College of Engineering
COMP 491 – Computer Engineering Design Project
Final Report

Multi-party WebRTC Videoconferencing over Software-Defined Networks

Baris Can KAYA & Berkan HIZIROGLU

Project Advisor:
Prof. A. Murat TEKALP

June 6th 2018

Table of Contents

1- Abstract

In our senior design project, our aim was to propose an architecture and implementation of third-party video service providers (VSP) to offer WebRTC videoconferencing services at a predictable and stable quality level in collaboration with network service providers (NSP) over a multi-operator SDN environment. Our multi-party WebRTC videoconferencing architecture uses a selective forwarding unit (SFU) where all peers send their video streams, and all streams are distributed to peers. In our architecture and implementations, clients perform motion-adaptive layer selection to adapt their send rates to the bandwidths reserved between the endpoints, and clients communicate through Google Chrome browser that supports VP9 codec, which is the most popular codec for scalable video coding and real time video communications. Our experiments show that the proposed framework yields excellent results with stable video quality when compared with the default service in terms of video quality parameters.

In our proposed managed WebRTC services, the VSP collaborates with the NSP, where the NSP implements network slicing to offer per-flow end-to-end quality of service (QoS) by computing paths between clients, with specified bandwidth and delay parameters, and performing queue management at switches. In traditional WebRTC, VSP does not collaborate with NSP, so the best-effort WebRTC Service has no control over the network.

In our test environment, we have analyzed the remote videos of peers in multi-party WebRTC videoconferencing. We made experiments to show the default service's and our managed service's reaction in the cases of high cross traffic and high motion content in remote videos. As we proposed, our managed WebRTC service performs stable and predictable video quality even if there is high cross traffic in the network.

2- Introduction

WebRTC is a popular protocol for real-time communications (RTC) over the Internet that allows browser-to-browser voice, video, and data communications using simple Javascript APIs. Moreover, WebRTC serves to the goal of the next generation services with its device independent access to 5G. Due to the growing video traffic and bandwidth limitations in the internet, providing a predictable and stable video quality for real time communications became a critical problem in both academia and industry.

Software-defined networking (SDN) is a central theme of the upcoming 5G standards. It emphasizes separation of data planes and control planes by gathering all control planes in one hand called SDN controller. SDN is also a flexible and feasible network architecture for real time video communications due to its directly programmable and centrally manageable qualities.

Although video traffic in the Internet is increasing dramatically with the years, this is not contributing to the revenues of the NSPs because NSPs cannot help VSPs for videoconferencing with stable video quality due to the high traffic. This causes NSPs to lose their customers who want to make videoconferencing with stable video quality. The advent of WebRTC services combined with NSPs deploying SDN provides an important opportunity for third-party VSPs and NSPs to offer managed real-time communications services to increase their revenues.

In our senior design project, our aim was to design and implement managed multi-party WebRTC videoconferencing over multi-SDN operator. Our architecture presents managed services that provide videoconferencing with better video quality than default services in the case of high cross traffic and high motion content in the streamed videos. As most of the users have different geographical conditions, we have also aimed to present a managed WebRTC videoconferencing architecture for users in different domains to enlarge our project's scope and increase the novelty of the project.

There are works on best-effort WebRTC videoconferencing with scalable video coding that include motion-adaptive resolution layer selection at clients for two-party point-to-

point conferencing [8] and motion-adaptive rate selection at clients and motion-adaptive layer selection at the SFU for multi-party conferencing [9]. There is a work on managed DASH-unidirectional video streaming services over SDN [12]. However, prior work on managed WebRTC videoconferencing services over SDN is very limited. Launching WebRTC services in a single-operator SDN environment was discussed in a concept paper on ”network as a service” [14], which did not have an implementation. Implementation of dynamic-network-enabled RTC on a proof-of-concept 5G network was discussed in [15] SDN environment, which extends [13] and [9]. As a result of literature review, our managed multi-party WebRTC conferencing over multi-SDN operator project has combination of new videoconferencing and networking concepts.

3- System Design

Firstly, we have implemented Traffic Engineering Manager (TEM) and new SDN controller modules for network slice orchestration in NSP. TEM is implemented as a separate project in JAVA from SDN Controller. We successfully implemented specialized Traffic Engineering Manager for a domain that finds the optimal path with respect to delay and satisfies bandwidth requirement agreed by clients and service providers. The path calculation simply follows constrained Dijkstra's algorithm where the constraint is delay, and we apply the constrained Dijkstra's algorithm for the network topologies. While we calculate the optimal path for videoconferencing, our algorithm only takes account for the links whose capacity satisfy the bandwidth requirement. The TEM in the NSP sends the optimal path to the SDN controller, and SDN controller updates the flow tables of the switches by writing rules to switches. The rules make the switches send the video packets to the next switch by following the route that TEM calculates. In detail, TEM firstly gets the network topology from SDN controller, port number of switches and bandwidth information of the links in the topology. When the TEM gathers all the required information from the SDN controller, TEM represents the topology as a graph, and runs our algorithm. When the controller sends the request, TEM returns a string array consisting of switch names in the path, and an integer array consisting of port numbers of the switches to provide the E2E communication. Moreover, when two peer make videoconferencing, where one of them is from the master domain which has VSP server, and the other one is from a different domain which is linked to the master domain via GRE Tunnel, TEM finds paths between *PEER1-SFU*, *SFU-END_SWITCH* (Switch in master domain and has GRE interface) and *END_SWITCH-PEER2*. TEM also finds 3 paths for PEER2 and PEER1. The six paths are sent to the SDN controller from TEM when the SDN controller sends a request to TEM to stitch these slices. Although TEM gets the network topology only once, SDN controller dynamically sends new path requests to TEM with current capacities of links. TEM calculates and creates the paths by using current bandwidth information. This process continues until the videoconferencing ends.

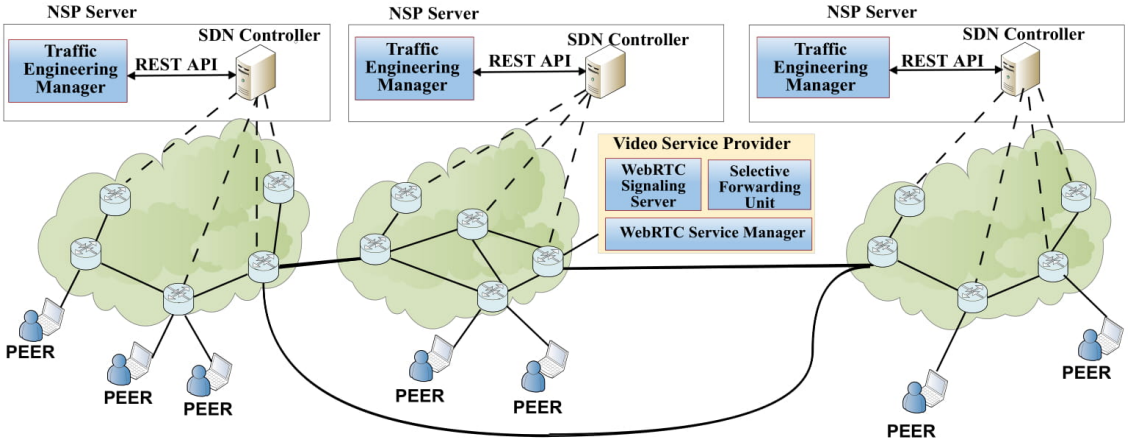


Figure 1: Multi-party WebRTC Service Architecture

The second important responsibility of NSP in our design is queue management to provide end-to-end slice reservation for specified bandwidth. Sub-slices within each NSP network are implemented by directing managed flows to special OpenFlow queues that are set up on switches by the SDN controller. The *QueuePusher* module in the Floodlight SDN controller allows us to open OpenFlow Queues. In collaboration with the *QueuePusher* module, we also used OVS database to open the queues. We opened an OVS manager that listens local 9091 port. By giving reference queue ids, we put the network flows to the queues.

In the side of VSP, we provide end-to-end WebRTC Service management. Our WebRTC service manage E2E network slice reservations by collaborating with the involved NSPs as mentioned before. Firstly, clients initiate a session by signing in to the WSS and exchanging SDP and ICE objects as they normally would do to initiate a WebRTC session. SDP includes client terminal type with their video capture/display resolutions. The agreed bandwidth guarantee is specified by also using display resolutions and terminal type. For example, a user makes videoconferencing via a mobile phone reserves lower bandwidth guarantee when compared with a user that makes videoconferencing via a laptop. In other words, the configurations of video encoders at clients are arranged by VSP by considering clients' SDP information.

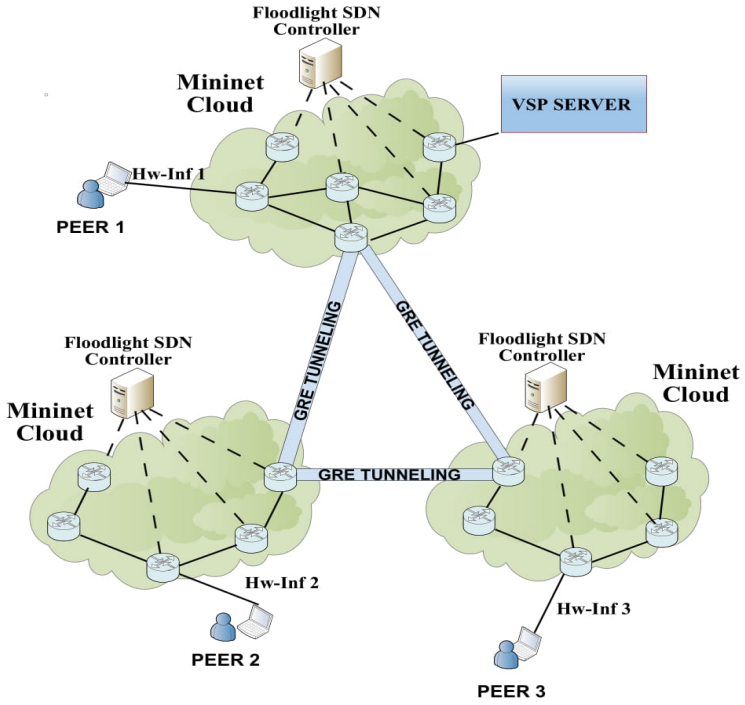


Figure 2: Multi-Domain Test Envoirment for Multi-Party WebRTC Videoconferencing

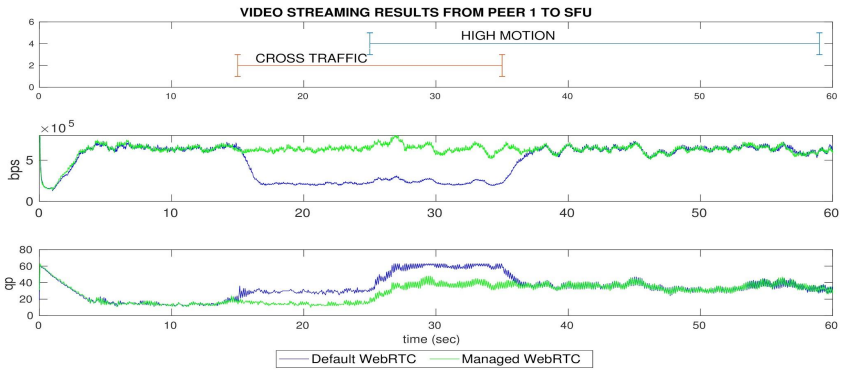
In our test environment, we composed two NSP domains, with 10 virtual switches each, using the Mininet running on a server. While we design topologies, we use Georgia Tech's network topology simulator to make similar our distribution of switches and links to real case scenarios. Then, we mapped the results of the simulator to the Mininet topologies. Each NSP domain has one border gateway, where border gateways are connected with each other using GRE tunnels as depicted in Figure 2. GRE tunnelling configuration requires change of MTU size of GRE tunnel interface. We have decided the correct MTU size by analyzing the UDP packets in Wireshark to find the overheaded bytes due to the using of GRE Tunnels. Usage of GRE tunnels creates a multiple interface problem that blocks NAT which is used for the sending IP address of hosts to SDN controller. As there are limited ways to connect multiple Mininet domains to each other, we start to send IP addresses to the controller via post requests from our local domain. Each NSP domain runs a separate instance of the Floodlight controller as SDN controller. The video service provider (VSP), which runs a WebRTC signaling server (WSS), a WebRTC service manager (WSM), and a Janus SFU [20], is a virtual host in one of the NSP domains. The WSS gathers IP addresses and terminal types of all WebRTC peers that are connected to it. The Janus application runs on the same virtual host that runs WSS and WSM applications. There are three WebRTC

peers that run WebRTC in Chrome browsers [17] on real hosts (three laptops), one in each NSP domain, which are connected to the server running the Mininet using ethernet to USB hardware interfaces (hw-intf). The WebRTC peers are directed to Janus Gateway VP9-SVC video room, which serves as the SFU in our test environment. We decided to use real hosts to run WebRTC in Chrome browsers on separate laptops rather than on virtual hosts because running the Mininet, SDN controller, WebRTC signaling server, and
multiple instances of VP9 encoder/decoder on the same server can cause real-time CPU performance limitations.

4- Analysis and Results

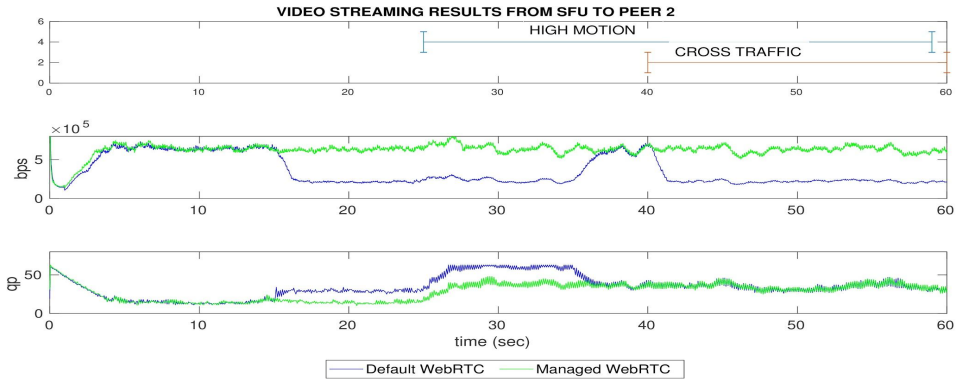
We compare the performance of the proposed managed multi-party WebRTC service using scalable VP9 video coding with that of the best-effort default WebRTC (provided by Google) in our test environment with three Chrome WebRTC peers running on real hosts as described in Section V-A.All clients perform scalable VP9 coding with 2 spatial layers, where the base layer is 640×480 and the enhancement layer is 320×240. The total bitrate for 2 layers is 600 kbps. Peers 1 and 2 receive 480p, and Peer 3 receives 240p video due to their terminal types. The reserved upload rate for all peers and download rate for peers 1 and 2 are 600 kbps. The download rate for peer 3 is 200 kbps. We record the local and remote videos as raw .yuv files for PSNR comparison. The results are compared in terms of video bitrate, resolution, quantization parameter (qp), and peak-signal-to-noise ratio (PSNR) in the presence of different amount of video motion and cross traffic. The frame rate is set to 30 Hz. We present results only for video from peer 1 to SFU, SFU to peer 2, and SFU to peer 3 due to space limitations.

Streaming from Peer 1 to SFU: The video has high motion between 25-60 sec. and low motion at other times. Cross traffic is emulated by limiting the bitrate from Peer 1 to SFU between 15-35 sec. using [21]. In the managed service, the reserved bitrate is not affected by the cross traffic.

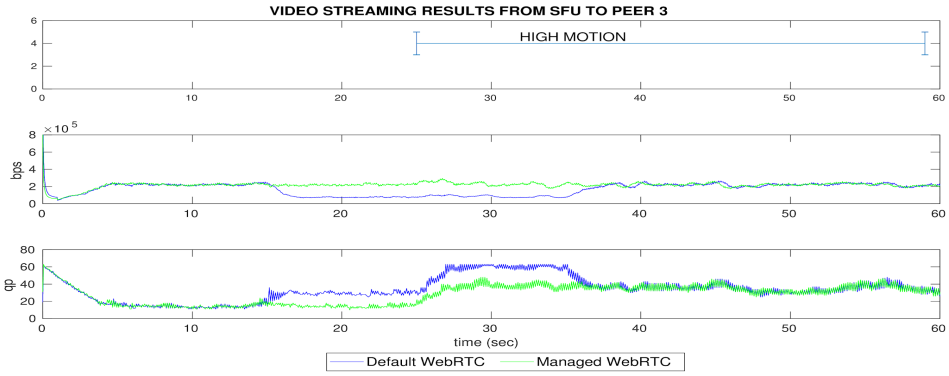


Streaming from the SFU to Peer 2: The cross traffic is applied between the SFU and Peer 2 during 40-60 seconds. In default service, SFU drops a spatial layer and decreases

the resolution to 240p to adapt to the network conditions. The results are depicted in Figure 4. Although the terminal type of Peer 2 is set to 480p, the received video quality of Peer 2 decreases remarkably.



Streaming from the SFU to Peer 3: Peer 3 receives 240p video due to its terminal type. When the cross traffic is applied between Peer 1 and SFU between 15-35 seconds, the available bandwidth between Peer1 and SFU decreases to 200 kbps in the default service. On the other hand, SFU drops a spatial layer to send 240p video. Finally, the available bandwidth for SFU-Peer 3 decreases to approximately 70 kbps as depicted in Figure 5.



Experimental results show that the managed WebRTC service provides excellent and steady video quality in all cases. However, in the default best-effort service, network congestion (cross traffic) negatively affects the quality of the video received by peers 2 and 3, where we observe that the video either stalls or becomes blurry.

5- Conclusions

We propose an architecture and an implementation for a third-party over-the-top VSP to provide managed WebRTC videoconferencing services over network slices with reserved bandwidth in collaboration with the NSP (operator). Since in a videoconference different peers can be in different geographical locations with different NSPs, the proposed architecture includes a WebRTC service manager that orchestrates E2E network slice provisioning among multiple NSPs. We provided experimental results to validate the proposed architecture in the Mininet environment using real OVS switches and Floodlight SDN controllers for each network operator domain. Our experimental results clearly demonstrate that the proposed managed WebRTC services overcomes well-known video quality limitations of today's best-effort WebRTC communications. For further design, TEM can be developed to serve for path calculation in the case of multiple GRE's.

In our current design, although there is only one GRE for each domain, TEM follows the switch has GRE mandatorily. If we could resolve the multiple interface problem, there would be multiple GRE's in a domain. Therefore, there will be lots of combinations for TEM and the optimization of TEM will increase by considering other GRE's. For further researches, implementation of multicasting for switches instead of SFU usage, may decrease delay time and improve the bandwidth consumption.

6- References

[1] H. Alvestrand,"Overview: Real Time Protocols for Browser-based Applications," draft-ietf-rtcweb-overview-19, Nov. 12, 2017.

[2] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, Analysis and design of the Google congestion control for web real-time communication (WebRTC)," Proc. 7th Int. Conf. on Multimedia Systems, 2016.

[3] A. Grange, P. de Rivaz, and J. Hunt, VP9 Bitstream and Decoding Process Specification, 31 March 2016.https://storage.googleapis.com/downloads.webmproject.org/docs/vp9/vp9-bitstream-specification-v0.6-20160331-draft.pdf

[4] J. De Cock, A. Mavlankar, A. Moorthy, and A. Aaron, "A large-scale video codec comparison of x264, x265 and libvpx for practical VOD applications," SPIE Proceedings Vol. 9971, Applications of Digital Image Processing XXXIX, San Diego, CA, Sep. 2016.

[5] VP9 video codec implementation, available online https://www.webmproject.org/vp9/

[6] Software-defined networking: The new norm for networks, Open Networking Fundation (ONF) White Paper, 2012.

[7] A. Eleftheriadis, M.R. Civanlar, and O. Shapiro, Multipoint videoconferencing with scalable video coding," Jou. of Zhejiang University Science A, vol. 7, no. 5, pp. 696-705, 2006.

[8] G. Bakar, R. A. Kirmizioglu, and A. M. Tekalp, "Motion-based adaptive streaming in WebRTC using spatio-temporal scalable VP9 video coding," IEEE Globecom, Singapore, Dec. 2017.

[9] R. A. Kirmizioglu, B. C. Kaya, and A. M. Tekalp, "Multi-party WebRTC videoconferencing using scalable VP9 video: From best effort over-the-top to managed value-added services," IEEE Int. Conf.
Multimedia and Expo (ICME), San Diego, CA, USA, July 2018.

[10] K.-F. Ng, M.-Y. Ching, Y. Liu, T. Cai, L. Li, and W. Chou, "A P2PMCU approach to multi-Party video conference with WebRTC," Int.Jou. of Future Computer and Communication, vol. 3, no. 5, Oct. 2014.

[11] S. Yoon, T. Na, and H.-Y. Ryu, "An implementation of Web-RTC based audio/video conferencing system on virtualized cloud," IEEE Int. Conf. on Consumer Electronics (ICCE), 2016.

[12] K.T. Bagci, K.E. Sahin, and A. M. Tekalp, Compete or collaborate: Architectures for collaborative DASH video over future networks,IEEE Trans. on Multimedia, vol. 19, no. 10, pp. 2152-2165, Oct. 2017.

[13] K. T. Bagci, S. Yilmaz, K. E. Sahin, and A. M. Tekalp, ”Dynamic end-to-end service-level negotiation over multi-domain software defined networks,” IEEE Int. Conf. on Commun. and Electronics (ICCE), Ha Long Bay, Vietnam, July 2016.

[14] A. Boubendir, E. Bertin, and N. Simoni, ”Network as-a-Service: the WebRTC case: How SDN and NFV set a solid Telco-OTT groundwork,” Int. Conf. on the Network of the Future (NOF), Montreal,
Canada, 30 Sep.- 2 Oct. 2015.

[15] S. Jero, V. K. Gurbani, R. Miller, B. Cilli, C. Payette and S. Sharma, ”Dynamic control of real-time communication (RTC) using SDN: A case study of a 5G end-to-end service,” Proc. of IEEE/IFIP Network Operations and Management Symposium (NOMS), 2016.

[16] E. W. Dijkstra ”A note on two problems in connexion with graphs,” Numerische Mathematik 1, 269-271.
[17] WebRTC source code, available online
https://chromium.googlesource.com/external/webrtc/

[18] Mininet Virtual Network, available online http://mininet.org/

[19] Project Floodlight Open-Source SDN Controller, available online
http://www.projectfloodlight.org/floodlight/

[20] Janus Gateway, online https://github.com/meetecho/janus-gateway

Project Github Repo: https://github.com/bhiziroglu/Video-Conferencing-over-SDN