Application-based QoS support with P4 and OpenFlow: A demonstration using Chameleon

Divyashri Bhat*†, Jason Anderson†, Paul Ruth‡, Michael Zink* and Kate Keahey§
University of Massachusetts Amherst*, University of Chicago†, RENCI‡, Argonne National Labs §
*dbhat,zink@ecs.umass.edu, †jasonanderson@uchicago.edu, ‡pruth@renci.org, §keahey@mcs.anl.gov,

***Abstract*—Although SD-WANs are now widely deployed by several production networks, they are largely restricted to traffic engineering approaches based on layer 4 (L4) of the network protocol stack that result in improved Quality-of-Service (QoS) of the network overall without necessarily focussing on a specific application. However, the emergence of application protocols such as QUIC and HTTP/2 needs an investigation of layer 5-based (L5) approaches in order to improve users' Quality-of-Experience (QoE). In this demonstration, we leverage the capabilities of flexible switches that incorporate protocol-independent packet processing in order to intelligently route traffic based on application headers. We use Adaptive Bit Rate (ABR) video streaming as an example to show how such an approach can not only provide flexible traffic management but also improve application QoE. Our prototype consists of an actual deployment in a research testbed, Chameleon, and a state-of-the-art orchestration and visualization tool, Jupyter, that we integrate with Chameleon in order to provide a single vantage point for SDN experimenters.**

I. Introduction

While application protocols such as HTTP have evolved to provide reduced latency and efficient use of network resources [1], traffic engineering paradigms such as Software Defined Networking (SDN) have simultaneously emerged to provide better Quality-of-Service (QoS) through flexible routing and centralized network management. Several large-scale production Content Distribution Networks (CDNs) such as Google [2] have implemented Software-Defined Wide Area Networks (SD-WANs) to efficiently perform application-aware routing at the peering edge. According to Cisco [3], downstream application traffic is predicted to account for 82% of all Internet traffic by 2021. Moreover, the same report predicts that SD-WAN traffic will account for 25% of all WAN traffic by 2021.

HTTP/2 incorporates several improvements over its predecessor, HTTP/1, which include a) multiplexing several streams into one TCP connection, b) server-push approaches, where content is delivered to a client without explicitly requesting it, and c) header compression for reduced latency. These improvements, particularly stream multiplexing, were devised to reduce page load time such that download requests for embedded objects such as images, video, etc., in a web page can be issued simultaneously (instead of sequentially). Similarly, the QUIC [4] protocol was introduced as a transport layer candidate for HTTP/2 with one basic difference: QUIC is based on UDP and can thus, be used to implement flexible congestion control as well. As protocols become more versatile to support high-bandwidth applications such as Adaptive Bit-Rate (ABR) video streaming and 360 Virtual Reality (VR), network architectures need to adapt in order to meet the demands of such applications worldwide. More recently, the introduction of flexible switch architectures such as [5] have paved the way for line-rate processing of application-layer headers [6]. Our demonstration investigates application-based QoS in centrally controlled networks. In particular, this work leverages the capability of protocol-independent packet processors (P4) [5] at the edge of the network to define a custom fixed-length application header and further, translate this into a Q-in-Q (802.1ad) tag [7] for the core network in order to perform QoS routing/provisioning.

In previous work [8], we demonstrated how HTTP/2-based multiplexing can be used to simultaneously fetch multiple qualities of video segments in order to improve the average bitrate quality and thereby, the Quality-of-Experience (QoE) of a client. In this demonstration, we show how HTTP/2 header information can be translated as a QoS requirement using P4-capable network elements to convert application layer header information into a Q-in-Q tag for differentiated routing via the core network using the Bring-Your-Own-Controller (BYOC) feature [9] provided by the Chameleon testbed [10]. Although we present a simple prototype using ABR streaming as an example, we believe the capabilities of such a system extend far beyond ABR segment retransmissions and can be used to implement systematic integration of Information Centric Networks (ICN) [11] with legacy networks and simultaneous transmission of 360 video viewports [12].

II. Design

A. Application header-based Traffic Engineering

1) Q-in-Q: The IEEE 802.1ad standard [7] double-tagging was introduced to allow network service providers to separate traffic from different VLANs as well as customers for better traffic management. Here, we use Q-in-Q tunneling to translate application-layer header information into link-layer headers at the edge before packets are forwarded to the core network. In particular, we focus on HTTP/2 application headers since they explicitly provide header fields that can be easily interpreted into Q-in-Q tags for better manageability.

2) HTTP/2 Header: As the number of objects embedded within a web page began to increase, the overhead due to variable length headers resulted in increased page load times for HTTP/1.1. Contrarily, HTTP/2 introduces a fixed length

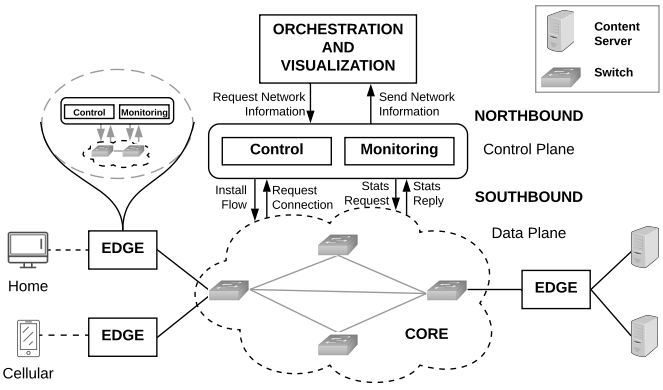

Fig. 1: Architecture for QoE to QoS translation at the edge, which is envisioned as SD-WANs

header and performs header compression in order to reduce perceived latency and increase goodput. It is interesting to note that HTTP/2 explicitly defines the *Stream ID* field to multiplex several streams into a single TCP connection and thus, can be re-interpreted as a Q-in-Q tag by any link-layer device. In this work, we redefine the outer customer tag (C-TAG) as a *Stream ID* tag using a flexible, protocol-independent packet processing language, P4 that can be programmed to interpret HTTP/2 headers. For the ABR video streaming application, *Stream ID* is used to differentiate between two distinct bitrate qualities that are simultaneously downloaded by the client as described in our previous work [8].

B. System Architecture

The main focus of our architecture is to translate application-layer header information into link-layer headers. We additionally includes a centralized component that allows network providers to orchestrate and visualize their network from a single interface. Figure 1 presents the architecture of our demonstration and consists of the following components:

1) The Core: For our architecture we assume a capability similar to that of a large-scale research testbed, ESNET[1], where the core or backbone network includes a programmable data-plane that performs fine-grained traffic engineering based on L2-L5 header information and is centrally controlled by an independent controller (denoted as *Monitoring* and *Control* in Fig. 1). However, we note that similar functionality can be incrementally deployed in production networks based on MPLS Traffic Engineering (MPLS-TE)[2] techniques as well.

2) The Edge: Innovation at the edge such as SD-WANs [2] is driven by the tremendous growth in downstream application traffic and the advent of cloud computing. Here, the edge network also includes a programmable data-plane of several flexible switches that are centrally controlled by an independent controller. However, these switches can perform fine-grained traffic engineering using L5 header information as well. In order to peer with the core, the edge controller must translate information in L5 headers to L2-L4 headers before sending packets out into the core.

3) Orchestration and Visualization: Our system also includes a centralized component that aggregates monitoring information from the various controllers in order to provide a visual representation of network performance.

The diagram also shows clients such as a wireless home network and cellular network that connect to an edge and receive requested content from a server located elsewhere on the edge network. The two edge networks are connected by the core. In the following section, we describe the setup for our demo including the platform and tools we use to run QoS translation experiments.

III. Setup

A. Chameleon Testbed

The Chameleon testbed is a deeply reconfigurable testbed that is suitable for large-scale prototyping of distributed systems and software defined networks. In this work, we leverage the recently released Bring-Your-Own-Controller feature along with previously existing capabilities of the Chameleon Cloud to create a prototype of our architecture. Figure 2 shows the setup of our testbed, which we describe in detail.

1) HTTP/2 application: For the HTTP/2 application we instantiate two bare-metal nodes: one that emulates a Web server using the open source `Caddy` (version=0.10.10) software and another that emulates an ABR video streaming client, `AStream`[3], that we modify to use an open-source Python-based library, `hyper`[4], that downloads video content using HTTP/2. Note that the cross traffic nodes are bare-metal machines used to create various network congestion scenarios using `Iperf3`[5] for controlled experiments.

2) P4 Switch[5]: We install the behavioral model, `BMV2`[6], software switch components in a bare-metal node, which emulates a P4-capable switch, and then use the `P4Runtime`[7] tool to programmatically install rules into each switch. In future, we plan to replace this with a hardware ASIC P4 switch[8] that has only recently become available.

3) OpenFlow Switch: OpenFlow [13], a widely-used implementation of SDN, is available to experimenters as a Virtual Forwarding Context (VFC), a functionality provided by Corsa switches, which enables each testbed user to provision a nearly-isolated instance of an OpenFlow (v1.3) switch. After HTTP/2 headers are translated into Q-in-Q tags as described in Sect. II-A2, the application packets are forwarded through the Corsa switch into the core network.

[1]https://www.es.net/network-r-and-d/experimental-network-testbeds/100g-sdn-testbed/

[2]https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/multiprotocol-label-switching-traffic-engineering/

[3]https://github.com/pari685/AStream

[4]https://github.com/Lukasa/hyper

[5]https://iperf.fr/iperf-download.php

[6]https://github.com/p4lang/behavioral-model

[7]https://github.com/p4lang/PI

[8]https://www.barefootnetworks.com/products/brief-tofino/

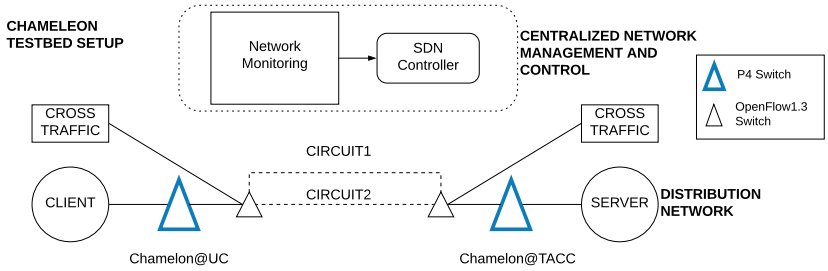

Fig. 2: Chameleon Testbed Setup: A HTTP/2 based video streaming client uses two disjoint paths to request multiple qualities of the same segment.

4) Core Network: In this demonstration, we provision two VLAN circuits (denoted as *Circuit1* and *Circuit2*) between University of Chicago (UC) and Texas Advanced Computing Center (TACC) using the Advanced Layer-2 Service (AL2S) implemented by Internet2, which is a network service provider for collaborative research[9].

5) Centralized Management and Control: For orchestration and visualization, we use Jupyter Notebooks [14], an open-source web tool particularly suited for reproducible experiments. For this demonstration, Jupyter runs inside Chameleon and provides us with a single interface not only to run the controller and the ABR video streaming application but also to visualize network traffic and QoE metrics.

IV. Demo

We will use the testbed setup described above to systematically compare QoE metrics such as average quality bitrate for two traffic engineering approaches: one where ABR video streaming retransmissions are differentially routed using a flexible switch and another where retransmissions are classified as regular HTTP traffic without any preferential treatment. Since all of the experiment components are located in a public cloud, for this demonstration we will require a large monitor with a HDMI connector to allow conference attendees to view as well as use the Jupyter web instance to interact with our experiment and two power outlets.

V. Conclusion

In this work, we show how flexible switches at the edge can be used to translate application layer header information into link layer headers to differentially route distinct qualities of ABR video segments in order to improve QoE of a HTTP/2-based application. Our demonstration is performed in a geographically distributed testbed, Chameleon, using open source orchestration and visualization tools that are easily available to researchers.

References

[1] R. P. Mike Belshe and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," Internet Requests for Comments, RFC Editor, RFC 7540, May 2015. [Online]. Available: https://tools.ietf.org/rfc/rfc7540.txt

[9]https://www.internet2.edu/products-services/advanced-networking/layer-2-services/#features-al2s

[2] K.-K. Yap, M. Motiwala, J. Rahe, S. Padgett, M. Holliman, G. Baldus, M. Hines, T. Kim, A. Narayanan, A. Jain, V. Lin, C. Rice, B. Rogan, A. Singh, B. Tanaka, M. Verma, P. Sood, M. Tariq, M. Tierney, D. Trumic, V. Valancius, C. Ying, M. Kallahalla, B. Koley, and A. Vahdat, "Taking the edge off with espresso: Scale, reliability and programmability for global internet peering," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 432–445. [Online]. Available: http://doi.acm.org/10.1145/3098822.3098854

[3] Cisco, "The zettabyte era: Trends and analysis," 2017. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.pdf

[4] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 183–196. [Online]. Available: http://doi.acm.org/10.1145/3098822.3098842

[5] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2656877.2656890

[6] X. Jin, X. Li, H. Zhang, R. Soulé, J. Lee, N. Foster, C. Kim, and I. Stoica, "Netcache: Balancing key-value stores with fast in-network caching," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 121–136.

[7] *Provider Bridges*, IEEE Std. 802.1ad, 2006.

[8] D. Bhat, R. Deshmukh, and M. Zink, "Improving qoe of abr streaming sessions through quic retransmissions: Preprint," 2018. [Online]. Available: https://drive.google.com/file/d/1aa8mRKxvEaZAqqrTc8q4--LnSs7gyLxw

[9] P. Ruth, "Software-defined networking with chameleon," 2018. [Online]. Available: https://chameleoncloud.readthedocs.io/en/latest/technical/networks/networks_sdn.html

[10] K. Keahey, P. Riteau, D. Stanzione, T. Cockerill, J. Mambretti, P. Rad, and P. Ruth, "Chameleon: a scalable production testbed for computer science research," in *Contemporary High Performance Computing vol. 3. Ed. Jeff Vetter.*, 2017.

[11] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox, "Information-centric networking: Seeing the forest for the trees," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, ser. HotNets-X. New York, NY, USA: ACM, 2011, pp. 1:1–1:6. [Online]. Available: http://doi.acm.org/10.1145/2070562.2070563

[12] M. Hosseini and V. Swaminathan, "Adaptive 360 vr video streaming: Divide and conquer," in *Multimedia (ISM), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 107–110.

[13] 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 Comput. Commun. Rev.*, vol. 38, no. 2, Mar. 2008.

[14] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay *et al.*, "Jupyter notebooks-a publishing format for reproducible computational workflows." in *ELPUB*, 2016, pp. 87–90.