Function Split Between Delay-Constrained Routing and Resource Allocation for Centrally Managed QoS in Industrial Networks

Jochen W. Guck, Martin Reisslein, *Fellow, IEEE*, and Wolfgang Kellerer, *Senior Member, IEEE*

***Abstract*—Industrial networks demand centrally controlled quality of service (QoS), often in the form of hard real-time guarantees. Software-defined networking (SDN) provides a convenient paradigm for central QoS control. However, existing SDN-based solutions cannot guarantee hard real-time QoS as they rely on a control loop over the forwarding (data) and control planes. We propose a novel SDN-based QoS control framework that maintains an accurate network model through network calculus to avoid a control loop over forwarding and control planes, allocates resources to and routes flows over a network of “queue links,” whereby each physical network link houses multiple queue links (with different QoS levels), and manages QoS through a function split between delay-constrained least-cost routing on the network of queue links and the resource allocation to the queue links. This function split greatly reduces the computational complexity while achieving hard real-time QoS with high bandwidth utilization. Our evaluation results indicate that our function split approach allows for online runtime admission control and can achieve bandwidth utilization above 80% while meeting deterministic real-time QoS requirements.**

***Index Terms*—Bandwidth utilization, industrial network, network calculus, real-time quality of service (QoS), software-defined networking (SDN).**

I. Introduction

A. Motivation: Industrial Networking Quality of Service

INDUSTRIAL networks carry critical messages, e.g., control signals, for large automated production facilities. Many of these critical messages must be delivered with tight deterministic real-time quality of service (QoS) [1]. A wide gamut of proprietary industrial communications technologies have emerged to provide these strict QoS [2]. These proprietary technologies are typically costly and lack a uniformly accepted standardized communication framework.

B. Basis: Software-Defined Networking

The emergence of software-defined networking (SDN) with a standardized OpenFlow protocol [3] provides a basis (general framework) for introducing a uniform QoS networking approach in the industrial networking domain. As elaborated in Section II, several SDN-based QoS networking approaches have recently been examined. However, the approaches developed to date are not suitable for industrial QoS networking due to slow or inaccurate QoS control or prohibitive computational effort for the QoS control.

In this paper, we introduce a novel SDN-based QoS networking approach that is fast and accurate for industrial QoS networking. Specifically, the proposed QoS networking approach is accurate by deterministically modeling resource (bandwidth, buffer) allocation and QoS parameter (delay budget) allocation through network calculus. The proposed approach is fast by avoiding a QoS control loop over the forwarding and control planes. Rather, we close the QoS control loop only over the model, load map, and routing blocks in the SDN control plane, as illustrated in the left part of Fig. 1.

C. Contribution: Function Split Between Routing and Resource Allocation

The core contribution of this paper is a novel function split between routing and resource allocation for achieving hard real-time industrial QoS with SDN networking. In order to enable this function split, we introduce a novel “queue link” network topology (see top-left of Fig. 1) that represents both the underlying physical link topology as well as the resource and QoS parameter allocations. In particular, the resources (transmission bit rate, buffer space) of a given link and QoS parameter budgets (e.g., delay budgets) are proactively allocated to a set of QoS queues that correspond to different levels of forwarding QoS at the link. A standard reactive QoS routing algorithm, such as delay-constrained least-cost (DCLC) routing [4], can then find routes through the queue link network that meet the hard real-time QoS. The route selection, thus, determines both the path that an admitted flow takes through the physical links as well as the QoS queues that the flow traverses in the individual physical links.

We compare this novel function split SDN-based QoS approach through simulations with a monolithic mixed-integer

Manuscript received June 22, 2015; revised May 10, 2016; accepted July 2, 2016. Date of publication July 20, 2016; date of current version December 6, 2016. This work was supported, in part, by the European Research Council under the European Unions Horizon 2020 research and innovation program (under Grant 647158 FlexNets) and by the A. von Humboldt Foundation through an F. W. Bessel Research Award. Paper no. TII-15-1721.R1. *(Corresponding author: M. Reisslein.)*

J. W. Guck and W. Kellerer are with the Department of Electrical and Computer Engineering, Technical University of Munich, Munich 80290, Germany (e-mail: guck@tum.de; wolfgang.kellerer@tum.de).

M. Reisslein is with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287-5706 USA (e-mail: reisslein@asu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2016.2592481

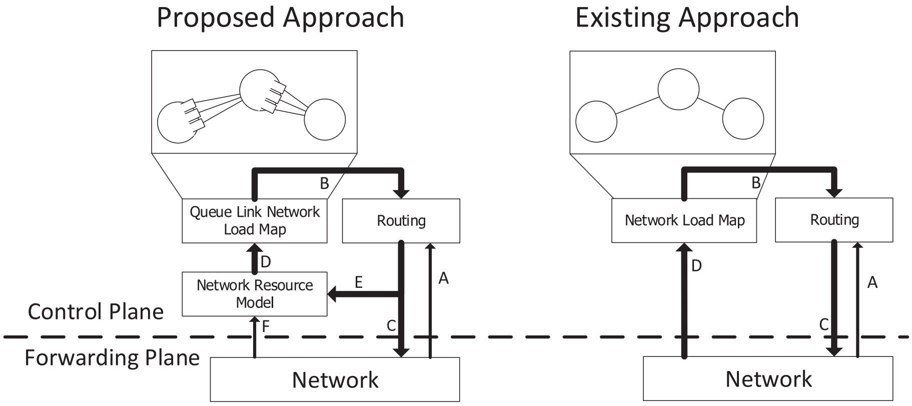

Fig. 1. Conceptual comparison of existing real-time Ethernet architectures (right) with our novel approach (left). We maintain an accurate network resource model in the SDN control plane, thus avoiding the signaling from the forwarding (data) plane to the control plane in existing architectures. While existing architectures route on a network load map of physical links, we route on a network load map of queue links; whereby multiple queue links for each physical link represent the different QoS levels of transmission over a given physical link. Call admission control involves the following steps and information exchanges: A: Receive connection request, including QoS requirements; B: Find best path based on latest load map; C: deploy path to network; D: update load map. Our proposed approach revises step D in the existing approaches (right side) by introducing the network resource model. The network resource model tracks the network state D by combining static network information F, such as physical topology, link rates, and buffer capacities, with the dynamics of the connection deployments E. The combined overall network resource model provides then the up-to-date queue link network load map, which is utilized for the online admission control and routing.

programming (MIP)-based approach that jointly solves the routing and resource/QoS parameter allocation problems. We find that the novel function split approach with online admission control achieves network connection carrying capacities and bandwidth utilizations that come relatively close to the computationally prohibitive monolithic MIP approach.

II. Related Work

A. Industrial QoS Networking

Decotingie [5] provided a broad overview of Ethernet-based real-time communication, including deterministic Ethernet standards. A common example of an Ethernet-based real-time communication system is Avionics Full DupleX Switched Ethernet (AFDX), which can provide deterministic QoS [6]. However, AFDX requires offline admission control, whereas our proposed approach allows for fast online admission control decisions at runtime. Moreover, AFDX typically utilizes a single queue for each physical link, whereas our approach features multiple priority levels (queue links) for each physical link, allowing flexible QoS management. The opportunity of cost reduction by using off-the-shelf commodity hardware for real-time communication instead of specialized proprietary communication systems was one trigger for the shift to Ethernet-based technology[2], [7]. However, the commodity hardware does not fulfill all requirements of real-time communication, which has led to the development of proprietary Ethernet-based solutions [8]. The proprietary industrial real-time communication solutions that fulfill hard real-time requirements require changes of the network protocol stack or topology restrictions or both, leading to expensive hardware and software.

B. QoS Networking Frameworks

Before the emergence of SDN, several middleware layer and interface architectures, e.g., [9], have been proposed for adaptive QoS control that exploits real-time scheduling approaches [10]. These architectures involve a control loop that is closed over the forwarding and control planes, as illustrated on the right side of Fig. 1. Closed-loop control over both the forwarding and control planes tends to introduce delays and measurement errors that are difficult to correct to achieve fast accurate QoS control.

SDN [3] has recently emerged as a flexible control plane paradigm, and several studies have begun to explore the implications of SDN for QoS networking. For instance, Jain *et al.* [11] report on an SDN-based traffic engineering system for efficient bandwidth usage in a backbone network, while Sharma *et al.* [12] propose SDN-based network QoS control frameworks. Policy Cop [13] is a closed-control loop system (over forwarding and control planes) for providing aggregated and per-flow QoS guarantees. Policy Cop determines the QoS level of each forwarding device through statistical analysis of the forwarding characteristics measured in the forwarding plane and is thus not designed to provide deterministic QoS. Moreover, the Policy Cop measurements in the forwarding plane introduce delays and measurements errors, which lead to slow and possibly erroneous QoS control actions. Sharma *et al.* [14] focus on the management of minimum and maximum bandwidth allocations for different QoS levels and do not consider delay requirements. Modifications to the OpenFlow protocol for representing QoS mechanisms have been formulated by Ishimori *et al.* [15].

C. Network Calculus

Network calculus [16] provides a mathematical framework for calculating deterministic upper delay bounds of a network. This mathematical model has been used for the dimensioning and offline planning of real-time systems [8]. In contrast, we employ network calculus for online modeling of the network at runtime. Duan [17] developed an SDN-based network as a service framework and derived network calculus to evaluate data rates needed for fulfilling delay requirements in such a general framework. Similarly, Guck and Kellerer [18] have conducted network calculus analysis to evaluate end-to-end delay guarantees in SDN environments. The network calculus results served as a basis for an offline MIP that evaluates admission control (resource availability) decisions and routes flows, albeit for prohibitive computational cost. In contrast, in this paper, we develop a computationally efficient online function split framework for runtime admission decisions and routing while maintaining the network-calculus-based real-time guarantees.

III. Setting: Industrial Real-Time Communication With SDN

A. Industrial Real-Time QoS Requirements

The industrial communication network has to fulfill hard real-time requirements [2]. This means that a data transmission has to be completed within a given time; otherwise, the connection is deemed as broken. There is also the soft real-time requirement class, which defines the deadline as a soft border. The connection counts only as broken if the deadline is violated several times. We focus on hard real-time requirements.

B. SDN-Based QoS Approach

SDN provides fine-granular access to the forwarding (data) plane. This fine-granular forwarding plane control can serve as the basis for the operation of real-time QoS communication mechanisms, including admission control, scheduling, and resource allocation mechanisms. Specifically, for achieving industrial real-time communication based on SDN, it is first necessary to define a model for all forwarding elements in the network. This network model creates an abstraction layer between SDN hardware and SDN applications. This abstraction should allow for the development of QoS-aware SDN applications independent from the network hardware.

The SDN hardware capabilities can include the queue buffer size, the number of queues at the links, the employed scheduling algorithm, the given data transmission rates, and the topology. On the other hand, an application may simply express its demands in terms of tolerable mean delay, maximum delay, jitter, and loss rate, as well as mean data rate and burstiness of the traffic flow. The abstraction layer should use these application parameters and the knowledge of the network to decide how to best fulfill the demands. This abstraction can, in principle, be done with any imaginable network model, such as statistical, deterministic, measurement-based, or calculation based-models, or combinations thereof.

C. Deterministic Network Calculus Model of Static Priority Scheduling

We employ deterministic network calculus [16] to bound the maximum (worst-case) delay per hop. The deterministic network calculus model for the worst-case delay allows for relatively easy admission control. In particular, due to the allocation of worst-case delay budgets to the priority queues (as detailed in Section VI-B), the models of the different priority queues at a given link (hop) are independent. Thus, admission decisions of low-priority queues do not have to be recalculated every time a flow is added to a high-priority queue.

We employ nonpreemptive static priority scheduling, which transmits a packet from a given priority class (queue) only if all higher priority queues are empty, but does not interrupt an ongoing packet transmission. Static priority scheduling is simple as it does not not require any parameter configuration beyond the number of priority queues, yet supports a broad range of delay bounds. We assume throughout this paper that the (computational) processing delay of each forwarding element is negligible.

IV. Function Split QoS Framework

Industrial QoS networking with a centralized SDN controller poses the general problems of path computation (routing) and resource allocation. The joint solution of the combined routing and resource allocation problem is NP-hard. This can be derived from the observation that the path computation problem corresponds to the well-known DCLC routing problem [4], which is NP-hard. Thus, the combined problem, which adds the resource allocation problem to the DCLC problem, is also NP-hard. The combined routing and resource allocation problem can be formulated (with simplifications of some quadratic equations [18]) as a MIP. The MIP solves both the routing and resource allocation problems jointly but incurs prohibitive computation times already for small networks (see comparisons in Section IX). The monolithic MIP solution approach cannot be employed for runtime decisions in realistic-sized industrial networks.

In order to reduce the computational effort for solving the general routing and resource allocation problem so as to operate SDN-based industrial QoS networks at runtime, we split the problem into two functions: an online function for DCLC routing [19], [20] and an offline function for resource allocation. We operate the online routing (as explained in Section VI) within the closed loop in the SDN control plane illustrated in the left part of Fig. 1. We model the state of the network through a queue link topology model introduced in Section V in conjunction with the network resource model introduced in Section VII. The queue link topology model represents the different priority queues that operate on each physical network link. We allocate each priority queue a fixed buffer size since buffer sizes are typically fixed in practical switches. We also allocate to each priority queue a fixed delay budget and consider the transmission bit rate as the main "resource" to be allocated to the individual priority queues at a link. Based on historical statistics or predictions and a resource allocation process running in the background (see Section VIII), we allocate the transmission bit rates offline to the individual priority queues (queue links).

V. Queue Link Network Topology

We consider origin (source) nodes connected by switching nodes and directed links to destination nodes. We denote $\mathcal{N}$ for the set of all nodes in the network. In our real-time QoS framework, each switching node may be viewed as a collection of queues that are allocated specific transmission bit rate and buffer space resources. We define the network formed by the

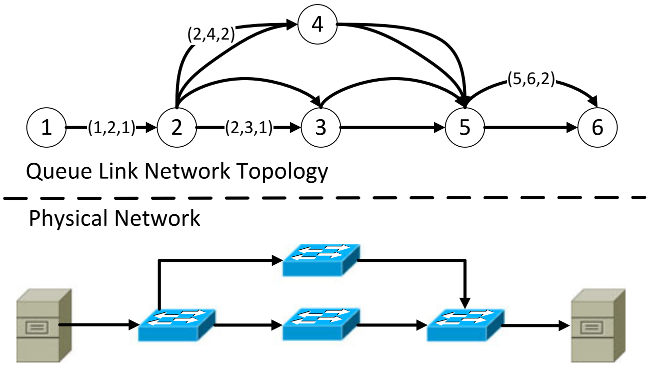

Fig. 2. Illustrative example of queue link network model. One origin node, four switching nodes, and one destination node are interconnected by physical links in the feedforward (left to right) direction. The nodes are indexed by integers 1 through 6, i.e., the set of network nodes is $\mathcal{N} = \{1, 2, \ldots, 6\}$. Each physical link emanating from a switching node contains $Q_{u,v} = 2$ priority queue links while source nodes have one queue. A queue link (edge) (u, v, q) connects priority queue q in node u to node v. Thus, the set of edges is $\mathcal{E} = \{(1, 2, 1), (2, 3, 1), (2, 3, 2), (2, 4, 1), (2, 4, 2), \ldots, (5, 6, 2)\}$.

queues in the switching nodes and the links interconnecting the switching nodes as a queue link network. A preliminary outline of the queue link network was presented in [21]. We denote $Q_{u,v}$ for the number of priority queues in a given switching node u for the physical link from switching node u to node v; origin nodes are considered to have one priority queue. (We note that the lowest priority queue can be defined as best effort queue.) We define the directed queue link (u, v, q) to connect queue q in node u via the physical link between nodes u and v to node v. The queue link concept is illustrated for the example of a simple feedforward network with $Q_{u,v} = 2$ queues in each physical link in Fig. 2. We define the set $\mathcal{E} = \{(u, v, q)\},\ u, v \in \mathcal{N},\ q = 1, \ldots, Q_{u,v}$, to denote the set of all queue links (edges) in a given queue link network, as summarized in Table I.

TABLE I
SUMMARY OF MAIN NOTATIONS

Network model	
$\mathcal{N}$	Set of nodes in network
$Q_{u,v}$	Number of priority queues in physical link (u, v)
(u, v, q)	Queue link from node u to node v via queue q in node u
$\mathcal{L}_{u,v}$	Set of queue links traversing physical link (u, v)
$R_{u,v}$	Transm. bit rate [bit/s] of physical link (u, v)
$\mathcal{E} = \{(u, v, q)\}$	Set of edges in queue link network
$\mathcal{G} = (\mathcal{N}, \mathcal{E})$	Queue link network graph
$\mathcal{I}(u)$	Set of incoming physical links to node u
Flow model	
$\mathcal{F}$	Set of active (accepted) flows in network
$\mathcal{F}_{u,v,q}$	Set of flows traversing queue link (u, v, q)
o_f, d_f	Origin and destination nodes of flow $f \in \mathcal{F}$
r_f	Avg. (mean) bit rate [bit/s] of flow f
b_f	Burstiness parameter [Byte] of flow f
t_f	End-to-end delay limit [s] of flow f
Network resource model	
$\mathbf{A}_R[u, v, q]$	Bit rate allocated to queue link (u, v, q)
$\mathbf{A}_B[u, v, q]$	Buffer capacity allocated to queue link (u, v, q)
$\mathbf{U}_R[u, v, q]$	Utilized mean bit rate of queue link (u, v, q)
$\mathbf{U}_B[u, v, q]$	Utilized buffer space of queue link (u, v, q)
$\mathbf{I}_R[n, u, v, q]$	Incoming agg. mean rate from prior node n via queue q of node u to node v
$\mathbf{I}_B[n, u, v, q]$	Incoming burst from prior node n via queue q of node u to node v
Queue link (edge) model	
$c(u, v, q) \mapsto \mathbb{R}^+$	Cost of using queue link (u, v, q)
$p(u, v, q)$ $\mapsto 1, 2, \ldots, Q_{u,v}$	Priority level of queue feeding into queue link (u, v, q)
$S(u, v, q)$	Transm. bit rate available to queue link (u, v, q) $(= R_{u,v} -$ bit rate alloc. to higher priority queues)
$\mathbf{T}[u, v, q]$	Delay budget (worst-case delay) of queue link (u, v, q)
Path model	
$\mathcal{P}(o, d)$	Set of all possible paths from origin $o \in \mathcal{N}$ to destination $d \in \mathcal{N}$ over edges in $\mathcal{E}$.
P_f	DCLC path $\{(u_p, v_p, q_p)\}_{p=0,1,\ldots,p_{\max,f}}$
$C(P_i)$	Cost of path $\mathcal{P}_i$, see Eq. (11)
$D(P_i)$	End-to-end worst-case delay of path $\mathcal{P}_i$, see Eq. (3)

VI. Online Routing and Admission Control

A. Background on a General DCLC Routing Problem

For a network graph $\mathcal{G}$ consisting of the set of network nodes $\mathcal{N}$ and edges $\mathcal{E}$, i.e., $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, let $\mathcal{P}(o, d)$ denote the set of paths between the given source (origin) node o and destination node d of flow f with delay limit (constraint) t_f. For a path cost function $C(P_i)$, the DCLC routing problem [4] is generally formulated with the objective function

$$\min_{P_i \in \mathcal{P}'(o,d)} C(P_i) \tag{1}$$

and constraint

$$\mathcal{P}'(o, d) = \{P_i \in \mathcal{P}(o, d) \mid D(P_i) \leq t_f\} \tag{2}$$

where $D(P_i)$ denotes the end-to-end delay of path P_i. That is, only the end-to-end paths $P_i \in \mathcal{P}'(o, d)$ that meet the delay limit t_f of flow f, i.e., the paths P_i that satisfy $D(P_i) \leq t_f$, are considered in the search for the minimum cost path. We denote DCLC Algorithm$(\mathcal{G}, t_f)$ for an algorithm, e.g., based on [4], [19], and [20], that solves the conventional DCLC problem represented by objective function (1) and constraint (2) in a centralized fashion in the SDN controller.

B. Proposed Online Routing and Admission Control

1) Offline Worst-Case Delay (Delay Budget) Allocation: Our function split QoS framework allocates to each queue link (u, v, q) a delay budget (worst-case delay) $\mathbf{T}[u, v, q]$ (see Section VIII). We base the delay budget $\mathbf{T}[u, v, q]$ on the worst-case queue link delay resulting from the offline queue link allocations of transmission bit rate $\mathbf{A}_R[u, v, q]$ and buffer capacity $\mathbf{A}_B[u, v, q]$ [see (12)]. Our online admission control evaluates the path delay based on the worst-case queue link delays (queue link delay budgets), i.e.,

$$D(P_i) = \sum_{(u,v,q) \in P_i} \mathbf{T}[u, v, q]. \tag{3}$$

2) Online Rate, Buffer, and Delay Budget Checking: The network calculus based evaluation of the queue link delay budget $\mathbf{T}[u, v, q]$ ensures that the actual packet delay at queue link (u, v, q) does not exceed $\mathbf{T}[u, v, q]$ as long as the utilized rate $\mathbf{U}_R[u, v, q]$ and buffer $\mathbf{U}_B[u, v, q]$ stay below the corresponding queue link allocations of rate $\mathbf{A}_R[u, v, q]$ and buffer $\mathbf{A}_B[u, v, q]$. For the online admission control, it suffices, therefore, to verify that a newly added flow f meets three constraints.

1) *Rate constraint:* The rate utilization due to already admitted flows $\mathbf{U}_R[u, v, q]$ plus the rate of the new flow r_f stays below the rate allocation $\mathbf{A}_R[u, v, q]$, i.e., $\mathbf{U}_R[u, v, q] + r_f < \mathbf{A}_R[u, v, q]$ at each queue link (u, v, q).
2) *Buffer (burst) constraint:* The buffer utilization stays below the buffer allocation, i.e., $\mathbf{U}_B[u, v, q] + b_f < \mathbf{A}_B[u, v, q]$ at each queue link (u, v, q).
3) *Delay constraint:* The worst-case path delay $D(P_i)$ is below the flow delay limit t_f, i.e., $D(P_i) < t_f$.

In the proposed online admission control, the worst-case path delay is based only on the offline rate and buffer allocations and is independent of the routes of the admitted flows through the queue link network. The rates and buffers utilized by the routes of the admitted flows through the queue link network are tracked (online) by the network resource model (see Section VII) and are used for checking the rate and buffer constraints. In summary, in the proposed online admission control, the checking of the delay constraint does not consider the utilization of the network. Rather, the checking of the rate and buffer constraints consider the current network utilization and ensure that the utilizations stay low enough to assure the (offline) allocated worst-case delays (delay budgets).

3) Implementation With Standard DCLC Algorithm: We implement our online admission control with a conventional DCLC algorithm as follows. First, we restrict the set of edges $\mathcal{E}^*(f)$ considered for the routing of a new flow f to the edges that can accommodate the rate r_f and burst size b_f in addition to the already utilized rate $\mathbf{U}_R[u, v, q]$ and buffer space $\mathbf{U}_B[u, v, q]$ while not exceeding the rate allocation $\mathbf{A}_R[u, v, q]$ and buffer allocation $\mathbf{A}_B[u, v, q]$, as specified in lines 2 and 3 of the online routing and admission control algorithm in Fig. 3. The resulting network graph $\mathcal{G}^*(f)$ (see line 4) along with the delay limit t_f is the input for a standard DCLC routing algorithm (line 5 in Fig. 3).

If no path is found, then the flow request is rejected (lines 6–8); otherwise, the flow is admitted. We denote P_f for the path of an admitted flow f. Specifically, we denote

$$P_f = \{(u_p, v_p, q_p)\}_{p=0,1,\ldots,p_{\max,f}} \tag{4}$$

for the ordered sequence of queue links that constitute the path, whereby (u_0, v_0, q_0) denotes the queue link from the source node to the first switching node and $(u_{p_{\max,f}}, v_{p_{\max,f}}, q_{p_{\max,f}})$ denotes the queue link from the last switching node on the path to the destination node.

VII. Network Resource Model

A. Overview

The network resource model tracks the usage of the link queue transmission bit rates and buffers, i.e., all the variables in the network resource model section of Table I. These resource usage metrics are required for the online admission decisions, i.e., the search for a DCLC path for a new flow. Based on the tracked resource usage metrics (which are maintained in the queue link network load map), the model allows for fast online admission control decisions in the control plane without interacting with the data plane.

```
 1: procedure ADD-FLOW($f$)
 2:     $\mathcal{E}^*(f)=\{(u,v,q)\in\mathcal{E}|\mathbf{A}_R[u,v,q]>\mathbf{U}_R[u,v,q]+r_f$
 3:                         $\wedge\mathbf{A}_B[u,v,q]>\mathbf{U}_B[u,v,q]+b_f\}$
 4:     $\mathcal{G}^*(f)\leftarrow(\mathcal{N},\mathcal{E}^*(f))$
 5:     $P_f \leftarrow$DCLC-Algorithm($\mathcal{G}^*(f), t_f$)
 6:     if $P_f == null$ then
 7:         return $False$
 8:     end if
 9:     for all $p=1,2,\ldots,p_{\max,f}$ do
10:         Upd. incom. rate, burst from prior node $n_p=u_{p-1}$:
11:         $\mathbf{I}_R[n_p,u_p,v_p,q_p]\leftarrow\mathbf{I}_R[n_p,u_p,v_p,q_p]+r_f$
12:         $\mathbf{I}_B[n_p,u_p,v_p,q_p]\leftarrow\mathbf{I}_B[n_p,v_p,u_p,q_p]+b_f$
13:     end for
14:     for all $p=1,2,\ldots,p_{\max,f}$ do
15:         Upd. rate, buffer util. at switch node $u_p$:
16:         $\mathbf{U}_R[u_p,v_p,q_p]\leftarrow\mathbf{U}_R[u_p,v_p,q_p]+r_f$
17:         $\mathbf{U}_B[u_p,v_p,q_p]\leftarrow B_{\max}(u_p,v_p,q_p)$
18:     end for
19:     return $True$
20: end procedure
```

Fig. 3. Algorithm for online routing and admission control. If a new flow f is admissible, i.e., a path is found that can accommodate the added rate r_f (line 2), added burst size b_f (line 3), and delay limit (deadline constraint) t_f (line 5), then the new flow f is added to the network calculus based network resource model (lines 9–18).

We present the algorithms for tracking the resource usage in Section VII-B. The resource tracking requires the available transmission bit rates $S(u, v, q)$ and the maximum buffer utilizations $B_{\max}(u, v, q)$ of a queue link (u, v, q) on a path considered for a new flow. We note that $S(u, v, q)$ and $B_{\max}(u, v, q)$ are auxiliary variables that are only computed for updating (tracking) the buffer usage $\mathbf{U}_B[u, v, q]$ in our network resource model. The evaluation of these auxiliary variables $S(u, v, q)$ and $B_{\max}(u, v, q)$ is detailed in Section VII-C. Section VII-D presents routing cost functions, including one cost function building on the maximum buffer utilization $B_{\max}(u, v, q)$.

B. Resource Tracking

If there is a suitable path P_f, then the incoming rate and burst variables $\mathbf{I}_R$ and $\mathbf{I}_B$ for the source node and all nodes along the path are updated (see Fig. 3, lines 9–13). Then, the bandwidth and buffer utilization variables $\mathbf{U}_R$ and $\mathbf{U}_B$ are updated, using the maximum buffer utilization $B_{\max}$ derived in Section VII-C (see Fig. 3, lines 14–18). Fig. 4 presents the algorithm for the removal of a flow f. The flow parameters have to be subtracted from every corresponding counter variable.

C. Evaluation of Available Transmission Bit Rate and Maximum Buffer Utilization

The available queue link transmission bit rate $S(u, v, q)$ is the (physical) link transmission bit rate R_{uv} minus the sum of

```
procedure REM-FLOW($f$)
    for all $p = 1, 2, \ldots, p_{\max,f}$ do
        $\mathbf{I}_R[n_p, u_p, v_p, q_p] \leftarrow \mathbf{I}_R[n_p, u_p, v_p, q_p] - r_f$
        $\mathbf{I}_B[n_p, u_p, v_p, q_p] \leftarrow \mathbf{I}_B[n_p, u_p, v_p, q_p] - b_f$
        $\mathbf{U}_R[u_p, v_p, q_p] \leftarrow \mathbf{U}_R[u_p, v_p, q_p] - r_f$
        $\mathbf{U}_B[u_p, v_p, q_p] \leftarrow B_{\max}(u_p, v_p, q_p)$
    end for
end procedure
```

Fig. 4. Algorithm for removing an existing flow f from the network resource model.

the transmission bit rates allocated to the queues with higher priority than the considered queue q:

$$S(u,v,q) = R_{u,v} - \sum_{l \in \mathcal{L}_{u,v}} \mathbf{A}_R[u,v,l]\, 1_{\{p(u,v,l) > p(u,v,q)\}} \quad (5)$$

where $1_{\{A\}}$ denotes the basic indicator function, which is one if A is true and zero otherwise.

Based on network calculus principles [16], [18], our network resource model evaluates a bound on the maximum buffer utilization $B_{\max}(u,v,q)$ for a queue link (u,v,q) as follows. We consider a given node u and first bound the amount of traffic arriving from incoming physical links from prior nodes $n \in \mathcal{I}(u)$. For a given prior node n, the amount of incoming traffic to queue link (u,v,q) is bounded by the incoming burst size $\mathbf{I}_B[n,u,v,q]$ and mean aggregate incoming traffic rate $\mathbf{I}_R[n,u,v,q]$. Specifically, the incoming traffic amount cannot exceed $\mathbf{I}_B[n,u,v,q] + \mathbf{I}_R[n,u,v,q]t$ for any positive time duration t. Moreover, the incoming traffic cannot exceed the transmission bit rate $R_{n,u}$ of the physical link (n,u). For positive burst size $\mathbf{I}_B[n,u,v,q]$ and a stable link with $\mathbf{I}_R[n,u,v,q] < R_{n,u}$, the overall bound

$$\alpha_n(t) = \min\{\mathbf{I}_B[n,u,v,q] + \mathbf{I}_R[n,u,v,q]t,\ R_{n,u}t\} \quad (6)$$

on the amount of traffic arriving over a time period t is concave. In order to reduce notational clutter, we omit the queue link notation (u,v,q) from $\alpha_n(t)$; however, it is understood that $\alpha_n(t)$ applies to a given queue link (u,v,q). The concave bound $\alpha_n(t)$ has a "knee point" at the time value where the two bounds in (6) intersect, i.e., at

$$\Delta[n,u,v,q] = \frac{\mathbf{I}_B[n,u,v,q]}{R_{n,u} - \mathbf{I}_R[n,u,v,q]}. \quad (7)$$

The aggregate amount of traffic arriving from the set of prior nodes $n \in \mathcal{I}(u)$ to queue link (u,v,q) is bounded by the sum of the bounds $\alpha_n(t)$ from the individual prior nodes n:

$$\alpha(t) = \sum_{n \in \mathcal{I}(u)} \alpha_n(t). \quad (8)$$

The concave bound $\alpha(t)$ on the amount of arriving traffic is illustrated for an example with two prior nodes in Fig. 5. The aggregate bound $\alpha(t)$ has knee points at times $\Delta(n,u,v,q)$, $n \in \mathcal{I}(u)$, i.e., there are as many knee points as prior nodes feeding into queue link (u,v,q) (whereby multiple knee points can coincide when the corresponding parameter values are identical).

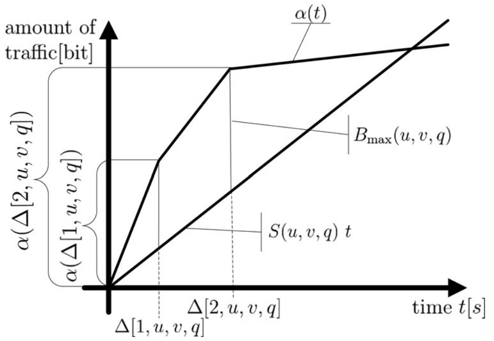

Fig. 5. Illustration of evaluation of maximum buffer utilization $B_{\max}$ at a queue link (u,v,q). The transmission bit rates $R_{n,u}$ of the incoming physical links as well as the incoming mean bit rates and burst sizes of the admitted flows determine the concave bound $\alpha(t)$ on the maximum amount of arriving traffic within a time interval of duration t. Comparing the differences between $\alpha(t)$ and the traffic transmitted with the guaranteed available transmission bit rate $S(u,v,q)$ at the "knee points" of the concave bound $\alpha(t)$ gives $B_{\max}$.

The buffer occupancy (utilization) at queue link (u,v,q) is bounded by the difference between the aggregate bound $\alpha(t)$ on the amount of arriving traffic and the amount of traffic transmitted with the available transmission bit rate $S(u,v,q)$ from (5). In particular, due to the concave shape of the aggregate bound $\alpha(t)$, the maximum buffer utilization $B_{\max}(u,v,q)$ of queue link (u,v,q) must occur at one of the knee points specified by (7). Thus

$$\begin{aligned} B_{\max}(u,v,q) = \max_{n \in \mathcal{I}(u)} \{&\alpha(\Delta[n,u,v,q]) \\ &- S(u,v,q)\Delta[n,u,v,q]\}. \end{aligned} \quad (9)$$

D. Routing Cost Functions

We introduce two per-hop cost components and corresponding cost functions. Our "simple" per-hop cost component counts the number of queues q with a priority level less than or equal to the priority level $p(u,v,q)$ of the queue q traversed by the considered flow. With $q = 1$ denoting the lowest priority and $q = Q_{u,v}$ denoting the highest priority, the number of queues with lower (or equal) priority than queue q is simply $c_s(u,v,q) = q$.

Our "buffer-aware" per-hop cost component counts the buffer usage $B_{\max}^{f_{\text{new}}} - B_{\max}$ due to a new flow f_{new} being added to the existing set of flows. This buffer usage is normalized by the allocated buffer space $\mathbf{A}_B[u,v,q]$ and the burstiness $b_{f_{\text{new}}}$ of the new flow:

$$c_b(u,v,q) = \frac{B_{\max}^{f_{\text{new}}}(u,v,q) - B_{\max}(u,v,q)}{\mathbf{A}_B[u,v,q]\, b_{f_{\text{new}}}}. \quad (10)$$

The total cost of an end-to-end path is the sum of the per-hop cost components $c(u,v,q)$ (e.g., $c(u,v,q) = c_s(u,v,q)$ or $c_b(u,v,q)$) along the traversed path, i.e.,

$$C(P_i) = \sum_{(u,v,q) \in P_i} c(u,v,q). \quad (11)$$

VIII. Offline Resource Allocation

A. General Considerations

With our network model, the resource allocation can readily be executed periodically in the background to optimize the queue link configuration of each physical link. Generally, a potential resource allocation algorithm has to optimize the allocation of the physical transmission bit rate to the different queues of a link to eliminate bottlenecks. Bottlenecks could arise for each of the three types of resources:

1) data rate, indicated through the utilized mean transmission bit rate $\mathbf{U}_R[u, v, q]$;
2) buffer space, indicated through the utilized buffer $\mathbf{U}_B[u, v, q]$;
3) delay limit utilization, indicated through the difference between the delay limits t_f and the worst-case path delays $D(P_f)$ (3).

For an actual network operating with the proposed function split approach, only the online routing needs to be executed when a new flow requests admission. As the arrivals of new flow requests or departures of existing flows occur typically on a slow time scale, significant changes in the resource utilization of the queue links occur only slowly. Thus, executing the resource allocation periodically in the background will typically be sufficient for conducting the routing (and admission control) with a resource allocation that is (nearly) up to date and close to optimal for the currently carried flows.

The resource allocation in the background should allocate transmission bit rates $\mathbf{A}_R[u, v, q]$ to the individual priority queue links so as to minimize a general resource allocation cost function based on the utilized rate, buffer, and delay budget. Based on the transmission bit rate allocations $\mathbf{A}_R[u, v, q]$, we evaluate then the worst-case delay (delay budget) $T[u, v, q]$ of a given queue link based on network calculus principles [16], [18]. Recall that $\mathbf{A}_B[u, v, l]$ specifies the available buffer size for queue link (u, v, l) and let $P_{\max}$ denote the maximum size of Ethernet packets (typically 1500 Byte). Moreover, recall from Section VII-C that $S(u, v, q)$ (5) denotes the transmission bit rate available at queue link (u, v, q). In the worst case, a given packet has to wait for transmission of all buffered traffic with the same or higher priority as well as, possibly, the completion of an ongoing transmission of a lower priority packet that cannot be interrupted, i.e.,

$$\mathbf{T}(u, v, q) = \frac{\sum_{l \in \mathcal{L}_{\mathrm{uv}}} \mathbf{A}_B[u, v, l]\, 1_{\{p(u,v,l) \geq p(u,v,q)\}} + P_{\max}}{S(u, v, q)}. \tag{12}$$

The optimal solution of the general resource allocation problem is beyond the scope of this paper. However, in order to illustrate a possible resource allocation approach for our overall function split QoS framework, we propose an elementary resource (rate) allocation algorithm in the next subsection.

B. Elementary Resource Allocation Algorithm

Initially, for the commencement of operation of a network, link rates are allocated according to historic traffic observations for similar networks and predictions of traffic patterns, similar to [22]. Then, the resource allocation algorithm in Figs. 6 and 7 is periodically employed in the background (offline) to sequentially examine rate allocations at the individual switching nodes σ. For a given switching node σ, each physically connected outgoing (next-hop) node $\omega \in \mathcal{O}_\sigma$ is considered. The current state of the network resource model (see Section VII) is copied over to a virtual network resource model, which is then used (in the background) for executing the resource allocation algorithm.

```
 1: procedure ResourceAllocation
 2:     for all Switching Nodes $\sigma \in \mathcal{N}$ do
 3:         for all Next-hop Nodes $\omega \in \mathcal{O}_\sigma$ do
 4:             Virtual Netw. Res. Model $\leftarrow$ Netw. Res. Model
 5:             Execute Steps on Virtual Netw. Res. Model
 6:             $\mathcal{T}_{\sigma,\omega}$ = Set of admitted flows traversing $\sigma, \omega$
 7:             for all $f \in \mathcal{T}_{\sigma,\omega}$ do
 8:                 Rem-Flow($f$)
 9:             end for
10:             $F_{\max} = 0$
11:             for all $\mathbf{A}_R[\sigma, \omega, q]_{q \in \mathcal{L}_{\sigma,\omega}} \in \mathcal{R}$ do
12:                 $F$ = FlowCountEst($\mathbf{A}_R[\sigma, \omega, q]_{q \in \mathcal{L}_{\sigma,\omega}}$)
13:                 if $F > F_{\max}$ then
14:                     $F_{\max} = F$
15:                     $\mathbf{A}_{R,\max} = \mathbf{A}_R[\sigma, \omega, q]_{q \in \mathcal{L}_{\sigma,\omega}}$
16:                 end if
17:             end for
18:             if $F_{\max} > |\mathcal{T}_{\sigma,\omega}|$ then
19:                 Netw. Res. Model $\leftarrow \mathbf{A}_{R,\max}[\sigma, \omega, q]_{q \in \mathcal{L}_{\sigma,\omega}}$
20:                 for all $(\sigma, \omega, q) \in \mathcal{L}_{\sigma,\omega}$ do
21:                     Evaluate $\mathbf{T}[\sigma, \omega, q]$ from Eqn. (12)
22:                 end for
23:                 for all $f \in \mathcal{T}_{\sigma,\omega}$ do
24:                     Rem-Flow($f$), Add-Flow($f$)
25:                 end for
26:             end if
27:         end for
28:     end for
29: end procedure
```

Fig. 6. Resource allocation algorithm executed periodically in background. The algorithm cycles through all switching nodes σ, and corresponding physically connected outgoing (next-hop) nodes ω. The current real network resource model is first copied over to a virtual network resource model to examine candidate resource (rate) allocations $\mathbf{A}_R[\sigma, \omega, q]_{q \in \mathcal{L}_{\sigma,\omega}}$ to the set $\mathcal{L}_{\sigma,\omega}$ of queue links traversing physical link (σ, ω). The candidate allocations $\mathbf{A}_R[\sigma, \omega, q]_{q \in \mathcal{L}_{\sigma,\omega}}$ are from a prescribed set $\mathcal{R}$. The resource allocation $\mathbf{A}_{R,\max}$ supporting the highest number of flows, as determined by the flow count estimation algorithm in Fig. 7, is implemented in the real network resource model (provided $\mathbf{A}_{R,\max}$ supports more flows than the current allocation).

The brute-force search in line 11 of Fig. 6 is over a prescribed reasonably large set $\mathcal{R}$ of rate allocations $\mathbf{A}_R[\sigma, \omega, q]_{q \in \mathcal{L}_{\sigma,\omega}}$ to all the queue links $q \in \mathcal{L}_{\sigma,\omega}$ traversing the considered physical link (σ, ω). For instance, in our numerical evaluations in Section IX-B, we consider the allocation of the total physical transmission bit rate 1 Gb/s in 5 Mbyte/s steps to the individual queue links.

For a given considered rate allocation to all the queue links $\mathbf{A}_R[\sigma, \omega, q]_{q \in \mathcal{L}_{\sigma,\omega}}$, the algorithm in Fig. 7 estimates the number

```
 1: procedure FLOWCOUNTEST($\mathbf{A}_R[\sigma,\omega,q]_{q\in\mathcal{L}_{\sigma,\omega}}$)
 2:     for all Admitted flows $f \in \mathcal{T}_{\sigma,\omega}$ do
 3:         if ADD-FLOW($f$) = False then
 4:             return 0
 5:         end if
 6:     end for
 7:     FlowCount = $|\mathcal{T}_{\sigma,\omega}|$
 8:     New flow $n$ = StatModel($\mathcal{T}_{\sigma,\omega}$)
 9:     while ADD-FLOW($n$) = True do
10:         New flow $n$ = StatModel($\mathcal{T}_{\sigma,\omega}$)
11:         FlowCount ++
12:     end while
13:     return FlowCount
14: end procedure
```

Fig. 7. Flow count estimator algorithm for testing a candidate resource (rate) allocation $A_R[\sigma,\omega,q]_{q\in\mathcal{L}_{\sigma,\omega}}$. The admitted flows that are currently traversing the examined physical link (σ,ω) are added one by one to the virtual network resource model. After all admitted flows have been successfully added, additional new flows are generated according to a statistical model of the flows currently traversing the link (σ,ω). The total number of flows (already admitted flows plus new flows from the statistical model) that the node can accommodate subject to the QoS constraints is returned to the resource allocation algorithm in Fig. 6.

of flows that can be supported in two main steps. In a first step (lines 2–6), the already admitted flows are added one-by-one back into the virtual network resource model. If all admitted flows can still be accommodated, the second step (lines 8–12) estimates the number of additional flows that the node can carry with the considered rate allocation. For this estimation, new flows are generated from a statistical flow model. The statistical flow model could, for instance, average the flow model parameter values (see Table I) of the admitted flows $\mathcal{T}_{\sigma,\omega}$ or uniformly randomly draw each added flow from $\mathcal{T}_{\sigma,\omega}$ (which is considered in our evaluations in Section IX).

We note that the resource allocation algorithm in Fig. 6 always starts from a valid state since the admission control has only admitted flows whose delay limits can be met with the currently considered network resource allocation. The search for a new rate allocation pattern on the virtual network resource model delivers a pattern that supports at least the currently admitted flows (and potentially additional new flows). Thus, already admitted flows are guaranteed to still have a valid path when a new resource allocation pattern is adopted in line 19 of Fig. 6. Line 21 uses (12) to update the queue link delay budgets $\mathbf{T}[\sigma,\omega,q]$, which are used to evaluate the delay constraints of new flow requests [see (3)]. The flows are shifted to their valid paths with the new resource allocation pattern through executing the flow removal and addition in line 24 of Fig. 6. Running this resource allocation algorithm periodically in the background following the general strategies for periodic background processes in networks [23] adapts the resource allocation to the admitted traffic flows and, if possible, allows for the admission of new flows.

IX. Simulation Evaluation

A. Network Setup and Traffic Mixes

We emphasize that the function split QoS framework proposed in the preceding Sections VI–VIII is applicable to arbitrary network structures that can be modeled with the queue link network topology introduced in Section V. For the simulation evaluations in this section, we consider a unidirectional ring, which is a typical elementary industrial network structure, to illustrate the performance characteristics of the proposed function split QoS framework. The ring has six switching nodes and six source/destination nodes (see Fig. 8). Each switching node output port (link) has four output queues operating with strict priority scheduling (without preemption). Note that the resulting queue link network has for three hops already 64 possible paths. Each flow belongs to one of the four service classes in Table II. Specifically, we consider the ensemble of 968 traffic mixes obtained by letting each of the four traffic service classes contribute 5, 10, 15,. . ., 85% of the total traffic. Independently of the service class, we draw for each flow a random hop distance according to a short (S) hop distance scenario with 70% of the flows transmitted over one hop, 20% over two hops, and 10% over three hops and a long (L) hop distance scenario with 10% one hop, 10% two hops, 20% three hops, 30% four hops, and 30% five hops flows.

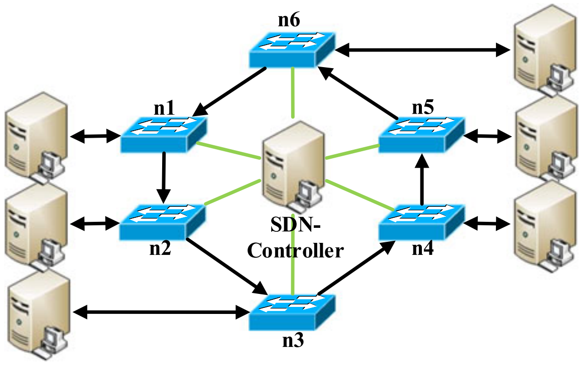

Fig. 8. Illustration of unidirectional ring physical link topology for evaluation of the end-to-end real-time QoS framework with function split between resource (transmission bit rate) allocation to priority queue links and routing through queue links. Each link has 1 Gb/s and four priority queues, each with 90 000 Byte buffer (as per NEC PF5240 switch specifications).

TABLE II
Traffic Characteristics and Delay Limits of the Considered Industrial Traffic Service Classes

Service Class	Mean Bit Rate r_c [kByte/s]	Burst Size b_c [Byte]	Delay Limit t_c [ms]
$c = 1$	10	100	5
$c = 2$	10	100	10
$c = 3$	10	100	20
$c = 4$	10	100	50

Each traffic class has a packet size of 64 Byte.

B. Evaluation Procedures

1) Online Routing and Admission Control: In order to obtain detailed insights into the performance of our online routing algorithm, including the online admission control for new flows, we consider the online routing initially in

TABLE III
Comparison of Computation Times for Short-Hop-Distance Scenario: Online Routing and Admission Control for a Single Flow Versus Offline (Background) Resource Allocation (One Iteration Through All Six Switching Nodes) Versus MIP for Resource Allocation and Routing.

	Online Routing and Adm. Control	Offline Resource Allocation	MIP: Res. Alloc. and Routing
25th percentile	$2.85\ \mu s$	24.9 s	148 s
mean	$3.47\ \mu s$	27.1 s	333 s
75th percentile	$3.95\ \mu s$	29.9 s	333 s

isolation from the offline resource allocation algorithm in Section VIII-B. In particular, we consider 2925 rate allocation patterns $\mathbf{A}_R[u, v, q]$ obtained by allocating the $R = 1$ Gb/s physical link rate in 5 MByte/s steps to the $Q_{u,v} = 4$ priority queue links ($q = 1, 2, 3, 4$) for each physical link. For each rate allocation pattern, we successively add in flows according to a given considered traffic mix (out of a total of 968 considered traffic mixes). For each new flow, we run the online routing and admission control algorithm. We gradually add in as many flows as admissible subject to the QoS requirements. We, thus, find the best and the worst resource allocation pattern, i.e., the pattern that leads to the most carried (admitted) flows and the fewest carried flows in the network among the 2925 considered rate allocation patterns.

2) Offline Resource Allocation: For the evaluation of the offline resource allocation from Section VIII, we simulate the continuous operation of the network for each of the 968 traffic mixes. Specifically, we initialize the network with a uniform (equal) split of the physical transmission bit rates to the queue links. Then, we continuously try to add in more flows of the considered traffic mix (using the online routing and admission control), while the resource allocation proceeds once in the background through all six switching nodes.

3) Comparison Benchmark: MIP Solution: We compare with the results generated by the offline MIP solution to the joint problem of routing and resource allocation in [18]. This comparison provides a benchmark for the performance of 1) our online routing and admission control algorithm, which is only optimal for the routing of each flow, and 2) our offline resource allocation algorithm which sequentially considers rate allocations at individual nodes. The MIP solution finds an overall optimum for the joint routing and resource allocation.

C. Computation Time

In Table III, we compare the computation time of the offline MIP algorithm [18] with the total computation time for the resource allocation algorithm from Section VIII-B and the online routing (and admission control) from Section VI-B. We consider the short-hop-distance scenario with up to three hops, for which the MIP is computationally feasible (the MIP becomes computationally prohibitive for the long-hop-distance scenario). All times were measured for computations on the same physical computing hardware in order to allow for meaningful computation time comparisons of the different approaches. The computation times in Table III are based on the 968 different traffic mixes, i.e., the variations of the computing times represent the variations due to the different traffic mixes. We observe that the mean total computation time (for a given arbitrary traffic mix) for the online routing and admission decision for a new flow seeking admission to the network is less than 5 μs. While the delay for making routing and admission decisions in the controller does not impact the packet delay of established flows on the forwarding (data) plane, the very fast routing and admission control computation in combination with real-time QoS control channel flows that could be established between switches and the controller could ensure delay bounds for flow establishment.

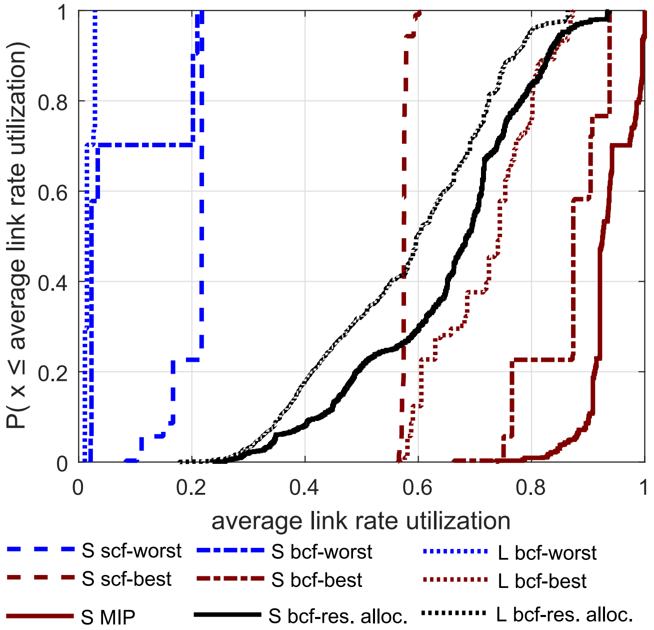

Fig. 9. Comparison of the CDF (across the 968 traffic mixes) of the average link (rate) utilization for the online routing evaluation for fixed rate allocations from Section IX-B1 (showing curves for the worst and the best rate allocation), the evaluation of online routing in conjunction with the background resource allocation algorithm from Section IX-B2 (bcf-res. alloc.), and the MIP benchmark evaluation from Section IX-B3. The short (S, up to three hops) and long (L, up to five hops) hop distances are considered. The scf and the bcf routing cost functions are considered for the evaluation from Section IX-B1.

One iteration of the resource allocation algorithm through all six switching nodes, which is executed in the background, i.e., does not impede the online routing and admission control, takes on the order of 30 s. In contrast, the MIP takes on the order of a few hundred seconds.

D. Utilization of Links, Buffers, and Delay Limits

We present cumulative distribution function (CDF) plots (across the 968 traffic mixes) in Figs. 9–11 for the following.

1) The average link (rate) utilization defined as the mean utilization of all links in the network (see Fig. 9).
2) The average buffer utilization defined as the mean utilization of all buffers in the network (see Fig. 10).

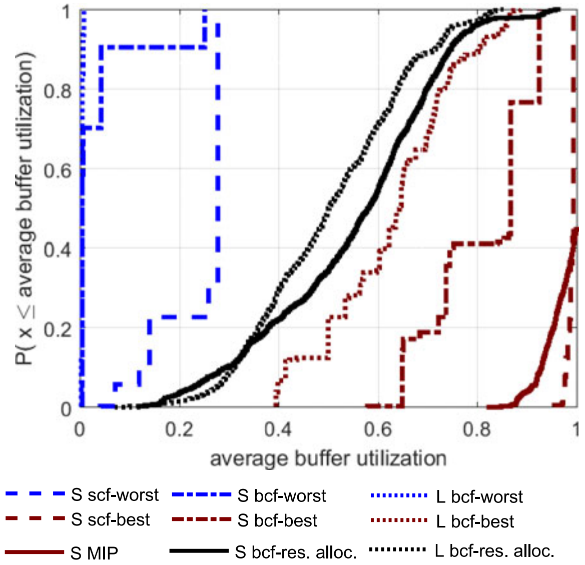

Fig. 10. Comparison of the CDF of the average buffer utilization of the overall network for the three evaluation procedures from Sections IX-B1–IX-B3 for the scf and bcf for short (S) and long (L) hop distance scenarios.

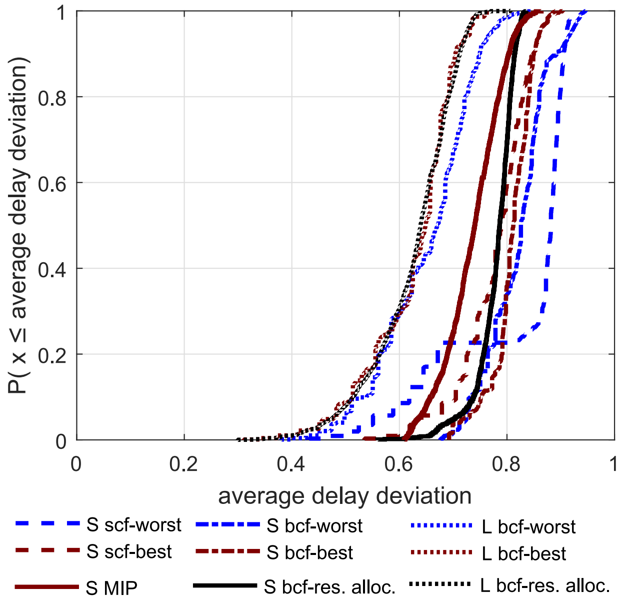

Fig. 11. Comparison of the average delay deviation (difference between delay limit t_f and worst-case path delay $D(P_f)$) of the overall network for the three evaluation procedures from Sections IX-B1–IX-B3 for the scf and bcf for short (S) and long (L) hop distance scenarios.

3) The average delay deviation defined as the average of the deviations of the end-to-end worst-case path delay $D(P_f)$ form the delay limit t_f in percent, i.e., the average of $(t_f - D(P_f))/t_f$, over all flows f carried in the network (see Fig. 11).

We focus initially on the online routing and MIP results for the link rate utilization in Fig. 9 and the buffer utilization in Fig. 10 for the short-hop-distance scenario. The results for the link utilization in Fig. 9 indicate higher bandwidth utilizations with the buffer-aware routing cost function (bcf) than the simple cost function (scf). For the best resource (rate) allocation, the bcf routing achieves up to 93% link utilization compared to up to close to 60% link utilization with scf routing (and compared to up to 100% with the MIP).

At the same time, we observe from Fig. 10 that the scf routing leads to a higher average buffer utilization than bcf routing. For the best resource allocation (that maximizes the number of flows), the scf routing almost fully utilizes the buffers, whereas bcf routing utilizes approximately 65–92% of the buffers. The scf (see Section VII-D) counts the number of queues with equal or lower priority and thus strives to route flows through low-priority queues. However, due to the principles of deterministic network calculus, the low-priority queues have only relatively low guaranteed available transmission bit rates S [see (5)], since the transmission bit rates allocated to higher priority queues are not considered to be available for the low-priority queue. The low level of guaranteed available transmission bit rate S leads to a high worst-case maximum buffer occupancy (utilization) $B_{\max}$ [see (9)] for the low-priority queues. In contrast, the buffer-aware cost function (bcf) considers directly the maximum buffer occupancy [see (10)] and accordingly routes flows through a mix of low- and high-priority queues so as to "save" buffer space. Comparing Figs. 9 and 10, we observe that for the considered networking scenario, the buffer space becomes the bottleneck resource for the scf routing: The scf routing utilizes nearly all available buffers for average link utilizations close to 60%. In contrast, bcf routing achieves up to 93% link utilization while requiring less than 92% of the available buffer for the worst-case $B_{\max}$ requirements.

We observe that for a given routing cost function, there is a wide gap between the utilization achieved with the best and worst resource (rate) allocations patterns $\mathbf{A}_R[u, v, q]$. We plot the utilization results for both the best and worst rate allocations to illustrate the impact of the rate allocation on the performance of the introduced function split QoS framework. Moreover, we observe from Fig. 9 that the elementary resource allocation algorithm from Section VIII-B (bcf-res. alloc. curve) also covers a fairly wide range of utilization levels. The elementary algorithm that considers rate allocations at individual nodes gives poor rate allocations for some traffic mixes, as indicated by the wide gap to the bcf-best curve at the bottom of Fig. 9. However, for over 60% of the traffic mixes, the elementary resource allocation is within 25% of the maximal supported number of flows of the bcf-best scenario. The refinement of the resource allocation within the function split QoS framework, e.g., by jointly considering rate allocations to several switching nodes, is an important direction for future research.

We observe from Figs. 9 and 10 that the long (L) hop distance scenario gives generally lower link and buffer utilizations than the short (S) hop distance scenario. The bcf routing with offline resource allocation (bcf-res. alloc. curve), for instance, has 5–10% lower link and buffer utilizations for the long-hop distance than for short-hop distance. The bcf-best approach achieves bandwidth utilizations up to around 84% for the long-hop-distance scenario compared to up to 93% for the short-hop-distance scenario. This is mainly because the long-hop-distance flows require rate and buffer allocations at more successive switch queue links and are, therefore, more difficult to accommodate, leading to fewer admitted flows and hence lower utilization levels.

Fig. 11 indicates that the bcf (for the best rate allocation) has a slightly higher average delay deviation between the delay limits and the worst-case path delays than the scf. This means that the bcf gives on average a shorter path delay $D(P_f)$, i.e., utilized less of the available delay limit t_f than the scf. However, even the MIP has an average deviation of over 60%, i.e., the path delays $D(P_f)$ are less than 40% of the delay limits t_f for all traffic mixes. The delay deviation curves are fairly close together, except for the scf (with the worst rate allocation) which utilizes on average only about 10% of the delay limits for roughly 70% of the traffic mixes. Overall, these delay deviation results indicate that the delay limits are on average only partly utilized, i.e., the packet traffic arrives typically well before the deadlines. We also observe from Fig. 11 that the long-hop-distance scenario utilizes 10–20% more of the delay budget than the short-hop-distance scenario.

Overall, the results indicate that the introduced split function QoS framework makes very quick runtime admission decisions while providing deterministic worst-case QoS guarantees. In conjunction with suitable offline (background) resource allocation, the connection carrying capacity and bandwidth utilization of our split function QoS network comes for short-hop distances in the considered ring topology example network to within 7% of the performance of a monolithic MIP that jointly optimizes routing and resource allocation.

X. Conclusion

We have developed a novel function split framework for achieving the hard real-time QoS required by industrial communication based on SDN. Our approach splits the online routing (and admission control) from the background allocation of transmission bit rates to priority queues. The function split operates in the context of a network of queue links that provide a range of QoS priorities at each switching nodes. Online DCLC routing selects the switching nodes as well as QoS priority queues that a flow traverses so as to meet its end-to-end delay requirement and to minimize a routing cost function. The function split framework can be implemented with any SDN controller that has a global view of the network and can set the queue level flow rules.

Our evaluations for two routing cost functions indicate that the proposed function split approach makes accurate routing and admission control decisions within a few microseconds, compared to hundreds of seconds required by a monolithic MIP approach that jointly optimizes routing and resource allocation. At the same time, our approach achieves up to approximately 93% average link utilization in an example industrial communication scenario compared to close to 100% utilization by the MIP approach.

There are several directions for future research within the split function QoS framework introduced in this article. One direction is to explore a wider range of routing cost functions and scheduling policies. Extensions to more complex scheduling policies than the static priority scheduling considered in this study could bring performance enhancements [24]. Another important direction is to investigate refinements of the background (offline) resource (transmission bit rate) allocation. Yet, another direction is to consider statistical (soft) real-time QoS which can achieve increased utilizations at the expense of rare deadline violations [25].

References

[1] L. Seno, F. Tramarin, and S. Vitturi, “Performance of industrial communication systems: Real application contexts,” *IEEE Ind. Electron. Mag.*, vol. 6, no. 2, pp. 27–37, Jun. 2012.

[2] P. Gaj, J. Jasperneite, and M. Felser, “Computer communication within industrial distributed environment—A survey,” *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 182–189, Feb. 2013.

[3] D. Kreutz, F. M. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[4] D. S. Reeves and H. F. Salama, “A distributed algorithm for delay-constrained unicast routing,” *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 239–250, Apr. 2000.

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

[6] H. Bauer, J.-L. Scharbarg, and C. Fraboul, “Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach,” *IEEE Trans. Ind. Informat.*, vol. 6, no. 4, pp. 521–533, Nov. 2010.

[7] J. Jasperneite, J. Imtiaz, M. Schumacher, and K. Weber, “A proposal for a generic real-time Ethernet system,” *IEEE Trans. Ind. Informat.*, vol. 5, no. 2, pp. 75–85, May 2009.

[8] T. Skeie, S. Johannessen, and O. Holmeide, “Timeliness of real-time IP communication in switched industrial Ethernet networks,” *IEEE Trans. Ind. Informat.*, vol. 2, no. 1, pp. 25–39, Feb. 2006.

[9] K. Nahrstedt, D. Xu, D. Wichadakul, and B. Li, “QoS-aware middleware for ubiquitous and heterogeneous environments,” *IEEE Commun. Mag.*, vol. 39, no. 11, pp. 140–148, Nov. 2001.

[10] L. Sha *et al.*, “Real time scheduling theory: A historical perspective,” *Real-Time Syst.*, vol. 28, nos. 2/3, pp. 101–155, 2004.

[11] S. Jain *et al.*, “B4: Experience with a globally-deployed software defined WAN,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[12] P. Sharma, S. Banerjee, S. Tandel, R. Aguiar, R. Amorim, and D. Pinheiro, “Enhancing network management frameworks with SDN-like control,” in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage*, 2013, pp. 688–691.

[13] M. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, “PolicyCop: An autonomic QoS policy enforcement framework for software defined networks,” in *Proc. IEEE SDN Future Netw. Serv.*, 2013, pp. 1–7.

[14] S. Sharma *et al.*, “Implementing quality of service for the software defined networking enabled future internet,” in *Proc. IEEE Eur. Workshop Softw. Defined Netw.*, Sep. 2014, pp. 49–54.

[15] A. Ishimori, F. Farias, E. Cerqueira, and A. Abelém, “Control of multiple packet schedulers for improving QoS on OpenFlow/SDN networking,” in *Proc. IEEE Eur. Workshop Softw. Defined Netw.*, 2013, pp. 81–86.

[16] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, vol. 2050. Berlin, Germany: Springer, 2001.

[17] Q. Duan, “Network-as-a-Service in software-defined networks for end-to-end QoS provisioning,” in *Proc. IEEE Wireless Opt. Commun. Conf.*, May 2014, pp. 1–5.

[18] J. W. Guck and W. Kellerer, “Achieving end-to-end real-time quality of service with software defined networking,” in *Proc. IEEE Int. Conf. Cloud Netw.*, Oct. 2014, pp. 70–76.

[19] Y. Han, M. Yao, and Z. Liu, “A scalable method for DCLC problem using hierarchical MDP model,” *Comput. Commun.*, vol. 36, no. 12, pp. 1310–1316, 2013.

[20] Z. Jia and P. Varaiya, “Heuristic methods for delay constrained least cost routing using κ-shortest-paths,” *IEEE Trans. Autom. Control*, vol. 51, no. 4, pp. 707–712, Apr. 2006.

[21] J. W. Guck, M. Reisslein, and W. Kellerer, “Model-based control plane for fast routing in industrial QoS network,” in *Proc. IEEE Int. Symp. Quality Serv*, Jun. 2015, pp. 65–66.

[22] A. Bianco, J. M. Finochietto, G. Giarratana, F. Neri, and C. Piglione, "Measurement-based reconfiguration in optical ring metro networks," *J. Lightw. Technol.*, vol. 23, no. 10, pp. 3156–3166, Oct. 2005.

[23] S. Floyd and V. Jacobson, "The synchronization of periodic routing messages," *IEEE/ACM Trans. Netw.*, vol. 2, no. 2, pp. 122–136, Apr. 1994.

[24] J. Liebeherr, D. E. Wrege, and D. Ferrari, "Exact admission control for networks with a bounded delay service," *IEEE/ACM Trans. Netw.*, vol. 4, no. 6, pp. 885–901, Dec. 1996.

[25] J.-L. Scharbarg, F. Ridouard, and C. Fraboul, "A probabilistic analysis of end-to-end delays on an AFDX avionic network," *IEEE Trans. Ind. Informat.*, vol. 5, no. 1, pp. 38–49, Feb. 2009.

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.

Martin Reisslein (A'96–S'97–M'98–SM'03–F'14) received the Ph.D. degree in systems engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 1998.

He is currently a Professor in the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ, USA.

Dr. Reisslein served as the Editor-in-Chief for the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS from 2003 to 2007 and as an Associate Editor for the IEEE/ACM TRANSACTIONS ON NETWORKING from 2009 to 2013. He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON EDUCATION as well as *Computer Networks* and *Optical Switching and Networking*.

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.