Achieving end-to-end real-time Quality of Service with Software Defined Networking

Jochen W. Guck and Wolfgang Kellerer
Lehrstuhl für Kommunikationsnetze
Technische Universität München
Munich, 80290
Email: {guck, wolfgang.kellerer}@tum.de

***Abstract*—Due to the distributed nature of the network protocols, the provisioning of end-to-end Quality of Service (QoS) in IP-based networks is a complex task. We argue that Software Defined Networking (SDN) delivers a key technology to improve QoS provisioning. The core idea of SDN is the establishment of a standardized interface between the control and forwarding planes, allowing a logically centralized controller to control each forwarding element in a unified way. This unified interface provides three key features of SDN: flexibility, a centralized view and programmability. In this paper we exploit these features to create an end-to-end real-time QoS communication service based on SDN. In particular, our concept allows to flexibly allocate each flow in a network to different priority queues in each hop along its way to optimize the resource usage while being able to keep the delay and bandwidth constraints. This concept can be used, for example, to realize a more fine granular access control for real-time communication services allowing potentially more flows to be accepted than with traditional QoS reservation concepts. For our concept, we introduce a deterministic network model using network calculus to compute the optimal paths for each flow through the priority queues. We validate our concept with a simulation based performance analysis for a selected network scenario. In this setting our concept improves the average link utilization from 30% to more than 60% compared to traditional QoS reservation concepts.**

Keywords*—*Software Defined Networking, Openflow, Quality of Service, real-time, priority queueing, access control.

I. Introduction

State of the art Quality of Service (QoS) mechanisms try to adapt the allocation of the network resources to the demands of services running over this network. The goal is to provide each service with the required network resources given the possibly limited overall available resources. Unfortunately, these services have different Quality of Service requirements, such as latency, jitter, error-rate, throughput, and redundancy. It is difficult to design a system that addresses all requirements. Two standardized frameworks attempt to overcome these problems [1] in the Internet. IntServ [2] was the first framework to provide network QoS. With IntServ, the network is able to provide end-to-end QoS connections per flow. However, the computational effort for establishing these end-to-end connections is huge. As a result, the scalability of this framework is poor. The next step in the standardization process was DiffServ [3]. This framework is designed for scalability but has low granularity in flow control. Furthermore, the implementation of DiffServ is highly proprietary, leading to difficulties in its management [4].

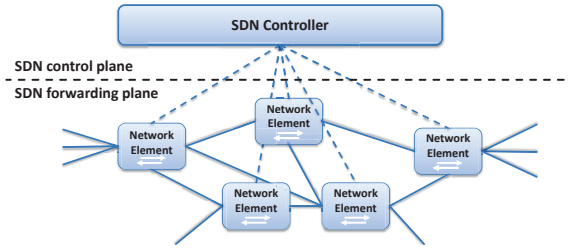

Fig. 1. The concept of network function split in Software Defined Networking

Software Defined Networking (SDN) is a new paradigm to increase the flexibility of networks. It introduces a standardized interface to access the forwarding plane of network nodes remotely. The OpenFlow protocol [5] is one candidate standard technology realizing SDN. The principle idea of SDN is to split a router or network switch into a forwarding plane and a control plane (Figure 1). The forwarding plane is located inside the network and is able to provide basic forwarding functionality such as classification, marking, metering and scheduling. The control plane is located on one or more servers and manages the forwarding plane remotely. This function split changes the way in which network functionality can be designed. In legacy networks, all algorithms have a decentralized character. With SDN, control functions have a centralized view and can be implemented in software in a centralized manner. In summary, the key features of SDN are flexibility, a centralized view and programmability [5], [6], [7].

SDN could be used to realize various QoS scenarios. Policy Cop [8], for example, implements Intserv and DiffServ like functionality using the SDN paradigm. This functionality is managed by a policy-based management system. VSDN [9] and other QoS routing frameworks [10] follow a centralized routing approach to improve video experience in a network. The disadvantage of the current state of the art is that these new SDN applications implement only those QoS management functions that are already available. These projects strongly correspond to the IntServ and DiffServ approach, with the advantage of having a standardized interface to the network hardware. However, the flexibility of SDN is not fully exploited.

To better demonstrate the flexibility of SDN we focus on a use-case that is not solvable with standard network hardware. We believe that it is possible to implement a deterministic real-time communication system with SDN. This type of real-

978-1-4799-2730-2/14/$31.00 ©2014 IEEE

time communication is important in order to realize real-time applications, for example online gaming and e-learning [11]. We put the focus on network domains that are limited in size such as data centers or enterprise networks, in particular. To run such a real-time application properly, it is necessary to control the communication channel and the servers [1]. Industrial Ethernet is the latest communication technology for connecting real-time critical automation devices. The ideas behind this technology can be easily transferred into other use cases such as real-time services in clouds and data centers. There are two big challenges in real-time communication. The first issue is that of simultaneousness, meaning that all devices have to be time-synchronized to fulfill their task. The second challenge is timeliness; all information that is important for the system has to be transmitted before a particular deadline. One of the main drivers of the Industrial Ethernet is to replace proprietary real-time communication solutions with standard Ethernet. Unfortunately, standard Ethernet mechanisms do not fulfill all the requirements of a real-time communication system. This has lead to the development of many different proprietary Ethernet derivates [12].

It is the objective of this work to build a QoS concept for real-time applications exploiting the benefits provided by SDN. The centralized view of SDN allows for a deterministic end-to-end QoS planning. The main part of this paper, introduces a network calculus based end-to-end traffic model to support a centralized QoS resource allocation planning. Our concept can be used for access control for real-time applications. We assess the performance of this model through a simulation based evaluation for a given network scenario.

II. State of the Art

PolicyCop [8] is a system that can provide aggregated and per flow guarantees, implementing a policy based flow management system. PolicyCop also implements measurement functionality to validate the status of the network. The QoS indicators of each forwarding device are determined by analyzing its statistical information, but the accuracy of the QoS measures provided is not mentioned. If a flow violates a policy, the system will determine this on the basis of these measurements and react automatically with configured actions. The reaction time mentioned by the author is of the order of several seconds. This control loop cycle is too slow for real-time requirements.

Other approaches tackle the case of dynamic video streaming, using the flexibility of SDN to improve the user experience of video content [9]. They introduce a use case specific network interface. This interface allows the streaming application to request QoS parameters for its video stream. This application also needs a control loop to regulate the network flows. In addition, the framework needs to modify the Openflow protocol to provide service guarantees.

The reaction time of a closed control loop would appear too slow to realize a real-time communication system. The alternative is a deterministic network model to implement access control from a centralized control point. Network calculus [13] provides a mathematical framework for calculating deterministic maximum delay bounds of a network. This mathematical model is commonly used for calculating the timeliness of real-time systems [14], [15], [16]. However it is used for dimensioning a real-time communication system and not to establish connections on run time. We propose to use the network calculus to realize joint access control and path planning at run time.

Schmitt [17] provides in his paper a method to use the deterministic network calculus to implement a simple local access control algorithm for an internet router. To achieve this goal, a strict priority scheduling mechanism is used because it has no configuration effort. This mechanism is modeled with network calculus. By transformation of the model, the author creates a very simple access control algorithm. However, he does not specify how to use this mechanism in a multi-hop network. The present paper therefore extends this model for a multi-hop network.

III. Problem and Contribution

Commercial real-time Ethernet solutions provide real-time performance by non-Ethernet standard extension such as TDMA medium access strategies. However, TDMA systems put a lot of burden in terms of implementational and computational complexity on the end devices of a network. Our approach is to provide real-time performance through network elements rather than end devices. The real-time performance of standard Ethernet or IP technology in general is poor. QoS framework such as DiffServ and IntServ provide some solution, but still lack either in scalability or granularity being based on distributed algorithms. In this paper, we exploit the flexibility provided by the centralized view for network control through Software Defined Networking to support real-time QoS provisioning.

The contributions of this paper are the following:

- Concept for end-to-end QoS path planning based on a centralized view (SDN) and a flexible flow to priority queue allocation planning for more efficient access control,
- Deterministic network model for delay bounds based on the network calculus,
- Extension of the model for multi-hop paths,
- Evaluation of the model for QoS path planning implemented by a Mixed Integer Program based on simulation.

IV. Model for real-time QoS based on SDN

To realize end-to-end real-time QoS with SDN it is necessary to define a model for all forwarding elements of a given network. The main task of this model is to create an abstraction layer between SDN hardware and real-time applications. This abstraction should allow the development of real-time applications, independent of the given hardware of the network. The SDN hardware capabilities can be queue length (buffer size), the number of queues at the links, the scheduling algorithm used, the available data rates and the topology. Instead of providing these values to a real-time application, the application should simply specify their demands and characteristics in terms of maximum delay, mean data rate and burstiness. The abstraction layer should use

this information and the knowledge of the network to decide how best to fulfill the demands. This abstraction layer can be provided by an SDN controller with a logically centralized view of the network.

For the realization of our scenario we use deterministic network calculus [13], calculating the maximum delay bound per hop. The goal of this model is threefold: (1) easy to calculate admission control, (2) use of multiple queues, and (3) to make the calculation independent of the influence of high priority queues on low priority queues. The latter independence is important because the deadlines of low priority queues would otherwise have to be recalculated every time traffic is added to a high priority queue. As a scheduling algorithm, we decided to use static priority queuing because it does not need configuration. This allows the dynamic usage by an SDN controller. In addition, it is possible to achieve a broad range of possible maximum delay bounds. We assume that the forwarding time of each forwarding element is negligible.

The network will be represented by a directed graph $G(N, L)$ with nodes $n \in N$ and links $l \in L$. These links are characterized by their origin node $a_l \in N$, their destination node $b_l \in N$ and their maximum capacity CAP_{max_l}. Additionally, a link contains Q_l queues, where queue 0 is the queue with highest priority.

$$l : [a_l, b_l, CAP_{max_l}, Q_l] \tag{1}$$

The network can carry flows $f \in F$. These are characterized by their origin $o_f \in N$, destination $d_f \in N$, and their service class $c_f \in C$. A QoS-aware SDN application should request this service at the controller and implement the admission control.

$$f : [o_f, d_f, c_f] \tag{2}$$

The service classes $c \in C$ contain the parameters of a traffic model. In our case, we choose the token bucket representation of the traffic, characterized by the average data rate r_f and burstiness b_c. This is a common characterization in traffic models that use the network calculus [13] to calculate the backlog and the maximum delay of a given network. An additional parameter is the end-to-end maximum deadline t_c; this deadline must be fulfilled inside the network.

$$c : [r_c, b_c, t_c] \tag{3}$$

There is a set of paths $p_f \in P$ from origin $o_f \in N$ to destination $d_f \in N$ for each flow $f \in F$ that associates (see Figure 2):

$$p_f = \left\{ p_f^j \right\} \tag{4}$$

p_f^j describes a specific path of the set p_f. PC contains the amount of paths that are chosen to carry a service.

$$PC = \left\{ p_f^j \right\} \tag{5}$$

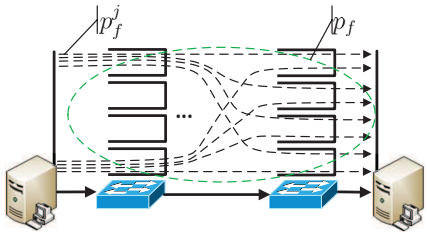

Fig. 2. Definition set of paths p_f

Each of these paths p_f^j has an associated binary matrix $M_f^j[Q_l, L]$ which indicates whether p_f^j uses queue q at each link l. That is, for the path p_f^j:

$$M_f^j[q, l] = \begin{cases} 1 & \text{queue } q \text{ in link } l \text{ is used by path } p_f^j \\ 0 & \text{else} \end{cases} \tag{6}$$

A. Network Calculus for access control in multi-hop networks

Schmitt [17] introduces an equation to calculate the maximum scheduling delay of priority queuing at each queue. It is straightforward to extend this formulation to a multi-hop scenario by considering the delay of every queue of every link (see equation 7). By assigning a fixed virtual data rate to each queue the maximum delay of each queue can be calculated. These delay values can be used by flows, which consume the available data rate and buffer space of the chosen queue. B_{lk} specifies the available buffer size for queue k at link l, b_{max} is the maximum size of Ethernet packets (typically 1500 byte), CAP_{max_l} is the maximum capacity of link l and R_{lk} is the data rate of queue k at link l. The parameter b_{max} takes into account that it is always possible that, on transmission of a high-priority packet, a packet of lower priority may already be in transmission. This transmission cannot be interrupted. The goal of this formulation is to simplify the management of each flow, with the delay T_{lq} being defined as a static value. To achieve this, the maximum data rates R_{lq} of each queue have to be fixed.

$$T_{lq} = \frac{\sum_{k=1}^{q} B_{lk} + b_{max}}{CAP_{max_l} - \sum_{k=1}^{q-1} R_{lk}} \tag{7}$$

To add a new flow (described by the following parameter r_{new}, b_{new} and t_{new}) to a link, Equation 8 checks the availability of the required data rate at each queue and link. This equation is a multi-hop extension of Schmitt [17].

$$R_{lq} \geq \sum_{p_f^j \in PC} M_f^j[q, l] r_c + r_{new} \tag{8}$$

The available buffer size is expressed by Equation 9. Therefore, the sum of all bursts must be smaller or equal to the buffer size of the queue (B_{lq}).

$$B_{lq} \geq \sum_{p_f^j \in PC} M_f^j[q,l] b_c + b_{new} \quad (9)$$

Finally, we have to consider whether the timing constraint (Equation 10) of their service class is fulfilled. To check the real-time constraint (t_{new}), the sum of all precalculated delays of each used queue must be smaller than the constraint.

$$t_{new} \geq \sum_{l \in L} \sum_{0 \leq q \leq Q_l} M_f^j[q,l] T_{lq} \quad (10)$$

The advantage of this formulation is the split of end-to-end real-time QoS routing into two sub problems. The first problem is to reserve the right amount of data-rate R_{lq} for every queue. This problem is not addressed into detail in this paper. The second problem is the QoS routing/access control problem which is formulated in the following. A flow can only be embedded if the constraints of Equation 8, 9 and 10 are fulfilled. This approach has the advantage that the calculation effort is small; the resource allocation can be achieved by simply summing the data rates and buffer spaces consumed by a flow and denying the acceptance of a new flow if no resources are left. The advantage of this formulation is that new flows can not lead to a violation of the QoS parameters of already embedded flows.

On the other hand, this approach is conservative because it is assumed that bursts from all services arrive at once at one network node, even when services share the link which leads to this node. To overcome this difficulty, termed the "following station problem" [17], it is necessary to extend this model. This can be done by calculating the necessary buffer size in a more accurate way.

B. Extension of the Network Calculus based access control

One opportunity to extend the above model to better take multi-hop characteristics into account could be to use deterministic network calculus to take the arrival rate of the flows into account. To achieve this, we have to define the input sets of a forwarding element. The set $cs_{nlq} \subseteq F$ contains all the flows f which are carried by each input link n to a queue q at link l. The set $N_{lq} \subseteq L$ contains all the input links n which leads to a queue q at link l.

We introduce a separate service curve for every queue at every link ($S_{lq}(t)$). Equation 11 is derived from Equation 7 in a way that every queue can only consume the data rate which is not allocated to higher priority queues. We assume that the forwarding delay of the forwarding element is negligible.

$$S_{lq}(t) = \left(CAP_{max_l} - \sum_{k=1}^{q-1} R_{lk} \right) t \quad (11)$$

The incoming link model introduced, $L_{cs_{nlq}}(t)$, limits the maximum possible burst to its maximum capacity CAP_{max_l}. Therefore, all arrival curves ($G_f(t)$) of the flows entering the queue over a specific link are summed.

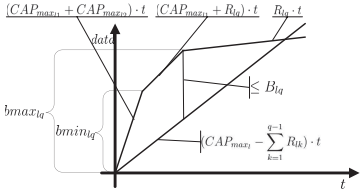

Fig. 3. Consider the incoming link capacity for the calculation of the buffer utilization

$$L_{cs_{nlq}}(t) = \sum_{s \in cs_{nlq}} min\left(G_f(t), CAP_{max_n} t\right) \quad (12)$$

The arrival curve $G_f(t)$ describes the token bucket characteristics of each flow.

$$G_f(t) = r_f t + b_f \quad (13)$$

Equation 14 shows the network calculus approach to determine the maximum buffer space needed to transmit without any loss. This value has to be smaller than or equal to B_{lq}:

$$B_{lq} \geq \sup_{t>0} \left(\sum_{n \in N_{lq}} L_{cs_{nlq}}(t) - S_{lq}(t) \right) \quad (14)$$

We are now able to calculate the maximum possible buffer space consumption for this more complex definition of a single queue at a link. The deterministic network calculus provides us with fixed maximum delay bounds. This model can be used to realize hard real-time conditions inside the network by simply taking more accurate account of the topological characteristics. This mathematical description can be expressed by its geometrical representation.

Equation 15 calculates the sum over the burstiness of the flows which come from input link n to the queue q of the link l.

$$b_{nlq} = \sum_{f \in cs_{nlq}} b_f \quad (15)$$

Equation 16 calculates the sum over the data rate of the flows which come from a input link n to the queue q of the link l.

$$r_{nlq} = \sum_{f \in cs_{nlq}} r_f \quad (16)$$

Equation 17 calculates the maximum burst time $tmax_{lq}$ for queue q of link l.

$$tm_{lq} = \max_{n \in N_{lq}} \frac{b_{nlq}}{CAP_{max_n} - r_{nlq}} \quad (17)$$

Figure 3 shows this representation for a network element with two input links. It is straightforward to see that under the

condition $CAP_{max_n} \leq CAP_{max_l} | n \in N_{lq}$ the max backlog is on the end of the last burst tm_{lq}. Together with Equation 17 the new access control scheme for the buffer space sums up the arrival curves of each incoming link $L_{cs_{nlq}}$ minus the service curve of the output queue S_{lq} at the end of the last burst tm_{lq}. This value has to be summed with b_{new} smaller equal B_{lq}.

$$B_{lq} \geq \sum_{n \in N_{lq}} L_{cs_{nlq}}(tm_{lq}) - S_{lq}(tm_{lq}) + b_{new} \tag{18}$$

The access control conditions for the extended access control scheme are Equation 8, 18 and 10. This access control scheme enables a better usage of the available buffer space by considering the incoming link rate for burst calculation.

C. Model summary

We have extended the calculation method of [17] for access control for a single aggregation network node to a multi-hop scenario. With this multi-hop access control we are able to calculate maximum end-to-end delay bounds for specific flows. The admission control functionality of the model has been improved by provisioning of the incoming link rate to achieve a higher number of flows.

V. Concept for QoS Path Planning and Evaluation

Goal of this evaluation is to show the principle gain that can be reached with our network calculus model. Therefore, we have implemented the end-to-end real-time QoS path planning with a greedy algorithm and a Mixed Integer Program (MIP) and show the comparison of both approaches. The greedy algorithm represents the behavior of a simple state of the art real-time QoS reservation concept with a fixed flow to queue allocation. The MIP shows the possible gains that can be achieved with our new SDN-based apprach with flexible flow to queue allocation.

A. Greedy algorithm (GA)

The greedy algorithm is a simple implementation of the described model. Depending on their deadline, the service classes are fixed to the queues from the beginning. Service class 1 has the highest priority queue and service class 4 the lowest. Based on this fixed allocation of the priority classes, the algorithm itself makes no routing decisions. The algorithm simply maximizes the amount of flows until one of the access control conditions is violated. The required buffer space, the link utilization and the end-to-end delays are therefore calculated. This calculation is used to search for the maximum number of flows where all border conditions are fulfilled. This is an approach which is common in traditional networks. This simple strategy is a result of the absence of centralized planning in today's networks.

B. Mixed Integer Program (MIP)

The potential gains of our model can be explored by means of a Mixed Integer Program as a first approach. We expect significant gains to be achieved through the joint planning of allocation of flows to priority queues and routing based on link capacities, buffer space and timing demands. This can be achieved through the central view of an SDN controller. The MIP has the ability to use the full path set p_f to find a path for a flow. This approach should lead to a higher utilization of the given network resources. MIP is expensive in terms of computational resources. However at this stage of our research, our aim is to provide a benchmark for future metrics.

TABLE I. Pattern table for the Mixed Integer Program to solve the reservation problem of R_{lq}

	in MByte/s				in ms			
Pattern	R_{l1}	R_{l2}	R_{l3}	R_{l4}	T_{l1}	T_{l2}	T_{l3}	T_{l4}
1	10	10	10	95	0.73	1.58	2.59	3.81
2	15	15	15	80	0.73	1.65	2.86	4.52
3	20	20	20	65	0.73	1.73	3.19	5.56
4	25	25	25	50	0.73	1.82	3.62	7.23
5	30	30	30	35	0.73	1.91	4.18	10.33
6	35	35	35	20	0.73	2.02	4.94	18.08
7	40	40	40	5	0.73	2.14	6.03	72.30
8	40	30	20	35	0.73	2.14	4.94	10.33
9	35	25	15	50	0.73	2.02	4.18	7.23
10	30	20	10	65	0.73	1.91	3.62	5.56

Unfortunately the the formulation as a MIP limits the possible mathematical formulation. So Equation 7 cannot be easily implemented in a Mixed Integer Program (MIP). The value of R_{lq} must be dynamic to find the best distribution of the total available data rate CAP_{max_l} for the different priority levels. The problem is that this dynamics needs to implement a division in a MIP. This can be done with a linear regression or with a lookup table. We decide to replace this equation by an lookup table (see Table I). This table is created with the assumption that the available buffer space in every queue is $B_{lq} = 90000byte$. This setting is common in state of the art SDN hardware. In addition we assume a homogeneous gigabit network which leads to $CAP_{max_l} = 125000000byte/s$ in every link. The value combinations of R_{lq} are chosen from previous experience and have not been tested for optimality. Given the value combination of R_{lq}, the value combination of T_{lq} is calculated using equation 7. The delay values could be calculated in the precalculation phase of optimization. During the optimization phase the MIP can choose one of these precalculated patterns.

Also the extended formulation (subsection IV-B) of the multi-hop model can only be implemented with some limitations. Figure 3 shows this representation for a network element with two input links (bi_{l1q},bi_{l2q}). For simplification we assume the term $(CAP_{max_{l1}} + R_{lq}) * t$ instead of the more accurate formulation $(CAP_{max_{l1}} + r_{2q}) * t$. This simplification enables us to formulate the problem of Equation 14 as a Mixed Integer Program. For simplification we consider only two input links (see Figure 3). In a next step, we consider the maximum $bmax_{lq}$ and the minimum burstiness $bmin_{lq}$.

$$bmax_{lq} = max(b_{1lq}, b_{2lq}) \tag{19}$$

$$bmin_{lq} = min(b_{1lq}, b_{2lq}) \tag{20}$$

For reasons of clarity in Equation 21, we calculate the serve rate sr_{lq} of the queue of a link:

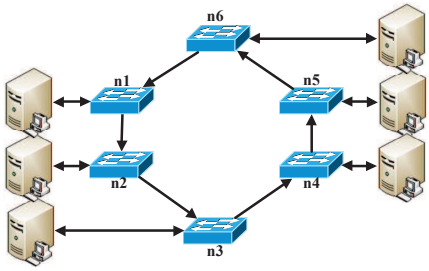

Fig. 4. Unidirectional ring topology for evaluation of the formulation of the end-to-end real-time QoS models

TABLE II. SERVICE CLASSES FOR THE EVALUATION

Name	r_c	b_c	t_c
$c = 1$	$10kByte/s$	$100Byte$	$5ms$
$c = 2$	$10kByte/s$	$100Byte$	$10ms$
$c = 3$	$10kByte/s$	$100Byte$	$20ms$
$c = 4$	$10kByte/s$	$100Byte$	$50ms$

$$sr_{lq} = CAP_{max_l} - \sum_{k=1}^{q-1} R_{lk} \tag{21}$$

Equation 22 shows the mathematical description of Figure 3:

$$\begin{aligned} B_{lq} \geq\ & bmin_{lq} - \frac{bmin_{lq} \cdot sr_{lq}}{CAP_{max_l}} + bmax_{lq} \\ & - \frac{sr_{lq}}{CAP_{max_l} + R_{lq}} \cdot (bmax_{lq} - bmin_{lq}) \end{aligned} \tag{22}$$

This access control equation 22 replaces the general equation 18. The necessary simplifications leads to a more conservative result.

C. Evaluation Scenario

For validation of our model for a real-time enabled SDN we compare the above GA with the MIP solution based on our network model. We introduce the following scenario. We analyze a six-node unidirectional ring structure (see Figure 4). Each node adds a traffic mix to the ring. Each port has four output queues combined with priority scheduling. The maximum hop length in the system is three and follows the following distribution:

- 70% of the traffic will be transmitted over one hop
- 20% of the traffic will be transmitted over two hops
- 10% of the traffic will be transmitted over three hops

This traffic mix contains four service classes shown in Table II. The distribution of the four traffic classes is varied in a way that all classes have to count at least five percent of the traffic. Traffic is varied in steps of five percent. The resulting set of traffic mixes forms the basis of the following calculation.

D. Evaluation Results

We compare four cases plotted as cumulative distribution functions (see Figure 5). The greedy algorithm (GA) is applied with the model of Schmitt (see Equation 9) and with the previously outlined extension (see Equation 18). Likewise, the Mixed Integer Program (MIP) is applied with the approach of Schmitt and with the extension. For the comparison, we focus at four values:

- The **total number of flows** is defined as the number of flows all services carry together. (Figure 5a)
- The **average link-rate utilization** is defined as the mean utilization of all the links of the topology. (Figure 5b)
- The **average buffer utilization** is defined as the mean utilization of all the buffers of the topology. (Figure 5c)
- The **average delay deviation** is defined as the average of the delay deviation for a specific traffic pattern. The delay deviation itself is the deviation of the planed delay form the maximum delay t_c in percent. (Figure 5d)

Figure 5a and 5b shows that the MIP scenarios handle much more traffic than the GA scenarios. The increase of the number of flows with both access control schemes is more than 100%. The maximum utilization reached is around 65%. This result shows that our real-time model provides huge benefits in terms of higher utilization of the overall topology. This improvement is reached without losing the deterministic characteristic of each flow.

Figure 5c shows in addition that Schmitt's access control scheme fully utilizes the buffers of the topology in the MIP case. In this scenario, this means that the buffer space is a bottleneck. Even our extended access control scheme reaches a high buffer utilization of over 80%. This behavior is a combination of two effects. On the one hand, there is an error caused by the quantization of Equation 7 through the pattern table (see Table I). On the other hand, there is a trade off between the buffer utilization, the link utilization and maximum deadline condition. This trade off could lead to the situation that the MIP optimization cannot find a pattern from Table I that better meets the requirements.

Figure 5d shows the average delay deviation of the topology. An average delay deviation of zero means that every service equals exactly its maximum delay. Under this condition Figure 5d shows that even the MIP cannot decrease the average delay deviation beyond 50%. This means that on average every service class will be handled 50% faster than is necessary. This behavior could be interpreted as meaning that the topology is not highly "utilized" in terms of delay deviation.

The average link-rate utilization, average buffer utilization and the average delay deviation are the three fundamental performance parameters, providing a good overview over the utilization of the topology. These performance values give an overview over bottleneck resources and can help to evaluate end-to-end real-time scheduling.

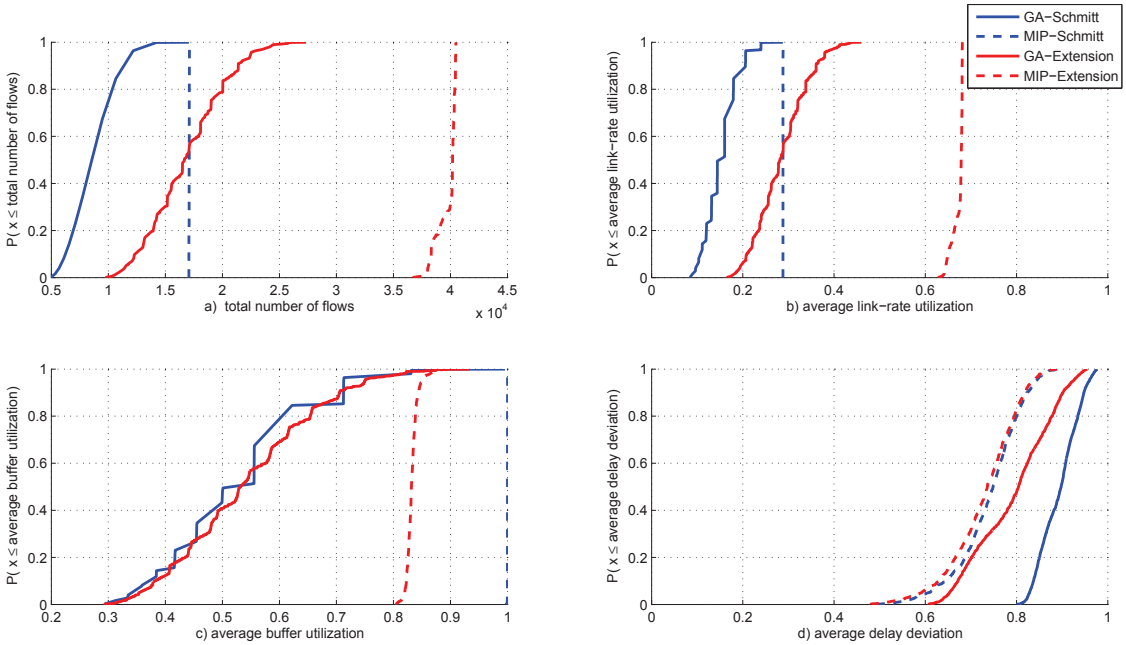

Fig. 5. Utilization of the topology (total number of flows, average link-rate utilization, average buffer utilization, average delay deviation)

VI. Conclusion

We propose a model for deterministic end-to-end real-time QoS for the realization of a centrally planned real-time communication service. This service implements a joint routing and access control system based on SDN leveraging the centralized view that its controller has on the network. Our model allows an admission control algorithm to be realized on the SDN controller. The path planning is carried out with a Mixed Integer Program to find an optimal solution. A overall link utilization of over 60% was possible in our validation scenario. This value was reached under fulfillment of the buffer, data-rate, and maximum deadline constraints. We achieved this high utilization with our extension of an existing network calculus based access control algorithm. The Mixed Integer Program might be too resource consuming for the realization of an online routing algorithm. However, it can be used as a benchmark for the efficiency of an end-to-end real-time QoS routing algorithm. In addition to the routing problem, a strategy needs to be developed to find a optimal pattern set.

References

[1] W. Zhao, D. Olshefski, and H. G. Schulzrinne, "Internet quality of service: An overview," 2000.

[2] R. Braden, D. Clark, and S. Shenker, "RFC1633: Integrated services in the internet architecture: an overview, june 1994," *Status: Informational*, 1994.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "RFC2475: An architecture for differentiated services," *An Architecture for Differentiated Services*, 1998.

[4] M. Welzl and M. Muhlhauser, "Scalability and quality of service: a trade-off?" *Communications Magazine, IEEE*, vol. 41, no. 6, pp. 32–36, 2003.

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

[6] S. J. Vaughan-Nichols, "Openflow: The next generation of the network?" *Computer*, vol. 44, no. 8, pp. 13–15, 2011.

[7] M. Jarschel, T. Zinner, T. Hossfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, attributes, and use cases: A compass for sdn," *Communications Magazine, IEEE*, vol. 52, no. 6, pp. 210–217, 2014.

[8] 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.

[9] 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.

[10] H. E. Egilmez, B. Gorkemli, A. M. Tekalp, and S. Civanlar, "Scalable video streaming over openflow networks: An optimization framework for qos routing," in *18th International Conference on Image Processing (ICIP)*. IEEE, 2011, pp. 2241–2244.

[11] 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.

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

[13] J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. Springer, 2001, vol. 2050.

[14] J. Loeser and H. Haertig, "Low-latency hard real-time communication over switched ethernet," in *16th Euromicro Conference on Real-Time Systems*. IEEE, 2004, pp. 13–22.

[15] 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.

[16] T. Skeie, S. Johannessen, and O. Holmeide, "Timeliness of real-time ip communication in switched industrial ethernet networks," *Transactions on Industrial Informatics, IEEE*, vol. 2, no. 1, pp. 25–39, 2006.

[17] 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.