คุณภาพการให้บริการสตรีมมิ่งที่มีการจัดความสำคัญบนเครือข่ายเอสดีเอ็น
Quality of Service of Streaming Service based on Priority Management over SDN

ปิยพงษ์ เคนเหลื่อม (Piyapong kenluem)[1] และชัยพร เขมะภาตะพันธ์ (Chaiyaporn Khemapatapan) [1]
[1]สาขาวิศวกรรมคอมพิวเตอร์และโทรคมนาคม วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรม มหาวิทยาลัยธุรกิจบัณฑิตย์
jacksand.pong@gmail.com, chaiyaporn@dpu.ac.th

บทคัดย่อ

เอสดีเอ็นเป็นเครือข่ายคอมพิวเตอร์แบบใหม่ที่สามารถจัดการส่งผ่านข้อมูลแบบพลวัตรผ่านโพรโทคอล OpenFlow โดยมี SDN Controller เป็นตัวควบคุมอุปกรณ์ซอฟต์สวิตช์ที่ทำงานด้วย OpenFlow ในเครือข่าย SDN งานวิจัยนี้จะแสดงขั้นตอนและการทดสอบ QoS ของการให้บริการแบบสตรีมมิ่งที่มีการส่งผ่านข้อมูลด้วยโพรโทคอล UDP บนเครือข่าย SDN ซึ่งมีขั้นตอนหลักคือการจัดการผ่านทาง Web service ที่เป็น REST API ไปยังตัว SDN Controller สำหรับผู้ใช้งานจะถูกแบ่งกลุ่มเป็นกลุ่มที่มีการให้ความสำคัญและกลุ่มผู้ใช้งานทั่วไปที่ไม่มีการให้ความสำคัญ ซึ่งกลุ่มที่มีความสำคัญจะประกันแบนด์วิดธ์ของเครือข่าย SDN ในขณะที่กลุ่มผู้ใช้งานทั่วไปจะพยายามใช้งานแบนด์วิดธ์ที่เหลือให้มากที่สุด การทดลองได้ใช้ Mininet, OpenDaylight และ PostMan ผลการทดลองที่ได้พบว่ากลุ่มผู้ใช้งานที่มีการจัดความสำคัญให้นั้นในสถาณการณ์ที่มีปริมาณทราฟฟิกในเครือข่ายมากที่สุดยังมี throughput ต่ำสุดใกล้เคียงกับค่าที่รับประกันไว้ และมี jitter ที่มีความแปรปรวนค่อนข้างต่ำ เมื่อเทียบกับกลุ่มผู้ใช้งานทั่วไปที่ไม่มีการจัดความสำคัญให้

คำสำคัญ: เอสดีเอ็น โอเพนโฟล์ มินิเนท โอเพนเดย์ไลท์ การให้ความสำคัญ สตรีมมิ่ง

ABSTRACT

SDN is a new computer network which can dynamically manage traffic flow using OpenFlow protocol. SDN controller is used to control soft switches operating using OpenFlow in SDN. In this paper, management processing and QoS testing of streaming service over SDN which transfers data via UDP protocol will be presented. The main process is to manage SDN controller via web service using REST API. Users will be separated into 2 groups: first group has a priority in order to guarantee bandwidth over SDN, another group is a normal user which consumes the rest bandwidth in best effort manner. Experiments are completed using Mininet, OpenDaylight and PostMan. Form experimental results, prioritized users can achieve nearly minimum guaranteed bandwidth when traffic in the network is maximum. Moreover, prioritized users also have a small variance of jitter in comparison with normal users with no priority.

Keywords: SDN, OpenFlow, Mininet, OpenDaylight, Priority, Streaming

1. บทนำ

การใช้บริการระบบสตีมมิ่งภาพและเสียงโดยทั่วไปนั้นมีการแบ่งกลุ่มผู้ใช้งานตามความสำคัญเช่นกลุ่มผู้ใช้งานที่มีความสำคัญหรือกลุ่มพรีเมียมกับกลุ่มผู้ใช้งานปกติ ซึ่งกลุ่มพรีเมียมนั้นต้องมีการบริการที่ดีกว่า แบนด์วิธถือเป็นทรัพยากรที่

ต้องมีการบริหารให้มีประสิทธิภาพ แต่ด้วยความหลากหลายของอุปกรณ์ทำให้เกิดปัญหาไม่สามารถควบคุมจัดการโดยระบบเพียงระบบเดียวหรือมีความยุ่งยากในการจัดการ [1]

อย่างไรก็ตามด้วยการใช้งานเครือข่าย Software Defined Network หรือ SDN ซึ่งมีการควบคุมการทำงานด้วย OpenFlow [2] การจัดระดับความสำคัญของกลุ่มผู้ใช้งานสามารถทำได้โดยง่าย ยืดหยุ่น และซับซ้อนกว่ามาก [3] สามารถตอบสนองต่อความต้องการที่เปลี่ยนแปลงได้ ดังนั้นในงานวิจัยนี้จึงเสนอขั้นตอนการจัดระดับความสำคัญของกลุ่มผู้ใช้งานเพื่อใช้บริการสตรีมมิ่งบนเครือข่าย SDN เพื่อรับประกันแบนด์วิดธ์ผ่านการควบคุมทาง North Bound ของ SDN ด้วยการเรียกใช้งาน web service ที่เป็น REST API เพื่อทำการการันตี แบนด์วิดธ์ซึ่งการทดสอบจะแบ่งผู้ใช้งานออกเป็น 2 กลุ่มคือกลุ่มที่ให้ระดับความสำคัญและการันตีแบนด์วิดธ์และกลุ่มผู้ใช้งานปกติที่ใช้งานทราฟฟิกจากแบนด์วิดธ์ที่เหลืออยู่ให้มากที่สุด

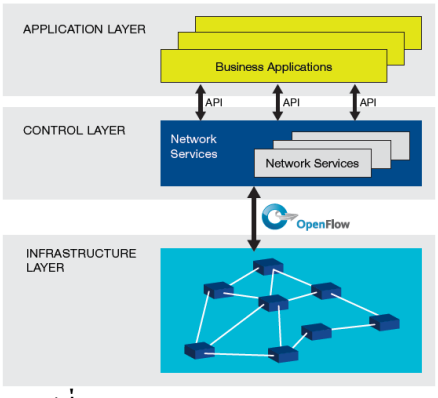
2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 Software Defined Network

Software Defined Networking (SDN) เป็นสถาปัตยกรรมเครือข่ายคอมพิวเตอร์แบบใหม่ที่ออกแบบมาเพื่อช่วยให้สามารถควบคุมการไหลข้อมูลผ่านเครือข่าย รวมถึงทรัพยากรในเครือข่ายโดยใช้ซอฟต์แวร์เป็นตัวกำหนดการทำงาน ซึ่งได้ทำการแยก Control plan และ Data plan ออกจากัน โดยโครงสร้างสถาปัตยกรรม SDN แบ่งออกเป็น 3 layer คือ Application Layer, Control Layer และ Infrastructure Layer

Application Layer ใช้ติดต่อกับผู้ใช้งานผ่าน application เพื่อกำหนดการทำงานโดยรวมของเครือข่ายเช่น การทำ QoS การหาเส้นทางที่ดีที่สุด การปรับเปลี่ยนเส้นทางเป็นต้น โดยใช้ API ทำหน้าที่ในการติดต่อ SDN Controller layer ผ่านทางจุดเชื่อมต่อที่เรียกว่า Northbound

Control Layer หรือ Control plan ทำหน้าที่เป็น Centralized control ควบคุมการไหลของข้อมูลโดยการส่งข้อมูลไปยัง Data plan อีกทั้งเป็นเหมือนสมองในการคำนวณหาผลลัพธ์ต่างๆ มีอุปกรณ์หลักคือ SDN Controller เช่น OpenDaylight [4] ซึ่งเป็น Open Source SDN Controller ชนิดหนึ่งและได้รับการยอมรับในผู้ผลิตอุปกรณ์เครือข่ายชั้นนำ เช่น Cisco [5] ทำหน้าที่ส่งข้อมูลควบคุมการไหลไปยังอุปกรณ์ในเลเยอร์ล่างถัดไปผ่านทาง Southbound ด้วยโพรโทคอล OpenFlow

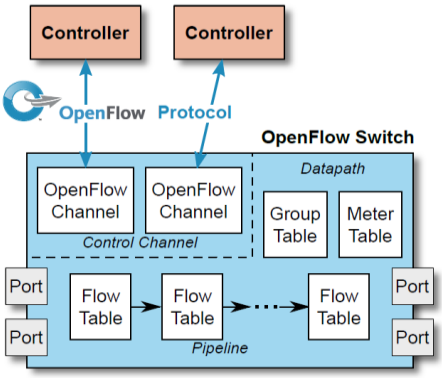


รูปที่ 1: สถาปัตยกรรมของ ONF/SDN [2]

Infrastructure Layer หรือ Data plan คืออุปกรณ์ที่ใช้สำหรับการส่ง Packet ตามข้อมูลที่อยู่ใน Flow Table และ Table อื่นๆ ตามข้อมูลที่ได้รับมาจาก Controller โดยปกติแล้วอุปกรณ์ใน layer นี้คือ OpenFlow Switch [6] ซึ่งมีมาตรฐานเปิดเรียกว่า Open vSwitch หรือ OVS ซึ่งเป็นอุปกรณ์ที่ทำงานในลักษณะ Virtual switch

2.2 OpenFlow

OpenFlow เป็นโพรโทคอลมาตราฐานที่ใช้แลกเปลี่ยนข้อมูลระหว่าง controller และ switch [7] ถูกนำมาใช้โครงการวิจัยในมหาวิทยาลัย Stanford ในปี 2008 โดยมีเป้าหมายเริ่มต้นคือการหาวิธีในการจัดการอุปกรณ์เครือข่าย เวอร์ชัน 1.0 เกิดขึ้นในปี 2009 OpenFlow อยู่ภายใต้การดูแลของ Open Networking Foundation (ONF) เวอร์ชันล่าสุดคือ 1.5.1 ออกเมื่อวันที่ 26 มีนาคม 2015



รูปที่ 2: องค์ประกอบของ OpenFlow switch [7]

OpenFlow Switch นั้นประกอบไปด้วย Flow Table และ Group Table [7] และ [8] ทำหน้าที่ในการตรวจสอบ packet ที่จะเข้าออกผ่าน Port กับเงื่อนไขของ Flow Table เพื่อที่จะกำหนดการกระทำต่อ packet เช่น ส่งต่อไปยัง Flow Table อื่นๆ Drop packet ทิ้งหรือทำการแก้ไขข้อมูลใน packet และในการติดต่อกับ Controller นั้นจะมี Openflow Channel ติดต่อกับ Controller เพื่อรับคำสั่งต่าง ๆ มาติดตั้งใน Flow Table โดยปกติโพรโทคอลนี้จะทำการเข้ารหัสข้อมูลที่เป็น Flow ควบคุมการทำงานของ switch ไว้ในรูปแบบ SSL/TLS

2.3 งานวิจัยที่เกี่ยวข้อง

การรับประกันคุณภาพการให้บริการบนเครือข่าย SDN [9] ได้นำเสนอรูปแบบโพรโทคอลแบบใหม่ที่จะควบคุม application ด้วยกลไก QoS แบบ end-to-end ที่ถูกสร้างอยู่บน SDN Controller Floodlight และจำลองเครือข่าย SDN ด้วย Mininet ผู้วิจัยได้ทำการสร้าง algorithm เพื่อใช้ในการตรวจสอบการทำงาน โดยให้ผู้ใช้งานกำหนดและสร้าง queues บนตัว OpenFlow Switches ก่อน จากนั้นผู้ใช้งานทำการ enable Qos Policy บน controller โดยใช้ Floodlight script และ controller จะทำการอ่าน script นั้นและสร้าง REST API จากนั้นติดตั้ง queues ลงบน switches ในการเลือกเส้นทางของ packet ผู้วิจัยได้ทำการคำนวนเส้นทางโดยใช้ Shortest Part ของ Dijkstra ฟังก์ชัน เพื่อกำหนดเส้นทางให้แต่ล่ะ queues อีกที เมื่อสิ้นสุดกระบวนการ controller จะทำการส่ง rules ไปยัง switches เพื่อที่จะเพิ่มลงไปใน Flow Tables ผลการวิจัยชี้ให้เห็นว่าผลจาก modified controller ตัว routing นั้น ทำให้การตัดสินใจในการเลือกเส้นทางมีความถูกต้อง การเลือก queues นั้นตรงตาม policy ที่ได้จากการคำนวนของ algorithm ทำงานได้ถูกต้องและแสดงให้เห็นว่าสามารถกำหนด QoS ของแต่ละทราฟฟิคที่ต้องการได้

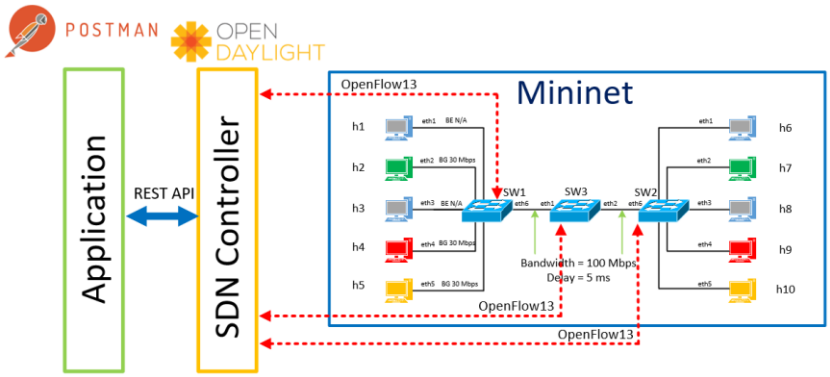
การสร้าง host และ switch โดยใช้ Floodlight เป็นตัว controller โดยจุดประสงค์ของงานวิจัยชิ้นนี้คือเพื่อหาข้อดีและข้อเสียของ Mininet ในงานวิจัย [10] พบว่าสมรรถนะของระบบที่ใช้ไปเมื่อทำการสร้างโทโพโลยีแบบต่างๆ พบว่าเวลาและหน่วยความจำที่ใช้งานที่ใช้เพิ่มขึ้นเยอะมากเมื่อสร้าง 511 node ซึ่งผลการทดสอบพบว่า ข้อจำกัดที่สำคัญที่สุดของ Mininet คือ ประสิทธิภาพของตัวระบบโดยเฉพาะเมื่อมีการโหลดของ cpu สูงขึ้น ปกติ Mininet ทำงานบนคอมพิวเตอร์เครื่องเดียวและ emulates โฮสต์ทั้งหมดบนทรัพยากรฮาร์แวร์แบบเดียวกัน ซึ่งเป็นข้อเสียที่สำคัญ แม้ว่า Mininet จะไม่เหมาะสมสำหรับการจำลองแบบขนาดใหญ่แต่เป็นตัวเลือกที่ดีในการสร้างต้นแบบเครือข่ายขนาดเล็กและขนาดกลาง นักศึกษาและนักวิจัยผู้ดูแลระบบเครือข่ายสามารถใช้ Mininet นี้ในการจำลองเครือข่ายได้ง่ายและรวดเร็วรวมถึงกำหนดโทโพโลยีที่ต้องการได้อีกด้วย

การศึกษา QoS บนเครือข่าย SDN [11] โดยเปรียบเทียบว่าการทำ QoS บน Core Switch และ Leaf Switch จะให้ผลอย่างไร ซึ่งกลไกที่นำมาทดสอบคือ Basic CBQ และ Source CBQ การทดลองนี้ผู้วิจัยได้ทำการแบ่งกลุ่ม client ออกเป็น 2 กลุ่มๆ ละ 50 Host และทำการสร้าง HTTP Server และ VLC Streaming Media Server เพื่อใช้ในการทดลอง และกำหนด QoS ไว้บนตัว Core Switch และ Leaf Switch จากนั้นทำการทดสอบซึ่งผลที่ได้คือการทำ CBQ ที่ Leaves มีประสิทธิภาพดีกว่า Basic CBQ ที่ Core อยู่ที่ 2% และดีกว่า Source CBQ 30%

3. ขั้นตอนการดำเนินงานวิจัย

3.1 ภาพรวมของระบบ

ระบบจะประกอบไปด้วย OpenDaylight ทำหน้าที่เป็น SDN Controller ในการเชื่อมต่อกับ Open vSwitch เพื่ออัพเดต flow table ในการค้นหาเส้นทางของ host โดยกำหนดลิงค์แบนด์วิดธ์ trunk ระหว่าง Open vSwitch ที่ 100 Mbps และมี delay 5 ms โดยมี Open vSwitch จำนวน 3 ตัว มีโฮสต์จำนวน 10 เครื่องซึ่งรับส่งข้อมูลที่เป็น Streaming ระหว่างกันจำนวน 5 คู่แบ่งเป็นกลุ่มผู้ใช้งานทั่วไป 2 คู่ คือ H1-H6 และ H3-H8 สามารถรับส่งข้อมูลด้วยแบนด์วิดธ์ที่เหลืออยู่ทั้งหมดในลักษณะ best effort (BE) และกลุ่มผู้ใช้งาน Privilege ที่มีการจัดระดับความสำคัญให้ 3 คู่คือ H2-H7, H4-H9 และ H5-H10 ซึ่งได้การรับประกันแบนด์วิดธ์ (bandwidth guarantee, BG) 30 Mbps ตามรูปที่ 3



รูปที่ 3: โทโพโลยีของระบบที่ใช้ทำการศึกษา

ขั้นตอนกระบวนการจำลองถูกแบ่งเป็น 4 ส่วนเพื่อแยกหน้าที่การทำงาน โทโพโลยีที่จำลองการทำงานจะถูกสร้างด้วยภาษา Python ไว้ใน topocustom.py เมื่อได้โทโพโลยีของระบบ SDN ที่ต้องการแล้ว จะกำหนดรูปแบบและคุณลักษณะการรับส่งข้อมูลผ่าน Application ซึ่งจำลองด้วยโปรแกรม Postman เพื่อสั่งการและแก้ไข flow table บนอุปกรณ์ซอฟต์สวิสซ์ซึ่งเป็นกระบวนแรกในการปรับปรุงทิศทางการไหลของข้อมูล ก่อนจะเข้าสู่กระบวนที่สองคือการคัดแยกทราฟฟิกซึ่งในขั้นตอนนี้จะสร้าง flow เพื่อจับคู่ทราฟฟิกชนิด UDP ที่วิ่งระหว่างผู้ส่งและผู้รับ กำหนดให้เป็น queue 0 1 2 และ 3 ตามลำดับผ่าน SDN Controller โดยการควบคุมผ่าน Northbound ซึ่งเป็น REST API

ตารางที่ 1: แสดง flow คัดแยกทราฟฟิกบน SW1

SW 1						
Flow	IP Source	IP Destination	ip-protocol	udp-destination-port	Action	Priority
11	10.0.0.2	10.0.0.7	17	12345	Set Queue id =1	100
12	10.0.0.4	10.0.0.9	17	12345	Set Queue id =2	100
13	10.0.0.5	10.0.0.10	17	12345	Set Queue id =3	100
14	10.0.0.1	10.0.0.6	17	12345	Set Queue id =0	100
15	10.0.0.3	10.0.0.8	17	12345	Set Queue id =0	100

กระบวนการที่สามการจัดลำดับความสำคัญให้แก่ Queue 1 2 และ 3 ให้ประกันแบนด์วิดธ์ที่ 30 Mbps ส่วน queue 0 ไม่มีการประกันทั้งนี้ การรับประกันแบนด์วิดธ์ใช้ traffic command ซึ่งซอฟต์สวิตซ์ถูกพัฒนาบนพื้นฐานลินุกซ์จึงรองรับ traffic command โดยระบุค่าพารามิเตอร์ที่สำคัญคือ min_rate=30 Mbps และ max_rate= 100 Mbps บนอินเตอร์เฟส SW1-eth6 และ SW3-eth2 และเข้าสู่กระบวนการที่สี่คือทดสอบและบันทึกผลการทดลอง

3.2 การจำลอง

ในขั้นตอนการทดสอบจะทำการจำลองการการสตรีมมิ่งด้วยโปรโตคอล UDP จากผู้ส่งไปยังผู้รับด้วย jPerf โดยกำหนดข้อมูลในส่วนของฝั่งผู้ส่งและเริ่มตามลำดับเวลาในตารางที่ 2

1. IP โปรโตคอล ใช้งาน UDP
2. Port Destination ใช้หมายเลข 12345
3. กำหนดทรูพุท 100 Mbit/s
4. กำหนดเวลาสตรีมมิ่ง 100 วินาที
5. กำหนด UDP buffer 110 KByte

ตารางที่ 2: แสดงลำดับการสตรีมมิ่ง

Time	Match	Type	Bandwidth	IP Source	IP Destination
0	h1-h6	BE	N/A	10.0.0.1	10.0.0.6
20	h2-h7	BG	30 Mbps	10.0.0.2	10.0.0.7
40	h3-h8	BE	N/A	10.0.0.3	10.0.0.8
60	h4-h9	BG	30 Mbps	10.0.0.4	10.0.0.9
80	h5-h10	BG	30 Mbps	10.0.0.5	10.0.0.10

4. ผลงานการทดลอง

การทดลองนั้นจะวัดปริมาณ throughput และ Jitter ที่แต่ละคู่สามารถส่งผ่านข้อมูลได้สำเร็จ โดยผลการทดลองแสดงตามรูปที่ 5 และ 6 และตารางที่ 1 จะเห็นว่า เมื่อเริ่มต้นการทดลองคู่ผู้ใช้งาน H1-H6 ในรูปที่ 5(a) สามารถรับส่งข้อมูลได้ตามปกติ

แต่เมื่อคู่ผู้ใช้งาน H2-H7 ที่มีการรับประกันแบนด์วิดธ์เริ่มต้นการรับส่งข้อมูลที่เวลา t=20s ตามรูปที่ 5(b) จะส่งผลกระทบโดยตรงต่อคู่ H1-H6 ตามรูปที่ 5(a) อย่างเห็นได้ชัดเจน โดยแบนด์วิดธ์ส่วนใหญ่ถูกกันไว้ให้กับผู้ใช้งานที่มีความสำคัญกว่า และเมื่อเริ่มต้นที่เวลา t=40s เมื่อคู่ H3-H8 ซึ่งเป็นผู้ใช้งานปกติเริ่มต้นการรับส่งข้อมูลในรูปที่ 5(c) พบว่าทั้งคู่ H1-H6 และ H3-H8 มี throughput ใกล้เคียงกัน ในขณะที่คู่ H2-H7 มี throughput ลดลงมาเล็กน้อยแต่ยังคงอยู่ในระดับที่สูงกว่ามากและสูงกว่าแบนด์วิดธ์ที่ได้รับการรับประกันไว้ สุดท้ายที่เวลา t=60s และ t=80s เมื่อคู่ H4-H9 และ H5-H10 เริ่มต้นการรับส่งข้อมูลดังแสดงในรูปที่ 5(d) และ 5(e) ตามลำดับ พบว่ากลุ่มผู้ใช้งานพรีเมี่ยมทั้ง 3 คู่มี throughput ประมาณ 30 Mbps ใกล้เคียงกับแบนด์วิดธ์ได้รับประกันไว้ ในขณะที่ผู้ใช้งานทั่วไปมี throughput ต่ำมากประมาณ 5 Mbps เท่านั้น จึงสรุปได้ว่าผู้ใช้งานที่ได้รับการรับประกันแบนด์วิดธ์มีการรับส่งข้อมูลในลักษณะ best effort ด้วยโดยมีแบนด์วิดธ์ขั้นต่ำใกล้เคียงกับแบนด์วิดธ์ที่ประกันไว้

ผลของ jitter ซึ่งเป็นตัวแปรสำคัญในการใช้งานสตรีมมิ่ง พบว่าผู้ใช้งานที่ได้รับการรับประกันแบนด์วิดธ์ มีลักษณะของค่า jitter ที่ดีมากมีค่าค่อนข้างต่ำและคงที่ กล่าวคือมีความแปรปรวนของ jitter น้อยมาก ตามรูปที่ 6(b) 6(d) และ 6(e) ซึ่งเป็นผลดีอย่างยิ่งต่อการใช้งานสตรีมมิ่ง สามารถแสดงผลได้อย่างราบเรียบต่อเนื่อง ในขณะที่ผู้ใช้งานทั่วไปมีการเปลี่ยนแปลงของ jitter ค่อนข้างมากโดยเฉพาะอย่างยิ่งในช่วงเวลาที่มีปริมาณทราฟฟิกมากที่สุดระหว่าง t=80s ถึง t=100s ตามแสดงในรูปที่ 6(a) และ 6(b) จากตารางที่ 1 พบว่าผู้ใช้งานทั่วไปมี jitter ต่ำสุด/สูงสุดเท่ากับ 0.075ms/18.097ms และ 0.139ms/7.586ms สำหรับคู่ H1-H6 และ H3-H8 ตามลำดับ ซึ่งมีความแปรปรวนมาก ไม่ส่งผลดีต่อการใช้งานในลักษณะสตรีมมิ่งเป็นอย่างมาก อาจส่งผลให้การแสดงผลมีอาการค้างกระตุกเป็นระยะ

5. สรุปผล

จากการศึกษาจะเห็นได้ว่า การจัดการระดับความสำคัญของผู้ใช้งานบน SDN นั้นทำได้ง่ายและมีความเป็นพลวัตรสูง สามารถสร้างรูปแบบการใช้งานเครือข่ายได้ตามการประยุกต์ใช้งานที่ต้องการ สำหรับการทดลองพบว่ากลุ่มผู้ใช้งานที่มีระดับความสำคัญสามารถทำงานได้ตามต้องการมีการส่งผ่านข้อมูลที่มี throughput ขั้นต่ำใกล้เคียงกับแบนด์วิดธ์ที่ได้รับการรับประกันและมี jitter ที่อยู่ในเกณฑ์ที่ดีมาก ในขณะที่กลุ่มผู้ใช้งานทั่วไปที่ไม่ได้จัดระดับความสำคัญให้สามารถใช้งานได้ แต่แบนด์วิดธ์จะถูกแบ่งปันไปให้กลุ่มผู้ใช้งานที่มีระดับความสำคัญกว่า ซึ่งเป็นไปตามวัตถุประสงค์ที่ได้ออกแบบไว้

การศึกษาต่อไปอาจสร้างกลุ่มผู้ใช้งานที่มีความหลากหลายและมีระดับความสำคัญที่แตกต่างกันมากขึ้น รวมทั้งการศึกษาการใช้งานผสมผสานระหว่างสตรีมมิ่งและการใช้งานอื่นๆ

เอกสารอ้างอิง

[1] Charles Krasic, Jonathan Walpole, Wu-chiFeng, “Quality-Adaptive Media Streaming by Priority Drop”, *Proc. NOSSDAV'03*, June1-3,2003, Monterey, California, USA.

[2] OpenFlow-enabled Transport SDN, *ONF Solution Brief* , May 27 2014

[3] Derrick D’souza et al., “Improving QoS in a Software-Defined Network”, *Capstone Research Paper, University of Colorado Boulder,* April 30, 2016.

[4] OpenDaylight Documentation , *Release Boron* , Aug 22, 2017

[5] Editorial: Cisco Open SDN Controller 1.2. (2015*). Cisco Public Information*. 1-6.

[6] http://docs.openvswitch.org/en/latest/ access on 22-Sep-2017

[7] OpenFlow Switch Specification , *Version 1.5.0 (Protocol Version 0x06)*, December 19 ,2014

[8] https://www.sdxcentral.com/sdn/definitions/what-is-openflow/ access 20-Sep-2017

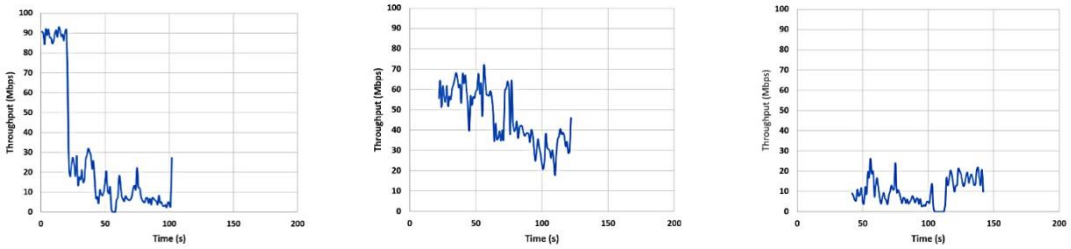
[9] Davide Adami , Lisa Donatini , Stefano Giordano, Michele Pagano , “A Network Control Application enabling Software-Defined Quality of Service” , *IEEE ICC 2015 – Communications QoS, Reliability and Modeling Symposium*

[10] RogérioLeão Santos de Oliveira, Ailton Akira Shinoda , “Using Mininet for Emulation and Prototyping Software-Defined Networks” , *in Proc. IEEE Colombian Conference on Communications and Computing (COLCOM)*, 2014.

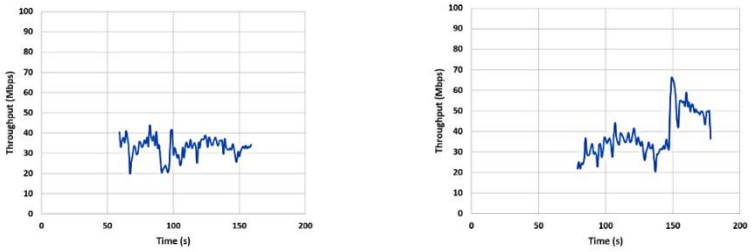
[11] Oliver Chato and William Emmanuel S. Yu, “An Exploration of Various Quality of Service Mechanisms in an OpenFlow and Software Defined Networking Environment”, *The 2016 3rd International Conference on Systems and Informatics (ICSAI 2016)*

ตารางที่ 3 ผลการทดลอง

Sender	Receiver	Traffic Management	Guaranteed Bandwidth	Period	Minimum Throughput	Maximum Throughput	Average Throughput	Minimum Jitter	Maximum Jitter
H1	H6	BE	N/A	0-100s	0 Mbps	92.9 Mbps	27 Mbps	0.075 ms	18.097 ms
H2	H7	BG	30 Mbps	20-120s	18 Mbps	71.3 Mbps	46 Mbps	0.1 ms	0.902 ms
H3	H8	BE	N/A	40-140s	0 Mbps	26.2 Mbps	10 Mbps	0.139 ms	7.586 ms
H4	H9	BG	30 Mbps	60-160s	21 Mbps	43.7 Mbps	33 Mbps	0.169 ms	1.405 ms
H5	H10	BG	30 Mbps	80-180s	21 Mbps	66.1 Mbps	39 Mbps	0.037 ms	1.618 ms

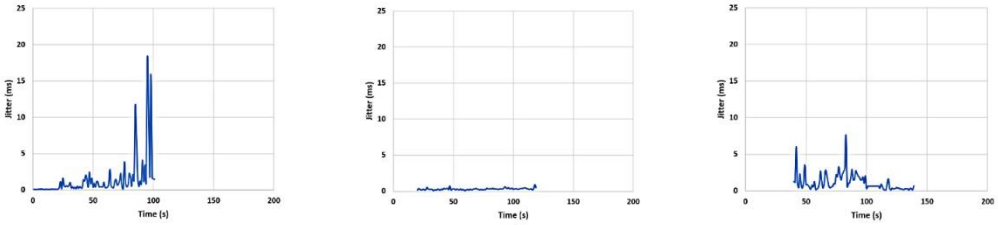


(a) H1-H6 BE (b) H2-H7 BG (c) H3-H8 BE

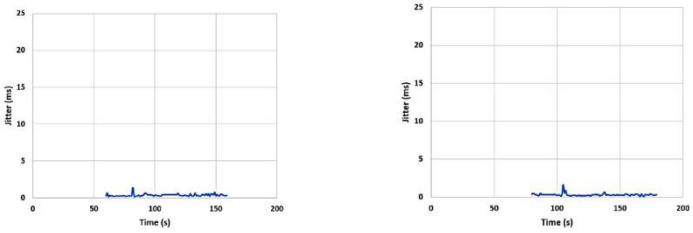


(d) H4-H9 BG (e) H5-H10 BG

รูปที่ 5: Throughput ที่ได้จากการทดลอง



(a) H1-H6 BE (b) H2-H7 BG (c) H3-H8 BE



(d) H4-H9 BG (e) H5-H10 BG

รูปที่ 6: Jitter ที่ได้จากการทดลอง