A Meter Band Rate Mechanism to Improve the Native QoS Capability of OpenFlow and OpenDaylight

Mohamed Saleh Al Breiki
Department of Computer Science
Middlesex University
London, UK
ma2634@live.mdx.ac.uk

Suiping Zhou
Department of Computer Science
Middlesex University
London, UK
suiping@mdx.ac.uk

Yuan Roger Luo
Department of Computer Science
Middlesex University
London, UK
y.luo@mdx.ac.uk

***Abstract*— The exponential growth of mobile connected devices with advanced multimedia features imposes a requirement to enhance quality of service (QoS) from heterogeneous systems and networks. In order to satisfy mission-critical multimedia QoS requirements new generation mobile networks must present content-optimized mechanisms in order to use valuable network resources efficiently and provide QoS requirements for each application. This research explores a novel solution for quality of service performance for streaming mission-critical video data in OpenFlow SDN networks. A Meter Band Rate Evaluator (MBE) Mechanism is proposed based on a new band rate description language to improve the native QoS capability of OpenFlow and OpenDaylight. Its design and development are presented and the mechanism is verified through a simulated experiment in an SDN testbed. The results revealed a significant percentage increase in QoS performance when the MBE was enabled. These findings provide support and validation for the effectiveness of the MBE to enhance the native capability of OpenFlow and OpenDaylight for efficient QoS provision.**

Keywords— ***software defined networks, OpenDaylight, OpenFlow, quality of service, meter band rate, cloud computing, network function virtualization, heterogeneous networks, 4G***

I. INTRODUCTION

Software Defined Networking (SDN) offers a highly promising paradigm to implement a novel architecture within a cloud computing environment to integrate quality of service (QoS) and guarantee video performance to mobile devices. The QoS design on which SDN and Network Function Virtualization (NFV) are based ensures reliability and the quality of mission-critical applications over multi-service networks (3G, 4G, 5G, Terrestrial Trunked Radio (TETRA) [1]. SDN is an emerging technology that is dynamic and adaptable providing enhanced network management features [2] and highly applicable given the increased network control required by cloud applications [3]. TETRA networks are unsuitable for hosting many data centric applications and services while accessing a widening range of real-time multimedia information from multiple sources is vital [4,5]. SDN is structured so that the control plane is abstracted from the forwarding/data plane allowing the control plane to be programmable [6,7]. This enhances network programmability and affords support for dynamic and advanced future network functions [8]. The purpose of this study is to investigate the suitability of SDN and OpenFlow architecture to achieve quality of service performance for video streaming to mobile devices in a 4G cloud computing environment. The central research question underpinning this research is: can SDN, QoS mechanisms, cloud and 4G technologies be integrated to deliver prioritized real-time video streaming according to QoS requirements? A QoS solution is designed and proposed to improve the native capability of OpenFlow and OpenDaylight to support QoS based on a novel meter band rate mechanism. This paper focuses on validating the meter band rate mechanism and presents the results of the evaluation tests. This study contributes new insight into the effectiveness of the custom solution implementing OpenDaylight SDN controller in conjunction with OpenFlow and OpenQoS within a cloud computing environment delivering real-time video and data to mobile end devices. A key contribution from this research is the advancement of a QoS solution to improve the native capability of OpenFlow and OpenDaylight to support QoS based on a novel meter band rate mechanism.

II. RELATED WORK

Enhancement of QoS has been the subject of several studies which have attempted to leverage the SDN network control functions by designation to software components located in a switch, which permits transparent interaction with network resources in accordance with the preferences of the application. Liyanage et al. [1] highlight that QoS provision cannot be automated according to application or service. Quality of service relates to the degree of service quality that is expected for a given application [9]. QoS requires the prioritisation of mission-critical applications over less critical applications [1]. OpenFlow is a protocol designed to update the flow tables in a switch, allowing the SDN controller to access the forwarding table of each member switch, or in other words to connect control plane and data plane in the SDN world [10]. The flexibility of the protocol means that network flows can be dynamically routed without causing interruption of traffic. This is accomplished through separation of the data and control plane which is where a controller coordinates network traffic and generates or modifies rules for networking devices [11]. Any software able to support OpenFlow can be employed to control the forwarding decisions in network devices configured with OpenFlow.

The potential for QoS management within the OpenFlow protocol has incited some research towards approaching the QoS issue at the network level. OpenFlow is able to contribute fine-granularity QoS support related to delay, throughput and jitter [12]. This is derived from the effective control of data delivery at the packet-level or flow-level through its controllers. The fine-granularity allows OpenFlow users to stipulate the management of individual flows, comparable to Integrated Services (IntServ) in Internet Engineering Task Force (IETF) definitions [13]. Users may also combine individual flows into classes.

Egilmez et al. [14] developed an OpenFlow protocol - based controller, OpenQoS, to attain end-to-end QoS for multimedia-based applications. OpenQoS is a key framework which safeguards the service delivery of application traffic on an optimal path while focusing principally on multimedia flows such as video streaming or voice over IP (VOIP) [15]. Traffic is categorized into data flows and multimedia flows, with the former following the shortest routing algorithm while the latter follows the dynamic QoS guaranteed routing algorithm. Shi and Chung [16] proposed a solution to achieve QoS in SDN by marking data-plane packets and management at the control-plane. The entire network is divided into a number of virtual networks (VN) and an autonomic manager monitors the requirement of the VNs and assigns physical resources accordingly [16]. An extension of OpenFlow is advanced as one area for improvement to enhance QoS. Work by Lu et al. [17] aimed to resolve the issue of satisfaction of QoS by traditional flow control mechanisms while maximizing throughput. Other research conducted in this area similarly fails to address the limitation of OpenFlow protocol to support QoS [16, 18, 19, 20].

III. Problem Formulation

QoS mechanisms such as IntServ, Differentiated Services (DiffServ) and Multiprotocol Label Switching (MPLS) have specific drawbacks being non-adaptive and non-global. SDN based wireless management tools introduced distributed and collaborative traffic classifier tools responsible for classifying traffic flows into different QoS classes, so it may be able to apply different service resource provisioning according to each application's QoS demands [21]. The OpenDaylight SDN controller utilizing OpenFlow protocol is one of the first successful implementations of SDN and has been adopted by many vendors globally. The Open Flow protocol is a key factor in SDN by controlling switches by means of OpenFlow signalling between the configured switches. It is critical in terms of the best effort paradigm and guarantees a dynamic QoS provision.

The key advantage of OpenFlow is that it may be adopted within existing network infrastructures without the need for fundamental design changes and is backwards compatible with many legacy flow-enabled network devices. To understand the problem it is first necessary to explain the existing capabilities of QoS Support in OpenFlow. QoS Support in OpenFlow for Software-Defined Networks is presently implemented in two ways using either per-flow QoS or DiffServ. With Per-port Queue, OpenFlow allows specification of a particular queue for a flow in a flow-entry. If a match occurs the corresponding action will output to the given queue. Each queue can be configured with properties. This determines how packets will be treated inside the queue. This per-flow queue feature of OpenFlow is used to define per-flow QoS. With per-flow meter table, OpenFlow allows attaching one or more flows to a meter table. Meters are used to measure the rate of packets and control the rate. A single flow can also be attached to multiple meter tables but using different tables. In a single table each flow can only be attached to one meter table. Per-flow queue combined with meter tables can be used to implement DiffServ.

The key limitations and areas for improvement of QoS in OpenFlow (versions 1.1-1.5.1.) can be identified as: limited QoS support through simple queuing mechanisms (e.g. min-rate, max-rate); no ability to configure queues inside OpenFlow; basic provision of support for DiffServ (e.g. Differentiated Services Code Point (DSCP) remark); QoS is only allowed at egress when using queue; and optional meters are not supported by most switches. The simple queuing mechanisms generate two key QoS issues. Firstly the per-port queue does not guarantee efficient utilisation of available bandwidth. Moreover, the specification of minimum bandwidth for traffic may require over-provisioning to prevent poor QoS during congestion. The behavior depends on the specific hardware implementation and is not predictable. Secondly there is limited scalability in the simple queueing mechanisms provided. The lack of ability to configure queues inside OpenFlow means that dynamic changes in queue configuration are not possible. This is because the SDN controller cannot be used to configure the queues, and other mechanisms such as cli, OF-Config, or Open vSwitch Database Management Protocol (OVSDB) must be deployed.

Using meter tables, fields can be modified however meters are not replacements to queues but only complement them and therefore cannot overcome their limitations. Meters can modify certain fields in a packet but do not support complete implementation of all types of per-hop-behavior (PHB). For example, appropriate queuing mechanisms such as weighted fair queueing (WFQ), or strict priority queuing (SPQ) are not deployable solutions for a PHB. Moreover, mechanisms such as Call Admission Control (CAC) are not feasible using meters. Allowing QoS only at egress when using queue is a problem as it is desirable to be able to drop traffic at ingress to save switch resources for other traffic when feasible. This is supported by meters however meters have a limited QoS capability as explained above. The widespread lack of switch support for meters means that there is no guarantee that any OpenFlow switch will support them.

IV. Experimental Study

An experimental approach was adopted to test and analyse quality of service (QoS) performance of video streaming under different scenarios. A SDN based network testbed was implemented using the Mininet emulator version 2.2.1 to conduct multiple tests based on OpenDaylight SDN controller and OpenFlow 1.3 and Open vSwitch 2.5. The environment was implemented in accordance with the system parameter and mechanisms as represented and outlined in Fig. 1. OpenDaylight Controller (ODL) was adopted as the implementation framework following evaluation of the different controllers. ODL is regularly updated in comparison to certain other controllers and also supports all versions of OpenFlow and possesses the required features to enable implementation of a traffic engineering platform in SDN. A key advantage is that the controller possesses multiple extant modules which can be used in conjunction with customized applications.

An initial set of experiments were conducted to evaluate QoS performance of the existing SDN/OpenFlow capabilities based on a high traffic environment using two separate sets of experimental conditions. The experiment supported identification and analysis of issues in terms of QoS and the limitations of SDN/OpenFlow that could be addressed in the next stage. These tests provide a benchmark of QoS metrics (packet delay variance (jitter), delay, packet loss and throughput).

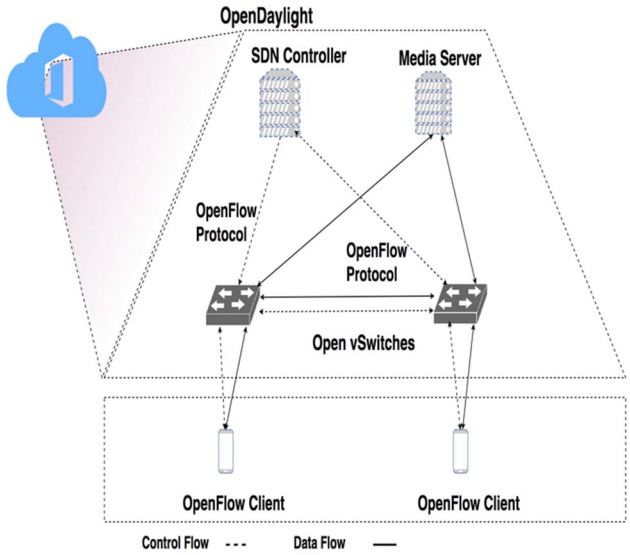

Fig. 1. Testbed for OpenFlow-based SDN Network

V. Proposed Design and Algorithm

This section outlines a novel Meter Band Rate Mechanism for the optimisation of QoS and explains the proposed Meter Band Rate Model and action structure. Fig. 2 provides an overview of the mechanism. The goal is to extend OpenFlow with a meter band rate evaluator. To obtain the dynamic states required by this band rate evaluation engine, a band description language is proposed. The proposed band rate description language has the following characteristics. For example, if there are i = 1 . . c types of classes in a network with priority of types as follows, 1>2>…>c therefore, the network is divided into c slices. The syntax of band description language allows specifying the rate of one slice relative to the rate of one or more other slices in the network. The syntax is as follows:

$$\text{rate}(x) = \{\text{MAX-}\}\{\text{rate}(y)\text{-}\} \; . \; . \; \{\text{rate\_const}\} \qquad (1)$$

Where, MAX is the total available capacity, rate(x) specifies rate of this traffic class, rate(y) specifies rate of any other traffic class and rate_const is any constant. The rate_const allows bandwidth for services to be reserved without having to explicitly assign them to any meter. For example, if there are two kinds of traffic in the network, namely voice and data, the data traffic can only be configured with meter and assigned the necessary bandwidth for voice traffic to rate_const. To illustrate the usefulness of this approach, an example of two flows in the network may be considered. One type of flow is voice traffic and the other is iSCSI traffic. Voice traffic has a fixed rate and therefore can be assigned a constant rate. Next, to assign rate for iSCSI traffic, rate(y)=rate(voice) is used and the rate_const component is not required. In this method, the band action is either DROP or ALLOW. Thus, using the band rate description language which is very simple to implement can overcome the limitation of existing meters and per-port queues which only allow constant max-rate and min-rate limiting. This scheme will allow maximisation of throughput while guaranteeing QoS requirement of each individual slice. Every flow will be assigned an OpenFlow meter by the SDN controller during flow installation phase.

In the SDN switch a "meter_id" uniquely identifies a meter within switch local scope. A "meter_id" is associated with every flow-entry. OpenFlow meters use meter bands to define behavior of meters on packets. This behavior is controlled by meter band type. Presently three band types are supported: OFPMBT_DROP; OFPMBT_DSCP_REMARK, OFPMBT_EXPERIMENTER. The third type (i.e. OFPMBT_EXPERIMENTER) is used to define a band rate evaluator in this model. The main components of OpenFlow meter bands are type, rate, burst and counters. The meter band is extended to incorporate new components: *constraint* and *guaranteed rate*. The *constraint* is a list of "meter_id" that has higher priority than this meter and defined using band description language given in Equation (1). More specifically, the *constraint* list of "band rate evaluator" is obtained from *rate(y) and rate_const* of "band description language". The *guaranteed rate* specifies the bandwidth guarantee for this meter. The *measure* field reflects the measured data rate by the meter. The above values can be specified in KBPS or packets per second as set in meter flags. As shown in Fig. 2 the flow-entries are associated with meters. Flow 1 is associated with Meter 1 having meter id "meter_id1" and Flow 2 is associated with Meter 2 having meter id "meter_id2". In the proposed architecture, a meter associated with a flow-entry is applied to every incoming packet of the flow at the ingress port. This incorporates a band rate evaluator logic, where the meter evaluates the band rate at which it should drop the packet. If the evaluated band rate is greater than present rate, the packet is passed, otherwise it is dropped.

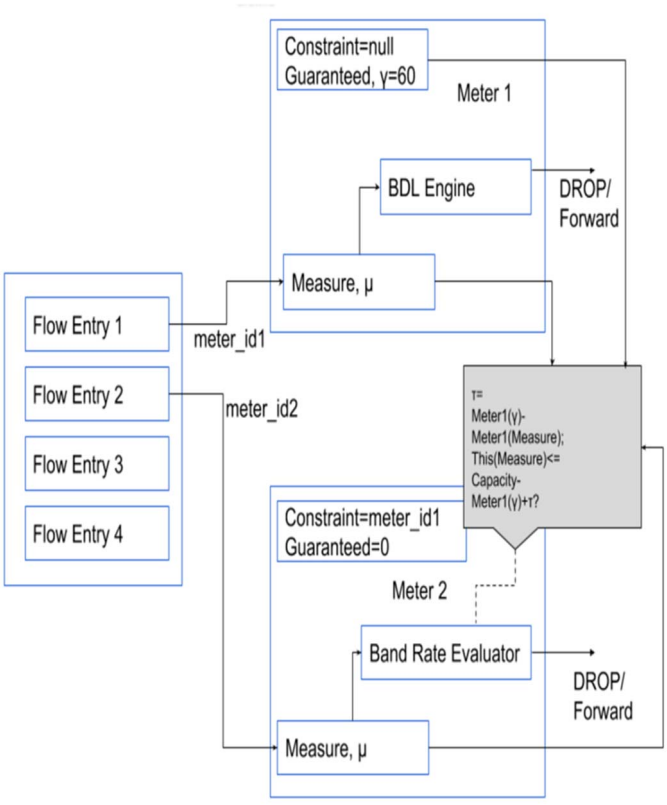

Fig. 2. Meter Band Rate Model

The evaluation process is implemented using a process based on *guaranteed* and *measure* components which are also called *superior constraint*. To explain further, the band

rate evaluator of a meter will take into consideration the *guaranteed* and *measure* parameters of the meters specified in the *constraint* field. As the meters specified these parameters are of "higher priority" or "superior" meters, they are referred to as *superior constraints*. The superior constraints are taken from the constraint meter list. Firstly, the measure field is subtracted from the guaranteed field of each meter in the constraint list, which provides the free bandwidth of each constraint meter; secondly, these free bandwidths are added; thirdly, the band rate evaluator adds the guaranteed fields and subtracts the result from the total port capacity; fourthly, the result from the second step is added to the result from the third step. Finally, the result from the fourth step is compared with the measure field; if greater, the packet is allowed to pass. For the process of compilation of the band description language into the constraint list used by band rate evaluator there are three types of traffic: voice, video and data. The expression for the data rates of these traffic types in band description language can be expressed as follows:

rate(voice) = 32Kbps (const)
rate(video) = 1Mbps
rate(data) = MAX – rate(video) – 32Kbps

The rate_const (=32Kbps) is used in expressing rate (data). No meters will be needed for voice traffic. Thus, there are two meters: meter_video and meter_data. The *constraint* and *guaranteed* fields of these meters are as follows:

meter_video: {constraint = null, guaranteed = 1Mbps}
meter_data: {constraint = meter_video, 32Kbps}

VI. Implementation of Meter Band Rate Evaluator

The proposed solution was implemented in OpenFlow Software Switch 1.3 (ofsoftswitch13). The implementation considers a simplified scenario of the proposed mechanism where one traffic is classified as priority traffic and where the remainder of the traffic is considered non-priority traffic. The core algorithm developed for this implementation is presented in Table I which provides a top level view of the core functions to realize the key components of the meter band rate evaluator. The full algorithm is deployed within the OpenFlow Software Switch (1.3) which was subject to modifications to enable the meter band rate evaluator.

The OFLIB component was modified across five files. Every OpenFlow message represented by the Oflib has a common header. This header struct contains only one member, which is the message type information. The initial struct for every message allows for the implementation of two general functions: marshalling (packing) and unmarshalling (unpacking). Ofl-struct.h was modified to redefine the meter_configuration to allow for meter statistics by adding a number of fields to monitor parent free tokens. A flag in meter configuration was added to facilitate enabling or disabling of the band evaluation engine.

Oflib-messages were modified to facilitate enabling or disabling of the band evaluation engine. Ofl-structs-print was adapted to allow for printing meter statistics, configuration and band evaluator status. The pack and unpack functions were used in Ofl-messages to transmit the new modifications through wire format. The udatapath component, specifically meter_entry.c is where the core of the algorithm is deployed to implement the meter band rate evaluation to enable parent monitoring support and to apply entries. The meter table is an element that addresses the performance of simple QoS operations.

TABLE I. MBE Core Algorithm

Handle packet at meter
Input: *packet, monitoringPeriod* Output: *decision* WHILE *parent* ← *nextParent* DO   IF *free(parent)* < *free*   *free* ← *free* + *free(parent)*   ENDIF ENDWHILE elapsedTime ← currentTime-startTime IF *elapsedTime = monitoringPeriod*   totalFree ← free   *startTime* ← *currentTime* ENDIF IF *available(meter)* > *size(packet)*   *decision* ← *ALLOW*   *available(meter)* ← *available(meter)-size(packet)* ELSIF *free* > *size(packet)*   *decision* ← *ALLOW*   *free* ← *free-size(packet)* ELSE   *decision* ← *DROP* ENDIF

This aspect of the component was modified to allow for additional fields that can allow for the actions defined in the new mechanisms to be applied. The meter table has responsibility for the creation and description and modification of meter entries as well as maintaining the counters for the statistics of packets processed. It is also core to the processing of packets according to the band operation. Meter_table.c was updated to enable the evaluation engine and modification of the structure size from 16 to 20. The design of the new mechanisms also required modification to the utilities/dpctl component to modify the BEEFLAGS. The Dpctl component allows for management and debugging of the tables of a single OpenFlow switch. It circumvents the need to add debugging code in a controller application. Dpctl uses Oflib to create and receive switch messages and also to print their contents and makes it possible to query the current flow table state.

VII. Results

Two sets of experiments were conducted with and without the Meter Band Rate Evaluator (MBE). The first experiment collected and analysed data on QoS in a simulated congested environment without the MBE to benchmark performance. Following installation and configuration of the MBE a second experiment was conducted to analyse the capability of the new solution to optimize QoS performance. The tests conducted in experiment 1 were repeated under the same network conditions over 4.3 minute periods. The average performance was calculated for each of the QoS measures and for each metric there was a significant improvement for delay, jitter, packet loss and throughput. A comparison between experiment 1 and experiment 2 results for each metric is presented in the following charts.

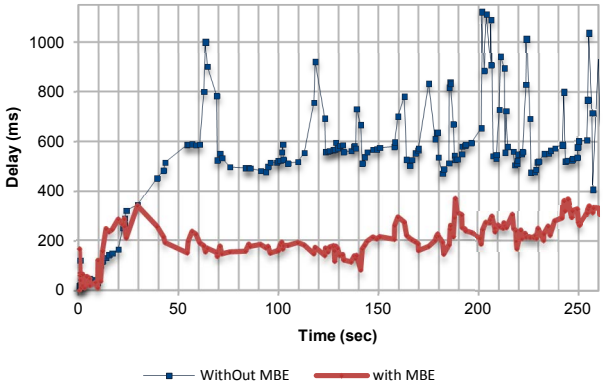

Fig. 3 Delay Performance with and without Meter Band Evaluator

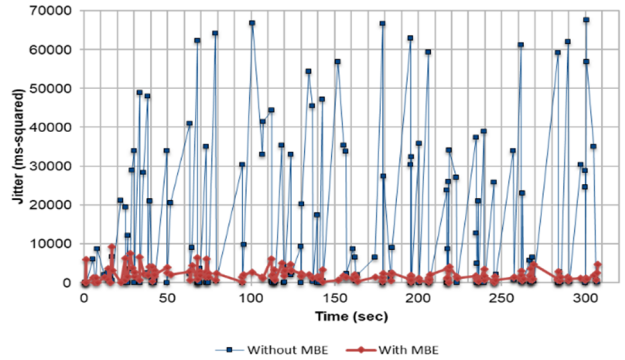

Fig. 4 Jitter Performance with and without Meter Band Evaluator

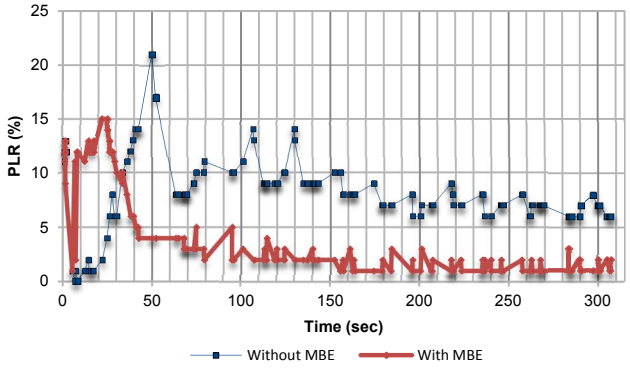

Fig. 5 Packet Loss Ratio Performance with and without Meter Band Evaluator

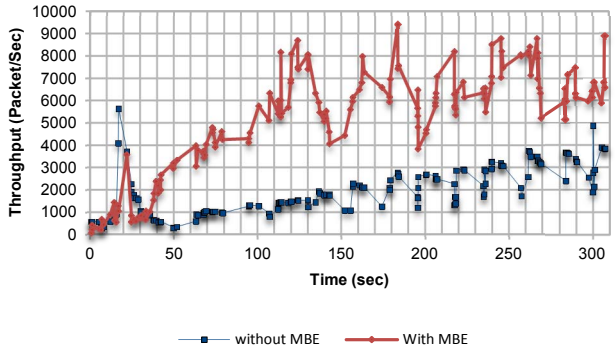

Fig. 6 Throughput with and without Meter Band Evaluator

Delay performance was enhanced significantly over the period. Fig. 3 shows a significant decrease in delay time over the 260 seconds period. Delay for experiment results without the MBE enabled ranged on average between 500-1100 ms delay compared to between 80-360 ms when the MBE was enabled. The results for average delay reveal an average delay of 5,467ms without the MBE compared to an average delay of 206.6ms with the MBE enabled. This impact of the MBE was evident for the other three performance measures which provide validation for this solution to significantly enhance OpenFlow QoS performance. A comparison of jitter data in Fig. 4 between the two modes showed that jitter was significantly minimized when the MBE was enabled, with a median average jitter of 1430 compared to 1906.5 when the MBE was disabled. Packet loss was also significantly reduced when the MBE was enabled. The results in Fig. 5 show that while packet loss without the MBE was on average across all the tests ranging between 5-15%, it was consistently under 5% when the MBE was enabled. The median average PLR was 8.3 compared to 2.1% when the MBE was active. The MBE also improved throughput performance, another key metric for QoS performance of video quality. Fig. 6 shows a marked increase in throughput in excess of 4000 packets/sec and reaching up to 9000 packets/sec which was significantly higher than without the MBE. The experiment results for packets/sec without the MBE fluctuated between 300-4000 packets/sec. For all four metrics the percentage change when the MBE was enabled was significant. Firstly, this represented a 94% average decrease in delay indicating that the MBE had substantially enhanced performance on this measure. For jitter a mean percentage decrease of more than 800% was indicated and accounting for extreme values a median average of 32% reduction in jitter. For packet loss ratio the mean and median average percentage reduction was 285% and 325% respectively. Finally, for throughput the MBE resulted in a mean and median average increase of 200% and 189% respectively. These results provide significant support and validation for the effectiveness of the MBE to enhance the native capability of OpenFlow QoS performance based on the measures tested.

VIII. Discussion

This research explored a novel solution for quality of service performance for streaming mission-critical video data in OpenFlow SDN networks. A Meter Band Rate Evaluator (MBE) mechanism was proposed. The QoS solution proposed focused on enhancing the native capability of OpenFlow and OpenDaylight to support QoS based on a new band rate description language to improve the native QoS capability of OpenFlow and OpenDaylight. The mechanism was verified through a simulated experiment in an SDN testbed. The results revealed a significant percentage increase in QoS performance when the MBE was enabled. These findings provide support and validation for the effectiveness of the MBE to enhance the native capability of OpenFlow and OpenDaylight for efficient QoS provision. This solution contributes to recent works focused on the lack of QoS guarantee which have explored marking data-plane packets and management at the control plane automatic manager for monitoring requirements of virtual networks [16] or a slice and forward method [17]. However such solutions do not examine the problem of supporting QoS in OpenFlow natively. Similarly other research conducted in this area fails

to address the limitation of OpenFlow protocol to support QoS [16, 18, 19, 20].

The solution extended OpenFlow with a meter band rate description language to improve the native capability of OpenFlow to support QoS efficiently. QoS requires a hierarchical relationship between different traffic classes. A higher position in the hierarchy is accorded to traffic classes with high priority. This approach is flexible as the relationship between any two or more classes in terms of QoS priority can be defined. In OpenFlow, meter band rates specify the point where meter band should be applied. Presently, meter band rates can be modified by OFPT_METER_MOD message and are constant 32-bit values specified either in packets/sec or kb/s. The constant value band rate limits the ability of meters to support an intelligent QoS scheme. This is because the flows in the network do not exist in isolation but rather co-exist with a large number of other networks. Therefore, in exploring the possibility of an expressive band description language, specification of band rates can be made that take other flows into account and help achieve intelligent QoS implementation in OpenDaylight networks powered by OpenFlow. The syntax of band description language allows the specification of the rate of one slice relative to the rate of one or more other slices in the network. Therefore, using the band rate description language provides a simple mechanism to overcome the limitation of existing meters and per-port queues which only allow constant max-rate and min-rate limiting. This paper demonstrates that the scheme is effective in maximizing throughput while guaranteeing the QoS requirement of each individual slice.

IX. Conclusion

This paper tested and validated a model for improving QoS by implementing a meter band rate mechanism to improve the native capability of OpenFlow and OpenDaylight for efficient QoS provision. The solution advanced in this research addresses the gap in QoS support in OpenFlow to improve the native capabilities of OpenFlow and OpenDaylight to support QoS. The solution proposed to exploit cloud computing by using emerging technologies such as Software Defined Networking (SDN) to enhance QoS to significantly increase performance across metrics for video streaming and multimedia traffic. The solution advances a new band rate description language that is simple to implement which can overcome the limitation of existing meters and per-port queues which only allow constant max-rate and min-rate limiting. The findings in this study verify the potential of the meter band rate mechanism to optimize QoS and provides further research avenues in this area.

Acknowledgment

The research has been supported and sponsored by the Ministry of Interior in the United Arab Emirates as part of an international research programme.

References

[1] M. Liyanage, A. Gurtov, and M. Ylianttila, *Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture*. London: John Wiley & Sons, 2015.

[2] D.L.C Dutra, M. Bagaa, T. Taleb, and K. Samdanis, "Ensuring end-to-end QoS based on multi-paths routing using SDN technology," in *GLOBECOM IEEE Global Comms Conf*., 2017, pp. 1-6.

[3] M. Banikazemi, D. Olshefski, A. Shaikh, J. Tracey, and G. Wang, "Meridian: An SDN platform for cloud network services," *IEEE Comm. Magazine,* vol 51, pp. 120-127, 2013.

[4] J. Dunlop, D. Girma, and J. Irvine, *Digital Mobile Communications and the TETRA System*. London: John Wiley & Sons, 2013.

[5] R. Blom, P. de Bruin, J. Eman, M. Folke, H. Hannu, M. Näslund, M. Stålnacke, and P. Synnergren, "Public safety communication using commercial cellular technology." in *IEEE Next Gen Mobile Applications, Services and Technologies,* NGMAST'08., 2008, pp. 291-296.

[6] M. Jammal, T. Singh, A. Shami, R. Asal,. and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks*, vol 72, pp.74-98, 2014.

[7] O. Coker, and S. Azodolmolky, *Software-Defined Networking with OpenFlow*. London: Packt Publishing, 2017.

[8] J. Shi, and S.H. Chung, "A traffic-aware quality-of-service control mechanism for software-defined networking-based virtualized networks," *Int. J. Dist. Sensor Networks*, vol 13, pp.1-13, 2017.

[9] J. Huang, P.J. Wan, and D.Z. Du, "Criticality-and QoS-based multi-resource negotiation and adaptation," *Real-Time Systems*, vol 15, pp. 249-273, 1998.

[10] T. Reza, *Learning OpenDaylight: A Gateway to SDN (Software Defined Networking) and NFV (Network Functions Virtualization) Ecosystem*. London: Packt Publishing Ltd, 2017.

[11] M. Sisov, "Building a software-defined networking system with OpenDaylight controller", BSc Thesis, Inf. Techn., Helsinki Metro. Uni. of Applied Sciences, 2016.

[12] S. Jivorasetkul, M. Shimamura, and K. Iida, "End-to-end header compression over software-defined networks: A low latency network architecture," in 4th International Conference on Intelligent Networking and Collaborative Systems (INCoS), pp. 493–494, 2012.

[13] F. Hu, Q. Hao, and K. Bao, K, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Comm. Sur. & Tutor.*, vol 16, pp. 2181-2206, 2014.

[14] H.E. Egilmez, S. Civanlar, and A.M Tekalp, "An optimization framework for QoS-enabled adaptive video streaming over OpenFlow networks," *IEEE Trans. Multimedia*, vol 15, pp. 710–715, 2013.

[15] A Mirchev, "Survey of concepts for QoS improvements via SDN," in Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM), Summer Semester 2015, Munich, Germany, pp. 33-39, 2015.

[16] J. Shi, and S.H. Chung, "A traffic-aware quality-of-service control mechanism for software-defined networking-based virtualized networks," *Int. J. Distrib. Sensor Networks*, vol 13, pp.1-13, 2017.

[17] Y. Lu, B. Fu, X. Xi, Z. Zhang, and H. Wu, "An SDN-based flow control mechanism for guaranteeing QoS and maximizing throughput," *Wireless Personal Comm.*, vol 97, pp.417-442, 2017.

[18] D. D'souza, K.P. Sundharan, S. Lokanath, V. Mittal, L. Perigo, and R. Hagens, *Improving QoS in a Software-Defined Network*. Boulder, Capstone: University of Colorado, 2016.

[19] R. Thenmozhi, T. Preethi, S. Sadhana, V. Shruthi, and B. Yuvarani, "Integrating multimedia services over software defined networking," *Int. Res. J. Eng. and Tech (IRJET)*, vol. 4, 2017.

[20] L. Han, S. Sun, B. Joo, X. Jin, and S. Han, "QoS-aware routing mechanism in OpenFlow-enabled wireless multimedia sensor networks," *Int. J. Dist. Sensor Networks*, vol 12, p.9378120, 2016.

[21] I.F. Akyildiz, S.C. Lin, and P. Wang, "Wireless software-defined networks (W-SDNs) and network function virtualization (NFV) for 5G cellular systems: An overview and qualitative evaluation," Computer Networks, vol 93, pp.66-79, 2015.