Dynamic Application-Aware Resource Management Using Software-Defined Networking: Implementation Prospects and Challenges

Thomas Zinner*, Michael Jarschel*, Andreas Blenk§, Florian Wamser*, Wolfgang Kellerer§
*Institute of Computer Science University of Würzburg, Germany.
Email: {zinner,michael.jarschel,florian.wamser}@informatik.uni-wuerzburg.de
§ Institute for Communication Networks, Technische Universität München, Germany.
Email: {andreas.blenk,wolfgang.kellerer}@tum.de

***Abstract*—Today's Internet does not provide an exchange of information between applications and networks, which may result in poor application performance. Concepts such as application-aware networking or network-aware application programming try to overcome these limitations. The introduction of Software-Defined Networking (SDN) opens a path towards the realization of an enhanced interaction between networks and applications. Hence, a more dynamic and demand-based allocation of network resources to heterogeneous applications can be realized. The implementation of the resource management action, however, may have an impact on the data transport and application quality. This paper summarizes resource management mechanisms provided by current SDN approaches based on OpenFlow and exemplary evaluates implementation prospects and challenges.**

***Keywords*-Application-Aware Networking, QoE-Aware Networking, SDN, Resource Management**

I. Introduction

Today's Internet does not feature a direct information exchange between applications and the network. Due to the varying requirements of the heterogeneous applications running on top of the network, this may result in a poor user-perceived quality of applications in certain situations. To overcome this limitation, applications must become network-aware, and networks must become application-aware.

Application-aware networking is traditionally realized using special purpose monitoring systems or network elements. For example, packets are classified within the ISP's network and processed according to the application requirements as specified in generic policies [1]. Other options are specialized entities like media-aware network elements [2]. Thus, the network is able to adjust application quality and optimize the allocation of the available network resources for a large number of heterogeneous and among themselves competing applications.

Network-aware applications measure the network performance and act based on the results. Typical examples of such applications are Skype [3] or MPEG-DASH [4], which perform adjustments on the application layer to overcome network problems, e.g., by adapting the used video codec or video quality, or, for Skype, by performing re-routing on the application layer. Each network-aware application tries to maximize their quality based on the available resources, but may also behave egoistically trying to supersede competing traffic and, thus, to increase its share of available resources.

The introduction of Software-Defined Networking (SDN) [5] opens a path towards the realization of an enhanced interaction between networks and applications. By introducing an external and programmable network control plane, SDN creates a flexible, adaptable, and open interface to the network data plane. Via the "Northbound-API" [6] applications may provide information to the network control plane, which then can control the data plane. In particular, such an application-aware control plane can be leveraged to augment the network control to improve the user perceived Quality-of-Experience (QoE) by changing, e.g., the forwarding behavior of switches, or by allocating more network resources to specific flows. Such management actions are possible on short time scales and in small networks on a per-flow basis.

A more dynamic and demand-based allocation of network resources to heterogeneous applications is seen as one possibility to achieve a network-wide QoE fairness [7], [8]. The implementation of dynamic resource management actions, however, may influence the behavior of the data transport and the application quality.

The contribution of this paper is twofold. First, we summarize different resource management mechanisms provided by current SDN approaches based on the OpenFlow protocol, version 1.0 to 1.4. Then, we evaluate the impact of dynamic resource allocation on a per-flow basis for an access network with two applications competing for limited network resources. We show the potential to increase the QoE of the involved applications and highlight possible side-effects affecting the end-to-end performance.

The rest of the paper is organized as follows. We discuss the background of this paper and related work in Section II.

This work has been performed in the framework of the CELTIC EUREKA project SASER-SIEGFRIED (Project ID CPP2011/2-5), and it is partly funded by the BMBF (Project ID 16BP12308). The authors alone are responsible for the content of the paper.

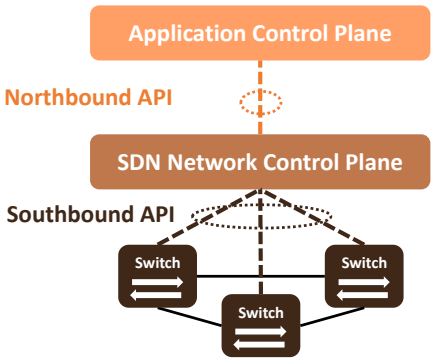


Fig. 1. SDN Interfaces

Section III highlights the investigated scenario and provides details on the conducted experiments. The results of our measurement study are presented in Section IV. Finally, the paper is concluded in Section V.

II. Background and Related Work

In this section, we give an introduction to Software-Defined Networking and discuss previous works on application-aware networking.

A. Introduction to Software-Defined Networking

The goal of SDN is to increase flexibility and innovation in the network and, thus, to improve the efficiency of network operation and the service quality as well as lead to reduce of CAPEX and OPEX. This is facilitated by the removal of the network control plane from the distributed network devices to a logically-centralized control entity, which enables the introduction of new open interfaces between the application, the data-plane, and the control plane [6]. With these interfaces, the network control plane can be realized as a freely programmable software, which can essentially be described as an operating system for the network. The network operating system, often called ”controller”, is responsible for all forwarding decisions within the network it controls. The network devices forward the traffic according to the rules set by the controller.

Figure 1 illustrates the relationship of involved control planes and interfaces. The ”Southbound-API” represents the interface between data- and control-plane. Current SDN implementations often use the OpenFlow protocol [9] as a realization of this interface. The OpenFlow protocol handles the communication between the individual network devices and the controller. Each of the network devices maintains a set of ”flow rules” matching individual network flows in so called ”flow tables”. The term ”flow” refers in this context to packets matching a general set of header fields either out of layers 2 to 4 of the ISO/OSI stack or headers defined by the operator of the network. Additionally, a flow rule contains a set of one or more actions that define how a packet matching the rule should be handled as well as flow statistics.

When a packet reaches an OpenFlow-enabled SDN switch, it is buffered and the packet header is checked against the rules in the flow table. In case of a successful match, the action(s) specified in the rule are executed. If there is no matching rule in the flow tables, the packet is either dropped or an OpenFlow ”packet-in” message containing the packet header is sent to the controller for processing. The controller calculates the action the network element should take with regard to the packet and communicates it. Furthermore, the controller can specify a flow rule and send it to the network element(s). This way all following packets of the flow are treated the same way by the network and the controller does not need to be involved any longer.

The controller can also introduce new flow rules or modify existing ones without being triggered by an incoming packet. For example, the controller may adhere to a pre-programmed schedule or implement a network policy. This is where the flexibility of SDN comes into play. Where traditional network devices would have to be reconfigured by an administrator, SDN enables the automatic and seamless implementation of changes in the forwarding behavior of the network. These changes can be triggered by external entities via the other key SDN interface - the ”Northbound-API” [6]. This interface makes application-awareness in the network feasible as it opens up a communication channel between the applications using the network and the controller, which can then utilize information provided by the applications to adapt its policy and the network traffic on different levels of granularity.

B. Previous Works on Application-Aware Networking

In [10], Wamser et al. describe a method to leverage application information from YouTube video streaming to enhance the quality of this particular application in an access network. They use an external entity called ”network advisor” to adapt the forwarding inside the network. Jarschel et al. [11] implement this concept using an OpenFlow-based SDN system, which yields similar results but simplifies the implementation significantly.

Google [12] exploits its knowledge about the applications running inside its global data center backbone to optimize and schedule the bandwidth usage inside the network with a centralized SDN-based traffic engineering system. The applications are categorized into priority classes according to their importance. In case of an overload situation, e.g. due to a failure, low priority packets are discarded. This way Google can maintain a bandwidth utilization on the data center interconnection links close to 100%.

Das et al. [13] show how the routing of aggregates can be improved using QoS parameters of applications in conjunction with an SDN approach. In [14], Jeong et al. introduce a QoS-aware extended Network Operating System (NOS) for Software Defined Networks. It is based on the IETF ForCES vision of SDN and leverages a variety of legacy networking protocols as well as virtualization techniques like MPLS. Egilmez et al. present a framework for enhancing OpenFlow networks to dynamically reroute QoS flows for scalable video

stream in [15]. However, none of these approaches take the effects of dynamic rerouting on the behavior of individual TCP flows into account.

In [16] Jarschel et al. demonstrate that in a data center environment it is possible to maintain the service quality of a video stream during the live migration of the virtual machine hosting the service by using the SDN Northbound-API to receive an advance notification for the migration from the cloud management system.

Georgopoulos [7] et al. introduce an SDN-based framework for QoE-guided fair scheduling of traffic in networks with limited resources and evaluate it using video as sample application.

C. OpenFlow Resource Management Capabilities

There are several ways to manage resources in an OpenFlow-based SDN as shown in Table I. The most flexible way are the per-flow meters introduced with OpenFlow 1.3. They allow the assignment of a fixed rate limit to individual flows. This enables a granular adjustment of individual flows inter- and intra-application. However, a lot of state information is required to maintain a queue for each individual flow. In particular, if the number of flows is large. Therefore, the implementation of this feature in hardware is still an open challenge.

The second way is the OpenFlow priority queue feature introduced with OpenFlow 1.0. It allows the assignment of flows to a forwarding queue with certain QoS parameters and queuing disciplines, e.g., a minimal forwarding rate or weighted fair queuing. The implementation and configuration of these queues is outside the scope of OpenFlow itself and therefore depends on the device or software. Here, it is no longer possible to discern between individual flows but only classes of flows that are assigned to the same queue.

The simplest method is the redirection of traffic away from one link to another in case multiple paths exists. No resource management is enforced on the links and therefore all flows are treated equally. The assignment of flows to the links is done according to the controller's policy, e.g. by assigning a class of flows to a dedicated link.

All methods have in common that the resource management for each flow can be changed by the controller network-wide, i.e., for heterogeneous hardware also from different vendors, on the fly. E.g. the assignment of a link, traffic class, or flow rate is not fixed and can be adapted live depending on the situation. Since flows can be aggregated in arbitrary granularity, SDN is such an attractive candidate to realize application-aware networking.

Method	OF Version	Granularity	Limiting Factor
flow meters	1.3	flow-level	fixed-rate
priority queues	1.0	class-based	bandwidth-share
traffic redirection	any	link-based	link bandwidth

TABLE I
OPENFLOW RESOURCE MANAGEMENT METHODS

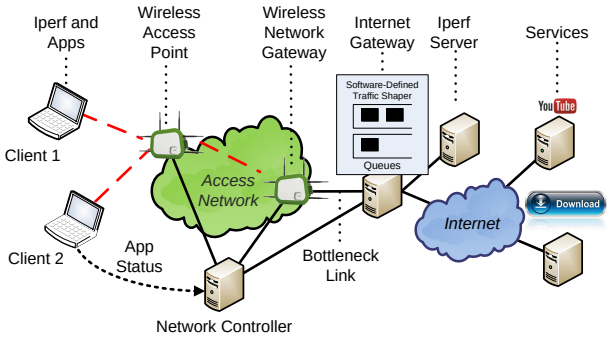


Fig. 2. Measurement Setup

III. APPLICATION-AWARE NETWORKING FOR VIDEO STREAMING

Today's application data is transmitted on a best effort basis. However, applications and users may show a highly dynamic behavior that leads to strongly varying, i.e., dynamic demands for network resources. Based on SDN, we propose a dynamic resource management for a multi-user scenario. In particular, the resource management considers application demands and may enhance the perceived user quality.

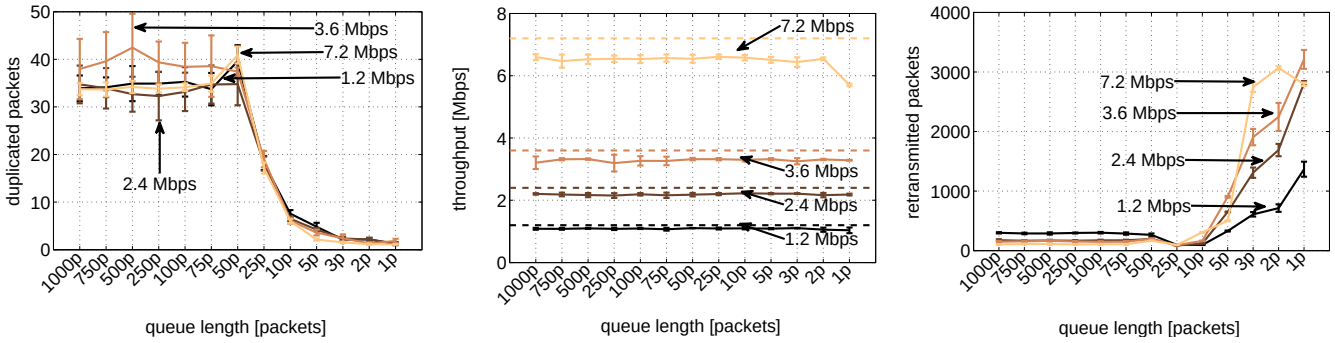
A. Scenario Description

In the investigated scenario, we assume that a large file download and a progressive video streaming application, YouTube, compete for resources on a bottleneck link. YouTube provides a good application quality as long as the playback buffer is sufficiently filled. If the buffer is empty, the video playback is interrupted and stalling occurs. Accordingly, we monitor the buffer state and increase the allocated network resources if necessary. If the buffer is sufficiently filled, i.e., a threshold is exceeded, the same amount of resources is allocated to each flow. We set up a small testbed as described in the next subsection in order to emulate the SDN behavior for the dynamic resource allocation use case.

B. Measurement Setup

All measurements were conducted within the setup that is illustrated in Figure 2. The setup can be split into two parts, the access network and the Internet. The access network consists of our own implementation of a network controller, two clients, two wireless mesh nodes and an Internet Gateway (IGW).

The clients are connected via a multihop wireless network to the Wireless Network Gateway (WNG). WNG is connected via a fixed link with the Internet Gateway (IGW). The controller is connected via wired links to the mesh nodes and IGW. As the wired link between WNG and IGW represents the bottleneck in our scenario, a Software-defined Traffic Shaper (SDTS) is installed on IGW and controls the uplink to WNG. Additionally, the rate of the link between WNG and IGW can be configured.



(a) Number of duplicated packets caused by moving the flow to a queue with higher priority.

(b) Perceived throughput on application layer.

(c) Overall number of packet retransmissions occuring by moving a flow to a higher priority.

Fig. 3. Amount of duplicate packets, amount of retransmissions, and achieved bandwidth of TCP Cubic for 1.2, 2.4, 3.6, and 7.2 Mbps.

In order to evaluate the transport layer performance, both clients are using Iperf [17] to generate TCP traffic. The performance of TCP CUBIC is analyzed as it is the most used TCP version by web servers [18]. In order to evaluate the application performance, both clients can run either a video streaming application, i.e. YouTube, or a file sharing application. Both services are accessed via the Internet. In case of file sharing, files with different sizes can be downloaded from a simple file server. An application monitor is installed on the clients to extract current application information. This information is sent to the controller via the access network. The controller commands IGW to prioritize network flows, which are passing IGW. On IGW, SDTS receives and executes the commands.

C. Evaluated Queuing Strategies and Performance Metrics

In the following, we explain our implementation of a priority queuing approach (PRIO) and a weighted fair queuing approach (WFQ).

1) PRIO: Three queues with descending priority are installed on IGW. If the highest priority queue contains packets, it is always served first. The queues do not have a minimum guaranteed data rate. The queue length, given in packets, is configured before each experiment and is varied between 1 and 1000 packets of a size of 1500 bytes.

2) WFQ: Two queues are installed on the IGW. Each queue has a minimum guaranteed data rate and a maximum data rate. The maximum data rate is given by the ceil parameter. The queue's size is 1000 packets for all measurements. If a queue does not use all allocated resources, the remaining resources are allocated to the remaining queue.

3) Metrics: The analysis of the measurements are based on transport layer metrics and application layer metrics. On the transport layer, throughput, retransmission and duplicated packets are used to evaluate the performance. On the application layer, the buffered playtime of videos is used to investigate the application performance.

IV. Measurement Results

First, we highlight the influence of the proposed resource management mechanisms on the TCP protocol namely the throughput and the occurrence of duplicated or retransmitted packets. A detailed investigation of WFQ showed no noticeable impact on these metrics. Thus, we omit WFQ and discuss PRIO for a single flow in the following. We conduct multiple runs to get statistically significant results. Each run lasts in total 40 seconds. After ten seconds the flow priority is increased, and after twenty seconds it is decreased again. We vary the size of the priority queues between 1 and 1000 packets and investigate its impact on the performance metrics mentioned above. Figure 3 depicts the results for rate limits of 1.2 Mbps, 2.4 Mbps, 3.6 Mbps, and 7.2 Mbps. The results are plotted with 95% confidence intervals. The x-axis of the subfigures depicts the adjusted queue length.

Figure 3(a) depicts the impact of the queue size on the number of duplicated packets. It can be seen that the influence of different rate limits on the average number of duplicated packets is negligible and keeps rather constant for queue sizes larger than 50 packets. With a decreasing queue size, the average amount of duplicated packets decreases and converges to 1 packet. The increased number of duplicated packets for larger queues is introduced by the change of the prioritization queue. Packets get stuck in the lower prioritized queue, are transmitted delayed and lead to out-of-order packet delivery at the destination possibly affecting the application. To minimize the impact of a queue change the queues should be kept as small as possible.

Figure 3(b) shows the impact of different queue sizes on the average throughput. The dashed lines illustrate the theoretical maximum. The average throughput remains constant for all queue size and rate limits despite the link rate of 7.2 Mbps. In this case, the throughput remains constant for queue sizes larger than 2 packets, however, it drops for a queue size of 1 packet. Although the throughput roughly remains constant for smaller queue size, the average number of retransmissions increases, i.e., packet loss occurs. This is depicted in Figure 3(c). For each rate limit, the average amount of retransmissions keeps roughly constant for a queue size bigger than 50 packets. Between 50 packets and 10 packets, the number of retransmissions decrease slightly, however, for queue sizes less than 10 packets the number of retransmissions, i.e the

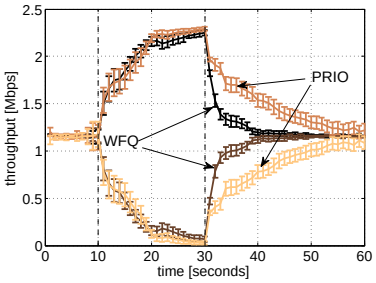
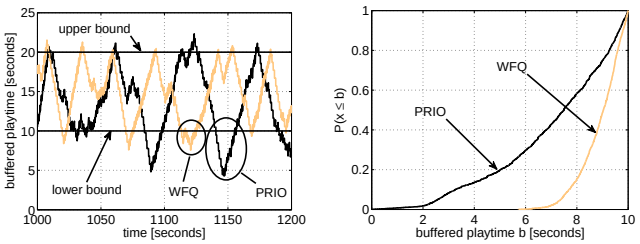


Fig. 4. Comparison of the behavior of two competing flows over time for PRIO and WFQ.



(a) Buffered playtime over time. (b) CDF of the buffered playtime.

Fig. 5. Impact of both resource management algorithms on the buffered playtime of an YouTube video.

packet loss as well, increases rapidly. To summarize, a small queue size leads to packet loss and retransmitted packets, and might decrease the throughput. However, a large queue induces out-of order delivery caused by the prioritization. For values between 10 and 50 packets these effects are minimized for the investigated scenario. Hence, we choose a queue size of 10 packets for the further evaluation.

A. Comparison of the Adaptation Dynamics

In this subsection, we take a closer look on the reaction times of TCP on resource management actions. For that, we investigate two concurrent TCP flows and adjust the available resources on a per flow basis. Both flows have the same priority for the first ten seconds. After that, one flow is prioritized for the following twenty seconds. Finally, both flows are put back to the same priority. For PRIO, this sequence is implemented with priority queues. For WFQ, we change the rate configuration for a flow, while always a minimum rate of 5 kbps is guaranteed per flow to avoid starvation. The results are illustrated for an overall rate of 2.4 Mbps, i.e., the ceil parameter is set to 2.4 Mbps. In total, we conducted 100 runs to get statistical evidence. The throughput trend per flow is depicted in Figure 4. Within the first ten seconds, the throughput is shared equally between both flows, independent from the traffic management mechanism. Between 10 sec and 30 sec, the throughput of the prioritized flow increases while the throughput of the other flow decreases. Further, it can be seen that the behavior of both flows is similar for PRIO and WFQ, and that the adaptation requires roughly ten seconds. Due to the minimum rate guarantees, the low priority flow performs slightly better in case of WFQ. After 30 sec, both flows have the same priority again. As illustrated in Figure 4, the fair usage of the available bandwidth is reached much faster for WFQ as for PRIO. Hence, we can conclude that a faster and more accurate bandwidth allocation to the flows can be achieved with the WFQ mechanism.

B. Impact on the Application Layer for the Example of YouTube

This subsection aims at understanding the impact of the proposed mechanisms PRIO and WFQ on the application performance. As use case we investigate a high load scenario with a YouTube and a downloading user. As performance metric the buffered playtime of the YouTube is chosen, and the interference of the resource management mechanisms on the application are evaluated. The test video is "HOME 2009 The Full Movie", with a resolution of 480p. The overall measured video bitrate including meta data and container overhead is 1189 kbps. Taking the additional transmission overhead into account we end up with an average required bit rate of 1240 kbps. As cross-traffic we emulate a file download which runs over the whole duration of the experiment, 1.5 hours. The maximum rate of the IGW is set to 1.5 Mbps, i.e., only the YouTube flow results in a network load of $\rho \approx 0.83\%$.

PRIO contains three queues with priorities ranging from 1, the highest priority, to 3, the lowest priority. While file traffic is always mapped to queue 2, the YouTube video traffic belongs either to queue 1 or queue 3.

WFQ provides one class for file traffic and one class for YouTube traffic. Ceil parameter and rate of the YouTube videos class is either set to 5 kbps or to 1.5 Mbps. The minimum rate is chosen in order to make the WFQ setup comparable to the PRIO setup, i.e., to guarantee a maximum of the available capacity in order to prioritize the video. The minimum rate of the file class is always 5 kbps, the ceil parameter is set to 1.5 Mbps.

For the prioritizing mechanism of the YouTube video, we set a lower bound at 10 sec and an upper bound at 20 sec. If the buffer state is lower than 10 sec, the video priority is higher than the download traffic, i.e, the video uses nearly the whole available capacity. If the buffer state is higher than 20 sec, the video gains a lower priority than the download.

The results of this experiment are illustrated in Figure 5. An extract of the buffer state for 200 sec is depicted in Figure 5(a). For both, WFQ and PRIO, the buffered playtime oscillates between the lower bound of 10 sec and the upper bound of 20 sec. However, the control delay may cause threshold underruns as illustrated in the Figure. In case of PRIO, these underruns get worse than for WFQ. The conditional cumulative probability of the buffered playtime $P(x < b|b < 10\text{s})$ is illustrated in Figure 5(b). It reveals that in case of the WFQ algorithm, the minimal buffered playtime for the customer

Home is available at http://www.youtube.com/watch?v=jqxENMKaeCU, last accessed at 08/01/2014

is always larger than 5 seconds, i.e., the video playback is never disturbed. However, in case of PRIO, smaller buffered playtimes occur, and the video playback may stall.

We can conclude that WFQ performs better, since it allows a faster and more accurate adaptation of the allocated flow resources. PRIO requires an appropriate dimensioning of the queue sizes and introduces TCP retransmissions.

V. Conclusion

Software-Defined Networking (SDN) can provide an enhanced interaction between networks and applications, and a more dynamic and demand-based allocation of network resources to heterogeneous applications. In this paper, we highlighted available resource management actions provided by the standardized communication interface between data and control plane and its realization in OpenFlow. We then evaluated the impact of two of the resource management actions in a specific scenario. We showed that a demand-based allocation of network resources based on the state of QoE-critical applications like YouTube video streaming is possible. However, managing network resources with an SDN based resource management may influence the TCP control loop and result in short-time performance degradations. The intensity of this degradation depends on parameters of the used shaping and TCP algorithm. Therefore, the impact of the dynamic resource allocation has to be analyzed and well understood before it can be applied in an SDN environment.

References

[1] T. V. Lakshman, K. Sabnani, and T. Woo, "Softrouter: An open extensible platform for tomorrow's internet services." Bell Labs, Alcatel-Lucent.

[2] R. Kuschnig, I. Kofler, M. Ransburg, and H. Hellwagner, "Design options and comparison of in-network h.264/svc adaptation," *J. Vis. Comun. Image Represent.*, vol. 19, pp. 529–542, Dec. 2008.

[3] T. Hoßfeld and A. Binzenhöfer, "Analysis of skype voIP traffic in UMTS: End-to-end qos and qoe measurements," *Computer Networks*, vol. Vol 52/3 pp 650-666,, 2008.

[4] ISO/IEC, *Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats*, Apr. 2012. ISO/IEC 23009-1:2012(E).

[5] Open Networking Foundation, "Software-defined networking: The new norm for networks," 2012.

[6] T. Zinner, M. Jarschel, T. Hoßfeld, P. Tran-Gia, and W. Kellerer, "A Compass Through SDN Networks," Tech. Rep. 488, University of Würzburg, Oct. 2013.

[7] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide qoe fairness using openflow-assisted adaptive video streaming," in *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*, (New York, USA), 2013.

[8] F. Wamser, D. Hock, M. Seufert, T. Zinner, and P. Tran-Gia, "Demonstrating the Benefit of Joint Application and Network Control Within a Wireless Access Network," Apr. 2013.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, p. 69, 2008.

[10] F. Wamser, D. Hock, M. Seufert, B. Staehle, R. Pries, and P. Tran-Gia, "Using Buffered Playtime for QoE-Oriented Resource Management of YouTube Video Streaming," *Transactions on Emerging Telecommunications Technologies*, vol. 24, Apr. 2013.

[11] M. Jarschel, F. Wamser, T. Höhn, T. Zinner, and P. Tran-Gia, "SDN-based Application-Aware Networking on the Example of YouTube Video Streaming," in *2nd European Workshop on Software Defined Networks (EWSDN 2013)*, (Berlin, Germany), Oct. 2013.

[12] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, *et al.*, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pp. 3–14, ACM, 2013.

[13] S. Das, Y. Yiakoumis, G. Parulkar, N. McKeown, P. Singh, D. Getachew, and P. D. Desai, "Application-aware aggregation and traffic engineering in a converged packet-circuit network," in *Optical Fiber Communication Conference and Exposition (OFC/NFOEC)*, pp. 1–3, IEEE, 2011.

[14] K. Jeong, J. Kim, and Y.-T. Kim, "Qos-aware network operating system for software defined networking with generalized openflows," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pp. 1167–1174, IEEE, 2012.

[15] H. E. Egilmez, S. Civanlar, and A. M. Tekalp, "An optimization framework for qos-enabled adaptive video streaming over openflow networks," *Multimedia, IEEE Transactions on*, vol. 15, no. 3, pp. 710–715, 2013.

[16] M. Jarschel and R. Pries, "An OpenFlow-Based Energy-Efficient Data Center Approach," *ACM SIGCOMM*, 2012.

[17] National Laboratory for Applied Network Research (NLANR) and Applications Support Team DAST, "Iperf," March 2008.

[18] P. Yang, W. Luo, L. Xu, J. Deogun, and Y. Lu, "TCP Congestion Avoidance Algorithm Identification," July 2011.