Multicast Extension for Network Calculus

Mark Schmidt, Thomas Schneck, and Michael Menth

Chair of Communication Networks (KN),University of Tuebingen
[mark-thomas.schmidt@,thomas.schneck@student,menth@]uni-tuebingen.de

Factory automation for Industry 4.0 leverages Industrial Ethernet to connect sensors and actuators for control purposes. This requires communication with little packet loss and delay. Such requirements can be supported with Network Calculus (NC) which calculates delay bounds for communication in packet-switched networks. However, multiple applications in factory automation rely on multicast communication which is currently not supported by existing NC theory and tools.

In this paper we extend a simple NC method such that delay bounds for multicast flows can be calculated and integrate it in our analysis tool DelayLyzer. We use the DelayLyzer to illustrate the dependency of delay bounds in a multicast tree on cross traffic and link bandwidths and compared it to other methods. We further applied the DelayLyzer to a use case in the area of substation automation to demonstrate the applicability of our method in practice.

1 Introduction

Industrial facilities of manufacturing and utilities currently witness a dramatic change towards more automation for which packet-switched networks are the base for communication. This machine-to-machine communication often relies on Industrial Ethernet and sometimes requires hard bounds on packet loss and delay. In such environments, communication patterns and traffic properties are rather predictable. Therefore, maximum delays can be estimated for a set of flows to ensure that network resources are sufficient to guarantee their delay requirements.

Network Calculus (NC) is a method specifically developed to determine upper delay bounds in communication networks and may be used in this context. However, typical industrial applications involve also multicast communication which cannot be treated by conventional NC algorithms. The contribution of this paper is an extension of NC to multicast traffic. It is integrated into the DelayLyzer [1] tool and evaluated with a realistic use case for electrical substation automation.

The paper paper is structured as follows. Section 2 gives a short introduction to NC. In Section 3 we review a simple NC algorithm to compute delay bounds for unicast traffic and and adapt it to multicast traffic. We illustrate the impact of traffic and network characteristics on calculated delay bounds for multicast traffic and compare it with a simple approximation through multiple unicast flows in Section 4. In Section 5 we provide a short overview about an electrical substation according to IEC 61850 and demonstrate that the proposed approach is applicable for the substation automation use case. Section 6 concludes this work.

2 Introduction to Network Calculus

We first explain basics about how network calculus computes performance metrics for a single link, and then we clarify how this method can be applied to an entire network.

2.1 Network Calculus Applied to a Single Link

NC is a mathematical framework developed by Cruz [2,3], Chang [4] and Le Boudec [5] to determine delay bounds in communication networks. NC models the maximum amount of data delivered by a flow to a link or a node within a given time. This behavior can be described by a token-bucket and modeled by a rate-burst curve. More complex models use generalized piece-wise linear functions of time. In a similar way, the delay added by the processing in links and nodes can be expressed by a rate-latency curve, the so-called service curve. NC provides operations on curves that eventually facilitate the computation of maximum delay and backlog on a link. In addition, an output bound for a flow can be calculated to characterize its timely behavior after having passed a specific link or node. The approach is scientifically backed by an algebraic theory.

2.2 Network Calculus Applied to a Network

To extend NC from a single link to a network, the output bound of a flow from a predecessor link is taken as arrival curve for the next link. Schmitt and Zdarsky developed simple algorithms to calculate the maximum delay bound for a flow when traversing a series of nodes and links in a feed-forward network [6, 7]. They proposed two simple variants: *Total Flow Analysis* (TFA) and *Separated Flow Analysis* (SFA). In addition, other authors proposed more complex algorithms leading to tighter delay bounds [8–11].

Forwarding and protection mechanisms in Ethernet networks like the spanning tree protocol (STP) are relatively simple and lead to feed-forward networks so that NC can be used for the computation of delay bounds in that context.

3 Multicast Extension for Network Calculus

There is a multitude of NC algorithms [12] that distinguish in accuracy, complexity and computational effort. We choose the rather simple TFA algorithm [6] as a base and extend it for multicast traffic. In a first step, we develop *linkwise TFA* (LTFA) by applying the idea of TFA to every link on the path of a flow, leading to increased computational effort and to slightly improved delay bounds. This modification is restricted to first-in-first-out packet scheduling, a condition fulfilled in typical Ethernet networks. In a second step, we define a point-to-multipoint data structure for multicast flows which allows the application of LTFA (MC-LTFA). As a result, an upper delay bound can be predicted for every link on the path of a multicast flow. This allows the computation of end-to-end delay for each destination within the multicast tree. Our approach results in rather low computational effort as many intermediate results may be reused and additionally in smaller delay bounds than for TFA.

In the following, we give some common notations. We present the helper function LowPrioOutputBound which is part of the original TFA and our novel LTFA algorithm. We first present the linkwise TFA (LTFA) for unicast flows and then the *multicast LTFA* (MC-LTFA) for multicast flows.

3.1 Notation

We use the following notations in the code listings presenting our algorithms. The set of all flows in a network is denoted by $\mathcal{F}$. The arrival curve of a flow f is named $\alpha(f)$. $\alpha_{low}[l]$ and $\alpha_{high}[l]$ indicate the arrival curves on link l for low-priority and high-priority traffic. The minimum service curve of link l for a set of low-priority flows is called $\beta_{low}[l]$. The path of a flow f is an ordered list of links by path(f), starting from the destination and ending at the source. For two neighboring links m and n the set of all flows using link m followed by link n is denoted $g(m, n)$. The function pred(f, l) yields the predecessor link of flow f relative to link l on its path path(f). The function firstLink(f) yields the first link of flow f on the path from source to destination.

3.2 "LowPrioOutputBound"

The algorithm *LowPrioOutputBound* computes the output bound for a set of low-priority flows $\mathcal{A}$ on a link l. It has been essentially presented in [6] but in a different notation. We modified the original algorithm to facilitate its application to multicast.

The algorithm is given in Algorithm 1. It has a set of low-priority flows $\mathcal{A}$ and a considered link l as arguments. It first computes the arrival curves $\alpha_{low}[l]$ and α_{high} for low- and high-priority flows. First, flows f entering the network at link l are considered by taking their original arrival curve $\alpha(f)$, then other flows are considered by recursively calculating their arrival curves as output bounds on their predecessor links. Then, the service curve $\beta_{low}[l]$ remaining for low-priority traffic $\mathcal{A}$ using the overall service curve $\beta[l]$ of link l. Finally, *LowPrioOutputBound* calculates the output bound for aggregate $\mathcal{A}$ on link l using algorithm *OutputBound* which is further described in [13].

In this algorithm, all arrival and service curves are globally available. However, only $\alpha(f)$ and $\beta[l]$ are given while other variables like $\alpha_{low,high}[l]$ or $\beta_{low}[l]$ are incrementally computed. Therefore, the contents of these variables depends on the calculation order.

Algorithm 1 LowPrioOutputBound: computes and stores arrival curve $\alpha_{low}[l]$ and minimum service curve $\beta_{low}[l]$ for traffic $\mathcal{A}$ on link l and returns its output bound.

Input: Link l, set of low-priority flows $\mathcal{A}$
  $\alpha_{low}[l] \leftarrow 0$
  $\alpha_{high}[l] \leftarrow 0$
    {Classify flows with l as first link**}**
  for all $f \in g(l,l)$ **do**
    if $l = \text{firstLink}(f)$ **then**
      if $f \in \mathcal{A}$ **then**
        $\alpha_{low}[l[\leftarrow \alpha_{low}[l[+\alpha(f)$
      else
        $\alpha_{high}[l[\leftarrow \alpha_{high}[l] + \alpha(f)$
      end if
    end if
  end for
    {Classify flows with l not as first link**}**
  for all $m \in pred(f)$ **do**
    $\alpha_{high}[l] \leftarrow \alpha_{high}[l]+$ LowPrioOutputBound$(m, g(m,l) \setminus \mathcal{A})$
    $\alpha_{low}[l] \leftarrow \alpha_{low}[l]+$ LowPrioOutputBound$(m, g(m,l) \cap \mathcal{A})$
  end for
  $\beta_{low}[l] \leftarrow [\beta[l] - \alpha_{high}[l]]^+$
Output: OutputBound$(\alpha_{low}[l], \beta[l], \beta_{low}[l])$

3.3 Linkwise Total Flow Analysis (LTFA)

The LTFA algorithm calculates the end-to-end delay of a flow and is described in Algorithm 2. It calculates the maximum delay incurred on each link of a flow's path. To that end, the arrival curve for the overall traffic on that link is computed by LowPrioOutputBound together with $\beta_{low}[l]$ equaling in this case $\beta[l]$. The link-specific delay is computed by $h(alpha_{low}[l], \beta_{low}[l])$. This is the maximum horizontal distance between $alpha_{low}[l]$ and $\beta_{low}[l]$, see [13] for more details. The sum of all link-specific delays yields the end-to-end delay.

This approach is valid only for FIFO traffic forwarding as we do not differentiate between low- and high-priority traffic on the links when calculating LowPrioOutputBound.

Compared with the original TFA in [6], LTFA leads to increased execution times because *LowPrioOutputBound* is called multiple times for links that are closer to a flow's source. However,

the modification is needed for the following multicast extension.

Algorithm 2 LINKWISE TOTAL FLOW ANALYSIS: computes end-to-end delay bound for (unicast) flows f according to LTFA.

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

3.4 Multicast Linkwise Total Flow Analysis (MC-LTFA)

A multicast flow f has a tree as path layout path(f) as it involves many destinations $\mathcal{D}(f)$. The path from a specific destination $d \in \mathcal{D}(f)$ to its source is obtained as an ordered list by path(f, d). The objective is now to compute delay bounds for all destinations $d \in \mathcal{D}(f)$ in a multicast flow f.

The algorithm for MC-LTFA is given in Algorithm 3. First, the delays $delay[l]$ for all links within a flow are calculated. Then, the delays $delay[d]$ for each destination $d \in \mathcal{D}(f)$ are first initialized with zero and then computed by summing the delays of the links contained in the flow's path from destination to source. This algorithm can be scaled for networks with many flows by first calculating the delay for all links and then calculating the delay for all destinations of all flows without calculating the delay for each link again.

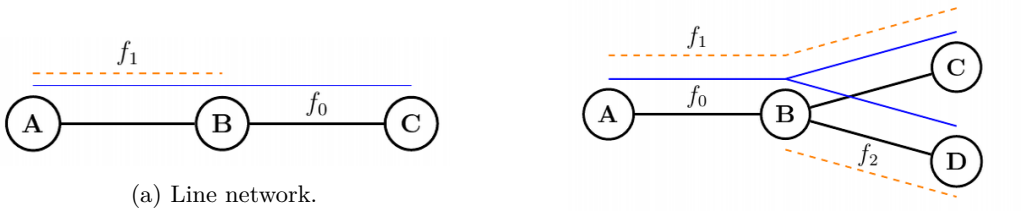
As an optimization all required intermediate results are reused during the calculation, so that the required operations of the algorithm and execution time can be reduced significantly.

Algorithm 3 MULTICAST LINKWISE TOTAL FLOW ANALYSIS: computes end-to-end delay bounds for a multicast flows f.

Input: f: multicast flow with $\mathcal{D}(f)$ destinations
  for all $l \in \text{path}(f)$ **do**
    LOWPRIOOUTPUTBOUND$(l, g(l,l))$
    $delay[l] \leftarrow h(\alpha_{low}[l], \beta_{low}[l])$
  end for
  for all $d \in \mathcal{D}(f)$ **do**
    $delay[d] \leftarrow 0$
    for all $l \in path(f, d)$ **do**
      $delay[d] \leftarrow delay[d] + delay[l]$
    end for
  end for
Output: End-to-end delays $delay$ for the $\mathcal{D}(f)$ destinations of the multicast flow f.

4 Comparison of (MC-)LTFA with Other NC Methods

In this section, we compare our novel (MC-)LTFA method with the two methods from [6] on a line network and on a Y-network. We perform the experiments with the DelayLyzer [1,13] tool in which we integrated our new methods LTFA and MC-LTFA.



(a) Line network.

(b) Y-network.

Figure 1: Topology and flows of investigated sample networks.

4.1 Comparison of LTFA with TFA and SFA in a Line Network

In the this experiment, we use the simple line network in Figure 1a consisting of three nodes (A, B, C) and the two links l_{AB} and l_{BC}. The links l_{AB} and l_{BC} between the nodes have a delay of 0.1 ms. The flows f_0 and f_1 share the link l_{AB}. Flow f_0 has a burst of 100 KB and a rate of 15 Mb/s and flow f_1 has a burst of 1 KB and a rate of 3 Mb/s. Link l_{BC} has a bandwidth of 30 Mb/s. In the following, we study upper bounds on the delay of flow f_0 depending on the bandwidth of link l_{AB} in a range from 20 Mb/s to 100 Mb/s.

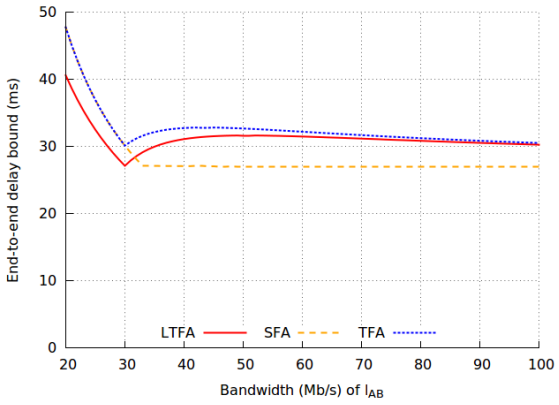


Figure 2: End-to-end delay bounds for flow f_0 in the line network depending on the bandwidth of link l_{AB}.

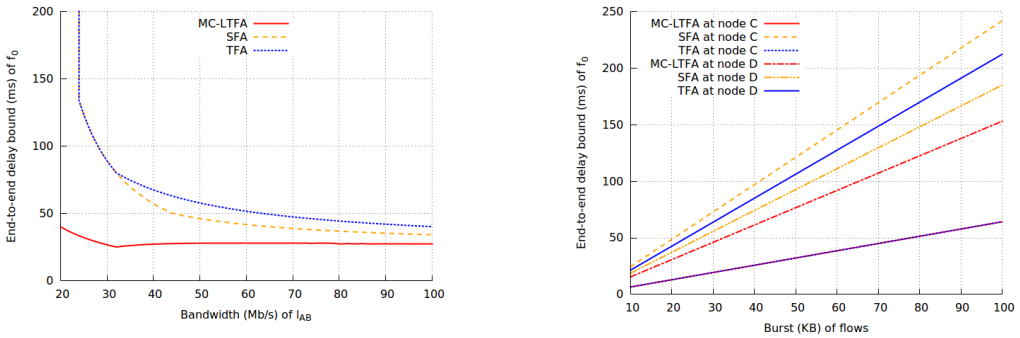
We compare the results from the NC methods LTFA, TFA, and SFA. The latter two are NC methods proposed in [6] and are simple but they produce delay bounds for blind multiplexing which leads to larger delay bounds than FIFO multiplexing. The results are presented in Figure 2. LTFA leads to lowest delay bounds for a bandwidth of l_{AB} smaller than 32 Mb/s because LTFA assumes all flows on link l_{AB} having the same priority while TFA and SFA assume that f_1 has higher priority than f_0. At a rate of 30 Mb/s on l_{AB}, link l_{BC} is the bottleneck. As a consequence, traffic may be delayed at l_{BC}, increasing the overall delay for TFA and LTFA from that point on. The SFA computes an end-to-end service curve, taking into account that f_0 has been already sufficiently shaped at l_{AB}. This effect is known as the pay busts only once (PBOO) phenomenon. Therefore, SFA calculates lower end-to-end delays than TFA and LTFA for a bandwidth larger than 33 Mb/s on l_{AB}. However, SFA is more complex and cannot be easily extended to multicast flows. Moreover, SFA leads to larger delays than LTFA in other scenarios because SFA assumes that cross traffic has higher priority than the considered end-to-end flow.

4.2 Comparison of MC-LTFA with TFA and SFA in a Y-Network

We now consider two approaches to adapt NC methods to multicast flows. The first approach treats a multicast flow as separate unicast flows. The second approach considers only a subtree of the multicast flow and requires multiple runs to compute end-to-end delay bounds for all destinations of the considered multicast flow. For both experiments we use the Y-network depicted in Figure **??**. It consists of four nodes A, B, C and D and 3 links l_{AB}, l_{BC} and l_{BD}.

4.2.1 Multicast Approximation by Multiple Unicast Flows

In the first experiment, we consider only flow f_0 with a rate of 12 Mb/s and a burst size of 100 KB. We assume that links l_{BC} and l_{BD} have a bandwidth of 32 Mb/s and a latency of 0.1 ms. We study the effect of the bandwidth of l_{AB} in the range from 20 Mb/s to 100 Mb/s on the delay bounds of the multicast flow at its destinations. They are shown in Figure 3b. The end-to-end delay bound computed with MC-LTFA is below 50 ms for all considered bandwidth values of link l_{AB}. It decreases if the bandwidth for link l_{AB} increases from 20 to 32 Mb/s and remains at about the same value for even larger bandwidths. This phenomenon is due to the fact that from 32 Mb/s on, links l_{BC} and l_{BD} are the bottlenecks so that increasing the bandwidth of l_{AB} cannot lower the end-to-end delay bound of f_0. We also compute delay bounds with the TFA and SFA method by assuming two simultaneous unicast flows from A to C and from A to D with the same properties of f_0. This is a naive adaptation of these methods to multicast flows. Figure 3b shows that the end-to-end delay bound is infinity for a bandwidth on l_{AB} of 24 Mb/s or lower because this equals the overall rate of the two simultaneous flows. The delay bounds decrease with increasing bandwidth of link l_{AB} but are still significantly larger than with MC-LTFA.



(a) Impact of the bandwidth of link l_{AB}; bandwidth of link l_{BC} is assumed 32 Mb/s, the rate of f_0 is 12 Mb/s, and its burst size 100 KB.

(b) Impact of burst sizes of flows f_0, f_1, and f_2; their rates are 12 Mb/s and the bandwidth of all links is 25 Mb/s.

Figure 3: End-to-end delay bounds of multicast flow f_0 in the Y-network.

4.2.2 Multicast Approximation by Multiple Evaluations of Selected Multicast Destinations

In the second experiments, flows f_0, f_1, and f_2 as shown in Figure 1b have a rate of 12 Mb/s and varying burst sizes and the links have a bandwidth of 25 Mb/s.

We now compute the end-to-end delay bounds for the multicast flow from A to C and D, respectively, with MC-LTFA, TFA, SFA. To compute the delay for destination C with TFA and SFA, we just cut off the branch from B to D to obtain a unicast flow. Figure 3b shows the results. MC-LTFA produces the lowest delay bounds. The delay bound for destination C is lower than for D. This is intuitive because the flow from A to C is multiplexed only once at node A with the competing flow f_1 while traffic does not suffer multiplexing delay at node B. This is different for destination D whose traffic is multiplexed twice: once with flow f_1 at node A and once with flow f_2 at node B which adds extra delay.

The results for TFA are similar. For destination C, TFA assumes both f_0 and f_1 to be low priority both on l_{AB} and l_{BC} so that TFA produces the same results as MC-LTFA. This is different for destination D because TFA assumes flow f_1 high-priority while the considered flow f_0 is assumed to be low-priority which leads to larger delay bounds on l_{AB} compared to MC-LTFA.

With SFA, the delay bounds are larger than for TFA and MC-LTFA. This is because SFA assumes both f_1 and f_2 to be scheduled with higher priority than f_0 leading to larger delay bounds than the other two methods. Moreover, SFA produces larger delay bounds for destination C than for destination D which is counter-intuitive. This can be explained as follows. For destination D, the considered flow f_0 is multiplexed with f_1 and with f_2, assuming them to be high-priority, but having their original flow description. With destination C, SFA computes the left-over bandwidth for f_0 taking into account f_1 with its original flow description on l_{AB} and taking into account f_1 with an aggravated flow description on l_{BC}.

Thus, MC-LTFA produces also smaller delay bounds than TFA and SFA when applied to sub-flows of the multicast flow, essentially because TFA and SFA are intended for blind multiplexing instead of FIFO multiplexing.

5 Use Case: Substation Automation

In this section we give a short introduction to communication in electrical substations which involves multicast traffic. We compute delay bounds for the traffic using MC-LTFA, showing that our method can be applied for realistic use cases.

5.1 Introduction to IEC 61850

An electrical substation is a facility that transforms high voltage powerlines coming from a power plant or transmission lines to lower voltage lines that are used by the consumers. IEC 61850 is a set of standards that describe the components and communication of an electrical substation. The parts [14–17] focus on communication.

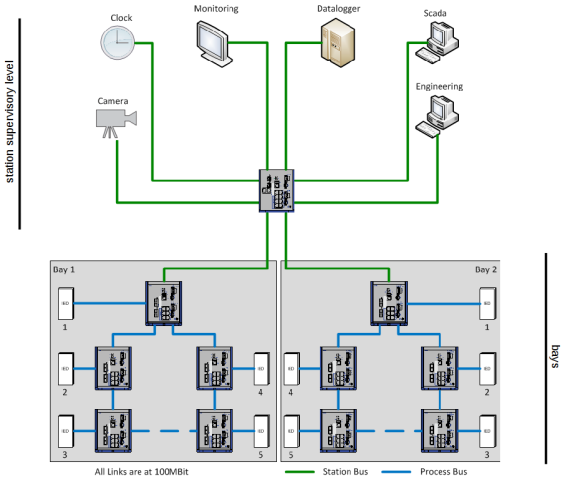


Figure 4: Topology of an electrical substation.

Figure 4 illustrates the typical setup of an electrical substation. It consists of a *station supervisory level* which is the management and maintenance part and *bays* which contain transformer stations and other electrical components in the field, called Intelligent Electronic Devices (IEDs). The station supervisory level provides work stations for employees and SCADA terminals to manage the bays. Additionally, it includes security cameras and monitoring devices used to check the substation and data-logging devices to store measured data. To manage the different components and ensure that all devices work synchronously, a precise clock synchronizes all devices

with the substation. The *Station Bus* facilitates communication within the station supervisory level and between the station supervisory level and the bays. The *Process Bus* interconnects the IEDs within a bay and is usually implemented through a ring network. The Media Redundancy Protocols (MRP) [18] is used in the bays to rearrange the forwarding in case of a link failure.

We now give an overview of the traffic within a substation. The flows between the different components are classified into different flow classes. Generic Substation Events (*GOOSE*) traffic is used for control purposes within a bay of a substation and is typically distributed by multicast. Control traffic among different bays is specified in the Manufacturing Message Specification (*MMS*) class and is distributed by unicast. Measurement values are needed by control components and are called Sampled Values (*SV*). They are transported from the EIDs using publish/subscribe mechanisms and are distributed by multicast. Traffic used to synchronize the clocks in all devices within the substation is called *CLOCK* and is sent either by unicast or multicast.

For all flows in the classes GOOSE, MMS, SV, and CLOCK, a flow specification including a priority, a traffic description, a maximum allowed delay, and communication protocol is provided. This facilitates the modelling of the communication among all devices.

5.2 Delay Analysis for an Electrical Substation

In this use case, the clock data is distributed via multicast to all other devices in the substation. It is used for synchronization purposes and has tight realtime constraints. Therefore, we determine the maximum delay from the clock towards all devices in the network, which should be done in the planning phase of the network before roll-out. We investigate the delay bounds for 100 Mb/s and 1 Gb/s Ethernet technology because 100 Mb/s technology is preferred for two reasons: network equipment for 100 Mb/s is cheaper and energy consumption is lower than for 1 Gb/s.

We performed the delay analysis with the DelayLyzer tool and modelled 19 flows in the different traffic classes of IEC 61850, including the multicast flow from the clock to all other devices. These flows are compiled in Table 1.

Flow	**Rate (kb/s)**	**Burst (KB)**	**Delay Bound (μs)**	**source**	**destination(s)**
GOOSE_IED1.2_to_IED1.1	0.8	0.1	500	IED_1.2	IED_1.1
GOOSE_IED1.5_to_IED2.4	0.8	0.1	500	IED_1.5	IED_2.4
GOOSE_IED2.1_to_IED2.4	0.8	0.1	500	IED_2.1	IED_2.4
GOOSE_IED2.3_to_IED2.2	0.8	0.1	500	IED_2.3	IED_2.2
MMS_IED1.1_to_Scada	10	0.25	1000	IED_1.1	Scada
MMS_IED1.2_to_Scada	10	0.25	1000	IED_1.2	Scada
MMS_IED1.3_to_Scada	10	0.25	1000	IED_1.3	Scada
MMS_IED1.4_to_Scada	10	0.25	1000	IED_1.4	Scada
MMS_IED1.5_to_Scada	10	0.25	1000	IED_1.5	Scada
MMS_IED2.1_to_Scada	10	0.25	1000	IED_2.1	Scada
MMS_IED2.2_to_Scada	10	0.25	1000	IED_2.2	Scada
MMS_IED2.3_to_Scada	10	0.25	1000	IED_2.3	Scada
MMS_IED2.4_to_Scada	10	0.25	1000	IED_2.4	Scada
MMS_IED2.5_to_Scada	10	0.25	1000	IED_2.5	Scada
SV_IED1.3_to_IED2.2	3200	0.1	250	IED_1.3	IED_2.2
SV_IED1.5_to_IED1.1	3200	0.1	250	IED_1.5	IED_1.1
SV_IED2.3_to_IED2.2	3200	0.1	250	IED_2.3	IED_2.1
SV_IED2.5_to_IED2.2	3200	0.1	250	IED_2.5	IED_2.2
Timesync_Clock	3.2	0.1	100	Clock	<ALL NODES>

Table 1: Properties of the considered flows in the electrical substation use case.

The DelayLyzer computes the cumulative delay bound of a flow from its source to its destination(s). To simplify the evaluation of numerical results, the DelayLyzer visualizes them in a colored network topology as presented in Figure 5. It shows the delays for a network consisting of 27 nodes and 28 links, each of them having a bandwidth of 100 Mb/s. The multicast flow originates at the node clock and any other device in the network receives the data. The color of the links and nodes visualizes the cumulative delay bound relative to the maximum allowed delay: green

is far below the maximum allowed delay, red is above the maximum allowed delay, and orange is close to the maximum allowed delay. Moreover, the cumulative delay bound is annotated on all links. Obviously, the path from the clock towards IED_2.3 in the lower right corner introduces most delay.

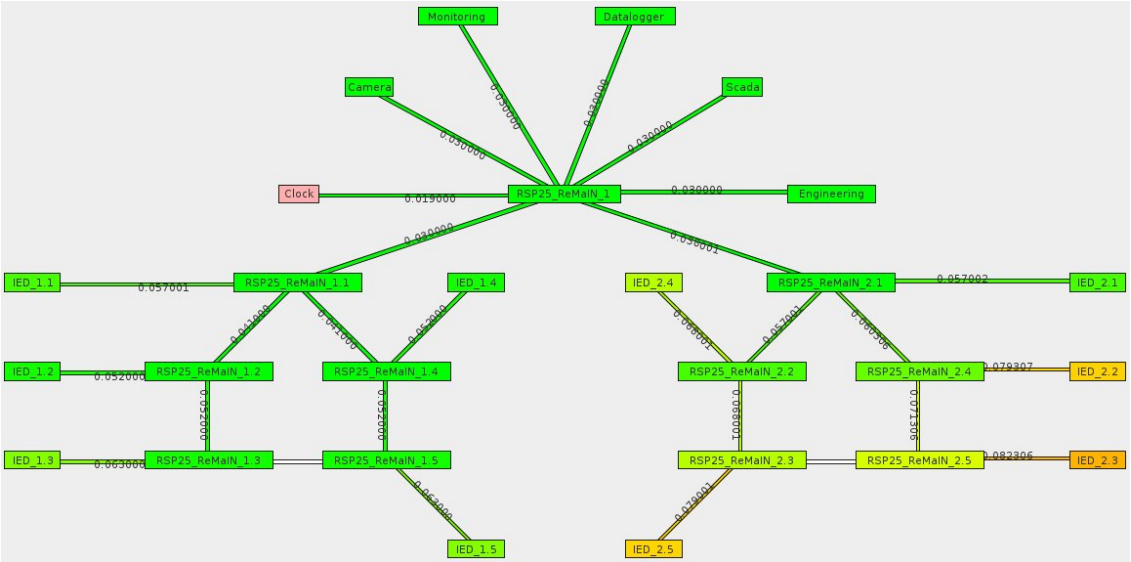


Figure 5: Network topology modelled in DelayLyzer showing a multicast flow from the clock towards all other devices for 100 Mb/s Ethernet technology.

We also calculate the delays of the clock traffic towards all IEDs for 1 Gb/s and compare it with 100 Mb/s technology. The resulting delays are compiled in Table 2. The maximum allowed delay for the clock traffic is 100 μs. The table shows that the maximum calculated delay bound is 82.3 μs for 100 Mb/s which well meets the delay requirement. Thus, the considered network could be rolled out using 100 Mb/s technology. Nevertheless, 1 Gb/s may be more future-proof as it can accommodate additional traffic.

An interesting finding is that 1 Gb/s technology reduces the delays only by a factor of less than 2 in spite of 10 times more capacity. This is because we assumed the same propagation delays and similar processing delays in the nodes.

Link capacity	**Delay towards IED (μs)**									
	1.1	**1.2**	**1.3**	**1.4**	**1.5**	**2.1**	**2.2**	**2.3**	**2.4**	**2.5**
100 Mb/s	57.0	52.0	63.0	52.0	63.0	57.0	79.3	82.3	68.0	79.0
1 Gb/s	35.4	44.8	55.8	44.8	55.8	35.4	47.2	57.4	46.4	57.4

Table 2: Delay bounds for multicast flow “Timesync_Clock” computed with MC-LTFA.

6 Conclusion

In this paper we presented Linkwise Total Flow Analysis (LTFA) as a modification of the existing Total Flow Analysis (TFA) algorithm to compute bounds for end-to-end delays in a packet-switched networks. LTFA leads to lower delay bounds for FIFO networks. We further proposed MC-LTFA as an extension of LTFA to calculate bounds for end-to-end delays for multicast flows in a simple way.

We implemented LTFA and MC-LTFA in our DelayLyzer tool. We showed that the use of MC-LTFA provides much lower delay bounds than other network calculus (NC) algorithms that

model multicast flows by multiple unicast flows. We further illustrated how different branches of a multicast flow are affected by cross traffic and that MC-LTFA produces lower delay bounds than SFA and TFA applied to unicast subflows.

Multicast is used in practical use cases, e.g., for substation automation based on IEC 61850. We modelled the communication scenario in a typical substation and showed that 100 Mb/s Ethernet technology meets the delay requirements of the selected use case. Thus, our algorithms are applicable to realistic networking scenarios.

References

[1] 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.

[2] R. L. Cruz, "A Calculus for Network Delay, Part I: Network Elements in Isolation," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 114–131, Jan. 1991.

[3] ——, "A Calculus for Network Delay, Part II: Network Analysis," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 132–141, Jan. 1991.

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

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

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

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

[8] 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, Oct. 2006.

[9] 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.

[10] 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*.

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

[12] 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.

[13] M. Menth, M. Schmidt, S. Veith, S. Kehrer, and A. Dreher, "Comparison of Delay Bounds Based on Simple Network Calculus Algorithms," in *IEEE ISCC'14*, 2014.

[14] IEC, "IEC 61850-5: Communication networks and systems in substations - Part 5: Communication requirements for functions and device models," 2013.

[15] ——, "IEC 61850-7-1: Communication networks and systems in substations - Part 7-1: Basic communication structure for substation and feeder equipment - Principles and models," 2003.

[16] ——, "IEC 61850-8: Specific communication service mapping (SCSM) - Part 8," 2003.

[17] ——, "IEC 61850-90-4: Communication networks and systems in substations - Part 90-4: Network engineering guidelines for substations," 2012.

[18] ——, "IEC 62439-2: Industrial communication networks - High availability automation networks - Part 2: Media Redundancy Protocol (MRP)," 2010.