows to compute an upper bound on the output flow

$$F^*(t) \leq (F \otimes \overline{\beta})(t) \tag{11}$$

because $F^*(t) \leq F(t-s) + \overline{\beta}(s)$ must hold for any $0 \leq s \leq t$. The maximum service curve $\overline{\beta}(t)$ describes the maximum amount of traffic arrived at the tail-end node at time t. It may also be expressed by a rate-latency curve. However, the interpretation is different. A byte arrived at a head-end node at time t can show up at the tail-end node *not earlier* than $t+T$. The constant rate R models a maximum transmission capacity of the link.

Figures 2 and 3 show two different maximum service curves $\overline{\beta}$. The one in Figure 2 reveals the same delay T as the minimum service curve β and differs only by a larger rate R. In contrast, the maximum service curve $\overline{\beta}$ in Figure 3 differs from the minimum service curve both by a lower delay T and a larger rate R.

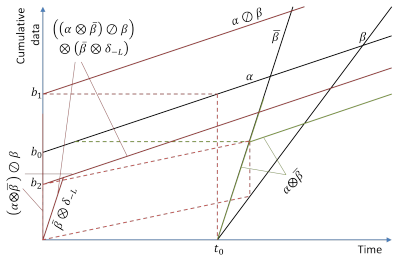

Fig. 2: The minimum service curve differs from the maximum service curve only by its rate R.

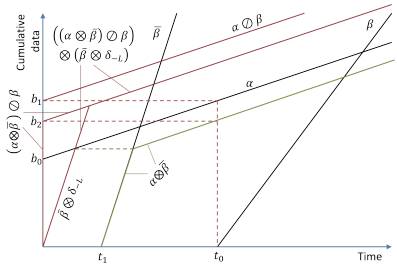

Fig. 3: The minimum service curve differs from the maximum service curve by its rate R and delay T.

F. Lower Output Bound due to Maximum Service Curves

The arrival curve $\alpha(t)$ is an upper bound on the traffic $F(t)$ arrived at the head-end node at time t and the maximum service curve $\overline{\beta}$ provides an upper bound on the transmission capability of a link. They allow for the computation of an upper bound on the traffic $F^*(t)$ arrived at the tail-end node by $(\alpha \otimes \overline{\beta})(t)$. The minimum service curve β also provides a lower bound on the output traffic $F^*(t)$. Therefore, a tighter bound of the output flow can be calculated by

$$\alpha^*(t) = ((\alpha \otimes \overline{\beta}) \oslash \beta)(t). \tag{12}$$

It is lower than the one given in Equation (10). Figure 2 shows that this output bound can be even tighter than the one for the input flow if the maximum service curve differs from the minimum service curve only by its rate R. According to Figure 3, it can also be larger than $\alpha(t)$ and close to the output bound given by Equation (10). The latter phenomenon is observed because the maximum service curve deviates a lot from the minimum service curve.

Schmidt and Zdarsky [14] showed that the output bound in Equation (12) can be improved by

$$\alpha^*(t) = (((\alpha \otimes \overline{\beta}) \oslash \beta) \otimes (\overline{\beta} \otimes \delta_{-L}))(t). \tag{13}$$

whereby $\overline{\beta} \otimes \delta_{-L}$ is a horizontal translation of the maximum service curve such that it becomes positive only for $t > 0$; in other words, the delay T of the service curve is removed. This additional convolution accounts for the fact that the maximum service curve has only a finite rate R. Figures 2 and 3 illustrate the effect. The bound for the output flow increases only with the maximum rate of the maximum service curve, bursts are not possible because any link speed is finite.

G. Multiplexing of Flows

When two flows f_0 and f_1 with arrival curves α_0 and α_1 are multiplexed on a link, they share this link's capacity. If the link provides a maximum service curve of β, then both flows experience that curve as maximum service curves $\overline{\beta_0}$ and $\overline{\beta_1}$. However, the experienced minimum service curves can be lower. If f_0 is prioritized over f_1 and β is strict[2], then f_0 experiences $\beta_0 = \beta$ as minimum service curve while f_1 experiences

$$\beta_1 = [\beta - \alpha_0]^+ \tag{14}$$

as "left-over" minimum service curve β_1 which has a larger delay and a lower rate than β. This is depicted in Figure 4. It leads to a larger virtual delay $h(\alpha_1, \beta_1)$ for f_1 compared to the virtual delay $h(\alpha_0, \beta_0)$ of f_0. Nevertheless, the maximum backlog $v(\alpha_0, \beta_0)$ of f_0 may still be larger than the one of f_1 ($v(\alpha_1, \beta_1)$). As the minimum service curve β_1 of f_1 is substantially lower than its maximum service curve $\overline{\beta_1}$, the output of f_1 may be more bursty than its input flow. We observe that in Figure 3 by the fact that all three differently computed output curves of f_1 quickly exceed their input curve.

Scheduling influences how capacity is shared among competing flows. If the scheduling discipline is unknown, priority scheduling must be assumed to guarantee bounds, which is called blind multiplexing [4, Sect. 6.2.1]. If the service discipline is first-in-first-out (FIFO), the left-over service curve β_1 is larger than for priority scheduling [4, Sect. 6.2.2].

[2]We say that a system offers a strict service curve β to a flow if, during any backlogged period of duration u, the output of the flow is at least equal to $\beta(u)$. [4, Def. 1.3.2]

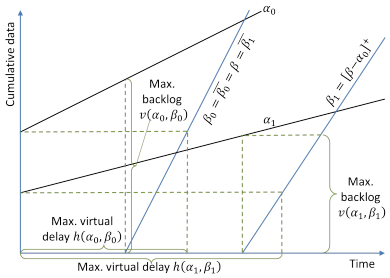

Fig. 4: Flows f_0 and f_1 are multiplexed on a common link whereby f_0 is served with absolute priority over f_1.

III. Performance Illustration

We perform experiments using the DelayLyzer [15] to show the impact of various parameters on delay bounds and backlog. We consider two competing flows f_0 and f_1, each of which has a rate of $r_0 = r_1 = 40$ Mb/s and a burst size of $b_0 = b_1 = 100$ kB by default. These parameters constitute arrival curves α_0 and α_1.

A. Impact of Priority Scheduling on Delay Bounds

We multiplex f_0 and f_1 onto a single link l with a rate of $R = 100$ Mb/s and a delay of $T = 1$ ms. These parameters constitute the maximum (and minimum) service curve $\overline{\beta} = \beta$ of the link. We vary the rates and burst sizes of f_0 and f_1 and evaluate their impact on delay bounds. First we compute the delay bound $D(f_0 + f_1)$ for the aggregate consisting of f_0 and f_1. Then we assume that f_0 is served with priority over f_1 (idealized with work-conserving service interruption) and compute the delay bounds $D(f_0)$ and $D(f_1)$ for f_0 (high-priority) and f_1 (low-priority). These metrics are calculated as follows:

$$D(f_0 + f_1) = h(\alpha_0 + \alpha_1, \beta) \tag{15}$$
$$D(f_0) = h(\alpha_0, \beta) \tag{16}$$
$$D(f_1) = h(\alpha_1, [\beta - \alpha_0]^+). \tag{17}$$

1) Constant Overall Traffic: In these series of experiments we keep the overall traffic aggregate constant but vary the traffic mix of high- and low-priority traffic by a parameter $0 < \lambda < 1$. First, we keep b_0 and b_1 at their default values and set the rates to $r_0 = \lambda \cdot 80$ Mb/s and $r_1 = (1 - \lambda) \cdot 80$ Mb/s.

Thus, λ is the rate fraction of f_0 in the overall aggregate, which is the fraction of high-priority traffic in case of priority scheduling. Figure 5 shows that the delays $D(f_0 + f_1)$, $D(f_0)$ and $D(f_1)$ depend on the traffic mix. As the traffic mix does not affect the characteristics of the overall aggregate, the delay bound $D(f_0 + f_1)$ is constant for varying λ and determined by the overall burst size b and the service curve β. The delay bound $D(f_0)$ for high-priority traffic is also constant because it depends only on the burst size b_0 and β. It is lower than $D(f_0 + f_1)$ because we have $b_0 < b$. In contrast, the delay bound $D(f_1)$ for the low-priority traffic rises with an increasing fraction of high-priority traffic because this increases the latency T_1 and reduces the rate R_1 of the left-over service curve β_1. The limit of the low-priority delay $D(f_1)$ for $\lambda \to 0$ is higher than the delay $D(f_0)$ of high-priority traffic. This is due to the fact that the high-priority

traffic still comes with a burst b_0 although its rate r_0 is very small because the values $\lambda \in \{0, 1\}$ are not considered in the figure.

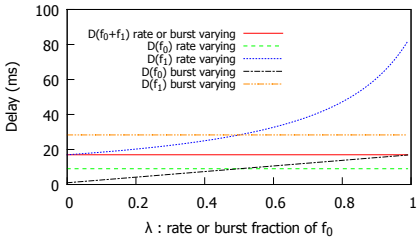

Fig. 5: Impact of traffic mix on delay bounds; f_0 is prioritized over f_1.

Second, we keep r_0 and r_1 at their default values and set the burst sizes to $b_0 = \lambda \cdot 200$ kB and $b_1 = (1 - \lambda) \cdot 200$ kB. Thus, λ is the burst fraction of f_0 in the overall aggregate. Figure 5 shows the delay bounds $D(f_0 + f_1)$, $D(f_0)$, and $D(f_1)$. The delay bound $D(f_0 + f_1)$ is constant for the same reason as above. The delay bound $D(f_0)$ for high-priority traffic increases linearly with the burst size b_0. The delay bound $D(f_1)$ for low-priority traffic does not depend on the traffic mix. On the one hand, an increased burst size b_0 of high-priority traffic reduces the left-over service curve β_1, but on the other hand it also decreases the burst b_1 according to the experiment setup, which exactly compensates the reduced service curve.

2) Constant Low-Priority Traffic: The output bound of a flow serves as input bound when the flow enters another link. Multiplexing can lead to extended output bounds which are maximum if flow f_0 stops sending when the maximum backlog of f_1 is reached. An output bound is described by a special curve rather than a single number. However, the backlog of the flow correlates with the initially increased rate of its output bound. Therefore, we use the backlog to visualize how the output bound of a flow can be increased by multiplexing. The backlog of f_1 is computed by

$$\beta_1 = [\beta - \alpha_0]^+ \quad (18)$$

$$B(f_1) = v(\alpha_1, \beta_1). \quad (19)$$

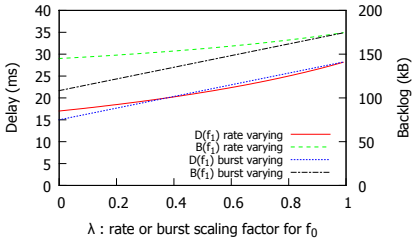

Fig. 6: Impact of high-priority traffic on delay bounds and backlog of the low-priority flow f_1.

We keep the rate and the burst of the low-priority flow f_1 constant at its default value. First, we vary the rate $r_0 = \lambda \cdot 40$ Mb/s of the high-priority traffic by the scaling factor $0 < \lambda < 1$. Figure 6 shows that the delay and the backlog of low-priority traffic increases slightly more than linearly with increasing high-priority traffic rate. Then, we keep the rate of the high-priority traffic constant at $r_0 = 40$ Mb/s and vary its burst size by $b_0 = \lambda \cdot 100$ kB, $0 \leq \lambda \leq 1$. Figure 6 shows that increasing the burst size significantly increases the delay and the backlog of f_1. Thus, multiplexing can increase the backlog of low-priority traffic and, thereby, its burstiness on a consecutive link. As a consequence, it is important to take the effect of multiplexing on prior links into account when calculating the output bound for a flow.

B. Impact of Multiplexing Multiplexed Traffic on Delay Bounds

As outlined above, multiplexing of two flows f_0 and f_1 on a link l_0 may increase the burstiness of the low-priority flow f_1. We consider that f_1 is multiplexed on the next link l_1 with another flow f_2. The output bound α_1^* of f_1 can be computed according to Equation (13) and may be used as input bound for f_1 on link l_1. Thus, the traffic characteristics of f_0 influence the delay experienced by f_2 although f_0 and f_2 do not share a common link. For ease of computation we assume that f_1 enjoys priority over f_2. Then, the delay bound for flow f_2 can be computed by

$$\alpha_1^* = ((\alpha_1 \otimes \beta) \oslash \beta_1) \otimes (\beta \otimes \delta_{-T_0}) \quad (20)$$

$$\beta_2 = [\beta - \alpha_1^*]^+ \quad (21)$$

$$D(f_2) = h(\alpha_2, \beta_2). \quad (22)$$

We quantify the impact of the traffic characteristics of f_0 on the delay of f_2 by two experiment series. The two links l_0 and l_1 have the same delay $T_0 = T_1 = 1$ ms and the same bandwidth $R_0 = R_1 = 100$ Mb/s. The two flows f_1 and f_2 have the same rate $r_1 = r_2 = 40$ Mb/s and burst size $b_1 = b_2 = 100$ kB.

First, flow f_0 has a burst size of $b_0 = 100$ kB and its rate is controlled by $r_0 = \lambda \cdot 40$ Mb/s, $0 < \lambda \leq 1$. Figure 7 shows that an increased rate of f_0 clearly increases the delay of f_2. In a similar experiment, we keep the rate of f_0 constant at $r_0 = 40$ Mb/s and control its burst size by $b_0 = \lambda \cdot 100$ kB, $0 \leq \lambda \leq 1$. The figure shows that an increasing burst size of f_0 also increases the delay of f_2.

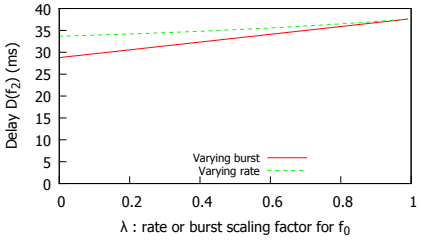

Fig. 7: Impact of traffic characteristics of f_0 on delay bounds of f_2.

C. The Pay-Bursts-Only-Once Phenomenon

We consider the transmission of a single flow f over two links l_0 and l_1 and are interested in the end-to-end delay bound $D(f, (l_0, l_1))$ of the flow. The flow is sent over two consecutive links l_0 and l_1. The associated service curves are $\beta(l_0)$ and $\beta(l_1)$. Link-specific delay bounds $D(f, (l_0))$ and $D(f, (l_1))$ may be computed for the flow. The delay bound $D(f, (l_0))$ is simple. The delay bound $D(f, (l_1))$ for the second link requires

the computation of the output bound α^* of the flow. The end-to-end delay bound $D^{lbl}(f,(l_0,l_1))$ is computed link-by-link as the sum of the link-specific delay bounds:

$$D(f,(l_0)) = h(\alpha,\beta(l_0)) \quad (23)$$
$$\alpha^* = (((\alpha \otimes \beta(l_0)) \oslash \beta(l_0)) \otimes (\beta(l_0) \otimes \delta_{-T_0}) \quad (24)$$
$$D(f,(l_1)) = h(\alpha^*,\beta(l_1)) \quad (25)$$
$$D^{lbl}(f,(l_0,l_1)) = D(f,(l_0)) + D(f,(l_1)). \quad (26)$$

Another way to compute the delay bound is to compute an end-to-end flow-specific service curve β_{e2e} which may be used to calculate an end-to-end delay bound $D^{PBOO}(f,(l_0,l_1))$ according to

$$\beta_{e2e} = \beta(l_0) \otimes \beta(l_1) \quad (27)$$
$$D^{PBOO}(f,(l_0,l_1)) = h(\alpha,\beta_{e2e}). \quad (28)$$

If the rate of the first link l_0 is larger than the rate of the second link ($R_0 > R_1$), then a burst of the flow may be shaped twice. In that case, $D^{lbl}(f,(l_0,l_1))$ may exceed $D^{PBOO}(f,(l_0,l_1))$. In fact, some traffic may face a maximum delay of $D(f,(l_0))$ on link l_0 and some other traffic may experience a maximum delay of $D(f,(l_1))$ on link l_1, but no traffic experiences an end-to-end delay longer than $D^{PBOO}(f,(l_0,l_1))$. This effect is known as "pay-bursts-only-once" (PBOO).

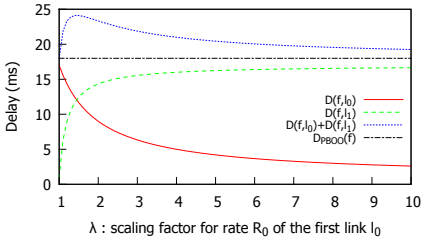

Fig. 8: Impact of the rate $R_0 = \lambda \cdot R_1$ of the second link on the result of two different delay bound calculations.

In the following we quantify this effect. The flow has rate $r = 80$ Mb/s and burst size $b = 200$ kB which provides an arrival curve α. The second link l_1 is the rate bottleneck with $R_1 = 100$ Mb/s and the rate of the first link is varied by $R_0 = \lambda \cdot R_1$ with $1 \leq \lambda \leq 10$, and each link has a delay of $T_0 = T_1 = 1$ ms. This constitutes $\beta(l_0)$ and $\beta(l_1)$. Figure 8 shows the maximum delay experienced by f on links l_1 and l_0 as well as the sum of both, depending on the scaling parameter λ for the rate R_0 of link l_0. The delay on link l_0 decreases with increasing bandwidth R_0, but the delay on link l_1 increases at the same time. The summarized end-to-end delay $D^{lbl}(f,(l_0,l_1))$ reveals a maximum. In contrast, the obtained values for $D^{PBOO}(f,(l_0,l_1))$ are independent of the rate R_0 and clearly smaller than the sum of the maximum delays experienced on individual links.

IV. Network Calculus Extensions for Networks

We calculate delay bounds for flows in a network. First, we outline required network inputs and clarify definitions. Then, we present three different methods to extend NC from a single link to an entire network: the Total Flow Analysis (TFA), a modified TFA (mTFA), and the Separated Flow Analysis (SFA) that leverage the concept of output bounds. The TFA and SFA variants were originally presented in [13]. We reformulate the algorithms so that they are easier to understand for networking engineers and present the improved mTFA variant. Furthermore, we compare the quality of the upper bounds obtained by the different variants in various experiments.

A. Notation

1) Representation of a Network: A communication network is represented by a set of nodes $\mathcal{V}$ and a set of directed communication links $\mathcal{E}$. Standard NC transforms such a representation into a graph of linked service nodes, the so-called server graph, where each link in the communication network becomes a node. This transformation is common in NC, but not necessary for our purpose. For the sake of simplicity, we stick to the topological structure of a communication network for the algorithm descriptions.

2) Some Useful Definitions: The descriptions of the algorithms require some standard notation about communication networks. The capacity of a link l is denoted by $c(l)$ and $d(l)$ is the sum of the processing delay of its head end node and the propagation delay of the link itself. The set of predecessor links of a link l is denoted by $Pred(l)$.

The analysis requires that all flows in the network are known. The source and destination node of a flow f are given by $src(f)$ and $dst(f)$. A path is a series of directed links $l_0,...,l_n$. The flow's path $path(f)$ can be derived from the forwarding based on the source and destination nodes of the flow. The function $lastLink(f)$ returns the last link of f within its path. The predecessor link of a specific flow f on link l is $pred(f,l)$. The aggregate of all flows sharing a subpath spanning from link m to link l is denoted by $g(m,l)$ which is well defined since we consider only a single operational path between any two nodes in the network.

The arrival curve $\alpha(f_x)$ of a flow f_x consists of a rate r_x and a burst size b_x. The service curve $\beta(l_y)$ of a link l_y consists of a bandwidth R_y and a delay D_y.

B. Output Bounds for Traffic from Predecessor Links

Algorithm 1 computes for a specific link l an output bound for the traffic aggregate $\mathbb{F}$ that is served with blind multiplexing in a network. To that end, this traffic aggregate is assumed to be served with low priority and all other traffic is served with high priority. The algorithm first summarizes the arrival curves $\alpha_{low}[l]$ and $\alpha_{high}[l]$ for high- and low-priority traffic. Required arrival curves for sub-aggregates $g(m,l)$ that do not originate on the same link are recursively computed, assuming they are served with low priority on previous links to maximize output bounds. Finally, the left-over service curve $\beta_{low}[l]$ for low-priority traffic is computed, as well as the output bound of the considered traffic aggregate using the procedure OutputBound; the latter may implement Equation (13).

During its recursive execution, the algorithm calculates and stores $\alpha_{low}[l]$, $\alpha_{high}[l]$, and $\beta_{low}[l]$ as global variables on every recursively considered link. Some links are visited even more than once. As the objective is to provide arrival curves and left-over service curves on the path of a considered flow f, the last stored variables must relate to traffic belonging to $\mathbb{F}$ and its competing traffic. To that end, recursive calls to

Algorithm 1 LOWPRIOOUTPUTBOUND: computes and stores arrival curve $\alpha_{low}[l]$ and minimum service curve $\beta_{low}[l]$ for low-priority traffic $\mathbb{F}$ on link l and returns its output bound.

Input: Link l, set of low-priority flows $\mathbb{F}$
$\alpha_{low}[l] \leftarrow 0$
$\alpha_{high}[l] \leftarrow 0$
{Classify flows with l as first link}
for all $f_g \in g(l,l)$ **do**
  if $l = firstLink(f_g)$ **then**
    if $f_g \in \mathbb{F}$ **then**
      $\alpha_{low}[l] \leftarrow \alpha_{low}[l] + \alpha(f_g)$
    else
      $\alpha_{high}[l] \leftarrow \alpha_{high}[l] + \alpha(f_g)$
    end if
  end if
end for
{Classify flows with l not as first link}
for all $m \in Pred(l)$ **do**
  if $g(m,l) \setminus \mathbb{F} \neq \emptyset$ **then**
    $\alpha_{high}[l] \leftarrow \alpha_{high}[l] +$ LOWPRIOOUTPUTBOUND$(m, g(m,l) \setminus \mathbb{F})$
  end if
  if $g(m,l) \cap \mathbb{F} \neq \emptyset$ **then**
    $\alpha_{low}[l] \leftarrow \alpha_{low}[l] +$ LOWPRIOOUTPUTBOUND$(m, g(m,l) \cap \mathbb{F})$
  end if
end for
$\beta_{low}[l] \leftarrow [\beta(l) - \alpha_{high}[l]]^+$
Output: OUTPUTBOUND$(\alpha_{low}[l], \beta(l), \beta_{low}[l])$

Algorithm 1 must first cover high-priority traffic and then low-priority traffic. The quantities $\alpha_{low}[l]$, $\alpha_{high}[l]$, and $\beta_{low}[l]$ will be used by following algorithms.

Note that Algorithm 1 terminates only if the paths of the flows cannot create any cycles. Therefore, the presented algorithms can be applied to feedforward networks only. This is a limitation for general routing, but not for Ethernet networks whose routing usually creates feedforward networks.

C. Total Flow Analysis (TFA)

We provide a reformulation of the TFA presented in [13]. TFA calculates the maximum delay that may be experienced by a considered flow f on any link along its path and adds these delays to provide an upper bound for the end-to-end delay of flow f. Thus, it does not implement the PBOO principle presented in Section III-C.

TFA calculates per-link delays that are experienced by the flow of interest f. To minimize obtained delay bounds, the delay is computed for a maximum aggregate that shares its path up to the destination of f. This phenomenon was illustrated in Section III-A2. This assumption is valid if all traffic within that aggregate is served FIFO. Algorithm 2 first calls LOWPRIOOUTPUTBOUND for the last link of f with all the traffic on that link, which calculates the arrival curves $\alpha_{low}[l]$ and service curves $\beta_{low}[l]$ that are experienced by aggregates sharing a common path with f up to its destination. The original version of TFA in [13] uses the line TFA in Algorithm 2, i.e., FIFO multiplexing is assumed only for the traffic that shares its path with the flow of interest up to its destination.

D. Modified Total Flow Analysis (mTFA)

We propose to assume FIFO multiplexing for all traffic on a link. This is modelled by the line $mTFA$ in Algorithm 2. This variant is called modified TFA (mTFA) and reduces link-specific delays compared to the TFA variant.

Algorithm 2 TFA_DELAYBOUND: computes end-to-end delay bound for flow f according to TFA. Either Variant TFA or Variant $mTFA$ is applicable.

Input: f: flow for which the delay bound is calculated; $\alpha_{low}[l], \beta_{low}[l]$: global variables for $l \in path(f)$
LOWPRIOOUTPUTBOUND$(lastLink(f), g(lastLink(f), lastLink(f))$
$delay \leftarrow 0$
for all $l \in path(f)$ **do**
  $< TFA : delay \leftarrow delay + h(\alpha_{low}[l], \beta_{low}[l]) >$
  $< mTFA : delay \leftarrow delay + h(\alpha_{low}[l] + \alpha_{high}[l], \beta(l)) >$
end for
Output: End-to-end delay $delay$ for flow f.

E. Separated Flow Analysis (SFA)

We provide a reformulation of the SFA presented in [13]. Algorithm 3 shows pseudo code for SFA. SFA calculates the end-to-end delay for a considered flow f on the basis of an end-to-end minimum service curve. This is computed by convoluting the minimum service curves for f on any link. They are determined by subtracting from the maximum service curve of a link the input bounds of the other traffic that shared the link with the considered flow f. These are again calculated using LOWPRIOOUTPUTBOUND. SFA implements the PBOO principle. However, the approach considers on each link any other traffic than f as high-priority so that the left-over service curve for f is rather low.

Algorithm 3 SFA_DELAYBOUND: computes end-to-end delay bound for flow f according to SFA.

Input: f: flow for which the delay bound is calculated
$\beta \leftarrow e_{\otimes}$
for all $l \in path(f)$ **do**
  $\alpha_{high} \leftarrow 0$
  for all $f_g \in g(l,l)$ **do**
    if $l = firstLink(f_g)$ AND $f_g \neq f$ **then**
      $\alpha_{high} \leftarrow \alpha_{high} + \alpha(f_g)$
    end if
  end for
  for all $m \in Pred(l)$ **do**
    $\alpha_{high} \leftarrow \alpha_{high} +$ LOWPRIOOUTPUTBOUND$(m, g(m,l) \setminus f)$
  end for
  $\beta_{low} \leftarrow [\beta(l) - \alpha_{high}]^+$
  $\beta \leftarrow \beta \otimes \beta_{low}$
end for
Output: End-to-end delay $h(\alpha(f), \beta)$ for flow f.

F. Comparison of the TFA and the SFA Method

We consider an example with two links l_0 and l_1 and with two flows f_0 and f_1. The flow f_1 crosses both link l_0 and l_1 while f_0 crosses only one of them. Both flows have a burst size of $b_0 = b_1 = 100$ kB and their overall rate is $r = 80$ Mb/s. In our experiments, we control the rate of f_0 by $r_0 = \lambda \cdot r$, $0 < \lambda < 1$. We calculate the delay bound with TFA, mTFA, and SFA and compare the results.

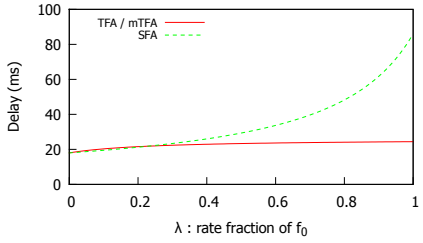

Fig. 9: $R_0 = 100$ Mb/s, f_0 crosses l_1.

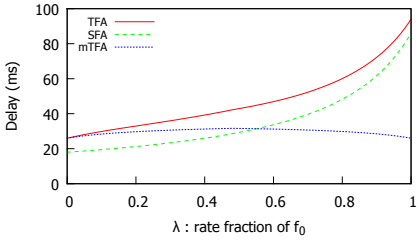

Fig. 10: $R_0 = 100$ Mb/s, f_0 crosses l_0.

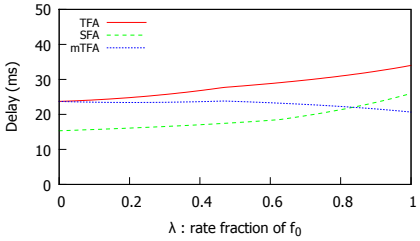

Fig. 11: $R_0 = 150$ Mb/s, f_0 crosses l_0.

1) Equal Bandwidths and Other Traffic on Last Link: In the first experiment, both links have a bandwidth of $R_0 = R_1 = 100$ Mb/s and a delay of $T_0 = T_1 = 1$ ms.

Flow f_0 crosses only link l_1. Figure 9 shows that TFA and mTFA lead to the same low delay while the delay computed with SFA increases with an increasing rate fraction of f_0. This is because SFA assumes that f_1 is scheduled with less priority than f_0 while TFA and mTFA assume that f_0 and f_1 are scheduled in a FIFO manner.

2) Equal Bandwidths and Other Traffic on First Link: In the second experiment, we use the same network, but flow f_0 crosses only link l_0. Figure 10 shows that SFA computes about the same large delays as before in Section IV-F1.

The delay computed by TFA even exceeds the one calculated by SFA because it cannot aggregate f_0 and f_1 for FIFO scheduling. The delay values of mTFA are somewhat larger than in the previous experiment. They are larger than those for SFA if the rate fraction of r_0 is low, but they are lower if the rate fraction of r_0 is high. This is due to the FIFO scheduling assumption of mTFA on any link for any traffic while SFA assumes low priority only for the flow of interest.

3) Unequal Bandwidths and Other Traffic on First Link: In the third experiment, link l_0 has an increased bandwidth of $R_0 = 150$ Mb/s, all other parameters are like in the experiment above (Section IV-F2).

Figure 11 illustrates that the delay values of TFA are lower than in the experiment before which is due to the increased bandwidth on link l_0. We make the same observation for SFA. Its delay bounds are lower than those of mTFA for a larger range of rate fractions of f_0 because it leverages the PBOO phenomenon.

V. Conclusion

In this paper we have revisited simple extensions of Network Calculus (NC) from a single link to an entire network that are applicable for feed-forward networks such as Ethernet. We consider FIFO multiplexing in the network, but algorithms assume high priority for certain sub-aggregates to compute output bounds.

We explained NC for a single flow on a single link, then multiplexing of a high-priority flow and a low-priority flow for both priority and FIFO scheduling, and illustrated performance tradeoffs. For the NC-based investigation of networks, we considered the TFA and the SFA approach and proposed another modified TFA (mTFA). The Total Flow Analysis (TFA) assumes FIFO for some sub-aggregates for the calculation of link-specific delay bounds, which reduces bounds for end-to-end delay. mTFA uses this principle even more. The advantage of the Separated Flow Analysis (SFA) is the fact that it implements the pay-bursts-only-once (PBOO) idea which improves delay bounds compared to TFA under some conditions. Our experiments showed that mTFA leads to tighter delay bounds than TFA. mTFA outperforms SFA in some cases as it makes use of the FIFO scheduling assumption where possible. In other cases, SFA outperforms mTFA since it leverages PBOO. Thus, the minimum of SFA and mTFA may be used for simple delay calculation in networks with FIFO service.

References

[1] R. L. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation," *IEEE Transactions on Information Theory*, vol. 37, pp. 114–131, Jan. 1991.

[2] ——, "A Calculus for Network Delay, Part II: Network Analysis," *IEEE Transactions on Information Theory*, vol. 37, pp. 132–141, Jan. 1991.

[3] C.-S. Chang, *Performance Guarantees in Communications Networks*. Springer, 2001.

[4] J.-Y. Le Boudec and P. Thiran, *Network Calculus*. Springer, 2004.

[5] M. Fidler, "A Survey of Deterministic and Stochastic Service Curve Models in the Network Calculus," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 1, pp. 59 – 86, 2010.

[6] L. Lenzini, L. Martorini, and G. Stea, "Tight End-to-End per-Flow Delay Bounds in FIFO Multiplexing Sink-Tree Networks," *Performance Evaluation*, vol. 63, no. 9, pp. 956 – 987, Oct. 2006.

[7] L. Lenzini, E. Mingozzi, and G. Stea, "A Methodology for Computing End-to-End Delay Bounds in FIFO-Multiplexing Tandems," *Performance Evaluation*, vol. 65, no. 11 – 12, pp. 922 – 943, Nov. 2008.

[8] J. B. Schmitt, F. A. Zdarsky, and M. Fidler, "Delay Bounds under Arbitrary Multiplexing: When Network Calculus Leaves You in the Lurch..." in *IEEE Infocom*, 2008.

[9] A. Bouillard, L. Jouhet, and E. Thierry, "Tight Performance Bounds in the Worst-Case Analysis of Feed-Forward Networks," in *IEEE Infocom*, Mar. 2010.

[10] F. Ciucu and J. Schmitt, "Perspectives on Network Calculus – No Free Lunch, but Still Good Value," in *ACM SIGCOMM*, 2012.

[11] J.-P. Georges, T. Divoux, and E. Rondeau, "Network Calculus: Application to Switched Real-Time Networking," in *VALUETOOLS*, 2011.

[12] S. Kerschbaum, K.-S. J. Hielscher, U. Klehmet, and R. German, "Network Calculus: Application to an Industrial Automation Network," in *GI/ITG MMB*, 2012.

[13] J. B. Schmitt and F. A. Zdarsky, "The DISCO Network Calculator – A Toolbox for Worst Case Analysis," in *VALUETOOLS*, 2006.

[14] J. B. Schmitt, F. A. Zdarsky, and I. Martinovic, "Performance Bounds in Feed-Forward Networks under Blind Multiplexing," University of Kaiserslautern, Tech. Rep. 349/06, Jul. 2006.

[15] M. Schmidt, S. Veith, M. Menth, and S. Kehrer, "DelayLyzer: A Tool for Analyzing Delay Bounds in Industrial Ethernet Networks," in *GI/ITG MMB*, Bamberg, Germany, 2014.