DetServ: Network Models for Real-Time QoS Provisioning in SDN-based Industrial Environments

Jochen W. Guck, Amaury Van Bemten, and Wolfgang Kellerer

***Abstract*—Industrial networks require real-time guarantees for the flows they carry. That is, flows have hard end-to-end delay requirements that have to be deterministically guaranteed. While proprietary extensions of Ethernet have provided solutions, these often require expensive forwarding devices. The rise of Software-Defined Networking (SDN) opens the door to the design of centralized traffic engineering frameworks for providing such real-time guarantees. As part of such a framework, a *network model* is needed for the computation of worst-case delays and for access control. In this article, we propose two network models based on network calculus theory for providing deterministic services (DetServ). While our first model, the *multi-hop model* (MHM), assigns a rate and a buffer budget to each queue in the network, our second model, the *threshold-based model* (TBM), simply fixes a maximum delay for each queue. Via a packet-level simulation, we confirm that the delay bounds guaranteed by both models are never exceeded and that no packet loss occurs. We further show that the TBM provides more flexibility with respect to the characteristics of the flows to be embedded and that it has the potential of accepting more flows in a given network. Finally, we show that the runtime cost for this increase in flexibility stays reasonable for online request processing in industrial scenarios.**

***Index Terms*—access control, real-time, industrial network, network modeling, network calculus, Quality of Service (QoS), Software-Defined Networking (SDN)**

I. Introduction

A. Motivation: Industrial Networking Quality of Service

Industrial communications (e.g., machine-to-machine (M2M) communications or production facilities networks) have strict Quality of Service (QoS) requirements, mainly in terms of end-to-end delay [1]. This means that flows have end-to-end delay bounds that must not be exceeded. In this article, such flows are referred to as *real-time flows*. A wide range of proprietary solutions [2] and extensions of Ethernet [3] have been developed for providing this strict QoS. However, these solutions typically require changes within the network protocol stack or impose restrictions on the topology that can be deployed, which leads to expensive forwarding devices.

B. Basis: Centralized Frameworks based on Software-Defined Networking

Software-Defined Networking (SDN) is a new networking paradigm that runs control functions on a centralized controller which is then able to program the Ethernet forwarding elements in the network using a standardized interface such as OpenFlow [4]. This central view offered by SDN allows to perform traffic engineering based on the global knowledge of the network. Because it only requires simple commodity SDN forwarding elements that can be changed and updated independently [5], SDN is considered as an inexpensive solution. Therefore, as elaborated in Sec. II, a plethora of work has been considering the usage of SDN for the provisioning of QoS [6]–[18]. However, the QoS control provided by these approaches is either too *inaccurate* or *slow* for industrial applications [18].

As initiated by Jasperneite et al. [19], Guck et al. [16]–[18] propose to overcome the two above-mentioned shortcomings by using *network calculus*, a mathematical modeling framework (introduced in Sec. III), to maintain a deterministic model of the network state in the control plane. First, network calculus being a deterministic framework, *accurate* bounds can be computed on a per-flow basis. Second, keeping a deterministic model in the centralized control plane allows to avoid the QoS control loop to go through the forwarding plane, thereby allowing to *quickly* provision new flow requests [17]. As such, the two drawbacks of existing approaches are overcome.

C. Contribution: DetServ: Network Models for Deterministic Worst-Case Delay Computation and Access Control

As elaborated in Sec. IV, a centralized industrial QoS framework requires a network model for the computation of worst-case delays and for access control. The core contribution of this article consists of two network models that can be used as part of such QoS frameworks for providing deterministic services (*DetServ*). The first model, the *multi-hop model* (MHM – Sec. V-D), assigns a rate and a buffer budget to each queue in the network. This allows to compute worst-case delays for any path in the network. This model corresponds to an updated version of a previously proposed model [16], [18] which was not considering buffer consumption and, hence, was potentially leading to packet loss. We show that the MHM requires an *a priori* choice regarding the characteristics of flows that are to be embedded based on the trade-off between rate, buffer capacity and delay. R1-4The second model, the *threshold-based model* (TBM – Sec. V-E), is the main contribution of this article. It simplifies this trade-off by only fixing a maximum delay for each queue in the network, thereby avoiding the *a priori* assignment of rate and buffer budgets. We show that the TBM automatically adapts the allocation of rate and buffer capacity based on the type of traffic (bandwidth or buffer demanding) and we find that this gives it the potential to outperform the MHM, i.e., to accept more flows and hence increase network utilization. However, this increase in flexibility leads to an increase in the request processing time by a factor corresponding to the number of priority levels in the network. Further, we propose an extension to both models that considers the shaping introduced by the limited capacity of the links in the network (Sec. V-G). While beneficial for both models, we show that it has a higher impact on the TBM,

J. W. Guck, A. Van Bemten, and W. Kellerer are with the Lehrstuhl für Kommunikationsnetze, Technical University of Munich, Munich, 80290, Germany (email: {guck, amaury.van-bemten, wolfgang.kellerer}@tum.de).

both in terms of increased runtime and performance. We find that this runtime increase is reasonable for industrial scenarios. Indeed, in our simulations, the total request processing time of the TBM remains lower than 350 ms in 99% of the cases and never exceeds 620 ms.

The power of the proposed models resides in the fact that they can be used with off-the-shelf switches supporting priority scheduling and any SDN protocol providing standard enqueuing and forwarding primitives, e.g., OpenFlow 1.0 [20].

II. Related Work

A. Legacy Industrial Networking Solutions

Initially, proprietary solutions (e.g., Profibus, Interbus or CAN) have been specifically developed for real-time industrial communications [2], [21]. These solutions often come with a complete proprietary communication stack which requires specialized and expensive hardware.

Later, Ethernet data transfer rates increased and Ethernet became ubiquitous in local area networks (LANs) and the Internet. Therefore, it attracted a lot of attention for industrial deployments. However, because of its non-deterministic medium access control (MAC) scheme, Ethernet was initially not considered as a suitable solution. The usage of full duplex point-to-point links along with Ethernet switches instead of shared buses and hubs allowed to avoid collisions and hence the negative impact of the Ethernet MAC protocol [3]. Nevertheless, this introduces buffering and possibly overflows, which were still considered to be a source of non-determinism [3]. Despite this, using Ethernet in industrial environments has major benefits, including simple and cheap deployment, easy connectivity towards office networks, the Internet or more generally any IP traffic, and usage of off-the-shelf communication hardware. Hence, many industrial control systems manufacturers decided to develop proprietary extensions of Ethernet to achieve determinism [21], [22]. A broad overview of Ethernet-based real-time technologies, including deterministic Ethernet standards, was provided by Decotignie [3]. Unfortunately, these solutions require changes within the network protocol stack or impose topology restrictions or both, which leads to more expensive forwarding devices than with standard Ethernet.

B. SDN-based QoS Networking Frameworks

The emergence of SDN as a new networking paradigm providing a global view of the network in a centralized control entity provided a new opportunity for traffic engineering. Hence, a wide range of work has been considering the usage of SDN for QoS networking. In this section, we present an overview of the state-of-the-art in QoS provisioning using SDN and highlight the contributions of this article with respect to the existing literature. We classify the existing approaches in six categories for which we list a few representative examples.

1) High-Level Architectural Proposals: Several proposals mainly focus on architectural issues such as interface design and requirements analysis [23]–[26]. These approaches mention that a method for access control and resource reservation is needed but do not tackle the problem. The models we propose in this article can be used as part of such frameworks.

2) OpenFlow Extensions: Other approaches consider the enhancement of the OpenFlow protocol with QoS-related features [10], [25], [27]. Because of the lack of standardization, this potentially leads to higher cost and/or effort. In contrast, we propose new models which can be used with any SDN protocol providing standard enqueuing and forwarding programming primitives, e.g., OpenFlow 1.0 [20].

3) TDMA Solutions: Systems using time division multiple access (TDMA) on top of Ethernet have also been proposed [28], [29]. These solutions can potentially lead to an optimal utilization of resources. However, because of the need for synchronization, changes in the protocol stack of endpoints might be needed, thereby leading to expensive solutions in terms of cost and effort. In comparison, our models do not require any change at the endpoints.

4) QoS Frameworks based on Data Rate Allocation: Another class of proposals, mainly tailored for Internet QoS, maps QoS requirements to equivalent minimum data rates [6]–[9]. Such systems typically do not consider the limited capacity of buffers and hence packet loss and queuing delay. These approaches provide the scalability and QoS level needed for wide-area networks but are not sufficient for industrial scenarios, which require strict buffer management as provided by our proposed models.

5) Measurement-based Frameworks: A wide range of proposals build the network state by retrieving it from the data plane [9]–[15]. This step adds a non-negligible delay to the flow request processing. Besides, these approaches suffer from possible measurement errors. Thus, they can only provide soft guarantees. While this is an efficient solution for multimedia traffic, it does not fulfill the requirements of industrial communications. On the other hand, the determinism of our models allow to provide hard, i.e., real-time, guarantees.

6) Model-based Frameworks: The present article falls into the category of model-based frameworks where a model of the resources usage is kept in the control plane [6], [8], [17], [30]. The state of the network can then be retrieved from the model itself, avoiding the request processing loop to go through the data plane, thereby reducing the request processing time. The model only has to communicate with the data plane at topology change events. While stochastic modeling could be used for soft QoS requirements, a deterministic model is needed for providing real-time guarantees. Duan [6] and Tomovic et al. [8] proposed models based on data rate allocation which, as elaborated in Sec. II-B4, are not suitable for industrial applications. For their part, Guck et al. [17] mentioned the need of a model but did not present one and King et al. [30] detailed a deterministic model but which requires a flow embedding procedure that can lead to high request processing time. The new DetServ models we propose in this article are deterministic models that can be used as part of a model-based QoS framework for fast request processing in industrial scenarios. One of the models was already partially described by Guck et al. [16], [18] but the limited capacity of buffers was not considered. In this article, we present an updated and more detailed version of this original model and further introduce a new second model providing more flexibility with respect to the characteristics of the flows to be embedded.

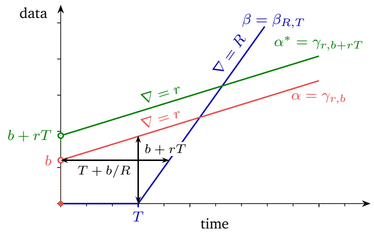


Fig. 1: Example of graphical computation of delay, backlog and output bounds using network calculus concepts. The delay and backlog bounds respectively correspond to the horizontal and vertical deviations between the arrival and service curves. In the particular case of an arrival curve $\gamma_{r,b}$ and a service curve $\beta_{R,T}$, the output bound α^* is obtained by shifting the initial arrival curve α up by rT.

It is worth mentioning that model-based approaches, and hence our proposed models, can be used as part of the path computation unit of Time-Sensitive Networking (TSN) approaches, the emerging real-time networking standards.

III. Modeling Background: Network Calculus

A. Basics: Theory Principles

In order to provide a deterministic model of the network, we propose to use *network calculus*. Network calculus [31] is a system theory for communication networks. From models of a considered flow and of the service a so-called *system* can offer, bounds on *(i)* the delay the flow will experience traversing the system, *(ii)* the backlog the flow will generate in the system, and *(iii)* the new model for the flow after it has passed the system can be computed. A system can range from a simple queue to a complete network. The theory is divided in two parts: *deterministic* network calculus, providing deterministic bounds, and *stochastic* network calculus, providing bounds following probabilistic distributions. Since we strive for deterministic modeling, we will only consider the former.

The modeling of a flow is done using a so-called *arrival curve* $\alpha(t)$. $\alpha(\tau)$ gives an upper bound on the amount of data a flow will send during any time interval of length τ. The α curve in Fig. 1 represents a *token bucket* flow: it is allowed to send bursts of up to b bytes but its sustainable rate is limited to r B/s. This type of arrival curve is denoted by $\gamma_{r,b}$.

The modeling of a network system is, for its part, done using a so-called *service curve* $\beta(t)$. Its general interpretation is less trivial than for an arrival curve [32]. The particular service curve β shown in Fig. 1 can be interpreted as follows. Data might have to wait *up to* T seconds before being served at a rate of *at least* R B/s. This type of service curve is denoted by $\beta_{R,T}$ and is referred to as a *rate-latency* service curve.

From these two curves, the three above mentioned bounds can be computed (Fig. 1). The delay and backlog bounds respectively correspond to the horizontal and vertical deviations between the arrival and service curves [32]. In the general case, the way to compute α^*, the arrival curve of the flow after having traversed the system, is not straightforward [32]. In the particular case where the arrival and service curves are $\gamma_{r,b}$ and $\beta_{R,T}$, we have $\alpha^* = \gamma_{r,b+rT}$ [32] (Fig. 1). This formula can be interpreted as follows. Since the flow can possibly wait up to T seconds before being served at a potentially infinite rate, its burst size can increase by up to rT bytes – the maximum amount of data that, by definition of the arrival curve of the flow, will arrive during these T seconds of potential waiting time.

B. Selected Results: Priority Scheduling

In the particular case of a non-preemptive strict priority scheduler with n queues traversed by token bucket flows [33], the service curve for priority queue i is given by [32]

$$\beta_i(t) = \left(Ct - t\sum_{j=1}^{i-1} r_j - \sum_{j=1}^{i-1} b_j - \max_{i+1 \le j \le n} \{l_j^{max}\} - l_i^{max} \right)^+, \tag{1}$$

where queue $i = 1$ is the highest priority queue, C is the capacity of the output link, and r_j, b_j and l_j^{max} are the rate, burst size and maximum packet size of the token bucket flow traversing queue j. This formula can be interpreted as follows. The service offered to a given queue i corresponds to the whole link capacity (first term) from which the capacity used by higher priority flows is deducted (second and third terms). Since we assume a non-preemptive priority scheduler, data in a high priority queue might have to wait for a packet of a lower priority queue to be transmitted before being served (fourth term). The fifth term models the store-and-forward behavior of switches. Indeed, the scheduler must wait for each packet to be completely received before serving it. Note that for cut-through switches, only the header length should be used here. Because the scheduler cannot provide negative service, the negative part of the resulting curve is reduced to zero ($(.)^+$ notation).

Eqn. 1 corresponds to a β_{R_i,T_i} curve where

$$T_i = \frac{\sum_{j=1}^{i-1} b_j + \max_{i+1 \le j \le n} \{l_j^{max}\} + l_i^{max}}{C - \sum_{j=1}^{i-1} r_j} \tag{2}$$

and

$$R_i = C - \sum_{j=1}^{i-1} r_j. \tag{3}$$

From Fig. 1, the delay and backlog experienced by the flow traversing queue i are respectively bounded by

$$d_i = \frac{\sum_{j=1}^{i} b_j + \max_{i+1 \le j \le n} \{l_j^{max}\} + l_i^{max}}{C - \sum_{j=1}^{i-1} r_j} \tag{4}$$

and

$$x_i = b_i + r_i T_i, \tag{5}$$

and the new burst of the flow after the system is given by

$$b_i^* = x_i, \tag{6}$$

while its rate remains unchanged.

IV. Context: Model-based QoS Framework

We present the model-based framework proposed by Guck et al. [16]–[18] (Sec. IV-A to IV-E). However, as mentioned in Sec. II, the models can be used with any model-based framework. This leads to the definition of an interface that the DetServ models have to implement (Sec. IV-F). Sec. V then describes how this interface is implemented for both models.

A. Parameter Considered: End-to-End Delay

There are numerous different QoS parameters that can be considered in industrial environments, e.g., resilience, packet loss, maximum jitter, average and maximum delay [34]–[36]. However, in most industrial cases, the most critical metric for applications is the response time [1], [36]. Though response time is also influenced by the processing time of the end hosts, we here only deal with the influence of the network and hence focus on guaranteeing maximum unidirectional end-to-end delay requirements of flows without packet loss. We refer to traffic requiring such guarantees as *real-time traffic*.

Along its path, a packet suffers from different types of delays: processing, queuing, transmission and propagation delays. Since the link characteristics are assumed to be known, the propagation delay for each link is known. The processing delay can usually be neglected. However, any assumption on the worst-case behavior of the hardware would allow to bound it at each node. Upper bounds on the queuing and transmission delays can, for their part, be computed using the network calculus results presented in Sec. III. The sum of all these components along the route of a flow makes up the total deterministic end-to-end worst-case delay bound for the flow.

B. Queue Link Network Topology

Obviously, the (queuing) delay a packet experiences on its way to its destination does not only depend on the path the packet follows but also on how the packet is scheduled at each output link. Because of its simplicity and ubiquity, we assume that non-preemptive strict priority scheduling is used.

From this, the route selection process for a flow must consider both the physical links the flow will traverse and the queues at which the flow will be buffered at each output link. As a consequence, Guck et al. [17], [18] introduced a *queue link* network topology. From the physical network topology, each directed physical link (u, v) is replaced by $Q_{u,v}$ queue links, where u and v are the source and destination nodes of the link and $Q_{u,v}$ is the number of priority queues at the scheduler of the link. Each link in the queue link network topology hence represents a physical link and a given queue at the ingress of this physical link, i.e., a different QoS level of transmission over this physical link. Route selection on this queue link network thus determines both the path that a flow takes through the physical network as well as the queue in which the flow will be buffered at each physical link.

Performing route selection on the queue link topology allows a flow to be assigned different priorities at each node, thereby increasing flexibility compared to other legacy [1] and SDN [7], [8], [13] approaches which usually assign fixed priorities to flows along their complete path. However, route selection is performed on a graph with a greater amount of edges, thereby increasing the routing procedure complexity.

C. Consideration of Best-Effort Traffic

One benefit of using Ethernet for guaranteeing real-time QoS is the interoperability with other IP networks such as a company's office network or the Internet itself. The traffic exchanged with these networks might not have such QoS requirements as the industrial traffic. The lowest priority queue of each link can be used for serving this so-called *best-effort* traffic. In this manner, the real-time traffic, which is only flowing through the higher priority queues, is not influenced by the best-effort traffic which is then only allowed to use resources which are left unused by the real-time flows.

Since best-effort traffic is allocated a single queue at each link, it can be routed using traditional SDN controller modules for routing (e.g., layer-two learning switch).

D. Problem Formulation

From a set of flows and the paths they follow in the queue link topology, the network calculus results presented in Sec. III allow to compute end-to-end delay bounds for each flow. Our initial problem is the following.

Problem 1: *For a set of real-time flows $\mathcal{F}$, find a route through the queue link topology for each flow $f \in \mathcal{F}$ such that the end-to-end delay requirement t_f of each flow is satisfied.*

As a result of the complexity increase due to the high number of edges in the graph on which route selection is performed, solving the problem using a mixed integer programming (MIP) formulation leads to intractable runtimes. Already hundreds of seconds or more are needed to solve the problem for small networks [17], [18]. Therefore, Guck et al. [17] proposed an online approach to solve the problem. Flows are taken one by one and embedded one at a time. They show that this approach can lead to results close to those of the MIP formulation in terms of number of embeddable flows, however having a much lower runtime. In such an approach, since the global goal of Problem 1 is to be able to embed all the candidate flows, each flow has to be embedded such that its consumption of resources is minimized, so as to maximize the probability of acceptance of forthcoming flow requests. As such, the following problem has to be solved.

Problem 2: *For a given flow f, find a route through the queue link topology such that* (i) *the end-to-end delay requirement t_f of the flow is satisfied,* (ii) *the end-to-end guarantees provided to previously embedded flows are still guaranteed, and* (iii) *the probability of future flow requests acceptance is maximized.*

Compared to the overall approach, this online approach has the additional advantage of being able to deal with scenarios for which the requests are not known *a priori* but are rather received at different points in time.

E. Interplay between Routing and Resource Allocation

As a result of this online approach, QoS routing is initiated by a query of the data plane. This can be done by contacting the northbound interface (NBI) of the SDN controller or by means of a *PACKET_IN* OpenFlow message [20]. The query should at least contain the flow characteristics (e.g., in our case, source, destination(s), burst, rate and maximum packet size) and QoS requirements (e.g., in our case, maximum delay). In case of queries coming from *PACKET_IN* messages, these parameters can be inferred from the packet header (port numbers, transport protocol, etc.). Based on this input and on the current state of the network, routing can then be performed.

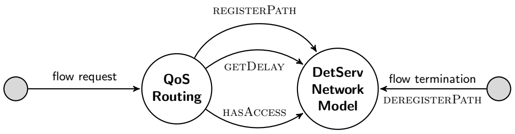


Fig. 2: Operation and interface of the DetServ network models. A flow request is handled by the QoS routing procedure whose task is to find a suitable route in the queue link topology for the corresponding flow (i.e., to solve *Problem 2*). While routing, the getDelay and hasAccess methods of the network model are used for the computation of worst-case delays and for access control. The registerPath and deregisterPath functions are for their part used to update the state of the network model to reflect the embedding or removal of a flow.

When this is done, the corresponding forwarding rules are pushed to the data plane.

The embedding of a new flow must not violate the delay guarantees provided to previously embedded flows. Indeed, as shown by Eqn. 1, embedding a new flow updates the service offered to other flows, which in turn updates the delay bounds for these flows (Eqn. 4), which might potentially in turn cause the violation of the end-to-end delay guarantees already provided to these flows.

As a result, resources usage has to be taken into account while routing. The approach proposed by Guck et al. [18] is to split the problem into two subproblems that can be solved separately.

- The *resource allocation problem*, which consists in finding the amount of resources to allocate to all the different queues at each link of the network, and
- the *routing problem*, which consists in finding a path in the queue link topology for which the delay of the new flow is guaranteed and that only uses resources that are still available, thereby ensuring that the guarantees of previously embedded flows are not violated.

F. Interface of a Generic DetServ Network Model

In this article, we consider that the resource allocation algorithm has allocated resources to the different queues in the network and that we have a routing algorithm able to look for a delay-constrained path in the network (*(i)* in Problem 2) using only resources that are still available (*(ii)* in Problem 2) and in a way that consumes the least amount of resources (*(iii)* in Problem 2). For *(iii)*, an option is for the routing algorithm to use a cost function whose minimization maximizes the probability of future requests acceptance. A delay-constrained least-cost (DCLC) routing algorithm is then needed. For *(i)* and *(ii)*, the network model has to provide an interface to the routing algorithm. This interface consists of the following four so-called *model functions*.

- getDelay: computes the worst-case delay of a given queue link edge.
- hasAccess: checks whether or not there are still enough resources available for a given flow at a given queue link edge.
- registerPath: updates the model state to reflect the embedding of a new flow.
- deregisterPath: updates the model state to reflect the removal of a previously embedded flow.

The processing of a flow request is then illustrated in Fig. 2. Upon receipt of a flow request, the QoS routing algorithm searches for a solution to *Problem 2*. While searching, the algorithm uses the getDelay and hasAccess methods to obtain the delay of an edge and to check if enough resources are available at an edge. Once a path has been found, the registerPath method is used to update the state of the model in order to reflect the embedding of the new flow. Similarly, the deregisterPath method is used upon the receipt of a flow termination notification in order to reflect the removal of the corresponding flow.

How these methods are implemented depends on how and which resources are allocated and managed at each queue. In the next section, we present our two DetServ models implementing these four model functions for providing deterministic guarantees.

V. DetServ: Network Models

A. Notations

The physical and queue link graphs are respectively denoted by $\mathcal{P}$ and $\mathcal{G}$. The indices E and N are used to refer to the set of edges and nodes of the graphs. For example, $\mathcal{P}_E$ corresponds to the set of edges of the physical graph. The capacity of a physical link $(u, v) \in \mathcal{P}_E$ is denoted by $R_{u,v}$. We assume a non-preemptive strict priority scheduler with $Q_{u,v}$ queues at the physical link $(u, v) \in \mathcal{P}_E$. Edges in the queue link network are denoted by (u, v, p), where (u, v) is the corresponding physical link and $p \in \{1, \ldots, Q_{u,v}\}$ is the priority of the corresponding queue at the physical link, $Q_{u,v}$ being the lowest priority.

The set of active (i.e., embedded) flows in the network is denoted by $\mathcal{F}$. For a given embedded flow $f \in \mathcal{F}$ or for a given flow f requesting an embedding,

- r_f denotes the rate (as defined in Sec. III-A) of the flow,
- $b_f[u, v, p]$ denotes the burst size (as defined in Sec. III-A) of the flow at queue link (u, v, p) (as we have seen in Sec. III-B that the burst of a flow changes at each hop),
- t_f denotes the end-to-end delay requirement of the flow,
- l_f^{max} denotes the maximum packet size of the flow, and
- $P_f \subseteq \mathcal{G}_E$ denotes the set of queue link edges through which the flow is routed (empty set if the flow is not embedded yet).

We denote the maximum packet size in the network by L^{max}. If it is not known, the maximum Ethernet frame size can be used.

For a given queue link edge $(u, v, p) \in \mathcal{G}_E$,

- $\mathcal{F}_{u,v,p} \subseteq \mathcal{F}$ denotes the set of flows routed through the queue link edge,
- $\mathbf{U}_R[u, v, p]$ denotes the sum of the rates of the flows routed through the queue link edge, i.e.,

$$\mathbf{U}_R[u, v, p] \triangleq \sum_{f \in \mathcal{F}_{u,v,p}} r_f, \tag{7}$$

- $\mathbf{U}_B[u, v, p]$ denotes the sum of the bursts of the flows routed through the queue link edge, i.e.,

$$\mathbf{U}_B[u, v, p] \triangleq \sum_{f \in \mathcal{F}_{u,v,p}} b_f[u, v, p], \tag{8}$$

- $l_{u,v,p}^{max}$ denotes the maximum packet size of the aggregate flow traversing the queue link edge, i.e.,

$$l_{u,v,p}^{max} \triangleq \max_{f \in \mathcal{F}_{u,v,p}} \{l_f^{max}\}, \tag{9}$$

- $\mathbf{T}[u,v,p]$ denotes the worst-case delay of the queue link edge,
- $B_{max}(u,v,p)$ denotes the worst-case backlog at the queue link edge, and
- $\mathbf{A}_B[u,v,p]$ denotes the buffer capacity of the queue corresponding to the queue link edge.

Using these notations, Eqn. 2, 3, 4 and 5 can be respectively rewritten as

$$T_{u,v,p} = \frac{\sum_{j=1}^{p-1} \mathbf{U}_B[u,v,j] + \max\limits_{p+1 \leq j \leq Q_{u,v}} \{l_{u,v,j}^{max}\} + l_{u,v,p}^{max}}{R_{u,v} - \sum_{j=1}^{p-1} \mathbf{U}_R[u,v,j]}, \tag{10}$$

$$R_{u,v,p} = R_{u,v} - \sum_{j=1}^{p-1} \mathbf{U}_R[u,v,j], \tag{11}$$

$$\mathbf{T}[u,v,p] = \frac{\sum_{j=1}^{p} \mathbf{U}_B[u,v,j] + \max\limits_{p+1 \leq j \leq Q_{u,v}} \{l_{u,v,j}^{max}\} + l_{u,v,p}^{max}}{R_{u,v} - \sum_{j=1}^{p-1} \mathbf{U}_R[u,v,j]}, \tag{12}$$

and

$$B_{max}(u,v,p) = \mathbf{U}_B[u,v,p] + \mathbf{U}_R[u,v,p] T_{u,v,p}, \tag{13}$$

where $\beta_{R_{u,v,p},T_{u,v,p}}$ is the rate-latency service curve offered by a queue link edge $(u,v,p) \in \mathcal{G}_E$.

B. Flows Requirements: Mathematical Formulation

First, in order to respect the QoS requirements of embedded flows, we must have,

$$\sum_{(u,v,p) \in P_f} \mathbf{T}[u,v,p] \leq t_f \qquad \forall f \in \mathcal{F}. \tag{14}$$

Second, in order to avoid any buffer overflow (and hence any packet loss), we must have

$$B_{max}(u,v,p) \leq \mathbf{A}_B[u,v,p] \qquad \forall\ (u,v,p) \in \mathcal{G}_E. \tag{15}$$

C. Requirement for the Models: Fixed Per-Queue Delay

Both bounds in Eqn. 12 and 13 depend on $\mathbf{U}_B[u,v,j]$, $\mathbf{U}_R[u,v,j]$ and $l_{u,v,j}^{max}$ for some j, i.e., on the burst size, rate and maximum packet size of other flows embedded on the same physical link. This means that, if a new flow is embedded on a link $(u,v) \in \mathcal{P}_E$, the worst-case delay (Eqn. 12) and buffer consumption (Eqn. 13) of some of the queues at the link will be updated, thereby possibly violating requirements of some previously embedded flows (Eqn. 14 and 15). As explained in Sec. IV-E, we do not want to check that the delay requirements of the already embedded flows are still satisfied (i.e., check Eqn. 14) after a new flow embedding. That means that the worst-case bounds $\mathbf{T}[u,v,p]$ have to be bounded independently of the status of the network. In such a way, if Eqn. 14 for a given flow f was satisfied when the flow was embedded, it will be kept satisfied for the whole runtime of the network.

The two different models we present in the next sections differ in the way they fix the $\mathbf{T}[u,v,p]$ bounds. While the *multi-hop model* upper-bounds the variable parts of Eqn. 12, the *threshold-based model* fixes $\mathbf{T}[u,v,p]$ itself and lets the variables vary until the fixed threshold is reached.

D. Multi-Hop Model (MHM)

Our first model, the *multi-hop model* (MHM), extends the access control scheme proposed by Schmitt et al. [33] for one aggregation node in order to consider multi-hop paths and physical buffer limits. This extension was already partially described by Guck et al. [18] but the limited capacity of buffers was not considered. We here present an updated version.

1) Network Calculus Developments: The model finds an upper bound for $\mathbf{T}[u,v,p]$ by replacing the variable components in Eqn. 12 with upper bounds for them.

Firstly, the packet size of a flow cannot be greater than the maximum packet size in the network. That is,

$$l_f^{max} \leq L^{max} \quad \forall f \in \mathcal{F}. \tag{16}$$

Secondly, the model assumes that the resource allocation algorithm allocates a data rate $\mathbf{A}_R[u,v,p]$ to each queue link edge. The rate of the aggregate flow traversing a queue is then limited by the access control scheme to the rate allocated to this queue. That is,

$$\mathbf{U}_R[u,v,p] \leq \mathbf{A}_R[u,v,p] \quad \forall\ (u,v,p) \in \mathcal{G}_E. \tag{17}$$

From Eqn. 12 and 13, Eqn. 16 and 17 allow to compute the following upper bounds for the worst-case delay and backlog at a queue link edge.

$$\mathbf{T}[u,v,p] \leq \frac{\sum_{j=1}^{p} \mathbf{U}_B[u,v,j] + 2L^{max}}{R_{u,v} - \sum_{j=1}^{p-1} \mathbf{A}_R[u,v,j]} \tag{18}$$

$$B_{max}(u,v,p) \leq \mathbf{U}_B[u,v,p] + \mathbf{A}_R[u,v,p] \frac{\sum_{j=1}^{p-1} \mathbf{U}_B[u,v,j] + 2L^{max}}{R_{u,v} - \sum_{j=1}^{p-1} \mathbf{A}_R[u,v,j]} \tag{19}$$

Finally, the burst of the aggregate flow traversing a queue has to be limited such that it does not generate any buffer overflow. Mathematically, combining Eqn. 15 and 19, we have

$$\mathbf{U}_B[u,v,p] + \mathbf{A}_R[u,v,p] \frac{\sum_{j=1}^{p-1} \mathbf{U}_B[u,v,j] + 2L^{max}}{R_{u,v} - \sum_{j=1}^{p-1} \mathbf{A}_R[u,v,j]} \leq \mathbf{A}_B[u,v,p]. \tag{20}$$

If we refer to the maximum allowed burst at a queue as $\mathbf{M}_B[u,v,p]$, i.e.,

$$\mathbf{U}_B[u,v,p] \leq \mathbf{M}_B[u,v,p] \qquad \forall (u,v,p) \in \mathcal{G}_E, \tag{21}$$

these $\mathbf{M}_B[u,v,p]$ bounds must be computed such that

$$\mathbf{M}_B[u,v,p] + \mathbf{A}_R[u,v,p] \frac{\sum_{j=1}^{p-1} \mathbf{M}_B[u,v,j] + 2L^{max}}{R_{u,v} - \sum_{j=1}^{p-1} \mathbf{A}_R[u,v,j]} \leq \mathbf{A}_B[u,v,p]. \tag{22}$$

Eqn. 22 allows to recursively compute the $\mathbf{M}_B[u,v,p]$ values independently of the state of the network. $\gamma_{\mathbf{M}_B[u,v,p],\mathbf{A}_R[u,v,p]}$

corresponds to the maximum arrival curve allowed to traverse a given queue link (u, v, p). We will denote it as $\mathbf{M}_\alpha[u, v, p]$.

As a result, Eqn. 18, can be rewritten as

$$\begin{aligned}\mathbf{T}[u,v,p] &\le \frac{\sum_{j=1}^{p}\mathbf{M}_B[u,v,j]+2L^{max}}{R_{u,v}-\sum_{j=1}^{p-1}\mathbf{A}_R[u,v,j]} \\ &\triangleq \mathbf{T}^{MHM}[u,v,p],\end{aligned} \tag{23}$$

where $\mathbf{T}^{MHM}[u, v, p]$ is the upper bound of the worst-case delay $\mathbf{T}[u, v, p]$ of a queue link $(u, v, p) \in \mathcal{G}_E$ used by the MHM and that is independent of the state of the network.

2) Model Operations: From these developments, the four model functions of the MHM are defined in Fig. 3.

```
function GETDELAY($(u, v, p)$)
    return $\mathbf{T}^{MHM}[u, v, p]$ (Eqn. 23)

function HASACCESS($f$, $(u, v, p)$)
    if $\mathbf{U}_B[u, v, p] + b_f[u, v, p] \le \mathbf{M}_B[u, v, p]$ and $\mathbf{U}_R[u, v, p] + r_f \le \mathbf{A}_R[u, v, p]$ then
        return true
    else
        return false

function REGISTERPATH($f$, $P$)
    for $(u, v, p) \in P$ do
        $\mathbf{U}_B[u, v, p] \leftarrow \mathbf{U}_B[u, v, p] + b_f[u, v, p]$
        $\mathbf{U}_R[u, v, p] \leftarrow \mathbf{U}_R[u, v, p] + r_f$

function DEREGISTERPATH($f$, $P$)
    for $(u, v, p) \in P$ do
        $\mathbf{U}_B[u, v, p] \leftarrow \mathbf{U}_B[u, v, p] - b_f[u, v, p]$
        $\mathbf{U}_R[u, v, p] \leftarrow \mathbf{U}_R[u, v, p] - r_f$
```

Fig. 3: The four model functions for the multi-hop model. The model uses $\mathbf{U}_B[u, v, p]$ and $\mathbf{U}_R[u, v, p]$ as state variables for each queue $(u, v, p) \in \mathcal{G}_E$. The registration and deregistration of a path in the network simply consists in updating these variables. For its part, the access control simply consists in checking that the state variables never exceed their respective limits, which are defined is such a way that, if the variables stay below these limits, *(i)* the maximum backlog at a queue will never exceed the buffer size of the queue, thereby avoiding any buffer overflow, and *(ii)* the maximum delay for a queue will never exceed the delay returned by GETDELAY for this queue.

The model uses $\mathbf{U}_B[u, v, p]$ and $\mathbf{U}_R[u, v, p]$ as state variables for each queue $(u, v, p) \in \mathcal{G}_E$. The registration and deregistration methods simply consist in updating these variables. The access control for a new flow simply consists in checking that Eqn. 17 and 21 are always satisfied. Based on the rate allocated by the resource allocation algorithm to each queue in the network, the $\mathbf{M}_B[u, v, p]$ and $\mathbf{T}^{MHM}[u, v, p]$ bounds can be computed once for each queue link edge $(u, v, p) \in \mathcal{G}_E$ and the four model functions then require low computation overhead.

An example of the detailed operation of the model at a given physical link $(u, v) \in \mathcal{P}_E$ is given as supplementary material. Basically, once the $\mathbf{M}_\alpha[u, v, p]$ curves have been recursively computed, flows will be accepted at a queue p of the link as long as the resulting aggregate arrival curve traversing the queue stays below the $\mathbf{M}_\alpha[u, v, p]$ limit curve.

3) Limitations of the Multi-Hop Model: The MHM requires a data rate to be allocated to each queue. These allocated data rates then define the maximum rate and burst allowed at each queue, as well as the maximum delay of each queue. The access control checks the availability of two resources: burst and rate. Hence, it can happen that the access to a queue is blocked because its rate budget is exhausted, while its burst limit is not reached. In such a situation, it would be beneficial to artificially reduce the buffer size $\mathbf{A}_B[u, v, p]$ of the queue. Indeed, this would, by Eqn. 22, reduce $\mathbf{M}_B[u, v, p]$ (which is not a problem since the remaining burst budget will never be used because of the data rate bottleneck) and lower priority queues could then either *(i)* see their maximum delay reduced (by Eqn. 23) or *(ii)* see their maximum allowed burst or rate increased (by Eqn. 22).

From this observation, the resource allocation algorithm should also assign a buffer capacity to each queue, thereby being allowed to artificially reduce the capacity of a buffer in order to trade it against lower delay or more rate or buffer for other queues. Note that the opposite situation could also happen. That is, the buffer capacity could be the bottleneck, in which case it would be beneficial to trade off rate in order to increase the maximum allowed bursts or reduce the maximum delays at other queues. In other words, the MHM requires the resource allocation algorithm to be responsible for adjusting the trade-off between the resources, that is, to make an *a priori* choice between buffer space, data rate and delay. However, adjusting this trade-off requires to know what is the bottleneck in the network or at a given link. Will flows be rejected because there is no buffer capacity available anymore, no data rate available anymore, or because their delay cannot be satisfied? Unfortunately, answering this question requires to know the traffic demand, which is, because of our online approach (see Sec. IV-D), not the case.

E. Threshold-Based Model (TBM)

The *threshold-based model* (TBM) solves the shortcoming of the MHM by choosing between buffer capacity and data rate as flows are added to the network, thereby allocating the rate and buffer capacity resources only when needed rather than pre-allocating them without knowing future flow requests.

1) Model Operations: In the TBM, the worst-case delay of each queue (Eqn. 12) is simply fixed by defining a threshold $\mathbf{T}^{TBM}[u, v, p]$. Then, flows are accepted in a queue as long as the worst-case delay of the queues at the same link do not exceed their respective thresholds.

This approach has two main benefits. First, as mentioned, the data rate and buffer space resources are allocated only when needed, rather than *a priori*, thereby leading to a better utilization of the resources. Second, the resource allocation algorithm is now simplified since it only has to optimize with respect to one variable (the time) rather than two (buffer space and data rate). In other words, the TBM replaces the three data rate, buffer space and delay resources by a *single* one: delay.

Unfortunately, this comes at the cost of a higher computational complexity for access control. Indeed, as $\mathbf{U}_R[u, v, p]$ is not bounded anymore, it is not anymore possible to compute a bound on the service curves offered to the different queues (i.e., on the $T_{u,v,p}$ and $R_{u,v,p}$ parameters). Adding a flow in a queue will update the service curve offered to lower priority queues (by Eqn. 10 and 11). Hence, when adding a flow in a queue (u,v,p), besides checking that $\mathbf{T}[u,v,p] \leq \mathbf{T}^{TBM}[u,v,p]$ for this queue, the access control mechanism has to check that the thresholds of lower priority queues are also not exceeded. That is, the access control mechanism has to check that

$$\mathbf{T}[u,v,j] \leq \mathbf{T}^{TBM}[u,v,j] \quad \forall\ j : p \leq j \leq Q_{u,v}. \tag{24}$$

Besides, the access control scheme has to make sure that no buffer overflow can be caused by the embedding of the new flow, i.e.,

$$B_{max}(u,v,j) \leq \mathbf{A}_B[u,v,j] \quad \forall\ j : p \leq j \leq Q_{u,v}. \tag{25}$$

Note that Eqn. 12 and 13 require the knowledge of the maximum packet size in lower priority queues. This means that, when embedding a flow in a queue, higher priority queues also have to be checked since the maximum packet size might have changed. However, because best-effort traffic flows through the lowest priority queue, we cannot keep track of this value and we hence replace it by L^{max}. From this, we have

$$\mathbf{T}[u,v,p] \leq \frac{\sum_{j=1}^{p} \mathbf{U}_B[u,v,j] + L^{max} + l_{u,v,p}^{max}}{R_{u,v} - \sum_{j=1}^{p-1} \mathbf{U}_R[u,v,j]}, \tag{26}$$

and

$$\begin{aligned} B_{max}(u,v,p) \leq\ & \mathbf{U}_B[u,v,p]\ + \\ & \mathbf{U}_R[u,v,p] \frac{\sum_{j=1}^{p-1} \mathbf{U}_B[u,v,j] + L^{max} + l_{u,v,p}^{max}}{R_{u,v} - \sum_{j=1}^{p-1} \mathbf{U}_R[u,v,j]}, \end{aligned} \tag{27}$$

which only depend on the state of higher priority queues. As a result, it is sufficient to only check lower priority queues when embedding a new flow.

The four model functions of the TBM are given in Fig. 4. As for the MHM, the registration and deregistration methods simply consist in updating the state variables. However, we here have one additional state variable: the maximum packet size at each queue. The delay of a queue link edge is now the one fixed by the resource allocation algorithm and the access control scheme simply verifies that Eqn. 24 and 25 are still verified for the subject queue and the lower priority queues if the flow is embedded.

An example of the detailed operation of the model at a given physical link is given as supplementary material.

2) Shortcomings of the TBM: The TBM, though having major advantages, presents two drawbacks. First, the complexity of the HASACCESS model function is increased by a factor of up to $Q_{u,v}$. Because the HASACCESS function is called each time the routing algorithm visits an edge, this might have a considerable influence on the overall request processing time. However, we will show in Sec. VI-B4 that the increase in runtime is acceptable for industrial scenarios. Second, the model presents an inherent blocking problem. Indeed, if a low priority queue is close to its delay threshold, it will block further embeddings in higher priority queues, even if these are still far from their own delay threshold. Consequently, the routing algorithm has now to operate cautiously when embedding flows in order to avoid such a blocking situation which would inevitably cause resource waste.

```
function GETDELAY($(u, v, p)$)
    return $\mathbf{T}^{TBM}[u, v, p]$

function HASACCESS($f$, $(u, v, p)$)
    for $i \in \{p, \dots, Q_{u,v}\}$ do
        $\mathbf{T}[u, v, i] \leftarrow$ Eqn. 26 including new flow
        $B_{max}(u, v, i) \leftarrow$ Eqn. 27 including new flow
        if $\mathbf{T}[u, v, i] > \mathbf{T}^{TBM}[u, v, i]$ or $B_{max}(u, v, i) > \mathbf{A}_B[u, v, i]$ then
            return false
    return true

function REGISTERPATH($f$, $P$)
    for $(u, v, p) \in P$ do
        $\mathbf{U}_B[u, v, p] \leftarrow \mathbf{U}_B[u, v, p] + b_f[u, v, p]$
        $\mathbf{U}_R[u, v, p] \leftarrow \mathbf{U}_R[u, v, p] + r_f$
        Update $l_{u,v,p}^{max}$

function DEREGISTERPATH($f$, $P$)
    for $(u, v, p) \in P$ do
        $\mathbf{U}_B[u, v, p] \leftarrow \mathbf{U}_B[u, v, p] - b_f[u, v, p]$
        $\mathbf{U}_R[u, v, p] \leftarrow \mathbf{U}_R[u, v, p] - r_f$
        Update $l_{u,v,p}^{max}$
```

Fig. 4: The four model functions for the threshold-based model. The threshold for the delay of a queue is chosen by the resource allocation algorithm. Access to a queue link edge $(u,v,p) \in \mathcal{G}_E$ is then checked by checking that the new worst-case bound does not exceed its threshold value. Besides, as the state of a queue influences the state of lower priority queues, the access control mechanism also has to check that the worst-case bounds of lower priority queues do not exceed their respective thresholds. Finally, the buffer capacity also has to be checked for the different queues.

F. Computation of the Burst Increase

1) Per-Flow Worst-Case Increase: Though we mentioned that the burst of a flow changes at each hop, we did not explain how these changes can be computed on a *per-flow* basis and how this impacts delay computations. From Sec. III, we know that an aggregate flow with arrival curve $\gamma_{\mathbf{U}_R[u,v,p],\mathbf{U}_B[u,v,p]}$ traversing a queue offering a service curve $\beta_{R_{u,v,p},T_{u,v,p}}$ will see its burst $\mathbf{U}_B[u,v,p]$ increased by $\mathbf{U}_R[u,v,p]T_{u,v,p}$, i.e.,

$$\mathbf{U}_B^*[u,v,p] = \mathbf{U}_B[u,v,p] + \mathbf{U}_R[u,v,p]T_{u,v,p}. \tag{28}$$

$\mathbf{U}_B^*[u,v,p]$ is the new burst of the entire aggregate. Nevertheless, the flows composing this aggregate might take different routes at the next hop and the individual burst increases of the individual flows composing the aggregate must be computed. From Eqn. 7 and 8, Eqn. 28 can be rewritten as

$$\mathbf{U}_B^*[u,v,p] = \sum_{f \in \mathcal{F}_{u,v,p}} \left(b_f[u,v,p] + r_f T_{u,v,p} \right), \tag{29}$$

which highlights the contribution of each individual flow to the burst increase. Therefore, the burst of a flow $f \in \mathcal{F}_{u,v,p}$ when entering a queue $(s,t,q) \in \mathcal{G}_E$ after having traversed queue $(u,v,p) \in \mathcal{G}_E$ is given by

$$b_f[s,t,q] = b_f[u,v,p] + r_f T_{u,v,p}, \tag{30}$$

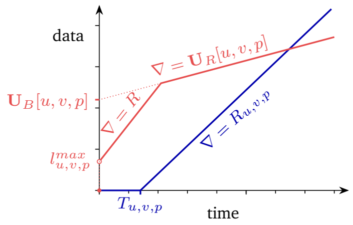


Fig. 5: Shaped arrival curve of an aggregate flow traversing a queue $(u,v,p) \in \mathcal{G}_E$ coming from an input link with rate R. The knowledge of the physical properties of the input link of the flow allows to limit the burst and rate of the aggregate respectively to the maximum packet size $l^{max}_{u,v,p}$ of the flow and to the maximum rate R of the link. Graphically, we can easily see that such a shaping reduces the values of the backlog and delay bounds.

which depends, through $T_{u,v,p}$, on other flows traversing the same physical link. This dependency of the burst increase on other embedded flows is problematic. Indeed, this means that, when a flow is embedded in a queue, the burst increases of other flows traversing the same link might change, possibly violating already performed access control checks. As explained in Sec. IV-E, such a situation must be avoided and the burst increase of a flow must therefore be, as the worst-case delay of a queue, independent of the network state. From Eqn. 10 and 12, it is straightforward that

$$T_{u,v,p} \leq \mathbf{T}[u,v,p] \quad \forall (u,v,p) \in \mathcal{G}_E. \tag{31}$$

Therefore, the burst increase of a flow f is such that

$$b_f[s,t,q] \leq b_f[u,v,p] + r_f \mathbf{T}[u,v,p], \tag{32}$$

and the MHM and TBM can compute $b_f[s,t,q]$ using $b_f[u,v,p] + r_f \mathbf{T}^{MHM}[u,v,p]$ and $b_f[u,v,p] + r_f \mathbf{T}^{TBM}[u,v,p]$, respectively, which are independent of the network state.

2) Exception: We note that, if the cycle time (or inter-arrival time of packets) of a flow is greater than its delay bound, then the burst increase can be neglected. Indeed, in such a case, a packet is ensured to reach its destination before the following packet is sent. As a result, packets of the same flow will not queue up at any queue and the burst of the flow will never increase.

G. Input Link Shaping (ILS)

1) Towards Lower Bounds: So far, we considered that the arrival curve of the aggregate flow entering a queue $(u,v,p) \in \mathcal{G}_E$ is $\gamma_{\mathbf{U}_R[u,v,p],\mathbf{U}_B[u,v,p]}$, that is, that the burst of the aggregate flow entering a queue is given by the sum of all the bursts of all the flows composing the aggregate (see Eqn. 8). Nevertheless, the individual flows come from physical links of finite capacity. Hence, the amount of traffic entering a given queue is further limited by the capacity of the links it is coming from. Considering this new bound on the traffic entering a queue, we can lower the corresponding arrival curves, yielding lower bound values and thereby potentially accepting more flows in the network.

The idea, to which we refer to as *input link shaping* (ILS), is illustrated in Fig. 5 for a given queue (u,v,p) traversed by a set of flows coming from a common input link of capacity R. From the knowledge of the physical properties of the input link, besides its traditional arrival curve, the aggregate flow is additionally constrained by a token bucket arrival curve with rate R and burst $l^{max}_{u,v,p}$. A better arrival curve for a flow constrained by two different token bucket arrival curves being the minimum of these curves [32], the new arrival curve of the aggregate flow is of the form shown in Fig. 5. We can see that the backlog and delay bounds will always be smaller than if shaping was not taken into account, highlighting the benefit of ILS.

2) ILS Does Not Contradict Network Calculus: In Sec. III, we have presented network calculus results for computing the output arrival curve of a flow after it has traversed a network node characterized by a given service curve. We now propose to cut off a part of this arrival curve by shaping it with the input link rate. Though this is intuitive, it might seem to contradict the network calculus results which say that a big burst could happen. The justification is the following. The results of network calculus theory are solely based on the arrival and service curve concepts. While the service curve gives a lower bound on the service a network node will offer to a flow, it does not specify anything regarding the maximum service the node could offer, hence potentially allowing infinite service, i.e., infinite rate. Taking this into account, network calculus results consider that an infinite service could instantly output the current backlog as a single burst, which is why, in Eqn. 6, the output burst corresponds to the worst-case backlog. As a matter of fact, we know more than what the service curve concept provides to network calculus theory. Indeed, we know that the service provided by the network node can never be higher than the link rate. The shaping we introduce is hence augmenting network calculus results, rather than contradicting them.

3) Adapting the Multi-Hop Model: In the MHM, the worst-case delay of a queue is made independent of the network state by statically defining the maximum arrival curves allowed at each queue. Therefore, to keep the worst-case delay of a queue static, ILS must be introduced in a way that is also independent of the network state. For a given queue-link edge $(u,v,p) \in \mathcal{G}_E$, the worst-case burst that could ever enter the queue is nL^{max} where n is the number of links entering node u. The worst-case rate is for its part given by the sum of the rates of the individual incoming links. Therefore, the arrival curve $\mathbf{M}_\alpha[u,v,p]$ considered so far can be replaced by

$$\mathbf{M}_\alpha^{ILS}[u,v,p] = \min\left\{ \sum_{x:(x,u)\in\mathcal{P}_E} \left(\gamma_{R_{x,u},L^{max}}\right), \mathbf{M}_\alpha[u,v,p] \right\}. \tag{33}$$

Two options are then possible.

First, one can compute the maximum allowed bursts $\mathbf{M}_B[u,v,p]$ without considering ILS and then shaping the obtained $\mathbf{M}_\alpha[u,v,p]$ curves according to Eqn. 33 in order to reduce the worst-case delay at each queue.

Second, one can compute the maximum allowed bursts $\mathbf{M}_B[u,v,p]$ using the already shaped curve. That is, $\mathbf{M}_B[u,v,p]$ is obtained as the maximum value such that the worst-case burst generated by $\mathbf{M}_\alpha^{ILS}[u,v,p]$ does not exceed the allocated buffer capacity $\mathbf{A}_B[u,v,p]$. Because the shaped arrival curve is lower or equal to the original arrival curve, the obtained maximum allowed burst $\mathbf{M}_B[u,v,p]$ will always be

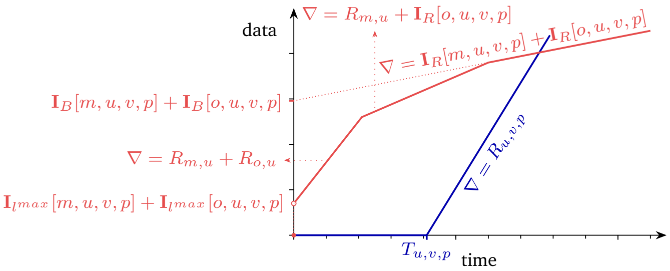


Fig. 6: Example of shaped arrival curve for the TBM. The aggregate flow traversing queue (u, v, p) comes from two input links (m, u) and (o, u). Each input link has shaped the traffic it carries as shown in Fig. 5 and the resulting aggregate, corresponding to the sum of the two shaped arrival curves, is composed of three segments with decreasing slopes. The backlog and delay bounds can then be reached at any angular point of both curves. The bounds will always be lower than if shaping was not taken into account.

greater than without considering ILS. The calculation of the worst-case delay is then also done using the shaped arrival curve $\mathbf{M}_{\alpha}^{ILS}[u, v, p]$.

These two options once more highlight the trade-off between the different resources in the MHM. While the first option reduces delay, the second increases the maximum allowed bursts.

Whichever option is considered, once these computations are done, the four model functions described in Fig. 3 are left unchanged.

4) Adapting the Threshold-Based Model: While present, the benefits of ILS for the MHM are limited. Indeed, since we only keep track of worst-case arrival curves, ILS also has to be done worst-case, i.e., considering the worst-case packet size and rates coming from each input link.

For the TBM, the arrival curves are computed live. Therefore, the maximum packet size and rate for each incoming link can also be computed on the fly. This can be done by introducing three new state variables $\mathbf{I}_R[m, u, v, p]$, $\mathbf{I}_B[m, u, v, p]$ and $\mathbf{I}_{l^{max}}[m, u, v, p]$ keeping track respectively of the rate, burst and maximum packet size of the aggregate flow coming from the physical edge (m, u) and traversing the queue-link edge (u, v, p). Instead of considering the arrival curve consisting of the sum of all the arrival curves of the flows entering the queue, the contribution of each input link can now be shaped individually. That is, the arrival curve considered at a queue (u, v, p) is now

$$\sum_{x:(x,u)\in\mathcal{P}_E} \left(\min\left\{\gamma_{R_{x,u},\mathbf{I}_{l^{max}}[x,u,v,p]}, \gamma_{\mathbf{I}_R[x,u,v,p],\mathbf{I}_B[x,u,v,p]}\right\}\right), \tag{34}$$

i.e., a sum of shaped arrival curves.

An example for two input links is shown in Fig. 6. One can see that the summed up arrival curve can have up to n knee points, where n is the number of physical input links.

For the same reasons as for the MHM, but with increased impact since shaping is done with the current real values, the computed worst-case delay and backlog values will be lower. As a consequence, the limits $\mathbf{T}^{TBM}[u, v, p]$ and $\mathbf{A}_B[u, v, p]$ will be reached later, thereby potentially allowing more flows to be accepted.

Obviously, the getDelay method in Fig. 3 does not change. The registerPath and deregisterPath methods have to be updated to keep track of the new state variables. For its part, the hasAccess method only has to be changed at lines 6-7. Since the arrival curves are not token buckets anymore, the formulas for computing the worst-case delay $\mathbf{T}[u, v, p]$ and backlog $B_{max}(u, v, p)$ are not valid anymore and these values have now to be computed geometrically (see Parag. V-G7).

5) Burst Increase with Shaped Arrival Curves: Unfortunately, when the arrival curve is shaped, the computation of the burst increase becomes mathematically much more complex [32]. In particular, its decomposition into the contributions by the different flows as in Sec. V-F becomes then much less trivial. For simplicity, we will therefore consider that the burst increase is still computed using Eqn. 32.

6) Impact on the Performance of the MHM: As mentioned, because the MHM performs shaping based on worst-case values, we expect the impact on the amount of flows that can be embedded to be quite low. Nevertheless, as everything is computed during initialization, the request processing time of the MHM should not be affected by ILS. Hence, for the MHM, ILS has only benefits, though limited.

7) Impact on the Performance of the TBM: On the contrary, the TBM performs shaping based on the current traffic. Hence, the impact of ILS on the amount of flows that can be accepted in the network is expected to be greater than for the MHM. While ILS does not slow down the MHM, the runtime of the TBM should be much more affected. Indeed, the increased amount of knee points in the arrival curves does not allow anymore the computation of the worst-case delay and burst with formulas. From the convexity of the region between the curves (see Fig. 6), the delay (resp. backlog) bound can be computed by comparing the horizontal (resp. vertical) deviation at each knee point of the two curves. This inevitably slows down the hasAccess method. Hence, ILS is expected to have a major impact on the TBM, both in terms of increased performance and increased runtime.

VI. Evaluation

The evaluation of the proposed models is separated in two parts. First, in Sec. VI-A, we run a packet-level simulation of one physical link managed by the different models and observe the amount of flows that can be accepted at the link and the delay experienced by the individual packets. The goal is to confirm that the models respect the delay guarantees provided to the different flows and to observe the higher flexibility of the TBM. Although the simulation is performed only at a single link, this also confirms that the models are valid for end-to-end delays. Indeed, if the worst-case delay of each queue is guaranteed, the end-to-end delay of each flow, corresponding to the sum of the individual worst-case delays of each queue visited by the flow, is also guaranteed. Second, in Sec. VI-B, we run a network-wide simulation by generating series of flow requests for different network settings and observe the request processing time for the different models, along with the amount of flows they can accept. The goal is to quantify the additional runtime required by the TBM and hence to determine whether or not it is viable for online request processing in industrial environments. Besides, we want to observe the impact of ILS and confirm our expectations formulated in Sec. V-G6 and V-G7. Note that, for the MHM with ILS, we used the first option described in Sec. V-G3.

A. *Packet-level Simulation: Confirming Correctness*

1) Setup: Saturated Link Simulation: We simulate the access control of a single 1 Gbps link with four priority queues and varying amount of input links (1, 2, 3, 5 and 10). For each model and amount of input links, we generate flow registration and termination requests during 100 seconds. We generate requests at a rate high enough for saturating the link (250 requests per second) and hence for experiencing rejections of requests.

2) Resource Allocation Algorithms: As we have seen in Sec. V-D and V-E, the two models require different types of resource allocation algorithms. We define two algorithms which lead to the same delays for the different queues. These delay values are chosen so that they lead to a nice distribution of QoS levels among the queues in both models. The algorithm for the TBM assigns the delays 0.487 ms (high priority), 1.437 ms, 3.035 ms, and 4.709 ms (low priority) to the different queues. The algorithm for the MHM assigns the rates 51.2 MB/s (high priority), 24.622 MB/s, 8.349 MB/s, and 3.953 MB/s (low priority) to these same queues and the buffer capacity of 60 KB to all of them.

3) First Configuration: The TBM Performs Better:

a) Request Types: In a first configuration, each request is defined by a data rate (between 50 KB/s and 150 KB/s), a burst size (between 70 B and 150 B), a maximum packet size (between 64 B and the burst of the flow) and a delay constraint (between 10 ms and 100 ms) which are uniformly randomly distributed in their respective ranges. These are values in line with traffic traces observed in an operational industrial wind park network in the context of the VirtuWind H2020 European Project [37]. We consider L^{max} as the maximum Ethernet frame size including preamble, VLAN tag and inter-frame gap, i.e., $L^{max} = 1542$ B. Because the delay constraint is always greater than the delay of any queue, the delay will not influence the rejection or acceptance of requests. The reason for this is that, since we are fully saturating the considered link, having requests rejected because of their delay constraint will not affect the amount of flows that can be embedded. The generated flow requests are evenly distributed among the different combinations of input link and queue of the considered link. Flow requests are characterized by a duration which is randomly generated from an exponential distribution with an average duration of 100 seconds, representing the long-duration characteristic of industrial flows.

b) Results: For each run, the amount of flows embedded at the link was sampled every second. The left diagram of Fig. 7 shows, for each amount of input link, the average and the standard deviation of these sampled values. We observe that the TBM considerably increases the amount of flows in the system – by around 50%. This shows the flexibility of the TBM. While it automatically adapted to the rate and burst characteristics of the requests, the MHM did not because of the *a priori* choice on the rate, buffer and delay trade-off. We observe that ILS does not provide any benefit for both models. For the MHM, since we use the first option mentioned in Sec. V-G3, ILS only reduces the delay of the queues. Since the delay does not influence the access control in our simulation, ILS has no impact on the MHM. For the TBM, ILS reduces both the delay and the maximum burst computation. However, as shown in Fig. 6, the maximum burst computation will be reduced only if one knee point of the arrival curve is after the knee point of the service curve. In our particular setup of requests distributed evenly among the combinations of input link and queue, the knee points of the arrival curves are always before the knee point of the service curve, thereby explaining why ILS has no impact in this configuration. During all the simulations, out of 909,267,506 transmitted packets, no packet loss was observed and the highest packet delay to deadline ratio was 1.07%.

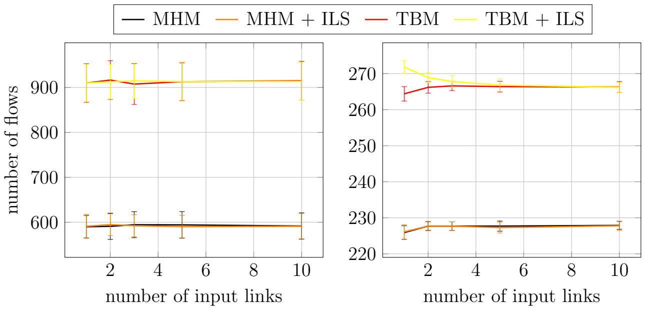


Fig. 7: On the left diagram, results of the packet-level simulation when flows are evenly distributed among the combinations of input link and queue. The TBM performs 50% better than the MHM and the ILS has no influence on the performance of both models. On the right diagram, one priority queue received more traffic from a given input link and the traffic was more bursty. The TBM still performs better than the MHM but the ILS now increases the performance of the TBM when the amount of input link is low. No packet loss nor deadline violation was observed in both scenarios.

4) Second Configuration: Impact of ILS:

a) Request Types: In a second configuration, we change the requests generation. The data rate and burst size are now varying between 7.086 KB/s and 8.086 KB/s and 879 B and 889 B, respectively. That is, the traffic is more bursty. Additionally, the requests are not anymore distributed evenly among the combinations of input link and queue but we generate 10 times more requests from the first input link for the highest priority queue than for all other combinations of input link and queue. In such a way, because more flows will be embedded in the high priority queue, the knee point of the corresponding shaped arrival curve will be shifted towards the right, thereby potentially reducing the maximum burst computation. Besides, since ILS shapes bursts, having more bursty traffic should increase the effect of ILS.

b) Results: The right diagram of Fig. 7 shows the result of the simulation for the second configuration. We can see that the TBM still behaves better than the MHM, confirming its higher flexibility: it adapted to the new characteristics of the requests. For the same reason as for the previous simulation, ILS has no impact on the MHM. On the other hand, ILS improves the performance of the TBM when the amount of input links is low. This is due to the fact that, when the amount of input link increases, the ratio of requests from the first input link for the high priority queue to the total of requests decreases. Therefore, as increasing the amount of input links leads to a more even distribution of requests among the combinations of input link and queue (as in the first simulation), the performance of ILS decreases. This shows that ILS behaves better when the flows at one link are not distributed evenly among the input links. During all the simulations, out of 36,747,129 transmitted packets, no packet loss was observed and the highest packet delay to deadline ratio was 0.47%.

B. *Monte Carlo Simulation*

The first part of our evaluation confirmed that our models are correct and showed that the TBM has the potential to outperform the MHM. Further, it has shown that the benefit of ILS grows when the traffic entering a link is not distributed evenly among the incoming links. However, we only observed the impact of ILS on the allowed bursts. In order to observe the impact of ILS on both the allowed bursts and the delay computation, a global network simulation is required. As part of a global QoS framework, the performance of a network model depends on the associated components (resource allocation and routing algorithms) and on the scenario (topology and type of flow requests). As such, with the aim of observing the influence of the network model only, we run a *Monte Carlo simulation* varying the different components (defined in Sec. VI-B1) and scenarios (defined in Sec. VI-B2) around the two models. In other words, we randomly vary the context in which the models are used in order to isolate their impact on the overall performance of the QoS framework.

1) Other Components: Resource Allocation and Routing Algorithms:

a) Resource Allocation Algorithms: For simplicity, resources are allocated among the queues identically for each link and following the resource allocation algorithms used in the first evaluation (Sec. VI-A).

b) Routing Algorithms: As proposed in Sec. IV-E, we use a DCLC algorithm. Among the plethora of such algorithms available in the literature, we consider *constrained Bellman-Ford* (CBF) [38] for its optimality, LARAC [39] for its good average performance [40] and Dijkstra computing the least-delay path (LDP) for its simplicity. We use different cost functions based on the priority of a queue link, the amount of average flows that can still be embedded in it or a combination of those.

2) Scenario:

a) Topologies: We define two network topologies based on lines and rings, which are typical structures for industrial networks. The first topology consists of a ring of size $m+1$ to which one programmable logic controller (PLC) and m lines composed of n remotes I/Os are attached. The second topology extends the first one by connecting another ring of size $m+1$ to the former loose ends of the remotes I/Os lines. The $(m+1)$th switch not connected to the lines is then connected to the PLC. Communication is only considered from the remote I/Os to the PLC. Both topologies can be scaled along the two n and m dimensions ($4 \leq n \leq 10, 4 \leq m \leq 10$).

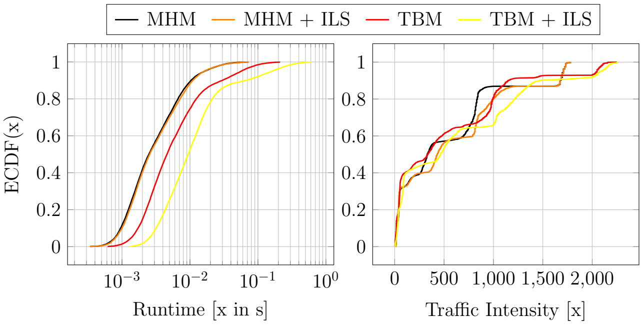


Fig. 8: Results of the evaluation. The left plot shows the empirical cumulative distribution function (ECDF) of the average runtime of one complete request life cycle (routing, embedding, deregistration) for the different models and their corresponding variations with input link shaping (ILS). The right plot shows the ECDF of the traffic intensity that the different models were able to reach. As expected, ILS has a greater impact on the TBM, both in terms of runtime and traffic intensity. We can observe that the TBM with ILS has the potential of reaching a high traffic intensity, but at the cost of a higher runtime.

b) Flow Requests: In order to generate a request for a given topology, a random remote I/O is selected to communicate with the PLC. Requests are defined as in Par. VI-A3a.

3) Evaluation Metrics: For a given iteration of the Monte Carlo simulation, i.e., for a given network model (and associated resource allocation algorithm), cost function, routing algorithm and topology, a binary search is started in order to find, for this scenario, the greatest *traffic intensity* for which every request can be embedded. Traffic intensity is defined as the arrival rate of flows multiplied by their average duration (100 s, see Par. VI-A3a), which also corresponds to the amount of active flows in the network (when the system converges). The traffic intensity associated to an iteration then corresponds to the maximum traffic intensity that could be reached. The runtime associated to an iteration corresponds to the average runtime of a request routing plus the average runtime of a path registration plus the average runtime of a path deregistration, i.e., to the average runtime of a request processing life cycle, that was observed during the complete binary search. The runtime was measured on a machine equipped with an Intel Xeon E5 2690v2 @ 3.00GHz processor.

4) Results: Fig. 8 shows the results of the Monte Carlo simulation. The left and right plot show the empirical cumulative distribution functions (ECDF) of, respectively, the runtime and the traffic intensity for the different models.

a) Runtime: As expected, the runtime of the MHM is not much affected by the introduction of ILS. Indeed, as we have seen in Sec. V-G6, the access control complexity of the MHM is the same with or without ILS. The small runtime difference in Fig. 8 is due to routing. As the delay values are changed by ILS, the routing algorithm will behave differently while searching for a path, hence possibly leading to slightly different running times.

We also observe that the TBM exhibits a higher runtime than the MHM. As mentioned in Sec. V-E2, this was expected and is due to the increased complexity of the access control method. More precisely, the TBM leads to an increase in the runtime by a factor of 2 to 4. This is consistent with the fact that the access control of the MHM checks only one queue, while the TBM checks up to $Q_{u,v}$ queues, which is 4 in our evaluation.

Contrary to the MHM, the runtime of the TBM is highly affected by the introduction of ILS (slowed down by a factor of around 2). As elaborated in Sec. V-G7, this was expected and is due to the increased complexity for computing horizontal and vertical deviations when introducing ILS to the TBM. However, the runtime stays lower than 350 ms in 99% of the cases and never exceeds 620 ms, which corresponds to a single-threaded worst-case performance of 1.6 requests per second, which is a reasonable performance for industrial applications.

Furthermore, because the runtime shift between the models stays roughly equal, Fig. 8 clearly shows that the network model is the main driver for the runtime of the system.

b) Traffic Intensity: We observe that the introduction of ILS brings a performance increase to both models, however more significant for the TBM. As elaborated in Sec. V-G4, this is due to the fact that the MHM performs ILS with worst-case values while the TBM performs ILS with the current flow values, which are inevitably lower. Because ILS does not affect the runtime of the MHM and sometimes improves its performance, this confirms that ILS is always beneficial for the MHM.

While Fig. 8 shows that the runtime is mostly influenced by the network model, we observe that this is not true for the traffic intensity. Indeed, the traffic intensity ECDFs present crossover points, which means that other components used in the Monte Carlo simulation have a significant impact on the performance of the models. This contrasts with the simulation in Sec. VI-A and shows that the MHM is able to outperform the TBM in some circumstances and hence that further study is required in order to identify which set of components (including the network model) is the most suitable for a specific scenario.

VII. Conclusions

In this article, we provided a detailed description of two network models (*DetServ*) for the provisioning of real-time QoS (e.g., for machine-to-machine (M2M) communications or production facilities) with SDN. The first model, the *multi-hop model* (MHM), assigns a rate and a buffer budget to each queue in the network. This model corresponds to an updated version of the model previously presented in [16], [18], which was not considering the buffer consumption of flows, i.e., not preventing packet loss. The second model, the main contribution of this article, simply fixes a maximum delay for each queue. We refer to this new network model as the *threshold-based model* (TBM). We have shown that, by avoiding an *a priori* choice on the trade-off between data rate and buffer capacity, the TBM is more flexible with respect to the characteristics of flows that are to be embedded in the network but that this comes at the price of an increase in the request processing time by a factor corresponding to the amount of priority levels in the network. We also gave an insight on how this increase in flexibility has the potential of reaching higher network utilization.

One major benefit of the proposed models is that they can be used with simple commodity switches supporting priority scheduling and any SDN protocol providing standard enqueuing and forwarding primitives, e.g., OpenFlow 1.0 [20].

We further introduced *input link shaping* (ILS), an extension to the two proposed models which takes into account the shaping of the traffic by the limited capacity of the links in the network. Our evaluations have shown that, while beneficial for both models, this extension has a much higher impact on the performance and runtime of the TBM. Our evaluations have additionally shown that the runtime cost of the higher flexibility and performance of the TBM with ILS stays reasonable for industrial scenarios. Indeed, the total request processing time never exceeds 620 ms.

In order to be part of a QoS framework, these models have to be combined with a routing procedure. This procedure was however not considered in this article and is a future research direction.

Acknowledgments

This work was supported by the European Unions Horizon 2020 Research and Innovation Programme under Grant 671648 (VirtuWind) and ERC Grant 647158 (FlexNets). We are grateful to Onur Ayan, Nemanja Deric, Murat Gürsu, Mu He, Alberto Martínez Alba, Petra Stojsavljevic, Samuele Zoppi, and the reviewers for their useful feedback and comments.

References

[1] “Communication delivery time performance requirements for electric power substation automation,” *IEEE Std 1646-2004*, pp. 1–24, 2005.

[2] T. Sauter, “The three generations of field-level networks – evolution and compatibility issues,” *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, pp. 3585–3595, 2010.

[3] J.-D. Decotignie, “Ethernet-based real-time and industrial communications,” *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1102–1117, 2005.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: enabling innovation in campus networks,” in *SIGCOMM Computer Communication Review*, vol. 38, no. 2. ACM, 2008, pp. 69–74.

[5] D. Henneke, L. Wisniewski, and J. Jasperneite, “Analysis of realizing a future industrial network by means of software-defined networking (SDN),” in *IEEE World Conference on Factory Communication Systems (WFCS)*. IEEE, 2016, pp. 1–4.

[6] Q. Duan, “Network-as-a-service in software-defined networks for end-to-end QoS provisioning,” in *23rd Wireless and Optical Communication Conference (WOCC)*. IEEE, 2014, pp. 1–5.

[7] S. Sharma, D. Staessens, D. Colle, D. Palma, J. Goncalves, R. Figueiredo, D. Morris, M. Pickavet, and P. Demeester, “Implementing quality of service for the software defined networking enabled future Internet,” in *Third European Workshop on Software Defined Networks*. IEEE, 2014, pp. 49–54.

[8] S. Tomovic, N. Prasad, and I. Radusinovic, “SDN control framework for QoS provisioning,” in *22nd Telecommunications Forum Telfor (TELFOR)*. IEEE, 2014, pp. 111–114.

[9] M. Shen, L. Zhu, M. Wei, Q. Zhang, M. Wang, and F. Li, “Joint optimization of flow latency in routing and scheduling for software defined networks,” in *25th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2016, pp. 1–8.

[10] W. Kim, P. Sharma, J. Lee, S. Banerjee, J. Tourrilhes, S.-J. Lee, and P. Yalagandula, “Automated and scalable QoS control for network convergence.” in *Internet Network Management Workshop/ Workshop on Research on Enterprise Networking (INM/WREN)*, vol. 10, no. 1, pp. 1–1, 2010.

[11] H. E. Egilmez, S. T. Dane, K. T. Bagci, and A. M. Tekalp, “OpenQoS: An openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks,” in *Asia-Pacific Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2012, pp. 1–8.

[12] M. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba *et al.*, “Policycop: An autonomic QoS policy enforcement framework for software defined networks,” in *SDN for Future Networks and Services (SDN4FNS)*. IEEE, 2013, pp. 1–7.

[13] A. V. Akella and K. Xiong, “Quality of service (QoS)-guaranteed network resource allocation via software defined networking (SDN),” in *12th International Conference on Dependable, Autonomic and Secure Computing (DASC)*. IEEE, 2014, pp. 7–13.

[14] D. Adami, L. Donatini, S. Giordano, and M. Pagano, “A network control application enabling software-defined quality of service,” in *IEEE International Conference on Communications (ICC)*, 2015, pp. 6074–6079.

[15] N. An, T. Ha, K.-J. Park, and H. Lim, “Dynamic priority-adjustment for real-time flows in software-defined networks,” in *17th International Telecommunications Network Strategy and Planning Symposium (Networks)*. IEEE, 2016, pp. 144–149.

[16] J. W. Guck and W. Kellerer, “Achieving end-to-end real-time quality of service with software defined networking,” in *3rd International Conference on Cloud Networking (CloudNet)*. IEEE, 2014, pp. 70–76.

[17] J. W. Guck, M. Reisslein, and W. Kellerer, “Model-based control plane for fast routing in industrial QoS network,” in *23rd International Symposium on Quality of Service (IWQoS)*. IEEE, 2015, pp. 65–66.

[18] ——, “Function split between delay-constrained routing and resource allocation for centrally managed QoS in industrial networks,” *IEEE Transactions on Industrial Informatics*, vol. 12, no. 6, pp. 2050–2061, Dec 2016.

[19] J. Jasperneite, P. Neumann, M. Theis, and K. Watson, “Deterministic real-time communication with switched Ethernet,” in *4th International Workshop on Factory Communication Systems*. IEEE, 2002, pp. 11–18.

[20] O. S. Consortium *et al.*, “Openflow switch specification version 1.0.0,” 2009.

[21] P. Gaj, J. Jasperneite, and M. Felser, “Computer communication within industrial distributed environment - A survey,” *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 182–189, 2013.

[22] J. Jasperneite and P. Neumann, “How to guarantee realtime behavior using Ethernet,” in *Information Control Problems in Manufacturing (INCOM): A Proceedings Volume from the 11th IFAC Symposium, Salvador, Brazil, 5-7 April 2004*, vol. 1. Gulf Professional Publishing.

[23] A. Kassler, L. Skorin-Kapov, O. Dobrijevic, M. Matijasevic, and P. Dely, “Towards QoE-driven multimedia service negotiation and path optimization with software defined networking,” in *20th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2012, pp. 1–5.

[24] P. Sharma, S. Banerjee, S. Tandel, R. Aguiar, R. Amorim, and D. Pinheiro, “Enhancing network management frameworks with SDN-like control,” in *International Symposium on Integrated Network Management (IM)*. IFIP/IEEE, 2013, pp. 688–691.

[25] I. Owens, A. Durresi *et al.*, “Video over software-defined networking (VSDN),” in *16th International Conference on Network-Based Information Systems (NBiS)*. IEEE, 2013, pp. 44–51.

[26] S. Gorlatch, T. Humernbrum, and F. Glinka, “Improving QoS in real-time internet applications: from best-effort to software-defined networks,” in *International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2014, pp. 189–193.

[27] A. Ishimori, F. Farias, E. Cerqueira, and A. Abelém, “Control of multiple packet schedulers for improving QoS on OpenFlow/SDN networking,” in *Second European Workshop on Software Defined Networks*. IEEE, 2013, pp. 81–86.

[28] E. Schweissguth, P. Danielis, C. Niemann, and D. Timmermann, “Application-aware industrial ethernet based on an SDN-supported TDMA approach,” in *World Conference on Factory Communication Systems (WFCS)*. IEEE, 2016, pp. 1–8.

[29] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, “Fastpass: A centralized zero-queue datacenter network,” in *SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 307–318.

[30] A. L. King, S. Chen, and I. Lee, “The middleware assurance substrate: Enabling strong real-time guarantees in open systems with openflow,” in *17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*. IEEE, 2014, pp. 133–140.

[31] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer, April 2012.

[32] A. Van Bemten and W. Kellerer, “Network calculus: A comprehensive guide,” *Technical University of Munich, Chair of Communication Networks, Technical Report No. 201603*, October 2016.

[33] J. Schmitt, P. Hurley, M. Hollick, and R. Steinmetz, “Per-flow guarantees under class-based priority queueing,” in *Global Telecommunications Conference*, vol. 7. IEEE, 2003, pp. 4169–4174.

[34] J. Åkerberg, M. Gidlund, and M. Björkman, “Future research challenges in wireless sensor and actuator networks targeting industrial automation,” in *9th International Conference on Industrial Informatics*. IEEE, 2011, pp. 410–415.

[35] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, “Smart grid technologies: communication technologies and standards,” *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 529–539, 2011.

[36] R. H. Khan and J. Y. Khan, “A comprehensive review of the application characteristics and traffic requirements of a smart grid communications network,” *Computer Networks*, vol. 57, no. 3, pp. 825–845, 2013.

[37] T. Mahmoodi, V. Kulkarni, W. Kellerer, P. Mangan, F. Zeiger, S. Spirou, I. Askoxylakis, X. Vilajosana, H. J. Einsiedler, and J. Quittek, “Virtuwind: virtual and programmable industrial network prototype deployed in operational wind park,” *Transactions on Emerging Telecommunications Technologies*, vol. 27, no. 9, pp. 1281–1288, 2016.

[38] R. Widyono *et al.*, *The design and evaluation of routing algorithms for real-time channels*. International Computer Science Institute Berkeley, 1994.

[39] A. Jüttner, B. Szviatovski, I. Mécs, and Z. Rajkó, “Lagrange relaxation based method for the QoS routing problem,” in *20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 2. IEEE, 2001, pp. 859–868.

[40] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer, “Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation,” *IEEE Communications Surveys Tutorials*, vol. PP, no. 99, pp. 1–1, 2017.

Jochen W. Guck received the Dipl.-Ing. degree in Ingenieurinformatik from the University of Applied Sciences Wuerzburg-Schweinfurt, Schweinfurt, Germany, in 2009, and the M.Sc. degree in electrical engineering from the Technical University of Munich, Munich, Germany, in 2011. In September 2012, he joined the Chair of Communication Networks at the Technical University of Munich as a member of the research and teaching staff. His research interests include real-time communication, industrial communication, software-defined networking, and routing algorithms.

Amaury Van Bemten was born in Liège, Belgium, in 1993. He received the B.Sc. degree in Engineering in June 2013 and the M.Sc. in Computer Science and Engineering in June 2015, both from the University of Liège (Belgium). In September 2015 he joined the Chair of Communication Networks at the Technical University of Munich (TUM), where he is currently pursuing the Ph.D. degree as a member of the research and teaching staff. His current research focuses on routing algorithms and the application of software-defined networking for resilient real-time communications in industrial environments.

Wolfgang Kellerer (M’96-SM’11) is a Full Professor with the Technical University of Munich, heading the Chair of Communication Networks with the Department of Electrical and Computer Engineering. Before, he was for over ten years with NTT DOCOMO’s European Research Laboratories. He received his Dr.-Ing. degree (Ph.D.) and his Dipl.-Ing. degree from Munich University of Technology, Munich, Germany, in 1995 and 2002, respectively. His research resulted in over 200 publications and 29 granted patents in the areas of mobile networking and service platforms. He currently serves as an associate editor for *IEEE Transactions on Network and Service Management* and on the Editorial Board of the *IEEE Communications Surveys and Tutorials*. He is a member of ACM and the VDE ITG.

DetServ: Network Models for Real-Time QoS Provisioning in SDN-based Industrial Environments

Supplementary Material: Examples of the Models Operations

Jochen W. Guck, Amaury Van Bemten, Wolfgang Kellerer

S.I. Introduction

This supplementary material provides two examples of the operations of the two DetServ network models proposed in the main article. Note that references to figures, sections or equations of the main article use the latter's numbering. Notations also correspond to those introduced in the main article (see Sec. III and V-A). Numbering for this supplementary material is reinitialized and preceded by *S*.

For both models, we consider a given physical link $(u, v) \in \mathcal{P}_E$ of capacity $R_{u,v} = 1$ Gb/s and with three priority queues scheduled by a non-preemptive strict priority scheduler. We assume $L^{max} = 1530$ B. We do not consider *input link shaping* (ILS) in these examples.

S.II. Multi-Hop Model Example

Before developing the example, we introduce two formulas which will be helpful. With similar developments as for obtaining Eqn. 23, Eqn. 10 and 11 yield the following upper bounds.

$$T_{u,v,p} \leq \frac{\sum_{j=1}^{p-1} \mathbf{M}_B[u, v, j] + 2L^{max}}{R_{u,v} - \sum_{j=1}^{p-1} \mathbf{A}_R[u, v, j]} \tag{S.1}$$

$$R_{u,v,p} \leq R_{u,v} - \sum_{j=1}^{p-1} \mathbf{A}_R[u, v, j] \tag{S.2}$$

The example for the *multi-hop model* (MHM) operation is illustrated in Fig. S.1. Let us assume that the resource allocation algorithm assigned half of the capacity to the high priority queue, a quarter of the capacity to the middle priority queue and an eighth of the capacity to the lowest priority queue, i.e., $\mathbf{A}_R[u, v, 1] = 500$ Mb/s, $\mathbf{A}_R[u, v, 2] = 250$ Mb/s and $\mathbf{A}_R[u, v, 3] = 125$ Mb/s. We further assume that the buffer size at each queue is 300 KB, i.e., $\mathbf{A}_B[u, v, p] = 300000$ B $\forall p \in \{1, 2, 3\}$, and that the resource allocation algorithm did not artificially reduce it.

For the highest priority queue (Fig. S.1a), the worst-case service curve parameters $T_{u,v,1}$ and $R_{u,v,1}$ are directly given by Eqn. S.1 and S.2. We have $T_{u,v,1} = 0.02448$ ms and $R_{u,v,1} = 1$ Gb/s. Based on this, $\mathbf{M}_B[u, v, 1]$ corresponds to the maximum burst that can be allowed to enter the queue such that the backlog bound stays smaller than the buffer capacity $\mathbf{A}_B[u, v, 1]$ at the queue, provided that the slope of the arrival curve is fixed by the resource allocation algorithm to $\mathbf{A}_R[u, v, 1]$. This corresponds to pushing the straight line with slope $\mathbf{A}_R[u, v, 1]$ up until its maximum distance to the service curve reaches $\mathbf{A}_B[u, v, 1]$. We have $\mathbf{M}_B[u, v, 1] = 298470$ B (computed with Eqn. 22). From the obtained maximum arrival curve $\mathbf{M}_\alpha[u, v, 1]$, the delay of the queue can be computed as the horizontal deviation between $\mathbf{M}_\alpha[u, v, 1]$ and $\beta_{u,v,1}$, i.e., with Eqn. 23. We have $\mathbf{T}[u, v, 1] = 2.41224$ ms.

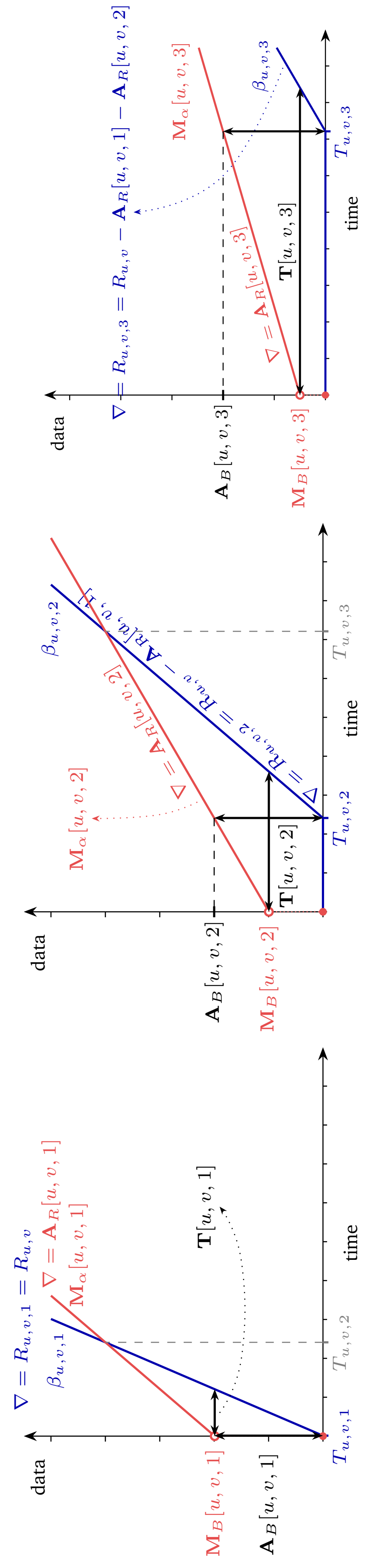
For the middle priority queue (Fig. S.1b), the service curve parameters $T_{u,v,2}$ and $R_{u,v,2}$ can now be computed with Eqn. S.1 and S.2, since $\mathbf{M}_B[u, v, 1]$ is now known. Note that, graphically, $T_{u,v,2}$ corresponds to the abscissa at which $\mathbf{M}_\alpha[u, v, 1]$ and $\beta_{u,v,1}$ intersect[1]. This can be intuitively understood. Indeed, the middle priority queue has to wait for the high priority queue to empty its backlog before being served. We will, hereafter, refer to this value as the *finishing time* of the queue. The computations yield $T_{u,v,2} = 4.82448$ ms and $R_{u,v,2} = 500$ Mb/s. $\mathbf{M}_B[u, v, 2]$ and $\mathbf{T}[u, v, 2]$ can then be computed as for the high priority queue. We have $\mathbf{M}_B[u, v, 2] = 149235$ B and $\mathbf{T}[u, v, 2] = 7.21224$ ms.

The process is then similar for the low priority queue, for which we obtain $T_{u,v,3} = 14.42448$ ms, $R_{u,v,3} = 250$ Mb/s, $\mathbf{M}_B[u, v, 3] = 74617.5$ B and $\mathbf{T}_{u,v,3} = 16.81224$ ms.

All the required values are now available for implementing the four model functions in Fig. 3. Basically, flows will be accepted at a queue p of the link as long as the resulting aggregate arrival curve traversing the queue stays below the $\mathbf{M}_\alpha[u, v, p]$ limit curve. This is illustrated in Fig. S.2. Let us assume that the current burst and rate utilization for the middle priority queue are $\mathbf{U}_B[u, v, 2] = 45000$ B and $\mathbf{U}_R[u, v, 2] = 106.115$ Mb/s. Let us consider that the routing algorithm would like to add a flow f_1 with rate and burst given by $r_{f_1} = 100$ Mb/s and $b_{f_1}[u, v, 2] = 150000$ B to this queue. In this case, access to the queue is refused because $\mathbf{U}_B[u, v, 2] + b_{f_1}[u, v, 2] = 195000 \text{ B} > \mathbf{M}_B[u, v, 2]$ (dashed line in Fig. S.2). If the routing algorithm then requests access to the queue for a flow f_2 with rate and burst given by $r_{f_2} = 200$ Mb/s and $b_{f_2}[u, v, 2] = 20000$ B, access to the queue will also be refused because $\mathbf{U}_R[u, v, 2] + r_{f_2} = 306.115$ Mb/s $> \mathbf{A}_R[u, v, 2]$ (dotted line in Fig. S.2). If the routing algorithm then requests access to the queue for a flow f_3 with rate and burst given by $r_{f_3} = 130$ Mb/s and $b_{f_3}[u, v, 2] = 15000$ B, the access to the queue will this time be granted because $\mathbf{U}_B[u, v, 2] + b_{f_3}[u, v, 2] = 60000 \text{ B} < \mathbf{M}_B[u, v, 2]$ and

[1] Strictly speaking, this is only the case if we neglect the l_i^{max} term in Eqn. 1, i.e., if the store-and-forward behavior of switches is neglected. Nevertheless, this term only *slightly* shifts the $T_{u,v,p}$ values, so we can consider, for understanding purposes, that the statement is true. Note that, in all the upcoming figures, the real values are shown. It can be seen that $T_{u,v,p}$ is always *very* close to the intersection of the two curves.

J. Guck, A. Van Bemten, and W. Kellerer are with the Lehrstuhl für Kommunikationsnetze, Technical University of Munich, Munich, 80290, Germany (email: {guck, amaury.van-bemten, wolfgang.kellerer}@tum.de).



(a) High priority queue.

(b) Middle priority queue.

(c) Low priority queue.

Fig. S.1: Example of service and maximum arrival curves for a non-preemptive strict priority scheduler with three queues using the multi-hop model. The process of computing the maximum allowed arrival curves is iterative, starting from the high priority queue towards the lower priority queues. The service curve parameters $T_{u,v,1}$ and $R_{u,v,1}$ are directly given by Eqn. S.1 and S.2. Then, based on the rate $\mathbf{A}_R[u,v,1]$ allocated to the high priority queue, the maximum allowed burst $\mathbf{M}_B[u,v,1]$ is the maximum burst such that the backlog at that queue stays smaller than the buffer capacity $\mathbf{A}_B[u,v,1]$ at the queue. This corresponds to pushing the straight line with slope $\mathbf{A}_R[u,v,1]$ up until its maximum distance to the service curve reaches $\mathbf{A}_B[u,v,1]$. From the obtained maximum arrival curve $\mathbf{M}_\alpha[u,v,1]$, the delay of the queue can be computed as the horizontal deviation between $\mathbf{M}_\alpha[u,v,1]$ and $\beta_{u,v,1}$. Now that $\mathbf{M}_B[u,v,1]$ is known, the same procedure can be applied to the middle priority queue to obtain the service curve parameters, the maximum allowed burst and the worst-case delay of the queue. The procedure then continues once more for the lowest priority queue. Note that the service curve parameter $T_{u,v,p}$ for a given queue p (approximately) corresponds to the time at which the service curve and the maximum arrival curve intersect for the priority queue $p-1$.

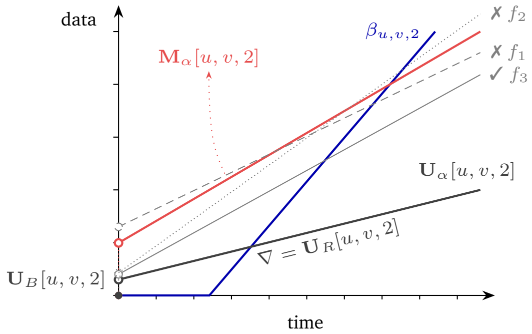


Fig. S.2: Operation of the HASACCESS method of the multi-hop model for the middle priority queue of Fig. S.1 (Fig. S.1b). The routing algorithm tries to embed three flows in the queue. The first one, f_1, is rejected because it would cause the maximum allowed burst to be exceeded (dashed line). The second one, f_2, is also rejected because it would cause the allocated rate to be exceeded (dotted line). The third one, f_3, is accepted because none of the two limits are exceeded when adding the flow to the queue.

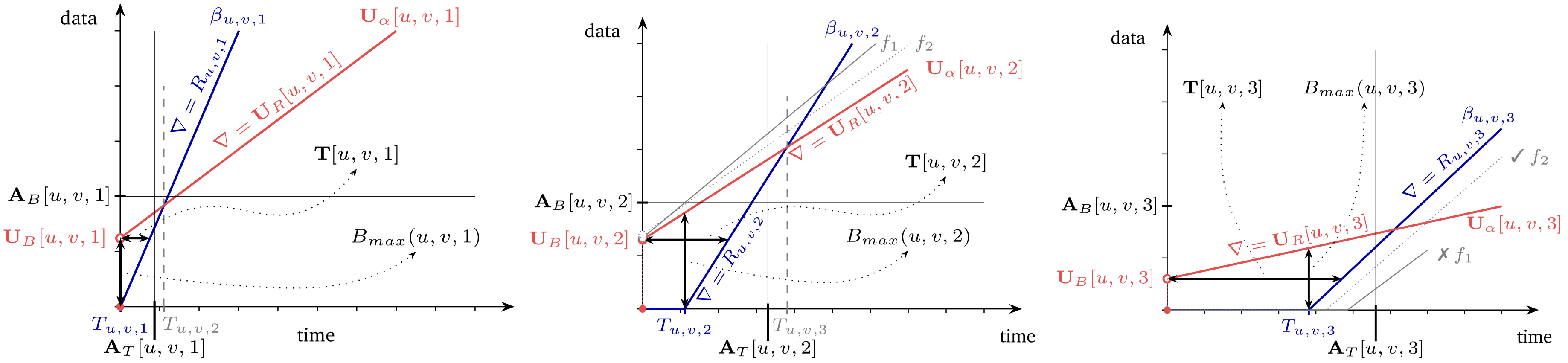
$\mathbf{U}_R[u,v,2] + r_{f_3} = 236.115$ Mb/s $< \mathbf{A}_R[u,v,2]$ (thin full line in Fig. S.2).

S.III. Threshold-Based Model Example

The example for the *threshold-based model* (TBM) operation is illustrated in Fig. S.3. Let us assume that the resource allocation algorithm assigned $\mathbf{A}_T[u,v,1] = 1.74$ ms, $\mathbf{A}_T[u,v,2] = 6.6$ ms and $\mathbf{A}_T[u,v,3] = 11.22$ ms as limit worst-case delays for the three queues and that these are in the following state. The high priority queue is traversed by an aggregate flow with parameters $\mathbf{U}_B[u,v,1] = 186$ KB, $\mathbf{U}_R[u,v,1] = 322$ Mb/s and $l^{max}_{u,v,1} = 700$ B. The middle and low priority queues are traversed by aggregate flows with parameters $\mathbf{U}_B[u,v,2] = 195$ KB, $\mathbf{U}_R[u,v,2] = 275$ Mb/s, $l^{max}_{u,v,2} = 400$ B and $\mathbf{U}_B[u,v,3] = 90$ KB, $\mathbf{U}_R[u,v,3] = 93$ Mb/s, $l^{max}_{u,v,3} = 1200$ B, respectively. The corresponding service and arrival curves for the three different priority queues are shown in Fig. S.3. The current buffer and delay usage $B_{max}(u,v,p)$ and $\mathbf{T}[u,v,p]$ are shown along with their limits $\mathbf{A}_B[u,v,p]$ and $\mathbf{A}_T[u,v,p]$.

Let us consider that the routing algorithm then requests access to the middle priority queue for a flow f_1 with parameters $b_f[u,v,2] = 5500$ B and $r_f = 82$ Mb/s. We assume the maximum packet size of the flow is smaller than the current maximum packet size of the aggregate, thereby leaving $l^{max}_{u,v,2}$ unchanged. The high priority queue is not concerned by this request. The updated arrival curve for the middle priority queue is shown in Fig. S.3b (thin full line). The delay and backlog thresholds of this queue are not exceeded. From the point of view of the middle priority queue, the flow can hence be embedded. Nevertheless, the low priority queue state also has to be checked. The updated service curve offered by the low priority queue is shown in Fig. S.3c (thin full line). Unfortunately, we can see that the worst-case delay limit $\mathbf{A}_T[u,v,3]$ would now be exceeded. As a result, f_1 has to be rejected from the middle priority queue because it would violate the delay threshold of the low priority queue.

The routing algorithm then requests access to the middle priority queue for a flow f_2 with parameters $b_f[u,v,2] = 15$ KB and $r_f = 30$ Mb/s. We once more assume that the maximum packet size of the flow is smaller than the current maximum



(a) High priority queue.

(b) Middle priority queue.

(c) Low priority queue.

Fig. S.3: Example of service and arrival curves for a non-preemptive strict priority scheduler with three queues using the threshold-based model. All the queues are constrained by two parameters. The first one, $\mathbf{A}_B[u,v,p]$, corresponds to the buffer space at the queue. The second one, $\mathbf{A}_T[u,v,p]$, assigned by the resource allocation algorithm, corresponds to the maximum worst-case delay of the queue. The queue has to refuse any traffic that makes the worst-case backlog and delay at this queue grow bigger than these two bounds. If bounds are still respected, bounds of lower priorities queues also have to be checked. A flow can then be embedded only if bounds of all lower priority queues are still satisfied. Fig. S.3b and S.3c show the updated arrival and services curves if a flow f_1 is added to the middle priority queue. From the point of view of the middle priority queue, none of its two bounds would be violated and the flow can be added. Nevertheless, f_1 cannot be accepted because the updated service curve of the lower priority queue would lead to the violation of its worst-case delay threshold. In dashed lines, Fig. S.3b and S.3c also show the updated arrival and services curves if another flow f_2 is added to the middle priority queue. In this case, none of the bounds in both queues will be violated and the flow can hence be accepted.

packet size of the aggregate. The high priority queue is still not concerned by the request. The updated arrival curve for the middle priority queue is shown in Fig. S.3b (thin dotted line). As for f_1, we can see that the delay and backlog thresholds of this queue are not exceeded. Before allowing the embedding of the flow to this queue, the updated state of the low priority queue also has to be checked. The updated service curve offered by the low priority queue is shown in Fig. S.3c (thin dotted line). We can see that the the worst-case delay limit $\mathbf{A}_T[u,v,3]$ would, in this case, not be exceeded. As a result, since all the worst-case limits are still respected, f_2 can be embedded in the middle priority queue.