Design and Implementation of SDN-based 6LBR with QoS Mechanism over Heterogeneous WSN and Internet

Tsung-Han Lee, Lin-Huang Chang* and Wei-Chung Cheng
Department of Computer Science, National Taichung University of Education
Taichung, Taiwan
[e-mail: thlee@mail.ntcu.edu.tw, lchang@mail.ntcu.edu.tw*, bcs103105@gm.ntcu.edu.tw]
*Corresponding author: Lin-huang Chang

Received August 31, 2016; revised December 11, 2016; accepted February 7, 2017; published February 28, 2017

Abstract

Recently, the applications of Internet of Things (IoTs) are growing rapidly. Wireless Sensor Network (WSN) becomes an emerging technology to provide the low power wireless connectivity for IoTs. The IPv6 over low-power wireless personal area networks (6LoWPAN) has been proposed by IETF, which gives each WSN device an IPv6 address to connect with the Internet. The transmission congestion in IoTs could be a problem when a large numbers of sensors are deployed in the field. Therefore, it is important to consider whether the WSN devices have be completely integrated into the Internet with proper quality of service (QoS) requirements. The Software Defined Network (SDN) is a new architecture of network decoupling the data and control planes, and using the logical centralized control to manage the forwarding issues in large-scale networks. In this research, the SDN-based 6LoWPAN Border Router (6LBR) is proposed to integrate the transmission from WSNs to Internet. The proposed SDN-based 6LBR communicating between WSNs and the Internet will bring forward the requirements of end-to-end QoS with bandwidth guarantee. Based on our experimental results, we have observed that the selected 6LoWPAN traffic flows achieve lower packet loss rate in the Internet. Therefore, the 6LoWPAN traffic flows classified by SDN-based 6LBR can be reserved for the required bandwidth in the Internet to meet the QoS requirements.

Keywords: WSN, SDN, 6LoWPAN, 6LBR, Contiki

A preliminary version of this paper was presented at APIC-IST 2016, and was selected as an outstanding paper. This version includes a concrete analysis and supporting implementation results on heterogeneous networks. This research was supported by research grants from Ministry of Science and Technology, Taiwan (MOST 105-2221-E-142 -001 -MY2 and 105-2221-E-142 -002 -MY2).

https://doi.org/10.3837/tiis.2017.02.025 **ISSN : 1976-7277**

1. Introduction

Wireless sensor network (WSN) [1] is widely used in Internet of Things (IoTs) [2] applications, such as industrial automation, environmental monitoring, and medical healthcare. These applications implemented with a larger numbers of small wireless sensor devices cover a wide area and require low power consumption. Many wireless sensor devices follow the IEEE 802.15.4 standard [3] which requires smaller size, less power consumption, faster computing and Internet networking capabilities. When these wireless sensor devices become IP-enabled devices by using Internet Protocol version 6 (IPv6) with 1280 bytes maximum transmission unit (MTU) to give each device an IP address, the IEEE 802.15.4 frame with maximum length up to 127 bytes cannot carry the whole IPv6 packet at once. Thus, the 6LoWPAN [4] adaptation layer has been proposed by IETF between the MAC and the network layers in order to support IPv6 over the IEEE 802.15.4 standard by providing header compression and fragmentation.

The Software Defined Networks (SDN) [5] is a platform to reduce the complexity of network elements. The essential ideas of SDN are the decoupling of the data and control planes from networks, and centralized control and management of the forwarding traffic in large-scale networks. The advantage of SDN is to reduce the management and maintenance costs by using the SDN controller to configure the traffic forwarding rules of SDN-enable switches dynamically in SDN networks. The OpenFlow [6] protocol defines the communication standard in SDN environments.

In this paper, we propose the design and implementation of SDN-enable 6LoWPAN Border Router (6LBR) [7] to recognize 6LoWPAN traffic flows and provide the higher transmission performance in 6LoWPAN for the heterogeneous wireless sensor networks and software-define networks. This research focuses on providing a guarantee bandwidth in SDN networks to enhance the transmission performance of 6LoWPAN traffic flows.

The rest of the paper is organized as follows. In Section 2, we review some of the relevant researches on 6LoWAPN and SDN. In Section 3, we describe the proposed SDN-enabled 6LBR for the wireless sensor networks. Section 4 aims at the transmission performance analyses for the best-effort 6LoWPAN flows and SDN bandwidth-guarantee 6LoWPAN flows. Finally, we draw conclusions and future works in Section 5.

2. Related Works

This paper established an environment for testing the packets across heterogenous networks. This section takes a deep discussion of the related researches, such as IEEE 802.15.4 and 6LoWPAN, Software-Defined Network and OpenFlow, 6LoWPAN border router, and the quality of service (QoS) mechanism in OpenFlow.

2.1 IEEE 802.15.4 and 6LoWPAN

Internet of Things trend to make each smart object to connect to the Internet depending on the IP-enabled embedded devices in wireless sensor network. These smart objects can connect different sensors to detect or collect environmental data. Once establishing an application in wireless sensor network, the huge number of nodes will run out the few existing IPv4 addresses. Therefore, using IPv6 to solve this problem is inevitable. IPv6 is the next generation Internet protocol which has many advantages, such as rich addressing resources,

high security, IP auto-configuration, and high mobility. It meets the need of large number addressing in IoTs. To encapsulate the IPv6 packet (with 1280 bytes MTU) into the IEEE 802.15.4 frame with maximum length up to 127 bytes, the 6LoWPAN protocol with an adaptation layer between IPv6 and IEEE 802.15.4 has been proposed by IETF, as shown in **Fig. 1**. The adaptation layer supports fragmentation, packet reassembly and header compression in 6LoWPAN. Through the 6LoWPAN adaptation layer, the IPv6 and IEEE 802.15.4 formats could be communicated and transferred. **Fig. 2** shows the packet format of uncompressed IPv6 header. Two compressing methods, HC1 and IPHC, of IPv6 header are shown in **Figs. 3** and **4**, respectively. In order to provide the transparency between WSN and SDN networks, we apply the 6LoWPAN uncompressed IPV6 header in our experiments.

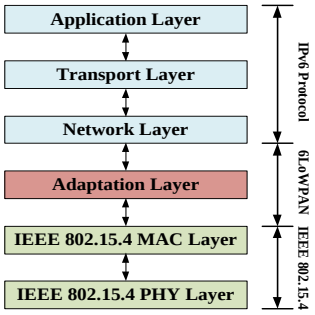

Fig. 1. 6LoWPAN Protocol Stack

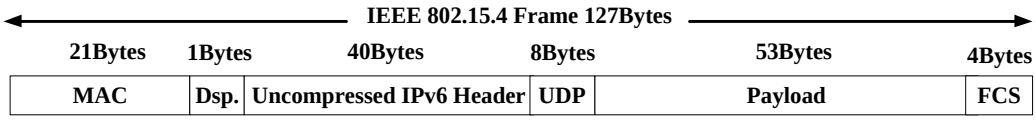

Fig. 2. The 6LoWPAN packet stack with uncompressed IPv6 header

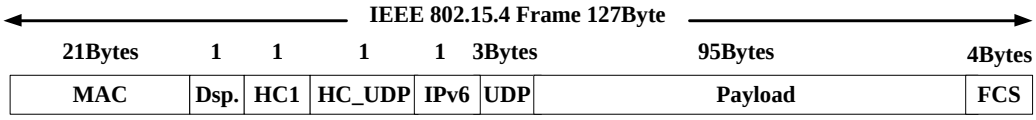

Fig. 3. The 6LoWPAN packet stack with HC1 compression header

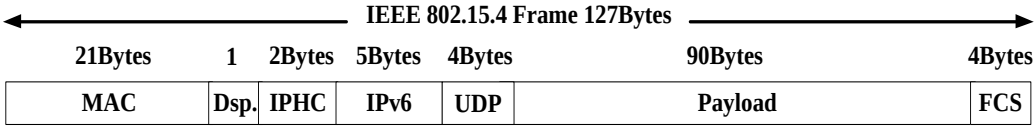

Fig. 4. The 6LoWPAN packet stack with IPHC compression header

2.2 Software-defined network and OpenFlow

SDN is a new networking architecture in recent years. The SDN networking concept is shown in **Fig. 5**. The SDN networking concept can be used in many different themes such as software-defined mobile networks [8] and software-defined wireless sensor networks [9].

However there is not simple extension from Internet to mobile network or WSN, because the radio access in mobile network or WSN is more complicated than wired network.

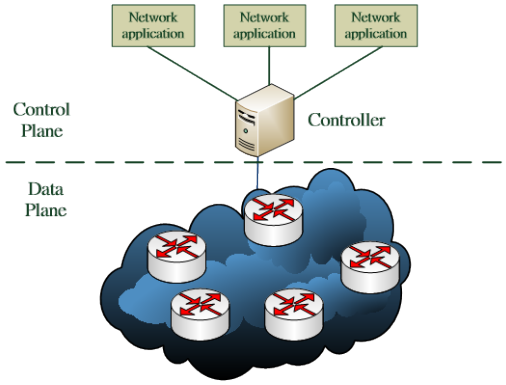

Fig. 5. SDN networking concept

OpenFlow is an open source protocol managed by the Open Networking Foundation (ONF). ONF defines OpenFlow as the communication interface between the controlling and forwarding layers in the SDN architecture. OpenFlow allows developer to design and define behaviors of OpenFlow compliant switches. In OpenFlow protocol, all packet transmission is defined by flow tables. Therefore, adding, deleting and modifing flow entry will control the actions of packet flows. In this paper, we use the Open vSwitch [10] as the OpenFlow compliant switch to realize SDN QoS setting, traffic monitoring, exceptional traffic flow, and other requirements. The research in [11] has been implemented the Raspberry-Pi embedded device to an Open vSwitch in the small-scale network. The research in [12] exploited the quantities returned by OpenFlow to predict the effect of the re-routing operation on less congested paths before they experience QoS violation.

2.3 6LoWPAN Border Router

Contiki [13] is an open-source operating system for low-power wireless devices which implement the 6LoWPAN adaptation layer for IEEE 802.15.4. The 6LoWPAN border-router (6LBR) is a necessary adaptation to establish the connection between 6LoWPAN devices and Internet. In [14] and [15], the 6LoWPAN border router were implemented into a 32-bit ARM Cortex M3 microcontroller with network-enabled operating system. The authors also tried to demonstrate the dynamic address configuration in 6LBR. The research in [16] proposed a smart constrained application protocol (CoAP)-based gateway with a border router for home safety services for seamless connection between a 6LoWPAN and the Internet.

2.4 The QoS mechanism in OpenFlow

The network traffics and applications, such as YouTube, KKBOX, Spotify, Twitch, and Skype, consume more and more bandwidth on Internet. The QoS of network applications, such as image lag and voice jitter, are affected by network bandwidth, delay, jitter, and packet loss rate. Therefore, to mitigate the network delay, jitter or packet loss, and/or provide acceptable bandwidth for different Internet applications would be an important issue on QoS

mechanism. Providing QoS service in the SDN network architecture through OpenFlow protocol could be one suitable solution. In [17], the authors proposed a QoS-aware Network Operation System (QNOX) in SDN architecture which provided several modules such as server element, control element, management element, and cognitive knowledge element to deploy the experimental architecture. The experimental results showed that the proposed architecture can be used in large-scale network. In [18], the authors employed Mininet to establish two different bandwidth-limited paths. They applied Open vSwitch (OVS) as the OpenFlow switch with Linux-HTB [19] to realize QoS service based on Traffic Control[20] in Linux. **Fig. 6** shows the basic topology of Linux-HTB.

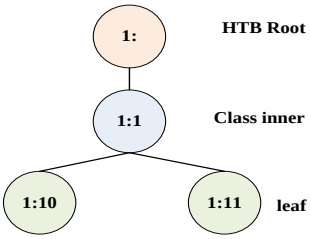

Fig. 6. Linux-HTB basic topology

Although it has been many researches conducted on the QoS issues in SDN, there are only few literatures available on SDN-based QoS mechanism for heterogeneous WSN and Internet. In this paper, by inserting the QoS Tag into 6LoWPAN WSN packets, the implemented SDN-enable 6LBR will recognize 6LoWPAN traffic flows to provide different QoS serivces in SDN-enable Internet.

3. Design of proposed Test-bed architecture

3.1 6LoWPAN Device

Fig. 7 shows an ATmega128RFA1 [21] wireless sensor node with Contiki OS which is used as the 6LoWPAN UDP client in the proposed test-bed. The ATmega128RFA1 microcontroller is a new generation of wireless integrated-chips being fully compatible with the Contiki OS and the IETF 6LoWPAN protocol.

Fig. 7. Zigduino Sensor Device

3.2 Proposed SDN-based QoS Mechanism

In this paper we proposed an SDN-based QoS mechanism to provide the extension of QoS service in heterogenous networks. We apply IPv6 flow label to insert QoS Tag in 6LoWPAN. The IPv6 flow label is kept intact after conversion from the 6LoWAPN format in WSN to IPv6 format in SDN Internet. **Table 1** lists the parameters of QoS Tag corresponding to 6LoWPAN.

It is the goal in this paper to compare the performance of different QoS levels with high priority, low priority, and best-effort in a congestion network.

Table 1. The parameters of 6LoWPAN QoS Tags

QoS level	Priority	QoS Tag	6LoWPAN IPv6 flow label
High	4	100	0x00100
Low	2	200	0x00200
Best-effort	0	0	0

Figs. 8 and **9** are the wireless transmitted packets captured by sniffer for unfragmented and fragmented packets, respectively. The IPv6 flow label in 6LoWPAN marked in red square is known as QoS Tag being used to confirm whether the packet is kept intact while transmitted from WSN with 6LoWPAN via air to the SDN Internet with IPv6.

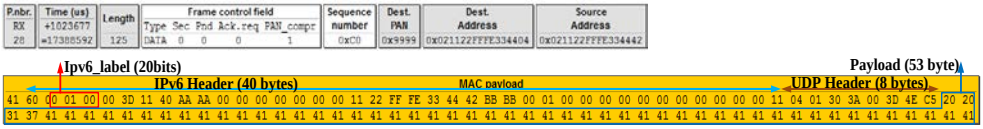

Fig. 8. Unfragmented packet in 6LoWPAN

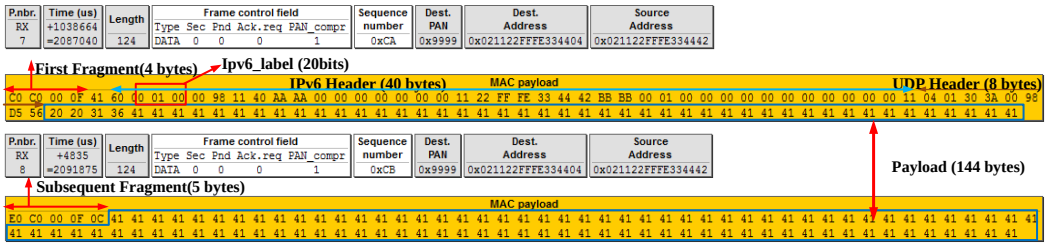

Fig. 9. Fragmented packets in 6LoWPAN

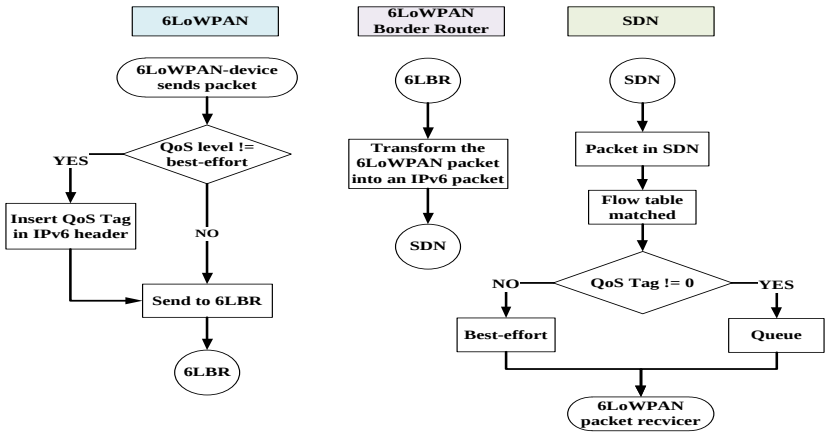

Fig. 10. The flow chart of proposed SDN-based QoS mechanism

Fig. 10 is the processing flow chart of proposed SDN-based QoS mechanism. As shown in **Fig. 10**, the 6LoWPAN UDP client first determines the QoS level of transmitted data by inserting the corresponding QoS Tag into the packet followed by the conversion of the packets from 6LoWPAN to IPv6 by 6LBR. Finally, the corresponding QoS Tag of converted packet is

recognized and processed for queueing or best-effort which is defined by SDN controller to achieve proper QoS service.

3.3 Proposed SDN-enabled 6LBR Test-beds

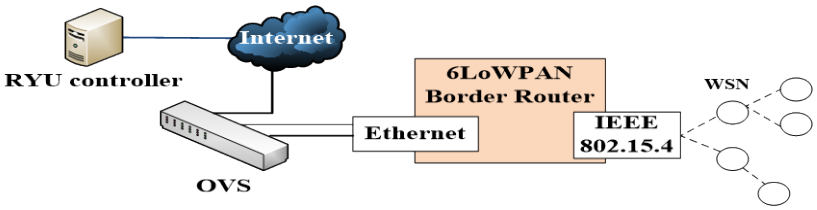

Fig. 11. The network topology of the proposed test-bed

Fig. 11 shows the basic network topology of the proposed test-bed. The SDN-enable 6LBR provides a communication between WSN and the IPv6 networks. The 6LBR acts as a router that separates the IPv6 and 6LoWPAN networks into different domains. Through OpenFlow protocol, SDN Controller can assign the actions for packets carrying the data of SDN-based QoS mechanism. In this paper, we will exam two scenarios in WSN and corresponding experimental environments, as shown in **Figs. 12** and **13**, which will be illustrated in more details in the next two sub-sections. We assume that in our experiment the CPU performance, memory and hard disk size are not the bottleneck. We use Iperf to construct the scenario where multiple UDP traffic flows from Iperf client to Server through two SDN-enable switches. The Iperf UDP client generates four UDP background flows, 2.5Mbits/sec each, to reach the link bandwidth limit of 10Mbits/sec. For the first scenario, we deploy a test-bed with single 6LoWPAN UDP client in WSN. For the second scenario, we extend the 6LoWPAN UDP client to multiple clients (in this scenario, it is three clients) with end-to-end transmission timers, and different SDN-based QoS Tags. The detailed experimental setups will be discussed in the next two sub-sections.

3.3.1 Single 6LoWPAN UDP client

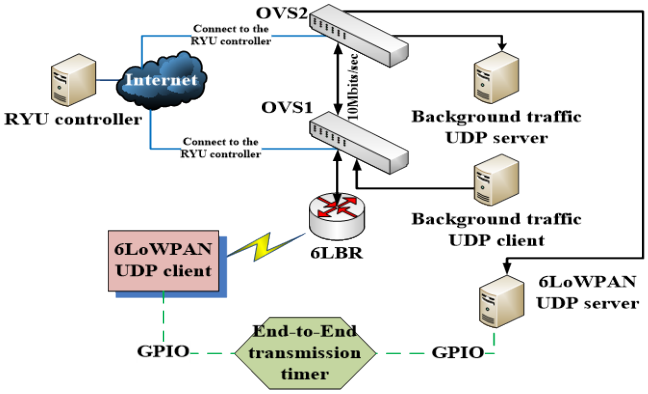

Fig. 12. SDN-based QoS management test-bed of single 6LoWPAN UDP client

As shown in **Fig. 12**, two SDN-enable switches, OVS1 and OVS2 are connected to the RYU controller using OpenFlow protocol. This proposed architecture supports end-to-end IP communication between any IPv6 host in SDN Internet and sensor device in WSN. The WSN sensor device, 6LoWPAN UDP client, sends IPv6/UDP packets encapsulated in IEEE 802.15.4 frames with 6LoWPAN header compression. The SDN-enabled 6LBR provides bridging capabilities and allows an external IPv6 host to discover and connect to sensors using

OpenFlow port forward and protocol conversion between IPv6 and 6LoWPAN. The SDN controller RYU [22] establishes SDN flow queues for corresponding QoS Tags sending from the 6LoWPAN UDP client (Zigduino node) to the 6LoWPAN UDP server to provide different QoS services.

3.3.2 Multiple 6LoWPAN UDP clients

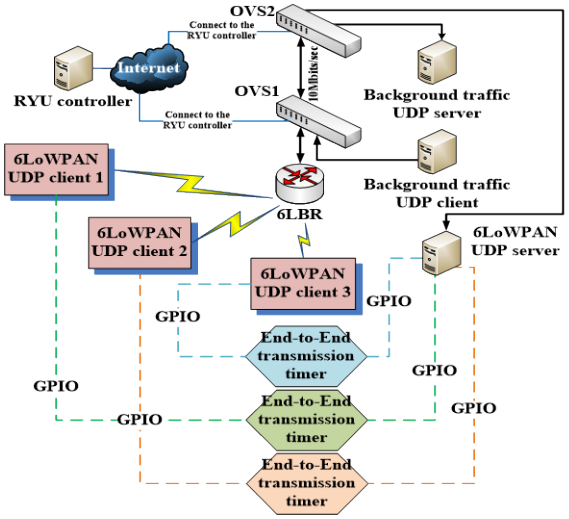

Fig. 13. SDN-based QoS management test-bed of multiple 6LoWPAN UDP clients

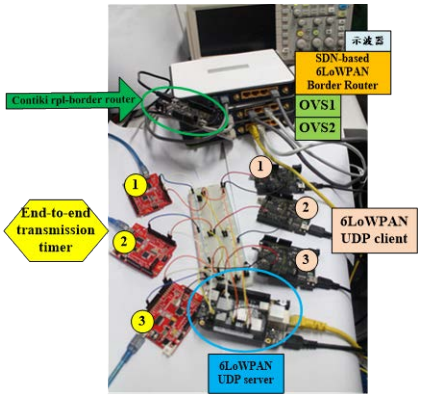

Fig. 14. The real 6LoWPAN test-bed

In order to emulate real 6LoWPAN UDP traffics from WSN to Internet environments, we further entend our experiments to handle the transmission for multiple 6LoWPAN UDP nodes with different QoS requirements. **Fig. 13** shows the architecture of multiple 6LoWPAN UDP clients test-bed. We deploy three 6LoWPAN UDP clients in WSN, each 6LoWPAN UDP client with its own end-to-end transmission timer and QoS Tag. Three 6LoWPAN UDP clients are triggered at the same time, and while sending out the first IPv6 UDP packet sequentially to make sure each IPv6 UDP packet can connect to 6LBR in one hop. Each 6LoWPAN UDP client records its own transmission time with its own end-to-end transmission timer by using GPIO signal. **Fig. 14** shows the construction of the real test-bed.

3.3.3 Test scenarios

Fig. 15 is the packet format of fragmented 6LoWPAN. The UDP packet will be fragmented if the sensing data is too big to be encapsulated in one 6LoWPAN packet. The first fragmented frame is tagged as FRAG1 and the remaining fragmented frames are tagged as FRAGN. From the experimental measurements, the maximum payload in different fragments are different for 6LoWPAN packets. **Table 2** lists the maximum payload size and packet size for 6LoWPAN packets with different fragments. In this research, we conducted the measurements for 6LoWPAN packets with different data sizes from 1 fragment to 14 fragments. **Table 3** lists four testing environments with different combinations of the transmission with/without background flows and with/without QoS mechanism supports. For each scenario, the 6LoWPAN UDP client (Zigduino node) sends out 1000 IPv6/UDP packets with 1 sec transmission interval to the 6LoWPAN UDP server. The testing flow charts for two

different experimental setups, single and multiple clients, are illustrated as follows respectively.

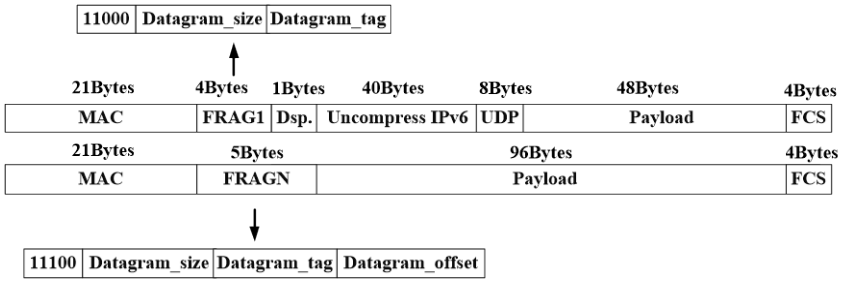

Fig. 15. The packet format of fragmented 6LoWPAN

Table 2. The maximum payload size and packet size for 6LoWPAN packets with different fragments

Fragment(s)	Payload size (bytes)	Packet size (bytes)
1	53	101
2	144	192
3	240	288
4	336	384
5	432	480
6	528	576
7	624	672
8	720	768
9	816	864
10	912	960
11	1008	1056
12	1104	1152
13	1200	1248
14	1232	1280

Table 3. The list of four testing environments

	No Background Flow	4UDP Background Flows
without Qos Mechanism	No Background Flow Environment without QoS Mechanism	4 UDP Background Flows Environment without QoS Mechanism
with Qos Mechanism	No Background Flow Environment with QoS Mechanism	4 UDP Background Flows Environment with QoS Mechanism

Single 6LoWPAN UDP client

Fig. 16 shows the process of SDN-based QoS mechanism for single 6LoWPAN UDP client environment. Depending on the QoS tags, the SDN-based QoS mechanism in 6LBR will take different actions. In single 6LoWPAN UDP client experiment, we assign QoS Tag as 100 to the 6LoWPAN UDP client. The QoS setting in OVS is shown in **Fig. 17**. **Fig. 18** shows the queue setting of SDN-based QoS mechanism, primary setting as 250Kbits/sec in this experiment.

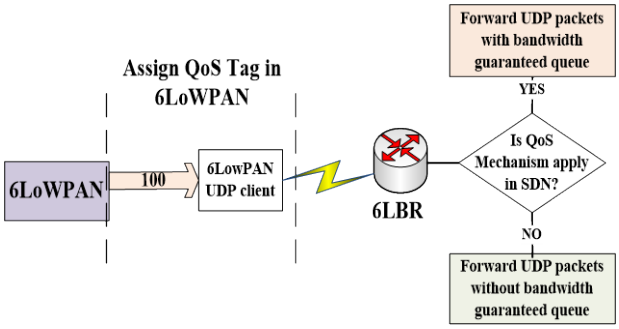

Fig. 16. The process of SDN-based QoS mechanism in single 6LoWPAN UDP client environment

```
root@OpenWrt:~# ovs-vsctl --db=tcp:192.168.1.112:9999 -- set port eth0.3 qos=@newqos -- --
id=@newqos create qos type=linux-htb other-config:max-rate=10000000 queues=111=@q1,112=@q2
 -- --id=@q1 create queue other-config:min-rate=20000 other-config:max-rate=250000 -- --id
=@q2 create queue other-config:min-rate=20000 other-config:max-rate=35000
df62214c-efc2-4d63-ba64-560c90e0ba4b
781f9b74-5365-497b-a84b-8cd050a9500c
aa07762e-4416-4251-a106-0c71927b6a4d
```

Fig. 17. The QoS setting of single 6LoWPAN UDP client in OVS

```
root@OpenWrt:~# ovs-vsctl --db=tcp:192.168.1.112:9999 list queue
_uuid               : 781f9b74-5365-497b-a84b-8cd050a9500c
dscp                : []
external_ids        : {}
other_config        : {max-rate="250000", min-rate="20000"}
```

Fig. 18. The queue setting of of single 6LoWPAN UDP client

Multiple 6LoWPAN UDP clients

Fig. 19 shows the process of SDN-based QoS mechanism for multiple 6LoWPAN UDP clients. In 6LoWPAN, we assign QoS Tag 100 as the high priority to the 6LoWPAN UDP client 1, QoS Tag 200 as the low priority to 6LoWPAN UDP client 2, and QoS Tag 0 as the best-effort to 6LoWAPN UDP client 3. Depending on the QoS tags, the SDN-based QoS mechanism in 6LBR will take different actions. The QoS setting of multiple bandwidth guaranteed queue in OVS is shown in **Fig. 20**. **Fig. 21** shows the queue setting of multiple bandwidth guaranteed queues with high priority queue as 20Kbits/sec and low priority queue as 9Kbits/sec in this experiment.

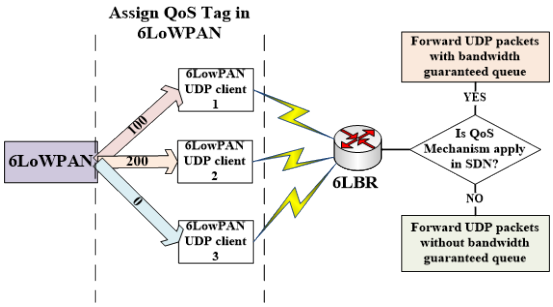

Fig. 19. The process of SDN-based QoS mechanism in multiple 6LoWPAN UDP clients environment

```
root@OpenWrt:~# ovs-vsctl --db=tcp:192.168.1.112:9999 -- set port eth0.3 qos=@newqos -- --id=@new
qos create qos type=linux-htb other-config:max-rate=10000000 queues=111=@q7,113=@q2 -- --id=@q7 c
reate queue other-config:min-rate=20000 other-config:max-rate=20000 other-config:priority=0 -- --
id=@q2 create queue other-config:min-rate=9000 other-config:max-rate=9000 other-config:priority=7
a0cead96-bf66-4c3f-a333-451b6244e256
34be3074-157d-4c6f-8233-18a2a231feb3
e5a0486c-c8b4-4cdd-8ed0-273e3a81345a
```

Fig. 20. The QoS setting of multiple bandwidth guaranteed queue in OVS

```
root@OpenWrt:~# ovs-vsctl --db=tcp:192.168.1.112:9999 list queue
_uuid               : e5a0486c-c8b4-4cdd-8ed0-273e3a81345a
dscp                : []
external_ids        : {}
other_config        : {max-rate="9000", min-rate="9000", priority="7"}

_uuid               : 34be3074-157d-4c6f-8233-18a2a231feb3
dscp                : []
external_ids        : {}
other_config        : {max-rate="20000", min-rate="20000", priority="0"}
```

Fig. 21. The queue setting of multiple bandwidth guaranteed queues

4. Experimental Results and Analyses

This section will present the experimental results of two testing scenarios as explained in section 3. The first testing scenario is the experimental measurements with four testing environments in single 6LoWPAN UDP client test-bed. For the second scenario with multiple 6LoWPAN UDP clients test-bed, we will compare four testing environments to investigate the performance of the proposed SDN-based QoS mechanism.

4.1 Experimental results of single 6LoWPAN UDP client

Table 4 presents the experimental result, in terms of end-to-end delay, packet loss and throughput, of no background flow environment without QoS mechanism for different fragmented packets sending from single 6LoWPAN UDP client to 6LoWPAN UDP server. The experimental results of different fragmented packets for four testing environments are carried out and the analyzed results are discussed in the following.

Table 4. The experimental results of single UDP client without background flow and QoS mechanism

No Background Flow Environment without QoS Mechanism					
Packet			1000 times		Throughput (packet size)
Number of fragmentation	Payload (only data)	Packet size	Average End-to-end transmission time(ms)	packet loss rate	bps
1	53	101	38.214	0.00%	808.000
2	144	192	67.768	0.00%	1536.000
3	240	288	98.631	0.00%	2304.000
4	336	384	129.467	0.00%	3072.000
5	432	480	160.284	0.00%	3840.000
6	528	576	191.102	0.00%	4608.000
7	624	672	221.949	0.00%	5376.000
8	720	768	252.770	0.00%	6144.000
9	816	864	283.574	0.20%	6898.176
10	912	960	314.657	0.00%	7680.000
11	1008	1056	345.782	0.20%	8431.104
12	1104	1152	376.619	0.20%	9197.568
13	1200	1248	407.425	0.00%	9984.000
14	1232	1280	418.781	0.00%	10240.000

Fig. 22 shows the average end-to-end transmission delay for packets sending from single 6LoWPAN UDP client to 6LoWPAN UDP server. The experimental results present four different testing environments, including no background flows with/without QoS mechanism and injecting UDP background flows with/without QoS mechanism. The result shows no significant difference for four different testing environments while the delay increase slightly when there are injected UDP background flows but no QoS mechanism support. **Fig. 23** shows the packet loss rate for four testing environments. The packet loss rate for injecting UDP Background flows environment without QoS mechanism is much higher than other three testing environments. This is because that the injected UDP background flows have occupied most of the OVC available bandwidth and related resources, consequently, resulting in significant packet loss. It reveals that the proposed QoS mechanism provide significant QoS service to prevent the packet loss during conjected network conditions. **Fig. 24** shows the

throughput results for four testing environments. The experimental results show that the throughputs with and without UDP background flows under the support of the proposed QoS mechanism are very close. That means the proposed SDN-based QoS mechanism preserve the bandwidth for the corresponding 6LoWPAN data flows.

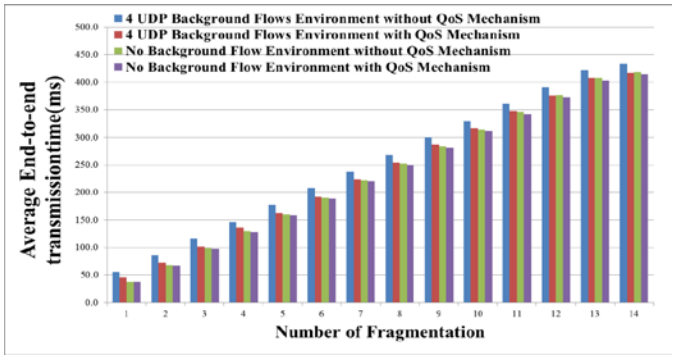

Fig. 22. The average end-to-end delay in four testing environments

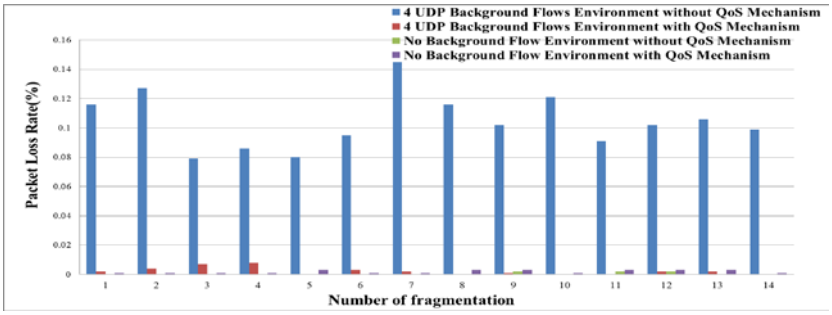

Fig. 23. The packet loss rate in four testing environments

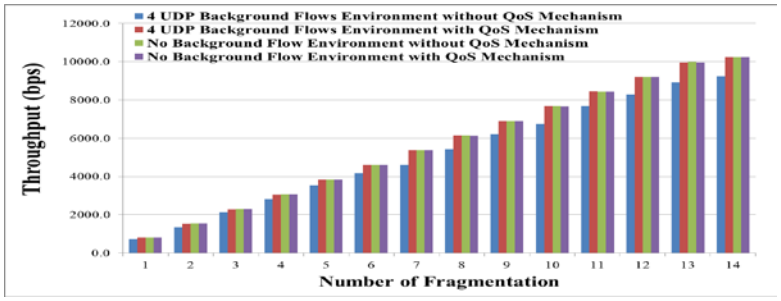

Fig. 24. Throughput in four testing environments

4.2 Experimental results of multiple 6LoWPAN UDP clients

Table 5 presents the experimental result of no background flow environment without QoS mechanism for three different 6LoWPAN UDP clients. The average end-to-end transmission delay and packet loss rate of no background flow environment without QoS mechanism for three different 6LoWPAN UDP clients are shown in **Figs. 25** and **26**, respectively. Without QoS support, the best-effort behavior of UDP packets show irregular packet drop. **Fig. 27** shows the throughput of no background flow environment without QoS mechanism for three different 6LoWPAN UDP clients. This testing environment indeed is an ideal environment with no any interference from background flow.

Table 5. The experimental results of no background flow environment without QoS mechanism

No Background Flow Environment without QoS Mechanism											
Packet			6LoWPAN UDP client 1 (Best-effort)			6LoWPAN UDP client 2 (Best-effort)			6LoWPAN UDP client 3 (Best-effort)		
			1000 times		Throughput (packet size)	1000 times		Throughput (packet size)	1000 times		Throughput (packet size)
Number of Fragmentation	Payload	Packet size	Average End-to-End transmission time(ms)	Packet Loss Rate	bps	Average End-to-End transmission time(ms)	Packet Loss Rate	bps	Average End-to-End transmission time(ms)	Packet Loss Rate	bps
1	53	101	18.455	0.10%	807.192	18.460	0.20%	806.384	18.449	0.20%	806.384
2	144	192	32.324	1.70%	1509.888	32.342	2.00%	1505.280	32.317	2.10%	1503.744
3	240	288	46.606	1.40%	2271.744	46.559	1.90%	2260.224	46.528	2.00%	2257.920
4	336	384	60.803	0.30%	3062.784	60.814	0.30%	3062.784	60.799	0.40%	3059.712
5	432	480	75.082	0.90%	3805.440	75.068	1.20%	3793.920	74.978	1.30%	3790.080
6	528	576	89.993	2.20%	4506.624	89.465	2.20%	4506.624	89.651	3.60%	4442.112
7	624	672	103.587	1.90%	5273.856	103.537	1.80%	5279.232	103.406	3.80%	5171.712
8	720	768	117.813	1.30%	6064.128	117.812	1.30%	6064.128	117.633	1.50%	6051.840
9	816	864	133.496	2.60%	6732.288	131.728	1.80%	6787.584	132.680	2.10%	6766.848
10	912	960	146.096	2.10%	7518.720	146.500	1.60%	7557.120	145.568	1.50%	7564.800
11	1008	1056	159.924	3.20%	8177.664	161.956	2.60%	8228.352	160.264	2.20%	8262.144
12	1104	1152	175.211	3.50%	8893.440	175.340	2.20%	9013.248	174.645	2.80%	8957.952
13	1200	1248	189.169	2.80%	9704.448	189.161	1.90%	9794.304	188.853	3.00%	9684.480
14	1232	1280	195.578	4.90%	9738.240	195.760	4.90%	9738.240	194.990	5.20%	9707.520

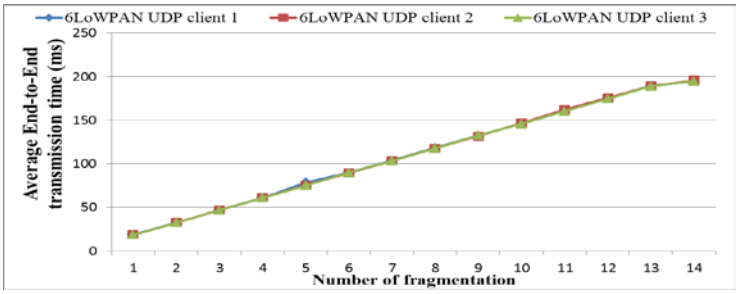

Fig. 25. The average end-to-end delay of no background flow environment without QoS mechanism

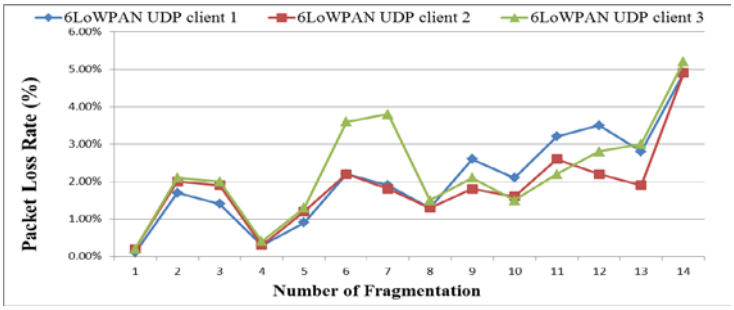

Fig. 26. The packet loss rate of no background flow environment without QoS mechanism

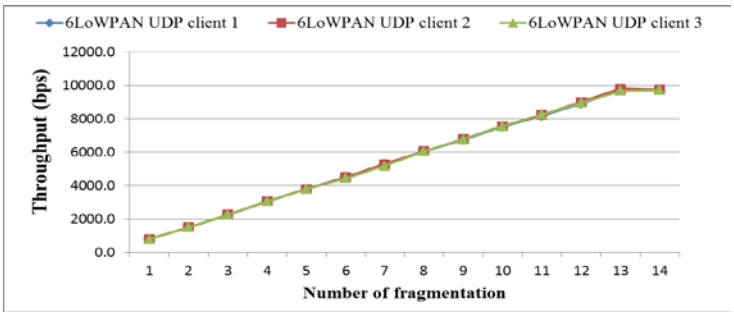

Fig. 27. The throughput of no background flow environment without QoS mechanism

Table 6 presents the experimental results of no background flow environment with QoS mechanism for three different 6LoWPAN UDP clients. The QoS levels for 6LoWPAN UDP client 1 (with high priority), 6LoWPAN UDP client 2 (with low priority), and 6LoWPAN UDP client 3 (best-effort) are different.

Table 6. The experimental results of no background flow environment with QoS mechanism

No Background Flow Environment with QoS Mechanism											
Packet			6LoWPAN UDP client 1 (High priority)			6LoWPAN UDP client 2 (Low priority)			6LoWPAN UDP client 3 (Best-effort)		
			1000 times		Throughput (packet size)	1000 times		Throughput (packet size)	1000 times		Throughput (packet size)
Number of Fragmentation	Payload	Packet size	Average End-to-End transmission time(ms)	Packet Loss Rate	bps	Average End-to-End transmission time(ms)	Packet Loss Rate	bps	Average End-to-End transmission time(ms)	Packet Loss Rate	bps
1	53	101	18.489	0.1%	807.192	18.464	0.2%	806.384	18.476	0.1%	807.192
2	144	192	32.348	1.1%	1519.104	32.345	1.4%	1514.496	32.330	1.2%	1517.568
3	240	288	46.612	1.4%	2271.744	46.576	2.3%	2251.008	46.556	2.4%	2248.704
4	336	384	60.882	3.4%	2967.552	60.842	3.1%	2976.768	60.793	2.8%	2985.984
5	432	480	78.622	1.7%	3774.720	75.445	2.7%	3736.320	75.009	2.6%	3740.160
6	528	576	89.833	3.0%	4469.760	89.339	5.0%	4377.600	89.209	4.8%	4386.816
7	624	672	103.662	2.60%	5236.224	103.599	4.00%	5160.960	103.465	4.0%	5160.960
8	720	768	118.288	2.80%	5971.968	117.676	4.60%	5861.376	117.68	4.5%	5867.520
9	816	864	132.328	4.40%	6607.872	133.308	4.50%	6600.960	132.252	6.1%	6490.368
10	912	960	145.660	8.20%	7050.240	146.104	7.20%	7127.040	146.124	13.0%	6681.600
11	1008	1056	160.645	0.90%	8371.968	160.685	1.10%	8355.072	160.423	1.7%	8304.384
12	1104	1152	175.268	2.70%	8967.168	175.044	1.80%	9050.112	174.132	2.3%	9004.032
13	1200	1248	189.352	1.20%	9864.192	188.496	1.20%	9864.192	188.688	2.1%	9774.336
14	1232	1280	194.703	2.30%	10004.480	194.882	2.50%	9984.000	194.790	2.9%	9943.040

The average end-to-end transmission delay and packet loss rate of no background flow environment with QoS mechanism for three different 6LoWPAN UDP clients are shown in **Figs. 28** and **29**, respectively. The packet loss of 6LoWPAN UDP client 3, with best-effort condition, is relatively higher than the other two clients due to the support of QoS service with higher priority in the other two clients. **Fig. 30** shows the throughput of no background flow environment with QoS mechanism for three different 6LoWPAN UDP clients. Without the interference from background flows, the throughput of three clients are about the same.

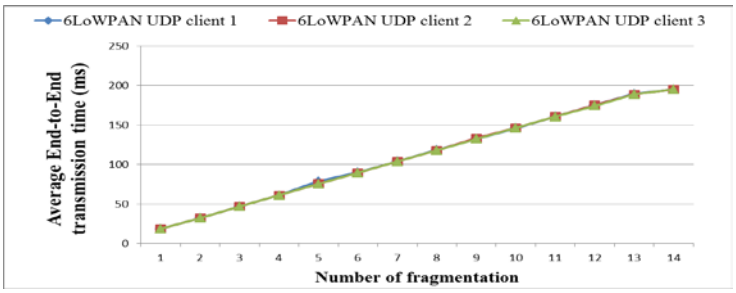

Fig. 28. The average end-to-end delay of no background flow environment with QoS mechanism

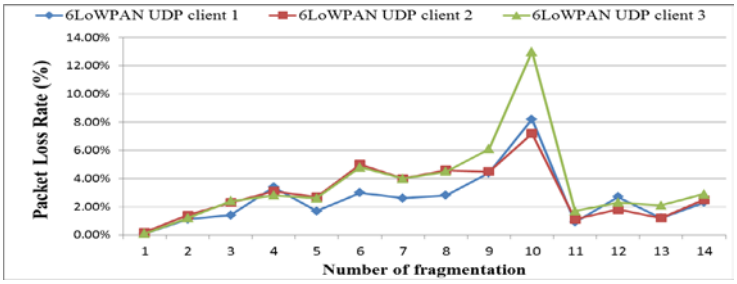

Fig. 29. The packet loss rate of no background flow environment with QoS mechanism

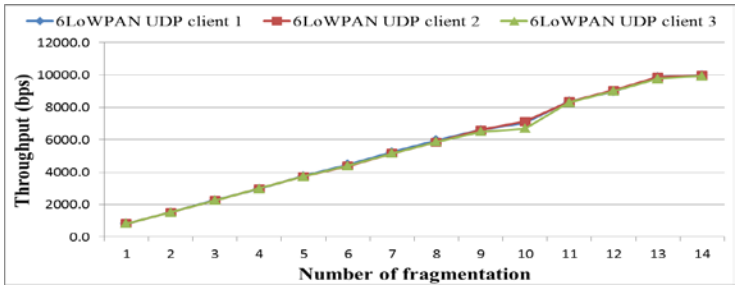

Fig. 30. The throughput of no background flow environment with QoS mechanism

Table 7 illustrates the experimental result of four injecting UDP background flows environment without QoS support for three different 6LoWPAN UDP clients. The average end-to-end transmission delay and packet loss rate of four injecting UDP background flows environment without QoS mechanism for three different 6LoWPAN UDP clients are shown in **Figs. 31** and **32**, respectively. Again, without QoS support, the best-effort behavior of UDP packets show relatively high (due to the interference of four injecting UDP background flows) and irregular packet drop (with no QoS support). **Fig. 33** shows the throughput of four injecting UDP background flows environment without QoS mechanism for three different 6LoWPAN UDP clients. This testing environment emulates the worst condition which provides no any QoS support under significant network congestion.

Table. 7. The experimental results of 4 UDP background flows environment without QoS mechanism

4 UDP Background Flows Environment without QoS Mechanism											
Packet			6LoWPAN UDP client 1 (Best-effort)			6LoWPAN UDP client 2 (Best-effort)			6LoWPAN UDP client 3 (Best-effort)		
			1000 times		Throughput (packet size)	1000 times		Throughput (packet size)	1000 times		Throughput (packet size)
Number of Fragmentation	Payload	Packet size	Average End-to-End transmission time(ms)	Packet Loss Rate	bps	Average End-to-End transmission time(ms)	Packet Loss Rate	bps	Average End-to-End transmission time(ms)	Packet Loss Rate	bps
1	53	101	36.885	10.2%	725.584	37.494	13.6%	698.112	34.329	2.1%	791.032
2	144	192	51.303	16.7%	1279.488	51.432	14.7%	1310.208	49.919	10.4%	1376.256
3	240	288	65.805	15.9%	1937.664	65.692	17.4%	1903.104	65.322	20.0%	1843.200
4	336	384	79.902	20.5%	2442.240	79.958	17.7%	2528.256	79.687	17.6%	2531.328
5	432	480	95.610	19.0%	3110.400	95.205	15.6%	3240.960	94.289	16.7%	3198.720
6	528	576	108.407	17.4%	3806.208	108.354	21.8%	3603.456	108.204	22.2%	3585.024
7	624	672	122.582	16.3%	4499.712	122.543	15.2%	4558.848	122.409	15.0%	4569.600
8	720	768	136.799	15.7%	5179.392	136.907	14.5%	5253.120	136.645	12.5%	5376.000
9	816	864	153.108	18.9%	5605.632	152.996	17.5%	5702.400	152.148	17.9%	5674.752
10	912	960	166.268	15.8%	6466.560	164.620	15.8%	6466.560	165.030	13.9%	6612.480
11	1008	1056	179.795	21.7%	6614.784	180.125	17.1%	7003.392	179.461	23.6%	6454.272
12	1104	1152	193.966	17.0%	7649.280	193.991	16.8%	7667.712	193.659	17.9%	7566.336
13	1200	1248	208.150	14.1%	8576.256	208.227	14.3%	8556.288	207.910	14.5%	8536.320
14	1232	1280	213.853	14.8%	8724.480	213.830	14.5%	8755.200	213.566	16.4%	8560.640

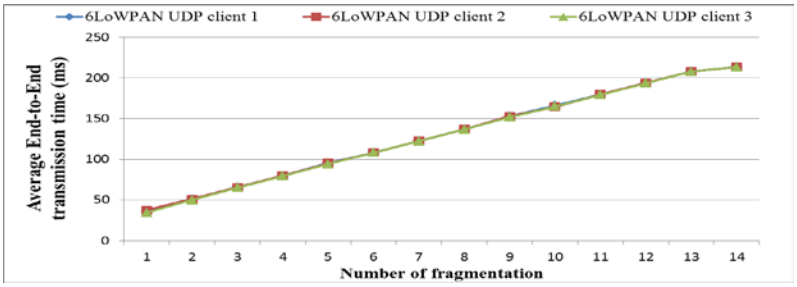

Fig. 31. Average end-to-end delay of 4 UDP background flows environment without QoS mechanism

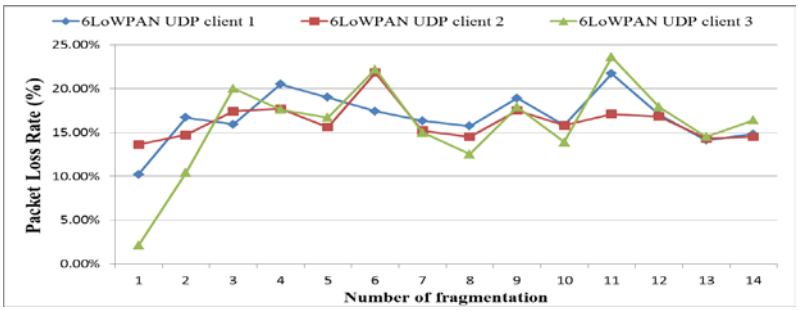

Fig. 32. The packet loss rate of 4 UDP background flows environment without QoS mechanism

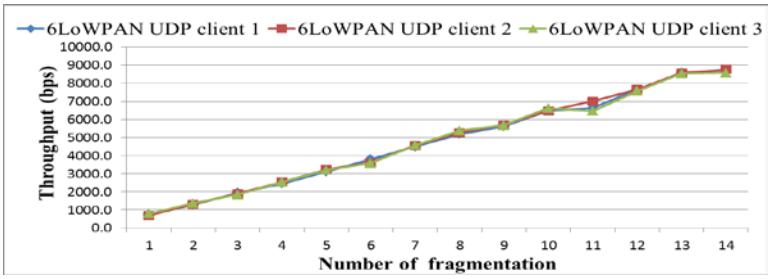

Fig. 33. The throughput of 4 UDP background flows environment without QoS mechanism

Table. 8. The experimental results of 4 UDP background flows environment with QoS mechanism

4 UDP Background Flows Environment with QoS Mechanism											
Packet			6LoWPAN UDP client 1 (High priority)			6LoWPAN UDP client 2 (Low priority)			6LoWPAN UDP client 3 (Best-effort)		
			1000 times		Throughput (packet size)	1000 times		Throughput (packet size)	1000 times		Throughput (packet size)
Number of Fragmentation	Payload	Packet size	Average End-to-End transmission time(ms)	Packet Loss Rate	bps	Average End-to-End transmission time(ms)	Packet Loss Rate	bps	Average End-to-End transmission time(ms)	Packet Loss Rate	bps
1	53	101	20.518	0.10%	807.192	20.471	0.20%	806.384	19.790	15.50%	682.760
2	144	192	33.996	0.70%	1525.248	33.368	1.00%	1520.640	33.352	13.60%	1327.104
3	240	288	46.913	1.90%	2260.224	46.954	2.90%	2237.184	46.701	11.50%	2039.040
4	336	384	61.136	3.50%	2964.480	61.139	5.40%	2906.112	60.936	12.70%	2681.856
5	432	480	75.416	3.80%	3694.080	75.377	5.90%	3613.440	75.172	13.40%	3325.440
6	528	576	89.861	3.90%	4428.288	90.253	6.70%	4299.264	89.857	17.90%	3783.168
7	624	672	107.933	4.00%	5160.960	107.493	6.50%	5026.560	106.750	28.80%	3827.712
8	720	768	117.640	6.60%	5738.496	119.516	12.60%	5369.856	118.000	21.00%	4853.760
9	816	864	140.080	3.20%	6690.816	143.932	4.70%	6587.136	132.360	23.40%	5294.592
10	912	960	146.216	2.60%	7480.320	148.612	3.30%	7426.560	145.924	18.90%	6228.480
11	1008	1056	160.528	1.20%	8346.624	162.220	2.00%	8279.040	160.412	15.00%	7180.800
12	1104	1152	175.888	5.20%	8736.768	175.252	8.70%	8414.208	174.268	19.80%	7391.232
13	1200	1248	190.540	1.30%	9854.208	189.032	1.80%	9804.288	188.912	13.00%	8686.080
14	1232	1280	197.505	8.30%	9390.080	197.706	8.60%	9359.360	197.497	28.00%	7372.800

Table 8 presents the experimental results of four injecting UDP background flows environment with QoS mechanism for three different 6LoWPAN UDP clients. Again, the QoS levels for 6LoWPAN UDP client 1 (with high priority), 6LoWPAN UDP client 2 (with low priority), and 6LoWPAN UDP client 3 (best-effort) are different. The average end-to-end transmission delay and packet loss rate of four injecting UDP background flows environment with QoS mechanism for three different 6LoWPAN UDP clients are shown in **Figs. 34** and **35**, respectively. As we can seen in **Fig. 35**, the packet loss rate of UDP client with best-effort is much higher than others. With QoS support, UDP client 1 with high priority shows the least pack loss as compared with others. Indeed, less than 5% of packet loss for most framentation cases of UDP client 1 is observed under significant network congestion in this experiment. **Fig. 36** shows the throughput of four injecting UDP background flows environment with QoS

mechanism for three different 6LoWPAN UDP clients. With the support of QoS mechanism, the 6LoWPAN UDP clients with high and low priorities maintain relatively high throughput.

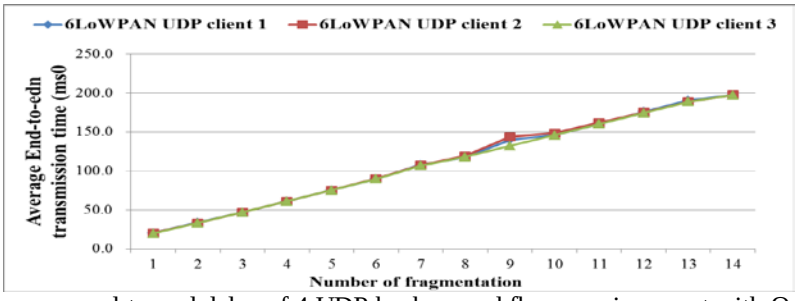

Fig. 34. The average end-to-end delay of 4 UDP background flows environment with QoS mechanism

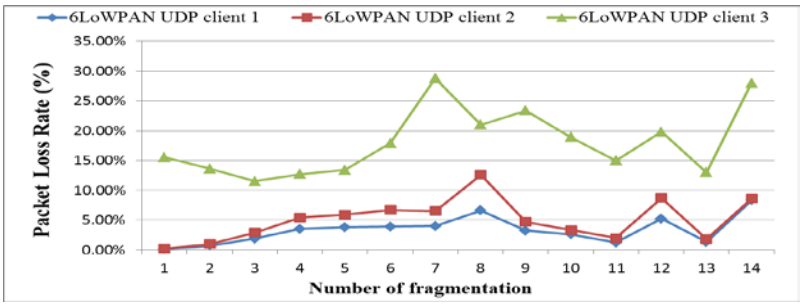

Fig. 35. The packet loss rate of 4 UDP background flows environment with QoS mechanism

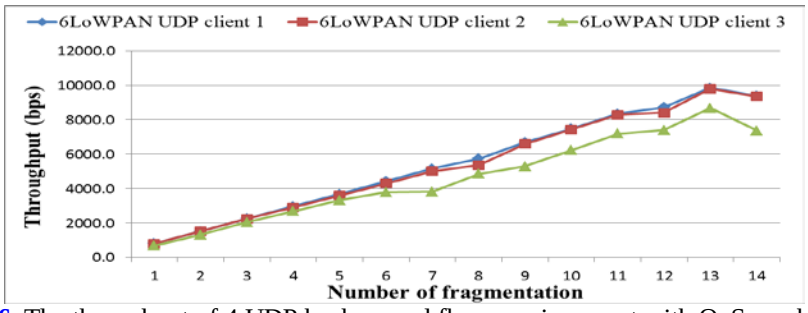

Fig. 36. The throughput of 4 UDP background flows environment with QoS mechanism

5. Conclusion

In this paper, we have presented an experimental research on the impact of end-to-end 6LoWPAN transmission performance on the best-effort IPv6 network. We have conducted experiments to analyze the actual 6LoWPAN end-to-end transmission delay, packet loss rate and throughput from the WSN client to the IPv6 server in a best-effort network environment. An SDN-enable 6LBR with QoS mechanism has been proposed and implemented to enhance the 6LoWPAN transmission performance. From our experimental results, the proposed SDN-based 6LBR with QoS mechanism communicated between WSNs and the Internet has brought forward the requirements of an end-to-end guarantee bandwidth to the SDN controller. It has shown that the SDN flow queue can reduce the 6LowPAN end-to-end packet loss rate effectively under significant network congestion. The proposed SDN-based QoS mechanism has been proved to work between heterogeneous WSN and IPv6 network. By using the QoS

Tag, the QoS mechanism has been extended from the Internet to the SDN-enabled WSN. By defining different QoS Tag in SDN-enabled WSN, the sensing data can be handled with different priorities, such as emergency packets.

In the future work, the bidirectional QoS mechanism between WSNs and the Internet will be investigated with the support of SDN. Furthermore, it could be an interesting approaching to combine the RPL routing protocol with SDN QoS Tag as a new routing scheme to provide multi-hopping in 6LoWPAN.

References

[1] G. Boggia, L. A. Grieco, M. Dohler, M. R. Palattella, N. Accettura, T. Watteyne, X. Vilajosana, "Standardized Protocol Stack for the Internet of (Important) Things," *IEEE Communications Surveys & Tutorials*, pp. 1389-1406, 2013. Article (CrossRef Link)

[2] ITU Internet Reports 2005: The Internet of things (ITU 2005 7th edition). Article (CrossRef Link)

[3] IEEE 802.15 Work Group, "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)," *ANSI/IEEE Std 802.15.4*, 2006. Article (CrossRef Link)

[4] D. Culler, G. Montenegro, J. Hui, and N. Kushalnagar, "Transmission of IPv6 Packets over IEEE802.15.4 Networks," in *Proc. of Internet Engineering Task Force*, RFC 4944, Sep. 2007. [Online]. Available: https://tools.ietf.org/html/rfc4944

[5] "SDN"[Online]. https://www.sdncentral.com/

[6] McKeown, et al., "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM CCR*, 38(2):69-74, Apr. 2008. Article (CrossRef Link)

[7] "6LBR" [Online]. https://github.com/cetic/6lbr/wiki

[8] Marja Matinmikko, Petri Ahokangas , Tao Chen, Xianfu Chen and Xuan Zhou, "Software Defined Mobile Networks: Concept, Survey, and Research Directions," *IEEE Communications Magazine*, Vol 53, Issue:11, pp.126-133, Nov., 2015. Article (CrossRef Link)

[9] Bruno Trevizan de Oliveira, Cintia Borges Margi and Lucas Batista Gabriel, "TinySDN: Enabling Multiple Controllers for Software-Defined Wireless Sensor Networks," *IEEE Latin-America Conference on Communications (LATINCOM)*, pp.1-6, Nov., 2014. Article (CrossRef Link)

[10] "Open vSwitch" [Online]. http://openvswitch.org/

[11] K. Hyunmin, K. Jaebeom, K. Young-Bae, "Developing a Cost-Effective OpenFlow Testbed for Small-Scale Software Defined Networking," *International Advanced Communication Technology (ICACT)*, pp. 758-761, 2014. Article (CrossRef Link)

[12] M. Cello, M. Marchese, and M. Mongelli, "On the QoS Estimation in an OpenFlow Network: The Packet Loss Case," *IEEE Communications Letters*, pp.554-557, vol. 20, Mar., 2016

[13] "Contiki"[Online]. http://www.contiki-os.org/

[14] E. Viktor, S. Thomas, "Development of a Contiki border router for the interconnection of 6LoWPAN and Ethernet," [Online]. https://www.researchgate.net/publication/269319668_Development_of_a_Contiki_border_router_for_the_interconnection_of_6LoWPAN_and_Ethernet

[15] M. Randy, P. Tanner, S. Matthew, T. Joseph, "Implementing Dynamic Address Changes in ContikiOS," in *Proc. of International Conference on Information Society (i-Society)*, pp.222-227, Nov. 2014. Article (CrossRef Link)

[16] H. Kim, J Seo, and J. Seo, "Performance Evaluation of a Smart CoAP Gateway for Remote Home Safety Services," *KSII Transactions on Internet and Information Systems*, pp. 3079-3089, vol. 9, Aug., 2015

[17] J. Kwangtae, K. Jinwook and K. Young-Tak, "QoS-aware Network Operating System for Software Defined Networking with Generalized OpenFlows," in *Proc. of IEEE Network Operations and Management Symposium (NOMS)*, pp. 1167-1174, Apr., 2012. Article (CrossRef Link)

[18] Phyo May Thet, Parichat Panwaree, JongWon Kim, and Chaodit Aswakul, "Design and Functionality Test of Chunked Video Streaming over Emulated Multi-Path OpenFlow Network," in *Proc. of 12th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTICON)*, pp. 1-6, June, 2015.
Article (CrossRef Link)
[19] Martin Devera aka devik, "Hierarchical Token Bucket Theory," [Online]. Available: http://luxik.cdi.cz/~devik/qos/htb/, 2002.
[20] Martin A. Brown, "Traffic Control HOWTO", [Online]. Available: http://linux-ip.net/articles/Traffic-Control-HOWTO, 2006.
[21] "ATMega rf128a1"[Online]. http://www.atmel.com/devices/ATMEGA128RFA1.aspx
[22] "Ryu"[Online]. https://osrg.github.io/ryu/

Tsung-Han Lee received his Ph.D. in Telecommunication Engineering from Queen's University of Belfast, United Kingdom. Currently, he is an Associate Professor in the Department of Computer Science at National Taichung University of Education, Taiwan. His research focus is on Internet of things, software-defined networking and wireless sensor networks.

Lin-Huang Chang received his Ph.D. degree in Electrical and Computer Engineering from State University of New York at Buffalo, Buffalo, New York, USA, in 1995. Currently, he is a professor and of the Department of Computer Science at National Taichung University of Education, Taiwan, where he has been with the department since July 2007. Professor Chang has authored over 100 pioneering research articles. He also held 6 patents, including 3 USA patents and 3 ROC patents. His research interests include the fields in wireless sensor networks, software-defined netowrking, multimedia communications and Internet of things.

Wei-Chung Cheng received his Master Degree in Computer Science from National Taichung University of Education, Taiwan. Currently, he is an employee in the National Chung-Shan Institute of Science & Technology, Taiwan. His research focus is on Internet of things, software-defined networking and wireless sensor networks.