Dynamic QoS on SIP Sessions Using OpenFlow

Charles Hubain

***Abstract*—With the increase of internet bandwidth and emergence of new multimedia internet applications, new technologies are needed to guarantee the Quality of Services. Traditionally QoS has been enforced using predefined SLAs which lack dynamic adaptability and flexibility. This article introduces an implementation leveraging the Software Defined Network protocol OpenFlow to dynamically adapt QoS to the network usage by analyzing SIP session negotiations. The first sections give a short introduction to OpenFlow, SIP and SDP while the last two sections explain how to implement dynamic QoS using those and show that line rate performances can be achieved on a HP 2920 switch. The implementation performed as expected, intercepting session negotiations and pushing QoS enforcing flow on-the-fly to the HP switch.**

***Index Terms*—SDN, OpenFlow, QoS, SIP, SDP**

I. Introduction

Over the last decade, with the increase of bandwidth, new multimedia internet applications have emerged such as Voice over Internet Protocol (VoIP) or Internet Protocol Television (IPTV). Those applications have strong requirements on the Quality of Service (QoS) in terms of bandwidth, latency, packet loss and jitter. To guarantee those requirements, predefined Service Level Agreements (SLA) have traditionally been used but they lack the flexibility to adapt dynamically to the clients needs. Software Defined Networking (SDN) aims to solve this problem by making the control plane of networking equipment programmable. OpenFlow is an open standard implementation of this concept supported by a wide variety of networking equipment vendors [1].

This article will present a QoS implementation using OpenFlow that analyzes Session Initialization Protocol (SIP) traffic to dynamically adapt QoS rules in networking equipment. SIP is already used in VoIP and IPTV applications to initiate multimedia streams and provides sufficient information to deduce QoS rules. Tests result on a real world HP 2920 switch implementing OpenFlow are presented at the end.

II. Related Works

Providing QoS for SIP based applications is not a new idea and has already been proposed in 2002 [2]. The idea is to extend the SIP protocol to encapsulate the Common Open Policy Service (COPS) protocol which could then be used in a DiffServ network. However it requires a complex network architecture and modifying existing SIP applications. In 2007 Park et al. proposed to use the Session Description Protocol (SDP) (which is a part of SIP [3]) to negotiate SLAs [4]. This has the advantage of requiring very limited modifications to existing applications but the authors did not expand on how the QoS would be enforced using this SLA negotiation. PolicyCop, a framework for autonomic QoS policy enforcement using SDN, was introduced in 2013 [5]. It uses OpenFlow to provide a dynamic, flexible, efficient and simpler alternative to DiffServ. Unfortunately the implementation is not yet complete and the paper only presents link failure tests.

III. OpenFlow

The driving idea behind OpenFlow is to separate the data plane, where packets flow according to rules, from the control plane which decides those rules. The networking devices form the data plane and connect to a programmable controller which is the control plane. The controller sends rules, called flows, to the network device which stores them in a flow table. Those flows define actions to be performed on every packets matching a specific pattern. The flow actions are used to forward the packets through the data plane but also allow to modify some header fields of the packets. If a network device can not find a match in the flow table for an incoming packet, that packet is sent to the controller which can then decide what to do with it and if additional flows need to be sent to the network device [6].

IV. SIP and SDP

The SIP protocol is used to negotiate multimedia sessions between two clients. The central component is the registrar server where the clients register themselves. Registered clients can attempt to open a session to another registered client with an INVITE request sent to the server with the username of that other client. The server behaves as a proxy between the two clients who do not know each other's IP address in the early stages of the session negotiation. Once the invited client accepts the session and respond with a code 200 OK, the actual session can be established [3]. This is shown on figure 1.

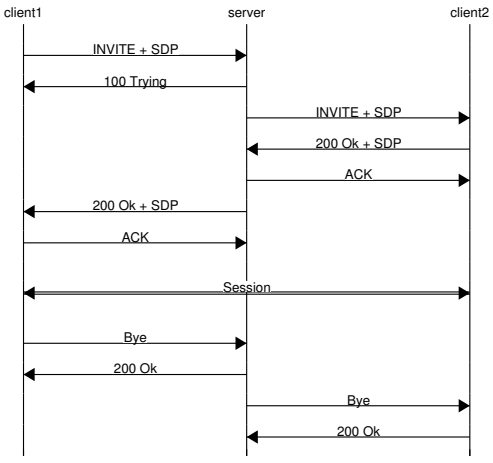

Fig. 1. Initiation and termination of a session between two clients.

Some requests carry an additional protocol, SDP, which is used to describe the multimedia codecs supported by the

clients but also to establish the connection parameters (protocol, IP address and port number) to be used for the session [7]. The two clients can directly connect to each other, but the server can also act as a proxy for the session in which case two sessions are negotiated. Figure 2 shows an example SDP packet for a the client "laptop" ready to accept audio RTP connection at IP address 10.0.0.2 on port 7078 and video RTP connection on port 9078.

```
v=0
o=laptop 285 284 IN IP4 10.0.0.2
s=Talk
c=IN IP4 10.0.0.2
t=0 0
m=audio 7078 RTP/AVP 124 111 110 0 8 101
a=rtpmap:124 opus/48000
a=fmtp:124 useinbandfec=1; usedtx=1
a=rtpmap:111 speex/16000
a=fmtp:111 vbr=on
a=rtpmap:110 speex/8000
a=fmtp:110 vbr=on
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
m=video 9078 RTP/AVP 103 99
a=rtpmap:103 VP8/90000
a=rtpmap:99 MP4V-ES/90000
a=fmtp:99 profile-level-id=3
```

Fig. 2. Example of an SDP packet.

The combination of the SDP information sent by the two clients during a session negotiation produces a pair of IP addresses, a protocol and a pair of port numbers uniquely identifying the session traffic.

V. Dynamic QoS

By sending a flow matching all packets sent from or to TCP or UDP port 5060 and requiring them to be sent to the controller, it is possible to intercept all the SIP traffic on the network. The controller can then decode the SIP packets and obtain the SDP information. Once it detects that a session has been established, it can send flows enforcing a QoS on the session to the networking devices.

There are several possibilities to enforce QoS using OpenFlow. OpenFlow has a specific action to enqueue packets on a specific output queue, guaranteeing a minimal bandwidth [6]. However its implementation is optional and there is no mechanism to configure those queues using OpenFlow. Enqueuing is implemented in OpenVSwitch but not on the HP 2920 switch [8]. Another possibility is to modify the header fields of the packets to influence how networking devices handle them. OpenFlow actions allow to set the IPv4 ToS field, set the VLAN ID or set the VLAN Priority Code Point (PCP) [6].

VI. Implementation

Flow on HP switch can either be stored in software, with limited performances, or in hardware which offer line rate performances. However some restrictions exist on the packet field that can be matched in hardware which imposes constraints on the implementation. For example, layer 2 and layer 3 fields can not be matched in the same flow [8].

As discussed in section V there are several possibilities to implement QoS using OpenFlow. On HP switches the VLAN PCP field is directly mapped to specific QoS output queues [9]. Changing it allows to give some packets a higher priority and thus to enforce QoS. Figure 3 and 4 show the effect of such QoS implementation on two UDP traffics that compete for the same 1 Gbps bandwidth but with one having priority over the other [1]. Around 80% of the bandwidth is allocated to the priority traffic. Also the jitter, which is crucial to real-time multimedia applications, is significantly lower compared to the non-priority traffic. This demonstrates that a line rate QoS using OpenFlow is possible on real world hardware. Moreover a standard QoS configuration, without using OpenFlow, produced the same result.

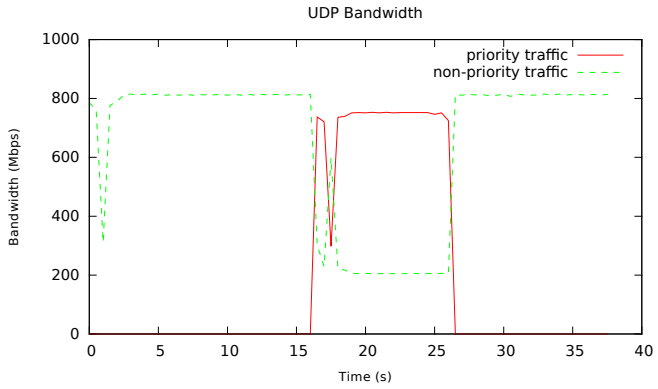

Fig. 3. Bandwidth of a priority and non-priority UDP traffic over time.

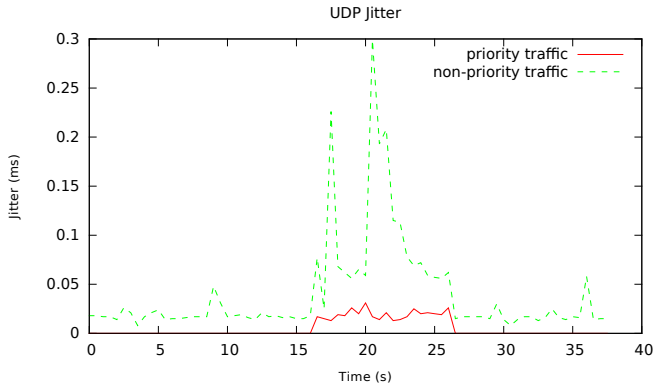

Fig. 4. Jitter of a priority and non-priority UDP traffic over time.

Piecing together the SIP traffic interception of section V, the SDP decoding of section IV and the QoS from this section, a dynamic QoS on SIP sessions using OpenFlow has been

[1]UDP traffic measurements were produced by iperf. Implementation is available at
https://github.com/haxelion/openflow_qos/blob/master/hp_qos_test.py

implemented [2]. It behaves as expected, intercepting session negotiations and pushing QoS enforcing flow on-the-fly to the HP switch.

VII. Conclusion

This article shows that SDN is an elegant solution to the QoS problem for modern internet multimedia applications. By intercepting SIP traffic, the OpenFlow controller was able to dynamically enforce QoS over negotiated sessions. Moreover line rate OpenFlow dynamic QoS performances were shown to be possible on real world hardware with results comparable to traditional static configurations.

References

[1] Open Networking Foundation. 13 April 2012. *Software-Defined Networking: The New Norm for Networks*. Whitepaper. 12p. https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf

[2] SALSANO Stefano, VELTRI Lucas. March 2002. "QoS control by means of COPS to support SIP-based applications" in *IEEE Networks*. Volume 16 Issue 2. Scientific paper. New York City, USA: IEEE. pp 27-33.

[3] ROSENBERG J., SCHULZRINNE H., CAMARILLO G., JOHNSTON A., PETERSON J., SPARKS R., HANDLEY M., SCHOOLER E. June 2002. *SIP: Session Initiation Protocol*. RFC 3261. https://www.ietf.org/rfc/rfc3261.txt

[4] PARK H.J., YANG J.H., CHOI J.K, KIM H.S. 12 February 2007. "QoS negotiation for IPTV service using SIP" in *The 9th International Conference on Advanced Communication Technology*. Volume 2. Scientific paper. New York City, USA: IEEE. pp 945-948.

[5] BARI M. Faizul, CHOWDHURY Shihabur Rahman, AHMED Reaz, BOUTABA Raouf, CHERITON David R. 11 November 2013. "PolicyCop: An Autonomic QoS Policy Enforcement Framework for Software Defined Networks" in *2013 IEEE SDN for Future Networks and Services*. Scientific paper. New York City, USA: IEEE. 7p.

[6] Open Networking Foundation. 31 December 2009. *OpenFlow Switch Specification*. Version 1.0.0. Technical specification. 44p. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf

[7] HANDLEY M., JACOBSON V., PERKINS C. July 2006. *SDP: Session Description Protocol*. RFC 4566. https://tools.ietf.org/rfc/rfc4566.txt

[8] Hewlett Packard. November 2013. *HP Switch Software OpenFlow Administrators Guide for K/KA/WB 15.14*. HP Manual. Palo Alto, USA: Hewlett Packard. 79p.

[9] Hewlett Packard. November 2014. *HP Switch Software Advanced Traffic Management Guide WB.15.16*. HP Manual. Palo Alto, USA: Hewlett Packard. 274p.

[2]Implementation available at https://github.com/haxelion/openflow_qos/blob/master/hp_flowqos.py