Dynamic QoS on SIP Sessions Using OpenFlow

Jérémy Pagé*, Charles Hubain† and Jean-Michel Dricot‡

OPERA Wireless Communications
Université libre de Bruxelles
Brussels, Belgium
Email: *jeremy.page@ulb.ac.be, †charles.hubain@ulb.ac.be, ‡jean-michel.dricot@ulb.ac.be

***Abstract*—With the increase in Internet bandwidth demand and emergence of new multimedia internet applications, new technologies are needed to guarantee the Quality of Service (QoS). Traditionally, QoS has been enforced using predefined Service Level Agreements (SLAs), which lack dynamic adaptability and flexibility. This article introduces an implementation leveraging the Software Defined Networking (SDN) protocol OpenFlow to dynamically adapt QoS to the network usage by analyzing Session Initiation Protocol (SIP) session negotiations. The implementation is verified on real world OpenFlow-enabled switches with different types of traffic QoS requirements.**

Keywords–SDN; OpenFlow; QoS; SIP; SDP.

I. Introduction

Over the last decade, with the increase of bandwidth demand, new multimedia internet applications have emerged such as, Voice over IP (VoIP), IP Television (IPTV), online gaming, etc. Those applications have strong requirements regarding the QoS in terms of bandwidth, latency, packet loss and jitter. To guarantee those requirements, predefined SLAs have traditionally been used but they lack the flexibility to adapt dynamically to the client needs. Besides those applications, the network environment can also have strong requirements, e.g., a medical-grade network regarding the transmission of delay-sensitive information [1], [2]. Indeed, when medical data or audio/video data transmission is required, the latency must be as low as possible and a bandwidth must be guaranteed.

In addition, the deployment of new network services in the operator networks comes at a high cost. Indeed the integration and operation typically come with separate hardware entities. Network Functions Virtualization (NFV) [3], [4] aims to address this problem by allowing to deploy network services onto virtualized industry servers, which can be located in data centers. Network functions can thus be deployed as virtualized instances without the need to install hardware equipment. By migrating the hardware to software, NFV is expected to lower not only the Capital Expenditure but also the Operational Expenditure [5]. The services can be deployed more flexibly and scaled up and down very quickly. As explained in [5], from an architecture perspective, NFV can be complementary to technologies, such as SDN and cloud computing.

Moreover, SDN aims to solve the lack of flexibility of the SLAs. SDN [6], [7] provides an abstraction between the data-plane forwarding (hardware) and the control-plane (software). It makes the control-plane programmable by a centralized SDN controller on a per data flow basis. The OpenFlow protocol [8] is an open standard implementation for the signaling between an SDN switch and an SDN controller.

Previous studies such as [5] has focused on the deployment of an Long-Term Evolution (LTE) Evolved Packet Core (EPC) system in an operator cloud environment with SDN as a network enabler. In the LTE EPC architecture, IP Multimedia Subsystem (IMS) is used for the VoIP and the SIP serves as the signaling protocol [9]. SIP is independent of the technologies chosen, and can be used regardless of IMS. Furthermore, SIP is the de facto standard for initializing multimedia communications between entities and, this protocol provides sufficient information to deduce QoS needs.

In a previous work [10], we demonstrated the integration of SDN and the 4G architecture towards 5G and we presented the benefits of SDN in the mobile environment in terms of performances and flexibility.

The paper is organized as follows. The next section presents the related works. Section III gives a short introduction to OpenFlow. Section IV describes the 5G architecture proposed in [10]. Section V introduces the protocols SIP and Session Description Protocol (SDP). Section VI explains how OpenFlow can be used to implement dynamic QoS. Section VII describes the implementation and verification that have been made over real world HP 2920 switches (implementing OpenFlow). The last section presents the conclusion and future works.

II. Related Work

Providing QoS for SIP-based applications started in 2002 [11]. 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 2006, the authors of [12] introduced an architecture for SIP-based QoS applications. This architecture combined both DiffServ and IntServ and generally works with standard SIP end systems.

In 2007, the authors of [13] proposed to use the SDP (included in some of the SIP messages) to negotiate SLAs. 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.

PolicyCop, a framework for autonomic QoS policy enforcement using SDN, was introduced in 2013 [14]. 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.

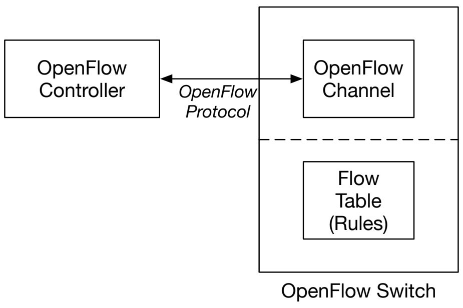

Figure 1. OpenFlow Controller and Switches

III. OpenFlow

As defined in the SDN architecture, OpenFlow separates the data-plane from the control-plane. The networking devices, i.e., the OpenFlow switches, form the data-plane where data packets flow according to rules (flows). Each rule is composed of matching parameters and a set of actions to execute when there is a match. Examples of matching parameters include: Internet Protocol (IP) address source/destination, Media Access Control (MAC) address source/destination, Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) source/destination port. Examples of actions include: sending the packet on a specific port, changing some header fields of the packet.

Every switch is connected to the controller and communicates with the OpenFlow protocol (Figure 1). The OpenFlow controller can inject flows, i.e., rules, into the switches in order to define the routing of specific packets. When the switch receives a new flow, it adds it in its flow table. When an incoming packet arrives, the flow table of the switch is looked up to match the packet according to the flows. If there is a match, the set of actions defined by the flow is executed. If there is no match, the packet is sent to the controller for inspection. The controller can then decide which action to take (e.g., create a new flow, drop the packet, send the packet to a specific port) [15].

IV. Previous Work: 5G OpenFlow Integration

In a previous work, the integration of OpenFlow in the core network of the mobile architecture between the Evolved NodeB (eNodeB) and the Packet Data Network Gateway (P-GW) has been demonstrated [10]. As it can be observed in Figure 2, the proposed solution removes the Serving Gateway (S-GW) and introduces the OpenFlow Controller. Hence, allowing for a faster circuit-switched transport from the antenna to the core network. The control path from the Mobility Management Entity (MME) to the P-GW and the data path from the eNodeB to the P-GW is comprised of OpenFlow Switches as underlying infrastructure. These switches are controlled by and connected to the controller.

When a new User Equipment (UE) connects to the network, it sends an *Attach Request* to the MME through the eNodeB (the antenna). The MME authenticates the user using information from the Home Subscriber Server (HSS) (the user database) and retrieves which services it can access to, e.g., IMS for VoIP, Internet. The MME selects the P-GW for the corresponding service and forwards the request to it. The

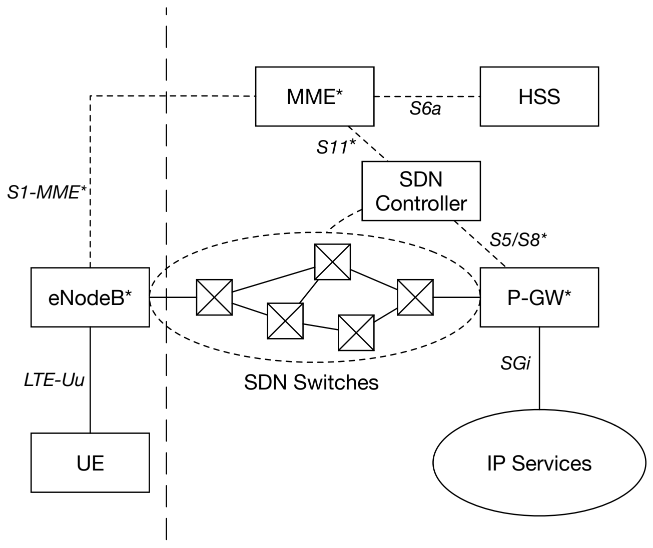

Figure 2. Proposed 5G Architecture [10]

controller is able to intercept and analyze the request in order to optimize the path and to proactively create the flows in the switches. These flows are created for the control path and for the data path. Each flow can have different QoS parameters attached to them, as it will be shown in the next sections, in order to satisfy some requirements. When the P-GW sends back a response, the flows are already created and the response is sent back to the UE, which is then successfully connected to the service.

V. SIP and SDP

The SIP [9] protocol is used to negotiate multimedia sessions between multiple clients. Figure 3 presents the SIP architecture comprising the SIP servers (*proxy* and *registrar*) and two UEs, Alice and Bob. First of all, the SIP clients need to register themselves to the registrar server. They send a `REGISTER` SIP request to their proxy which forwards it to the registrar. When a client is registered, it opens a multimedia session with another registered client by sending an `INVITE` SIP request to the proxy with the username of the other client.

The proxy behaves as an intermediary between the clients who do not know each others' IP addresses. It forwards the `INVITE` SIP request to the recipient. If the receiver accepts the session, i.e., responding with a `200 OK` SIP response, the session is established. Nevertheless, SIP is only providing the signaling and not the media transfer. When the session is established, the data path is going directly from end to end, not passing through the proxy. Real-time Transport Protocol (RTP) [16], for example, is a protocol used to exchange multimedia data. Figure 4 shows an example of the initiation and termination of a SIP session.

In order for RTP to transfer media from end to end, SDP [17] can be used to describe the media codecs supported by the clients and the connection parameters (IP addresses, port numbers, protocol). The SDP part is embedded in the `INVITE` request and in the `200 OK` corresponding response. The clients can send media streams from end to end as shown, but the server can also act as a media proxy. Figure 5 shows an

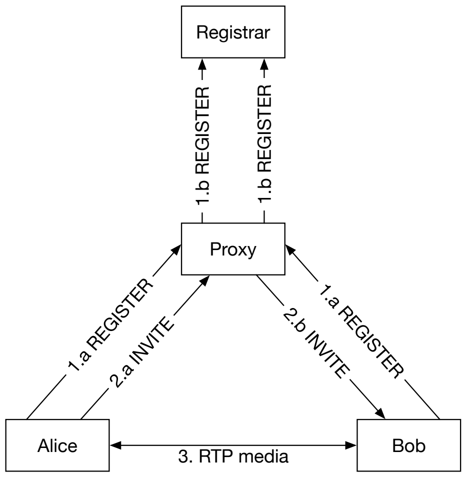

Figure 3. SIP Architecture

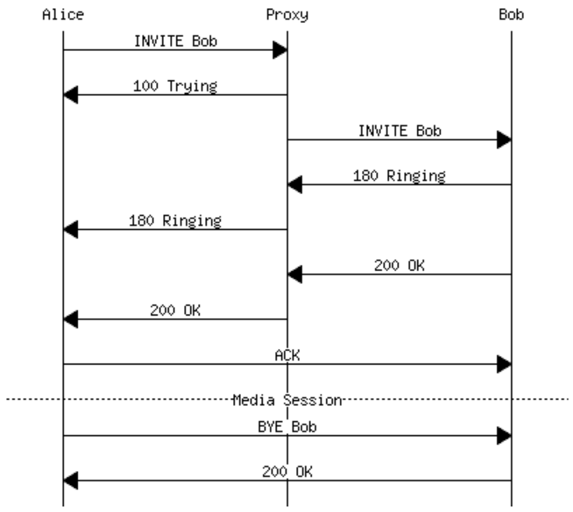

Figure 4. SIP Flow Example

```
INVITE sip:bob@example.com SIP/2.0
Via: SIP/2.0/UDP client1.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@example.com>;tag=9fxced76sl
To: Bob <sip:bob@example.com>
Call-ID: 3848276298220188511@example.com
CSeq: 1 INVITE
Contact: <sip:alice@client1.example.com;transport=udp>
Content-Type: application/sdp
Content-Length: 144

v=0
o=alice 2890844526 2890844526 IN IP4 client1.example.com
s=-
c=IN IP4 192.0.1.100
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Figure 5. SIP Message Example with SDP Part

example of a SIP `INVITE` message with a SDP part where the IP address is `192.0.1.100` and the port number is `49172`.

The session traffic between two clients is uniquely identified by combining the SDP information, which produces a pair of IP addresses, a pair of port numbers and a protocol.

VI. Dynamic QoS with OpenFlow

The flexibility of OpenFlow allows to dynamically attribute a specific QoS for incoming packets by creating flows with the attributed QoS. First, in order to intercept all the SIP packets in the controller, a flow is created in each switch where the matching is done on the port 5060 (the default SIP port) and the actions set is composed of one action: send the packet to the controller. If the switches are configured to send the packets to the controller when there is no match, this flow is optional because the default behavior of the switch would be to send the packet to the controller.

By intercepting all the SIP signaling, the controller is able to analyze their content and to attribute a QoS to each one of the sessions. This attribution is done by extracting the IP address, port numbers and media parameters from the SDP parts of the SIP messages. Once the controller detects that a session has been established (a `200 OK` following an `INVITE`), it can enforce the attributed QoS for this session by creating flows in the switches along the path of this session. The matching part of the flows contains the IP addresses and port numbers extracted from the SDP part. The actions set of the flows contains an action attributing the QoS parameter and action outputting the packet to the right destination.

There are several possibilities to enforce QoS using OpenFlow. OpenFlow defines a specific action to enqueue packets on a specific output queue, guaranteeing a minimal bandwidth [15]. However its implementation is optional, and there is no mechanism to configure those queues with OpenFlow. The HP 2920-24G switch used for the verification does not implement the enqueue method defined in OpenFlow [18, p. 12]. A generic workaround has been implemented by the modification of the IP version 4 (IPv4) header fields to attribute a QoS class. Indeed, the cited switch is compliant with IEEE 802.1p [19, p. 14]. OpenFlow defines actions to set the IPv4 Type of Service (ToS) field, the Virtual Local Area Network (VLAN) Identifier (ID), or the VLAN Priority Code Point (PCP) [15].

On HP switches the VLAN PCP field is directly mapped to specific QoS output queues [20]. Changing it allows to give some packets a higher priority and thus enforce QoS. On other hardware, the method used to enforce a specific QoS must be adapted.

VII. Implementation and Verification

The verification architecture is presented in Figure 6 and can be described as follows. 3 HP 2920 switches are connected to an OpenFlow Controller running on a server (Ubuntu 14.04 LTS). Each one of the switches is also connected to a host computer (Ubuntu 14.04 LTS). The topology used can be extended to a more complex one, by including more switches and hosts. When the network grows, the question of the scalability can arise. According to [21], the scalability concerns are not unique to SDN and solutions exist. However, the scalability issues are still studied and are not trivial [22]. HyperFlow [23], e.g., is a distributed control plane for OpenFlow and provides scalability.

The verification procedure was conducted as follows. First, Host 2 connects to Host 1 and maximize the utilization of the

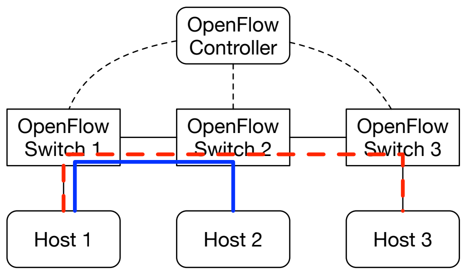

Figure 6. Verification Architecture

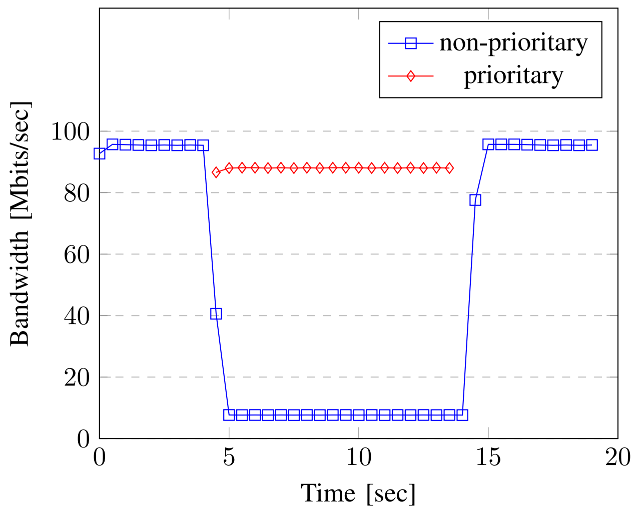

Figure 7. Bandwidth of a Priority and non-Priority UDP Traffic over Time

available bandwidth for a period of 20 seconds. During this period, some measurements are made on the bandwidth used and the jitter observed. After approximately 5 seconds, Host 3 connects to Host 1 and tries also to maximize the utilization of the bandwidth, but for a period of 10 seconds. The traffic generated by the Host 3 has priority over the traffic generated by the Host 2. The tool used to generate the traffic and to analyze the bandwidth and jitter is *iPerf* [24].

Figure 7 and 8 show the effect of such QoS implementation on the two UDP traffics that compete for the same 100 Mbps bandwidth. The results indicate that around 80% of the bandwidth is allocated to the priority traffic. Also the jitter, which is crucial to real-time multimedia applications, is significantly limited with respect to the non-priority traffic. This demonstrates that a line rate QoS using OpenFlow is possible on real world hardware.

When this dynamic QoS implementation is used in conjunction with the detection of new media traffic by analyzing the SIP signaling, the controller can enforce a specific QoS for new media sessions based on the SDP parameters.

VIII. Conclusion and Future Work

This article presents and verifies that SDN is an elegant solution to the QoS problem of modern internet multimedia applications. By analyzing on the fly the SIP signaling, the OpenFlow controller was able to dynamically enforce QoS over negotiated sessions. The integration of OpenFlow in the mobile architecture with a dynamic QoS for the SIP traffic towards IMS allows to be more proactive and flexible. Indeed, the approach is entirely software and the granularity is higher. The network can be programmed in software and the packets can be inspected until the fourth layer. Moreover line rate OpenFlow dynamic QoS performances were shown to be possible on real world hardware with results comparable to traditional static configurations.

Future work will focus on the integration of this implementation on a full-featured IMS platform (Voice over LTE (VoLTE)). Also, further tests will include various traffic and the proposed approach will be included with different kinds of protocols and networks.

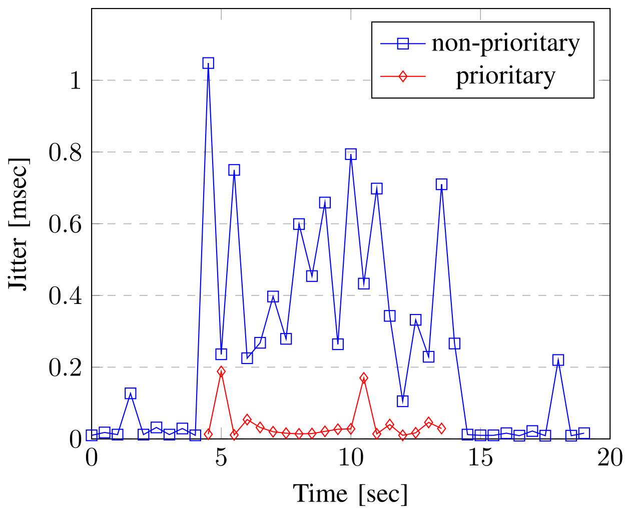

Figure 8. Jitter of a Priority and non-Priority UDP Traffic over Time

Acknowledgment

This work is supported by a grant of Innoviris, the Brussels Institute for Research and Innovation.

References

[1] L. Skorin-Kapov and M. Matijasevic, “Analysis of QoS requirements for e-Health services and mapping to Evolved Packet System QoS classes,” Int. J. Telemedicine Appl., vol. 2010, Jan. 2010, pp. 9:1–9:18.

[2] A. Zvikhachevskaya, G. Markarian, and L. Mihaylova, “Quality of service consideration for the wireless telemedicine and e-health services,” in WCNC, 2009, pp. 3064–3069.

[3] SDN and OpenFlow World Congress, “Network Functions Virtualization,” White Paper, Oct. 2012.

[4] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, “NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC),” IEEE Network, vol. 28, no. 6, 2014, pp. 18–26.

[5] A. Basta, W. Kellerer, M. Hoffmann, K. Hoffmann, and E.-D. Schmidt, “A virtual SDN-enabled LTE EPC architecture: a case study for S-/P-Gateways functions,” in 2013 IEEE SDN for Future Networks and Services (SDN4FNS), Nov. 2013, pp. 1–7.

[6] Open Networking Foundation, “Software-Defined Networking: the new norm for networks,” ONF White Paper, Apr. 2012.

[7] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” IEEE Communications Surveys & Tutorials, vol. 16, no. 3, 2014, pp. 1617–1634.

[8] N. McKeown et al., “OpenFlow: enabling innovation in campus networks,” ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, Apr. 2008, pp. 69–74.

[9] J. Rosenberg et al., “SIP: session initiation protocol,” Tech. Rep., 2002.

[10] J. Pagé and J.-M. Dricot, “Software-Defined Networking for low-latency 5G core network,” in 2016 International Conference on Military Communications and Information Systems (ICMCIS). IEEE, May 2016.

[11] S. Salsano and L. Veltri, “QoS control by means of COPS to support SIP-based applications,” IEEE Network, vol. 16, no. 2, 2002, pp. 27–33.

[12] E.-H. Cho, K.-S. Shin, and S.-J. Yoo, “SIP-based QoS support architecture and session management in a combined IntServ and DiffServ networks,” Computer Communications, vol. 29, no. 15, 2006, pp. 2996–3009.

[13] H. Park, J. Yang, J. Choi, and H. Kim, “QoS negotiation for IPTV service using SIP,” in The 9th International Conference on Advanced Communication Technology, vol. 2. IEEE, 2007, pp. 945–948.

[14] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, “PolicyCop: an autonomic QoS policy enforcement framework for software defined networks,” in 2013 IEEE SDN for Future Networks and Services (SDN4FNS). IEEE, 2013, pp. 1–7.

[15] OpenFlow Switch Specification, “Version 1.0. 0 (Wire Protocol 0x01).” Open Networking Foundation, Dec. 2009.

[16] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” RFC 3550 (INTERNET STANDARD), Internet Engineering Task Force, Jul. 2003, URL: http://www.ietf.org/rfc/rfc3550.txt [accessed: 2016-08-30].

[17] M. Handley and V. Jacobson, “Session Description Protocol,” Apr. 1998.

[18] Hewlett Packard, “HP Switch Software OpenFlow administrator guide for K/KA/WB 15.17,” Jun. 2015.

[19] ——, “HP 2920 Switch Series.”

[20] ——, “HP Switch Software advanced traffic management guide WB.15.17,” Jun. 2015.

[21] S. Yeganeh, A. Tootoonchian, and Y. Ganjali, “On scalability of software-defined networking,” IEEE Communications Magazine, vol. 51, no. 2, 2013, pp. 136–141.

[22] B. J. van Asten, N. L. M. van Adrichem, and F. A. Kuipers, “Scalability and Resilience of Software-Defined Networking: an overview,” CoRR, vol. abs/1408.6760, 2014.

[23] A. Tootoonchian and Y. Ganjali, “HyperFlow: A distributed control plane for OpenFlow,” in Proceedings of the 2010 internet network management conference on Research on enterprise networking, 2010, p. 3.

[24] NLANR/DAST: iPerf - the network bandwidth measurement tool. URL: https://iperf.fr [accessed: 2016-08-30].