Product Line-Based Network Calculus Analysis in Vehicular Communications

Nurbek Halikulov, Anna Arestova, Kai-Steffen Hielscher, Reinhard German
Computer Networks and Communication Systems
Friedrich-Alexander-Universität Erlangen-Nürnberg
{nurbek.halikulov, anna.arestova, kai-steffen.hielscher, reinhard.german}@fau.de

***Abstract*—Train communication systems are essential in the approval and safety processes of railway operations. In computer and packet-switched environment, the Network Calculus (NC) provides a robust mathematical framework which is used for analyzing the boundaries in particular systems or networks. The application of the Pay Multiplexing Only Once (PMOO) methodology allows for the achievement of accurate solutions within NC. In the context of (complex) system development, the principle of Product Line Engineering (PLE) has emerged by leveraging reusability concepts. This strategy can be achieved through the use of components that can be reused and customised. The paper explores the intersection of NC and PLE, utilizing the former to model and evaluate the performance of train network through prospective addition of new devices and flows to the current system variant. We introduce a pattern-based method to predict and manage network performance metrics associated with deterministic end-to-end flow latency analysis of extended variants. We utilize the Disco Deterministic Network Calculator (DiscoDNC) framework to derive such deterministic flow delays by evaluating the network topology. The evaluations demonstrate that the worst-case characteristics of data flows can be computed efficiently, avoiding redundant recalculations across various network configurations.**

***Index Terms*—Product Line Engineering, Network Calculus, Pay Multiplexing Only Once, Feed-forward Networks, DiscoDNC**

I. Introduction

The German Federal Ministry of Transport and Digital Infrastructure is forecasting a significant rise in passenger rail transport capacity, with an estimated increase of almost 20% by 2030 in Germany [1]. In this context, the Product Line Engineering (PLE) methodology facilitates the reuse of components across various products. As rail transport capacity expands, there is a need for infrastructure that is both scalable and modular to handle the increasing number of passengers.

Product line refers to a family of products derived from a common core, but varying in specific features [2]. Traditional systems engineering methods have limited themselves to a single variant or iteration of a product. Conversely, the design and development of the entire product in PLE has to be engineered as a whole family with shared components and design principles. The shared components can be customized due to the inclusion of variation points and offers more versatility in managing different variants of train communication system.

This work has been funded by the Federal Ministry of Economic Affairs and Climate Action of Germany as part of the MBPLE4Mobility project.

In order to understand the network behavior of such communication systems, a system theory specifically designed for computer networks, comes into play. Network Calculus (NC) [3], [4] has been widely applied across various domains, particularly in the Internet's Quality of Service (QoS) through frameworks such as Integrated Services (IntServ) and Differentiated Services (DiffServ) [5]. Using NC, we determine the longest end-to-end delay experienced by either a single data flow or multiple data streams within a given network topology. This process is optimized by the Pay Multiplexing Only Once (PMOO) network analysis tool [4], effective in feed-forward networks [6] where network traffic progresses linearly without revisiting nodes. PMOO's core principle is to account for the multiplexing (when multiple flows share the same network resources) of flows just once, resulting in more accurate delay bounds compared to methods that may overestimate due to considering multiplexing multiple times. PMOO ensures that each flow should pay its share of the link capacity only once, regardless of the number of network elements it traverses.

In this study, we present a pattern-based approach for determining worst-case delay bounds for network flows in both the original and extended versions of a network communication system's topology. This method is applied to a real-world scenario from a train communication network, incorporating end devices and data flows. By combining PLE with NC to assess worst-case performance bounds of network streams, our main benefit lies in the creation of a refinement algorithm. This algorithm enables performance to remain within acceptable limits without the need to recompute each variation from scratch. This technique has proven effective in various scenarios, marking our research as one of the first to integrate PLE with NC in this domain. The ability to guarantee network bounds is crucial and it provides the flexibility to adapt to any modified network variants. This approach reduces computational overhead, allowing for quicker analysis and more efficient system updates in dynamic environments.

The paper is structured as follows: Section II provides an overview about NC fundamental principles and its analytical framework. Section III describes the use of PMOO analysis in assessing network performance. Section IV evaluates the network expansion and its delay impacts, including practical real-world application. We will go through related work in Section V and conclude the paper in Section VI.

979-8-3315-2096-0/25/$31.00 ©2025 IEEE

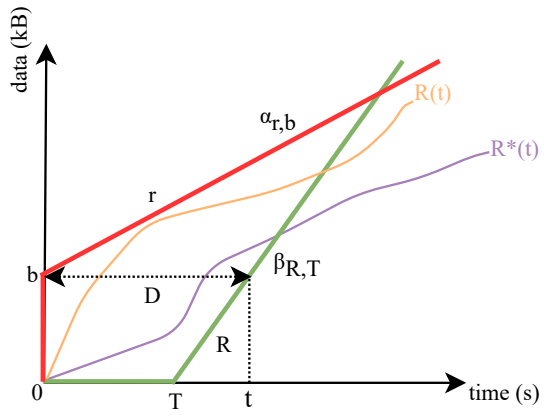


Fig. 1: NC curves and delay element

II. Background

A. Network Calculus

NC is a system theory that enables the assessment of the worst-case performance in communication networks and adopts a min-plus approach [3], [4], [7]. In this study, NC analysis will require the topology of the feed-forward network [6] which refers to the graphical representation of the flow paths in the network with physical connections between the network elements.

Fundamentals: Among the fundamental requirements of deterministic NC is the accurate modeling of network flows and network elements through mathematical curves, namely arrival curves and service curves respectively [8]. The input and output network flow is expressed as a cumulative function and refers to the set F of non-negative functions:

$$F = \{\forall t \geq s : f(t) \geq f(s), \forall t \leq 0 : f(0) = 0\} \quad (1)$$

Arrival curve represents a data flow, which is characterized by its specific data volume and transmission rate.

Definition 1. *(Arrival curve): Arrival curve α for a flow R determines the upper limit for the amount of data arriving:*

$$\forall s \leq t : R(t) - R(s) \leq \alpha(t - s) \quad (2)$$

Service curve defines the minimum service that a network guarantees to deliver.

Definition 2. *(Strict service curve): Service curve β determines the lower limit for the service during any backlogged period of duration $t - s$:*

$$\forall s \leq t : R^*(t) - R^*(s) \geq \beta(t - s) \quad (3)$$

Definition 3. *(Maximum output arrival curve): Output arrival curve describes the maximum number of data transmitted and processed at time t:*

$$\alpha^* = (\alpha \oslash \beta)(t) = \inf_{s \geq 0}\{\alpha(t + s) - \beta(s)\} \quad (4)$$

The performance is characterized by delay and backlog [9]. Deterministic delay bound is represented by a horizontal deviation between the arrival and service curves, Fig. 1.

Definition 4. *(Delay bound): $D(t)$, experienced by an input flow at network element offering a service curve:*

$$D(t) \leq h(\alpha, \beta). \quad (5)$$

To model the network element guarantees, it is important to define the following operations [7]:

Definition 5. *(Concatenation): Flow crosses two servers sequentially, described in Fig. 2a. Both servers offer its own strict service curves β_1 and β_2, and its concatenation offers a new service curve to arrival process [8]:*

$$\beta_{total} = (\beta_1 \otimes \beta_2)(t) = \sup_{s \geq 0}\{\beta_1(t + s) - \beta_2(s)\} \quad (6)$$

Definition 6. *(Leftover service curve): When two flows, f_1 and f_2 pass through the same server, described in Fig. 2b., the leftover service curve for each individual flow:*

$$\beta_{1,\text{leftover}} = [\beta - \alpha_2]^+, \ \beta_{2,\text{leftover}} = [\beta - \alpha_1]^+ \quad (7)$$

The leftover service curve specifically represents the service available to a particular flow after considering the impact of other cross flows on the shared network resources.

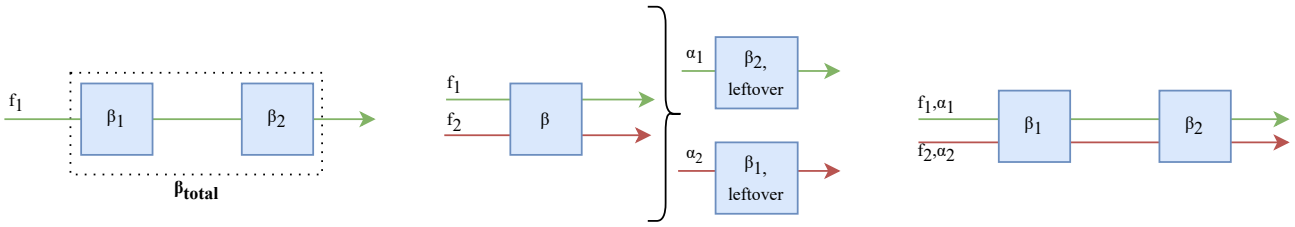
B. Network Delay Analysis

In a multi-hop network, where packets travel through multiple network elements, NC can be used to determine the worst-case delay experienced by flow of interest through a network, from the time it enters the network until it leaves. By using the arrival and service curves, we can calculate the maximum delay a data packet might experience as it passes through the network [10]. We will review the most common network analysis methods.

1) Total Flow Analysis (TFA): TFA [10] aggregates all the flows together, treating them as a single flow. The end-to-end delay bound of a flow is derived by summing up the individual server delay bounds on its path. Given that P represents the path of the individual flow of interest and s denotes a network element on that path, D_s refers to the delay through an individual server, as defined in equation (5). Total delay D of the flow of interest is expressed as the sum of delay bounds D_s across all network elements s in the path P:

$$D = \sum_{s \in P} D_s. \quad (8)$$

According to Zhou et al. [10], TFA method is not widely favored in practical applications. This is because TFA evaluates network performance by aggregating all flows together without separating the characteristics of each individual flow



(a) Concatenation of service curves (b) Leftover service curve (c) SFA and PMOO analysis difference

Fig. 2: Comparative analysis of service curve operations

and treating them as a single cumulative flow. Omission of concatenation shows an obvious weakness of the TFA [9].

2) Separate Flow Analysis (SFA): SFA analyzes each flow separately and calculates the residual service curve for the flow of interest at every network element along its path. The concatenation theorem is used to create an end-to-end service curve for the flow of interest's path and contributes to a tighter delay bound. The issue with this method lies in the calculation of delay bounds, which tend to be conservative when a flow undergoes multiple multiplexing [11].

3) Pay Multiplexing Only Once (PMOO): PMOO [12], [13] states that multiplexing is paid only once. Multiplexing can occur in a blind [14] or arbitrary manner [15] when the node multiplexes several flows. This method highly depends on arrival curves for cross flows, which is also called cross traffic arrival bounds. The distinction between SFA and PMOO methodologies lies in the sequence of operations involving the concatenation of service curves and the computation of leftover service curves. In SFA, the process begins by determining the leftover service curve for each individual network element. Following this, the concatenation of service curves is performed. The aggregate leftover service for flow f_1 is obtained by concatenating the individual leftover service curves for each server, as depicted in Fig. 2c:

$$\beta_{overall}^{SFA} = [\beta_1 - \alpha_2]^+ \otimes [\beta_2 - (\alpha_2 \oslash [\beta_1 - \alpha_1])]^+. \quad (9)$$

Conversely, PMOO approaches this by initially concatenating the service curves of the network elements and calculates the overall leftover service curve by subtracting the cross traffic arrival curve. For the flow f_1, the corresponding equation is given by [10]:

$$\beta_{overall}^{PMOO} = [\beta_1 \otimes \beta_2 - \alpha_2]^+. \quad (10)$$

The Disco Deterministic Network Calculator (DiscoDNC) [16], [17] incorporates all three aforementioned NC methods for computing network delays after the configuration of flows and network element in the network. Feed-forward architecture arranges network elements sequentially, allowing data to flow forward without loops, thereby avoiding cyclic dependencies.

III. Application Based NC

This section presents a structured application of NC for analyzing network delay bounds and divided into three interconnected parts: modeling arrival and service curves, applying the PMOO analysis method, and examining a practical network topology. The subsections are organized to build from theoretical modeling to practical application to demonstrate how the methodology is applied.

A. Arrival and Service Curve Models

NC is employed to model incoming traffic, network components, and the delay boundaries for the flow of interest. We adopt a strict priority scheduling mechanism, where flows assigned the highest priority are prioritized and can mainly interfere with flows having the same priority. High-priority flows, while having precedence in queues, can still be preempted by a lower priority flow during transmission. Each flow is represented as a token bucket with a certain refill rate (arrival curve) and a capacity (maximum burst size). Token-bucket arrival curve [8] represented with

$$\alpha(t) = b + rt \quad (11)$$

where b denotes the burst size and r is the flow rate. We consider periodic flows, which are characterised by sending data at discrete intervals rather than continuous flows. For each period P within each transmission period of the flow, there is a maximum number of bits transmitted during each period. Strict service curve is described by rate-latency function [8]:

$$\beta(t) = C \cdot [t - T]^+ \quad (12)$$

where link capacity C represents data processing capability of server, where latency T denotes a delay factor accounting for the transmission of a maximum transmission unit (MTU) sized lower priority packet just an instant before a higher priority packet arrived. MTU refers to the maximum size of the payload for a packet and the necessary network headers [18]. The equation $T = \frac{MTU}{C}$ represents the time it takes to transmit an MTU packet over a network link. The link capacity is uniform throughout the topology.

B. Practical Application of Ring Network Topology

We concentrate on open ring network topology with specific point in a circular arrangement of network switches which involves disabling a switch at one of its points to prevent network looping. Network traffic originates from the central control unit CCU and is distributed to decentralized peripherals DP, which transmit and receive data from sensors and actuators within the network. Frequently utilized in industrial

(a) Cross flows on shared network resources

(b) Path is extended by adding a new node

Fig. 3: Placement of a new network component.

and railway settings, DP play a crucial role in the network. The data flows initiated by the CCU travel through these DP before reaching their final destination nodes. This network configuration includes control loops which are responsible for managing different functions including speed control, temperature regulation, door operations, and other safety mechanisms. These loops function through the data collected by sensors connected to the DP, which is then relayed back to the CCU.

C. PMOO Analysis for End-to-End Delay

PMOO method is employed to calculate the total delay for a specific data flow within networks characterized by nested flows. Instances of nested flow scenarios are documented in [10], [12] and [19]. Bouillard et al. [4] define a nested network as a network such that the path of each flow is included in the path of the other. Fig. 3a illustrates a nested network where CCU is generating separate flows for each of the three network elements. PMOO leftover service curve in nested networks involves concatenating the service curves of each network element and subtracting the impact of the cross traffic. Flows share network paths and are multiplexed at certain points in the network when they pass through the common DP. The flow of interest originating from the CCU passes through DP_1 and DP_2 before arriving at the destination node. To calculate the total delay for the flow of interest, as shown in (8), we utilize the PMOO leftover service curve $\beta_{PMOO}^{f.o.i}$ in conjunction with the flow of interest's arrival curve $\alpha^{f.o.i}$

$$D_{foi}^{PMOO} = D(\alpha^{foi}, \beta_{foi}^{PMOO}) \tag{13}$$

where the arrival curve and the PMOO leftover service curve for the flow of interest are defined as

$$\alpha^{foi} = b_{foi} + r_{foi}t,\ \beta_{foi}^{PMOO} = \beta_{R_{foi}^{PMOO}, T_{foi}^{PMOO}}. \tag{14}$$

PMOO service curve is determined by identifying the lowest leftover service rate R_{foi}^{PMOO} and the greatest latency T_{foi}^{PMOO} the flow of interest might experience in the network with shared resources and cross traffic. PMOO operation, according to [7], imposes specific conditions on the types of curves used: arrival curves should conform to the token-bucket model (11), and service curves to the rate-latency type (12):

$$\beta_{foi}^{PMOO} = R_{foi}^{PMOO}(t - T_{foi}^{PMOO}). \tag{15}$$

We should find the earliest time instant t at which the available service [9] is enough to accommodate the burstiness of the flow of interest

$$R_{foi}^{PMOO}(t - T_{foi}^{PMOO}) = b_{foi} \tag{16}$$

where t represents the total delay for the flow of interest, occurring when the PMOO leftover service curve matches the burst size of the flow of interest, as illustrated in Fig. 1:

$$D = \frac{b_{foi}}{R_{foi}^{PMOO}} + T_{foi}^{PMOO} \tag{17}$$

Flow of interest's PMOO leftover service curve [4], [20]: It is important to note that when calculating the total delay, the flow rate of $r_{foi}t$ as outlined in (14) is not taken into account. This is because our focus is on the leftover service curve, which inherently considers the effects of cross traffic utilising the same network resource. Leftover link capacity available for the flow of interest is defined by considering the total available link capacity C_{DP_i} at each node and the total flow rate of all cross traffic $\sum r_f$. It is calculated by finding the lowest value (infimum) resulting from the differences calculated between these respective capacities and flow rates:

$$R_{foi} = \bigwedge_{i=1}^{N_{\text{DP}}} \left(C_{DP_i} - \sum_{f=1}^{N_{\text{cross}}} r_f \right) \tag{18}$$

Subtracting the flow rates from the link capacities determines the available capacity of the network for the flow of interest. This subtraction assumes the total rate of cross traffic does not surpass the individual link capacities of DP. The value obtained from this calculation represents the maximum data transmission rate the flow of interest can achieve through the most constrained part of the network path. Leftover link capacity indicates the minimum guaranteed service level for the flow of interest at any point along its network path.

Flow of interest's PMOO latency [4], [20]: To calculate the PMOO-induced latency T_{foi}, we first account for the direct latencies the flow of interest incurs across each network component, with an initial assumption of no cross traffic:

$$T_{foi} = T_{DP_1} + T_{DP_2} + T_{DP_{end}} \tag{19}$$

Next, we factor in the extra latency due to cross traffic sharing these elements, incorporating their burstiness b_f and rate r_f parameters:

$$\ldots + \frac{b_{f1} + r_{f1} \cdot T_{DP_1}}{R_{foi}} + \frac{b_{f2} + r_{f2} \cdot (T_{DP_1} + T_{DP_2})}{R_{foi}} \tag{20}$$

and when there is no cross traffic across the node, we consider it as zero. Using the results, we define the end-to-end delay for the flow of interest using (17):

$$D = \frac{b_{foi}}{R_{foi}} + T_{foi} \tag{21}$$

IV. Variant based NC

A. Delay change in extended network − no flow added

Having calculated the total delay for the flow of interest, we enhance the network by incorporating new network elements. We first check the addition of a new network element without an increase in data transmission from the CCU to the destination node. In other words, there is no generation of a new flow as a direct result of this expansion. As illustrated in Fig. 3b, the introduction of this new DP_3 potentially affects the total delay experienced by the flow of interest. The new leftover link capacity, R_{foi}, will remain unchanged, given that no additional cross flows are introduced. However, introduction of a new network element increases both the actual and the additional latencies. This is captured through equations that sum up the latencies across all traversed nodes

$$T_{foi}^{''} = T_{DP_1} + T_{DP_2} + T_{DP_3} + T_{DP_{end}} \tag{22}$$

and calculate the new additional latencies considering the impact of the new element

$$\ldots + \frac{b_{f1} + r_{f1} \cdot T_{DP_1}}{R_{foi}} + \frac{b_{f2} + r_{f2} \cdot (T_{DP_1} + T_{DP_2} + T_{DP_3})}{R_{foi}} \tag{23}$$

Ultimately, the increased delay for the flow of interest will be:

$$D^{''} = \frac{b_{foi}}{R_{foi}} + T_{foi}^{''} \tag{24}$$

Based on given calculations from (19), (20), (22), (23) we can derive a generic formula for the flow of interest's latency

$$T_{foi} = \sum_{i \in \mathcal{N}_{DP}} T_{DP_i} + \sum_{f \in \mathcal{F}_{cross}} \left(\frac{b_f + r_f \cdot \sum_{i \in \mathcal{N}_f} T_{DP_i}}{R_{foi}} \right) \tag{25}$$

where

- $\sum_{i \in \mathcal{N}_{DP}}$: sum of set of DPs the flow of interest intersects
- $\sum_{f \in \mathcal{F}_{cross}}$: sum of set of cross flows, denoted by f
- $\sum_{i \in \mathcal{N}_f}$: sum of set of DPs latency that cross flow f intersects

The difference between the original (21) and updated (24) end-to-end delays for the flow of interest will determine the factor of delay increase caused by the integration of a new element into the existing network topology. ΔD signifies the change in delay, making it clear how the delay has been affected by the network's modification:

$$\Delta D = T_{DP_3} + \frac{r_{f2} \cdot T_{DP_3}}{R_{foi}} \tag{26}$$

We developed a generalized formula to calculate the delay change when the network is extended by adding new nodes to the flow of interest's path

$$\Delta D = \sum_{k \in \mathcal{N}_{DP}} T_{DP_k} + \sum_{f \in \mathcal{F}_{cross}} \left(\frac{r_f \cdot \sum_{k \in \mathcal{N}_f} T_{DP_k}}{R_{foi}} \right) \tag{27}$$

- $\sum_{k \in \mathcal{N}_{DP}}$: sum of set of newly added DPs in the extended network

Algorithm 1 Flow Delay Bound Calculation

$\mathcal{F}$, Φ = flowParser($\mathcal{L}_1$, $\mathcal{L}_2$);
while *flow* f *in* Φ **do**
  if $f_i \in \mathcal{F}$ **then**
    $oldDelay \leftarrow \mathcal{L}_1$
  else if $additionalHops > 0$ **then**
    $extendedDelay \leftarrow oldDelay + \Delta D$
  else if $f \notin \mathcal{F}$ *and* $f \in \Phi$ **then**
    $pmoo.performAnalysis(f)$
    $newDelay$=$pmoo.getDelayBound()$
end
return $newDelay$, $oldDelay$, $extendedDelay$

- $\sum_{f \in \mathcal{F}_{cross}}$: sum of set of cross flows, denoted by f
- $\sum_{k \in \mathcal{N}_f}$: sum of set of the latencies of the newly added DPs that cross flow f intersects

B. Delay change in extended network − flow is added

In most cases, when new nodes are added to the network, a new flow is established from the source to the added node. Suppose a new flow, $f3$, is introduced from the CCU to the newly added DP_3 in Fig. 3b. The end-to-end delay for the flow of interest, using (21), becomes:

$$D^{newflow} = \frac{b_{foi}}{R_{foi}^{newflow}} + T_{foi}^{newflow} \tag{28}$$

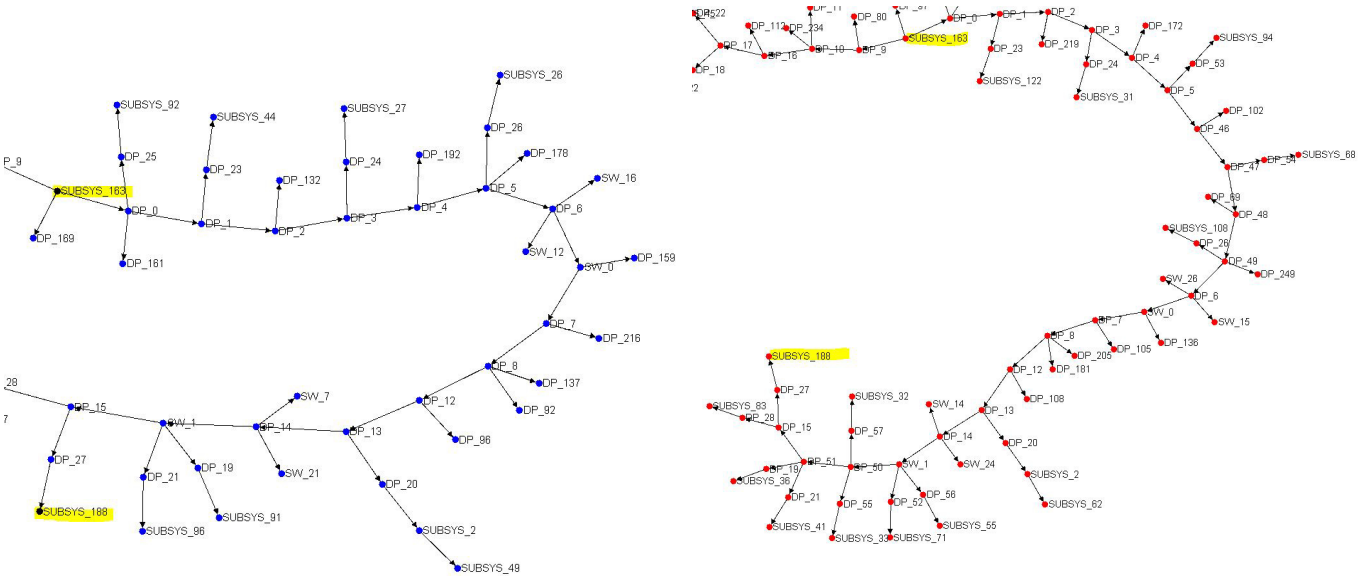
where $R_{foi}^{newflow}$ changes due to the additional cross traffic r_{f3}, leading to a new $T_{foi}^{newflow}$ from the contribution of shared nodes. The updated form of (27) for calculating the delay change between the non-extended and extended networks, including a new node and flow, is:

$$\begin{aligned} \Delta D^* = & \sum_{k \in \mathcal{N}_{DP}} T_{DP_k} + b_{foi} \left(\frac{1}{R_{foi}^{newflow}} - \frac{1}{R_{foi}} \right) \\ & + \sum_{f \in \mathcal{H}_{cross}} \left(\frac{b_f + r_f \cdot \sum_{k \in \mathcal{N}_f} T_{DP_k}}{R_{foi}^{newflow}} \right) \\ & - \sum_{f \in \mathcal{H}_{cross}} \left(\frac{b_f + r_f \cdot \sum_{k \in \mathcal{N}_f} T_{DP_k}}{R_{foi}} \right) \end{aligned} \tag{29}$$

- $\sum_{k \in \mathcal{N}_{DP}}$: sum of set of newly added DPs in the extended network
- $\sum_{f \in \mathcal{H}_{cross}}$: sum of set of cross flows, denoted by f
- $\sum_{k \in \mathcal{N}_f}$: sum of set of the latencies of the newly added DPs that cross flow f intersects

C. Delay Bound Analysis with Train Network Example

We have a real-world example of a train network topology, and we will assess how our findings can help us bypass the need to recalculate delay bounds for extended flows. The real-world flows used have been collected from research data as part of the MBPLE4Mobility project. Algorithm 1 describes the delay bound calculation process to determine how extending the network topology affects flow delays. The



(a) Original train network topology section

(b) Extended train network topology section

Fig. 4: Real train network topology example with flow extension and generation of new flows

algorithm takes in two versions of the network topology, where $\mathcal{L}_1$ contains 120 flows and $\mathcal{L}_2$ contains 160 flows. The extended network topology is modified for two main reasons: the introduction of new flows and the expansion of existing flow paths. This expansion occurs because new nodes (hops) have been added, and these nodes not only support the newly generated flows but also lengthen the paths of some pre-existing flows. As a result, both the network's complexity and its traffic dynamics have increased, requiring adjustments to the flow management and delay calculations across the system. The analysis begins by parsing the flows from the original and extended topologies. The flows from both topologies are parsed resulting in two sets of flows – $\mathcal{F}$ from $\mathcal{L}_1$ and Φ from $\mathcal{L}_2$. The flows are classified into three categories:

- *No hops:* flows have a direct connection from the source to the destination without any intermediate nodes
- *One hop:* flows have a single intermediate hop between the source and destination
- *Multiple hops:* flows include two or more hops, representing multiple intermediate nodes between the source and destination

For flows that do not change their path in the extended version, the old delay bound from the original topology is retained. If additional hops are introduced to the flow path without any flow incurred from the source, the extended delay is calculated by adding the old delay and the value from (27). However, if a new flow is introduced along with the added hops, the extended delay is calculated using (29). Newly introduced flows in the extended topology, which are not part of the path extensions of previously analyzed flows, undergo standard PMOO analysis to determine their delay bounds. As an example, we take a brief look at a small segment of the overall train network topology. As shown in Fig. 4, the flow from source *SUBSYS_163* to destination *SUBSYS_188* extended with six new nodes that result in additional hops:

Original Flow Information, Fig. 4a:
`Burstiness: 122 bits, Rate: 15.25 bps, Linkrate: 1.0E9 bps, Packettime: 0.12336 milliseconds`
`Source: SUBSYS_163, Destination: SUBSYS_188`
`Path (hops): DP_0, DP_1, DP_2, DP_3, DP_4, DP_5, DP_6, SW_0, DP_7, DP_8, DP_12, DP_13, DP_14, SW_1, DP_15, DP_27`

Extended Flow Information, Fig. 4b:
`Burstiness: 122 bits, Rate: 15.25 bps, Linkrate: 1.0E9 bps, Packettime: 0.12336 milliseconds`
`Source: SUBSYS_163, Destination: SUBSYS_188`
`Path (hops): DP_0, DP_1, DP_2, DP_3, DP_4, DP_5, DP_46, DP_47, DP_48, DP_49, DP_6, SW_0, DP_7, DP_8, DP_12, DP_13, DP_14, SW_1, DP_50, DP_51, DP_15, DP_27`

Delay Results:
`Flow f55 {Source: SUBSYS_163, Destination: SUBSYS_188}`
Flow was extended by 6 hops
Original worst-case delay bound: 2.22048984 ms
Extended worst-case delay bound: 2.960649842 ms

For clarity, Table I collects most of the mathematical notations used in this paper.

TABLE I: List of Notations

Symbol	Description
C	Link capacity
T	Latency of the node
$R(t)$	Input flow
$R^*(t)$	Output flow
α^*	Maximum output arrival curve
D	Total end-to-end delay
D_s	Delay bound through an individual node
foi	Flow of interest
β_{foi}	Leftover service curve for the flow
R_{foi}	Leftover link capacity for the flow
C_{DP}	Total link capacity of each node
b_f	Cross traffic burst size
r_f	Cross traffic flow rate
T_{DP}	Individual node latency
T_{foi}	Total latency for the flow
$T_{foi}^{''}$	Total latency after adding new nodes
$D^{''}$	Increased delay after after adding node
$D^{newflow}$	Increased delay after adding node and flow
ΔD	Change in delay bound after adding node
ΔD^*	Change in delay after adding node and flow

V. Related Work

This research paper studies the worst-case performance limits for data flows. NC has been a cornerstone for analyzing network performance, particularly in deterministic settings. Foundational principles of NC were laid in [3], [4], [7], [8] offering a comprehensive framework for assessing network service guarantees and performance bounds. This framework evaluated the worst-case scenarios in variant-based network architectures, including real-time systems where ensuring timely data delivery is crucial. PMOO technique introduced in [4], [12], [13] have refined delay bound calculations. The DiscoDNC toolbox introduced by Bondorf et al. [16] is utilized to derive these deterministic flow delays by evaluating network topology. We examined the real train network topology and calculated delay bounds for extended train topology.

VI. Conclusion

We have introduced a comprehensive method for predicting and managing the impact of network changes on the end-to-end delay of data flows in train communication systems. By using a mathematical model to represent both data flows and network components, we can efficiently compute flow worst-case delays. This approach proves especially effective in scenarios where new network devices are integrated, as it allows us to determine if and how specific flows interact with these added devices, either individually or collectively. This targeted analysis eliminates the need for redundant calculations, while clearly demonstrating how specific modifications affect overall delay reduction. While some existing flows experienced extended delays due to the expanded topology, the introduction of new flows came with distinct delay characteristics. This method offers several key benefits, including optimized computation and streamlined network recalibration. By allowing precise delay predictions, it enhances network performance management and ensures smoother integration of additional devices. Looking forward, we plan to refine our recalibration techniques to handle increasingly complex network topologies, making this method even more robust for dynamic and large-scale systems.

References

[1] G. Tasler and V. Knollmann, "The introduction of highly automatic operation - towards fully automatic train operation," pp. 6–14, June 2018.

[2] C. Krueger and P. Clements, "Systems and software product line engineering," *Encyclopedia of Software Engineering*, vol. 2, pp. 1–14, 2013.

[3] J.-Y. L. Boudec and P. Thiran, *NETWORK CALCULUS: A Theory of Deterministic Queuing Systems for the Internet*. Berlin, Heidelberg: Springer, Apr. 2012.

[4] A. Bouillard, M. Boyer, and E. Le Corronc, *Deterministic Network Calculus*. John Wiley & Sons, Oct. 2018.

[5] M. Barreiros and P. Lundqvist, *QOS-Enabled Networks: Tools and Foundations*. John Wiley & Sons, Ltd, 2011.

[6] A. Bouillard, L. Jouhet, and E. Thierry, "Tight performance bounds in the worst-case analysis of feed-forward networks," in *2010 Proceedings IEEE INFOCOM*, 2010, pp. 1–9.

[7] S. Bondorf, "Worst-Case Performance Analysis of Feed-Forward Networks - An Efficient and Accurate Network Calculus," Ph.D. dissertation, Kaiserslautern University of Technology, Germany, 2016.

[8] A. Van Bemten and W. Kellerer, "Network Calculus: A Comprehensive Guide," Chair of Communication Networks, Technische Universität München, Technical, Oct. 2016. [Online]. Available: https://mediatum.ub.tum.de/doc/1328613/1328613.pdf

[9] 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 - The 27th Conference on Computer Communications*, 2008, pp. 1669–1677.

[10] B. Zhou, I. Howenstine, S. Limprapaipong, and L. Cheng, "A survey on network calculus tools for network infrastructure in real-time systems," *IEEE Access*, vol. 8, pp. 223 588–223 605, 2020.

[11] T. Wasner, M. Helm, and D. Scholz, "What is deterministic network calculus?" *Network*, vol. 61, 2019.

[12] J. B. Schmitt, F. A. Zdarsky, and I. Martinovic, "Improving performance bounds in feed-forward networks by paying multiplexing only once," in *14th GI/ITG Conference-Measurement, Modelling and Evalutation of Computer and Communication Systems*. VDE, 2008, pp. 1–15.

[13] S. Bondorf and F. Geyer, "Generalizing network calculus analysis to derive performance guarantees for multicast flows." in *VALUETOOLS*, 2016.

[14] J. B. Schmitt, F. A. Zdarsky, and I. Martinovic, "Performance bounds in feed-forward networks under blind multiplexing," Citeseer, Tech. Rep., 2006.

[15] J. B. Schmitt, F. A. Zdarsky, and M. Fidler, "Delay bounds under arbitrary multiplexing," *Rapport technique, University of Kaiserslautern*, vol. 121, pp. 124–140, 2007.

[16] S. Bondorf and J. B. Schmitt, "The DiscoDNC v2 – A Comprehensive Tool for Deterministic Network Calculus," in *Proc. of the International Conference on Performance Evaluation Methodologies and Tools*, ser. ValueTools '14, December 2014, pp. 44–49.

[17] NetCal/DNC: The NetworkCalculus.org Deterministic Network Calculator. [Online]. Available: https://github.com/NetCal/DNC

[18] D. Thiele and R. Ernst, "Formal worst-case performance analysis of time-sensitive ethernet with frame preemption," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2016, pp. 1–9.

[19] J. B. Schmitt and F. A. Zdarsky, "The disco network calculator: a toolbox for worst case analysis," in *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, 2006, pp. 8–es.

[20] A. Bouillard, B. Gaujal, S. Lagrange, and É. Thierry, "Optimal routing for end-to-end guarantees using network calculus," *Performance Evaluation*, vol. 65, no. 11-12, 2008.