การสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถีซึ่งประสานงานด้วยเอสดีเอ็น

นางสาวปาริฉัตร ปันวารี

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2557

SDN-Coordinated Multi-Path Chunked Video Streaming

Miss Parichat Panwaree

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2014

หัวข้อวิทยานิพนธ์	การสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถี ซึ่งประสานงานด้วยเอสดีเอ็น
โดย	นางสาวปาริฉัตร ปันวารี
สาขาวิชา	วิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร.เชาวน์ดิศ อัศวกุล
อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม	ศาสตราจารย์ ดร.คิม จองวอน

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.บัณฑิต เอื้ออาภรณ์)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.สุภาวดี อร่ามวิทย์)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร.เชาวน์ดิศ อัศวกุล)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม
(ศาสตราจารย์ ดร.คิม จองวอน)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ชัยเชษฐ์ สายวิจิตร)

.....กรรมการภายนอกมหาวิทยาลัย
(รองศาสตราจารย์ ดร.ภูมิพัฒ แสงอุดมเลิศ)

ปาริฉัตร ปันวารี : การสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถีซึ่งประสานงานด้วยเอสดีเอ็น (SDN-Coordinated Multi-Path Chunked Video Streaming) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ. ดร. เชาวน์ดิศ อัศวกุล, อ.ที่ปรึกษาวิทยานิพนธ์ร่วม : ศ. ดร. คิม จองวอน, 76 หน้า.

วิทยานิพนธ์นี้ได้ศึกษาสมรรถนะของโครงข่ายโอเพนโฟลว์ซึ่งประสานงานด้วยตัวควบคุมเอสดีเอ็นในการรับ-ส่งแพ็กเก็ตวีดิทัศน์ที่สตรีมผ่านหนึ่งวิถีและพหุวิถี โดยได้สร้างระบบทดสอบเพื่อใช้ในการทดลอง 2 ชนิด ได้แก่ ระบบทดสอบโครงข่ายโอเพนโฟลว์เสมือนที่ถูกจำลองด้วยโปรแกรมมินิเน็ตและระบบทดสอบโครงข่ายโอเพนโฟลว์ระดับห้องปฏิบัติการที่จำลองด้วยกลุ่มคอมพิวเตอร์ที่ถูกติดตั้งโอเพนวีสวิตช์ ระบบทดสอบทั้ง 2 ระบบมีทอพอโลยีอย่างง่ายเหมือนกัน อันประกอบด้วยโอเพนวีสวิตช์ 4 ตัว ตัวควบคุมพอกซ์ภายนอก ตัวบริการสตรีมวีดิทัศน์ และตัวเล่นสตรีมวีดิทัศน์ นอกจากนี้ระบบทดสอบที่จำลองด้วยกลุ่มคอมพิวเตอร์ยังได้ถูกรวมเข้ากับมิดเดิลบอกซ์เพื่อใช้ในการรวมสตรีมของกลุ่มก้อนวีดิทัศน์ที่ถูกส่งให้แก่ตัวเล่นวีดิทัศน์ผ่านวิถี 2 วิถีที่แยกจากกันอย่างพร้อม ๆ กัน โอเพนวีสวิตช์ที่เชื่อมต่อกับตัวบริการวีดิทัศน์จะถูกสั่งการโดยพอกซ์เพื่อทำหน้าที่แบ่งกลุ่มก้อนวีดิทัศน์และส่งไปยังวิถีทั้ง 2 วิถี มิดเดิลบอกซ์และตัวควบคุมพอกซ์ในวิทยานิพนธ์นี้ถูกพัฒนาขึ้นโดยภาษาไพทอนและโปรแกรมสำเร็จที่ใช้ในการจัดการแพ็กเกต การทดลองถูกทำขึ้นเพื่อให้เข้าใจส่วนประกอบแต่ละส่วนของระบบทดสอบ การทดลองแรกแสดงการสตรีมวีดิทัศน์ทีซีพีและยูดีพีบนระบบทดสอบเสมือนที่ถูกจำลองด้วยมินิเน็ตและระบบทดสอบที่จำลองด้วยกลุ่มคอมพิวเตอร์ ซึ่งพบว่าระบบทดสอบเสมือนมีสมรรถนะดีกว่าระบบทดสอบที่จำลองด้วยกลุ่มคอมพิวเตอร์ โดยเห็นได้จากการสูญเสียแพ็กเกตและความหน่วงที่น้อยกว่า ทั้งนี้เนื่องจากข้อจำกัดทางฮาร์ดแวร์ในอินเตอร์เฟซทางกายภาพของโครงข่ายที่ถูกใช้ในระบบทดสอบที่จำลองด้วยกลุ่มคอมพิวเตอร์ การทดลองที่สองกล่าวถึงการพัฒนากำหนดการของแพ็กเกตที่มิดเดิลบอกซ์ด้วยบัฟเฟอร์เข้าก่อนออกก่อนจำนวน 2 บัฟเฟอร์เพื่อใช้สำหรับเก็บแพ็กเกตที่มาจากวิถีแต่ละวิถี กำหนดการที่ถูกโปรแกรมนี้จะส่งต่อแพ็กเกตไปให้ตัวเล่นวีดิทัศน์อย่างเป็นไปตามกำหนดเวลา โดยจะเลือกจากแพ็กเกตแรกในบัฟเฟอร์ใดบัฟเฟอร์หนึ่งที่มีตราเวลาอาร์ทีพีที่น้อยกว่า อัตราส่วนในการแบ่งกลุ่มก้อนวีดิทัศน์ของวิถีทั้ง 2 วิถี จะคำนวณได้จากการวิเคราะห์ความหน่วงของวิถีที่สมดุลกันโดยอ้างอิงระบบแถวคอย M/M/1 ด้วยการตั้งค่าทั้งหมดนี้พบว่า ระบบทดสอบที่จำลองด้วยกลุ่มคอมพิวเตอร์สามารถใช้สตรีมกลุ่มก้อนวีดิทัศน์ผ่านวิถี 2 วิถีได้อย่างมีประสิทธิภาพ ด้วยฮาร์ดแวร์ในระบบทดสอบที่จำลองด้วยกลุ่มคอมพิวเตอร์ขณะนี้ ได้พิสูจน์ให้เห็นว่าสามารถใช้สตรีมวีดิทัศน์ด้วยอัตราการเข้ารหัส 500 กิโลบิตต่อวินาทีได้อย่างเป็นผลสำเร็จ ในอนาคตได้คาดหวังว่าการสตรีมวีดิทัศน์ด้วยความละเอียดที่สูงขึ้นจะสามารถทำได้โดยการเพิ่มซีพียูและความจุโครงข่ายของเครื่องคอมพิวเตอร์ที่ใช้เป็นมิดเดิลบอกซ์ นอกจากนี้การทดลองสตรีมวีดิทัศน์ผ่านระบบทดสอบโอเพนโฟลว์ระดับระหว่างประเทศ เช่น TEIN ยังถือเป็นเรื่องที่น่าศึกษาต่อไปในอนาคตด้วยเช่นกัน

ภาควิชา วิศวกรรมไฟฟ้า	**ลายมือชื่อนิสิต** .....
สาขาวิชา วิศวกรรมไฟฟ้า	**ลายมือชื่อ อ.ที่ปรึกษาหลัก** .....
ปีการศึกษา 2557	**ลายมือชื่อ อ.ที่ปรึกษาร่วม** .....

5570286521 : MAJOR ELECTRICAL ENGINEERING
KEYWORDS: OPENFLOW/ SDN/ CHUNKED VIDEO STREAMING/ SINGLE PATH/MULTI-PATH/ TESTBED.

PARICHAT PANWAREE : SDN-COORDINATED MULTI-PATH CHUNKED VIDEO STREAMING . ADVISOR: ASST. PROF. CHAODIT ASWAKUL, Ph.D., CO-ADVISOR: PROF. KIM JONGWON, 76 pp.

This thesis has studied the performance of Openflow network coordinated by an SDN controller in the transmission and reception of video packet stream via single-path and multi-path scenarios. Two types of experimental testbeds have been built, namely, a Mininet-emulated virtual OpenFlow network testbed and a laboratory-scale OVS (Open vSwitch)-installed PC-cluster OpenFlow network testbed. Both testbeds share the same simple topology with four OVS's, an external POX controller, a video streaming server and a video streaming client. Moreover, the PC-cluster testbed has also been integrated with a middlebox for combining streams of video chunks travelling towards the video client via two disjoint paths simultaneously. Here, the OVS connecting to the video server has been commanded by a POX controller to take turn splitting video packets and sending to both paths. The middlebox and the POX controller have been both developed in this thesis by using Python and its packet-manipulation packages. Testings have been conducted for understanding each of the testbed components. Firstly, based on TCP and UDP video streaming in the Mininet virtual testbed and the PC-cluster testbed, it has been found that the virtual testbed outperforms the PC-cluster testbed with resultant lower packet loss and delay. This is due to the limitation in physical network interface hardware employed within the PC-cluster testbed. Secondly, the experiment is based on a packet scheduler implementation at the middlebox with two first-in-first-out buffers each dedicated for storing packets from each path. The scheduler is here programmed to forward towards the video client periodically the packet, waiting first in either of the buffers, with the earliest RTP timestamp. The ratio of splitting packets on both paths has been calculated from an M/M/1-based path delay equalisation analysis. With all these settings, it has been found that the PC-cluster testbed can be used to stream the chunked video packet streams effectively via the two paths. Based on the current hardware deployed in the developed PC-cluster testbed, up to 500-kbits/s video encoding bit rate has been successfully demonstrated. In the future, it is expected that higher resolution of video streaming can be achieved by upgrading the CPU and networking capacity of the computer running the middlebox. Also, an elaborate video streaming experiment via an international-scale OpenFlow network testbed such as TIEN is warrant of future worthy investigation.

Department : Electrical Engineering	**Student's Signature** .....
Field of Study : Electrical Engineering	**Advisor's Signature** .....
Academic Year : 2014	**Co-advisor's Signature** .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี ด้วยความช่วยเหลือจาก ผศ. ดร.เชาวน์ดิศ อัศวกุล อาจารย์ที่ปรึกษาวิทยานิพนธ์ และ ศ. ดร.คิม จองวอน อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม ซึ่งได้ให้คำแนะนำและข้อคิดเห็นต่าง ๆ อันเป็นประโยชน์อย่างยิ่งในการทำวิจัย ช่วยแก้ปัญหาที่เกิดขึ้นระหว่างการดำเนินงาน อีกทั้งตรวจทานงานวิทยานิพนธ์ฉบับนี้ด้วยดีเสมอมา ผู้วิจัยจึงขอกราบขอบพระคุณมา ณ ที่นี้ ขอขอบพระคุณ ผศ. ดร.สุภาวดี อร่ามวิทย์ ประธานกรรมการสอบวิทยานิพนธ์ ผศ. ดร.ชัยเชษฐ์ สายวิจิตรและ รศ.ดร.ภูมิพัฒ แสงอุดมเลิศ กรรมการสอบวิทยานิพนธ์ที่ได้สละเวลาตรวจสอบและให้คำแนะนำเพื่อให้วิทยานิพนธ์ฉบับนี้สมบูรณ์ยิ่งขึ้น และขอขอบพระคุณคณาจารย์ทุกท่านในสาขาวิชาไฟฟ้าสื่อสาร ที่ได้ประสิทธิประสาทความรู้อันเป็นพื้นฐานในการศึกษาและทำวิทยานิพนธ์นี้

งานวิจัยชิ้นนี้ได้รับทุนสนับสนุนจากโครงการขับเคลื่อนการวิจัย กองทุนรัชดาภิเษกสมโภช (Special Task Force for Activating Research (STAR) ภายใต้กลุ่มวิจัยโครงข่ายไร้สายและอินเทอร์เน็ตอนาคต (Wireless Network and Future Internet Research Group) จุฬาลงกรณ์มหาวิทยาลัย

ขอขอบคุณกลุ่มวิจัยโครงข่าย (Network Reserch Group) ซึ่งดูแลโดย ผศ. ดร.เชาวน์ดิศ อัศวกุล และ ผศ. ดร.ชัยเชษฐ์ สายวิจิตร ที่จัดกิจกรรมเพื่อส่งเสริมการเรียนรู้และการทำงานของผู้วิจัยให้มีประสิทธิภาพที่ดียิ่งขึ้น รวมถึงให้ความอนุเคราะห์อุปกรณ์เครื่องมือในการทำงานแก่ผู้วิจัย ทำให้งานวิทยานิพนธ์นี้สำเร็จได้อย่างสะดวกราบรื่น

ขอขอบพระคุณ ผศ. ดร.เชาวน์ดิศ อัศวกุล ที่ให้การสนับสนุนผู้วิจัยให้ได้มีโอกาสไปศึกษาแลกเปลี่ยน ณ สถาบัน GIST ประเทศเกาหลีใต้ และขอบพระคุณ ศ. ดร.คิม จองวอน ที่ให้คำแนะนำและคำปรึกษาที่มีประโยชน์แก่งานวิจัย อีกทั้งยังดูแลผู้วิจัยเป็นอย่างดี ตลอดระยะเวลาการทำวิจัยในประเทศเกาหลีใต้

ขอบคุณเพื่อนพี่น้องนักวิจัยทุกคน รวมถึงเจ้าหน้าที่ บุคลากรที่อยู่ในภาควิชาวิศวกรรมไฟฟ้า สาขาโทรคมนาคม จุฬาลงกรณ์มหาวิทยาลัย ที่ได้ให้ความช่วยเหลือในเรื่องต่าง ๆ และเป็นกำลังใจที่ดียิ่งต่อผู้วิจัย

สุดท้ายนี้ขอขอบพระคุณครอบครัวของผู้วิจัย ซึ่งได้ให้การสนับสนุนและเป็นกำลังใจให้แก่ผู้วิจัยเสมอมาจนสำเร็จการศึกษา

สารบัญ

บทที่ หน้า

สารบัญตาราง

หน้า

สารบัญรูป

หน้า

หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

โอเพนโฟลว์ (OpenFlow) [1] เป็นโพรโตคอลที่ใช้ในสถาปัตยกรรมเอสดีเอ็น (SDN : software-defined networking) [2] ซึ่งโอเพนโฟลว์ถือเป็นปรากฏการณ์ใหม่ของอุตสาหกรรมโครงข่าย และงานวิจัยพัฒนาวิศวกรรมโครงข่าย ปัจจุบันทั่วโลกให้ความสนใจโอเพนโฟลว์เป็นจำนวนมาก ประเทศหลายประเทศได้สร้างระบบทดสอบ (testbed) เพื่อใช้ทดลองงานวิจัยโอเพนโฟลว์โดยเฉพาะ เช่น OFELIA (OpenFlow in Europe: linking infrastructure and applications) [3] ในทวีปยุโรป GENI (global environment for network innovations) [4] ในประเทศสหรัฐอเมริกา หรือ NWGN (new-generation wireless network) [5] ในประเทศญี่ปุ่น เป็นต้น สำหรับงานวิจัยนี้ได้ทดลองสร้างระบบทดสอบโอเพนโฟลว์ โดยอ้างอิงจากโครงการความร่วมมือของกลุ่มวิจัยโครงข่าย ห้องปฏิบัติการวิจัยโทรคมนาคม ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย และเครือข่ายการวิจัยโอเพนโฟลว์ระหว่างประเทศ ภายใต้โครงการ OF@TEIN ที่เป็นระบบทดสอบโอเพนโฟลว์ ระหว่างสาธารณรัฐเกาหลีใต้ และ กลุ่มประเทศเอเชียตะวันออกเฉียงใต้ 5 ประเทศ ได้แก่ ไทย เวียดนาม มาเลเซีย อินโดนีเซีย และฟิลิปปินส์

ผู้วิจัยได้สนใจศึกษา และทดลองสร้างระบบทดสอบของโอเพนโฟลว์ขึ้นที่จุฬาลงกรณ์มหาวิทยาลัย โดยยกกรณีของการส่งรับวีดิทัศน์ผ่านโครงข่ายมาเป็นประเด็นศึกษา ซึ่งมีความน่าสนใจเนื่องจาก ในอนาคตมีการคาดการณ์ว่า ทราฟฟิกของอินเทอร์เน็ตส่วนใหญ่จะมาจากวีดิทัศน์ โดยเพิ่มขึ้นจาก 57% ของทราฟฟิกอินเทอร์เน็ตทั้งหมดในปี พ.ศ. 2555 เป็น 69% ในปี พ.ศ. 2560 [6] ซึ่งการมีทราฟฟิกจำนวนมาก จะทำให้เกิดปัญหาความคับคั่งภายในโครงข่ายอย่างหลีกเลี่ยงไม่ได้ และความคับคั่งนี้จะทำให้ผู้รับได้รับวีดิทัศน์ที่มีคุณภาพลดลงด้วย ซึ่งโอเพนโฟลว์เป็นเทคโนโลยีที่เอื้อต่อการจัดการความคับคั่ง เนื่องจากในโอเพนโฟลว์ระนาบควบคุม (control plane) ได้ถูกแยกออกจากระนาบข้อมูล (data plane) ทำให้ผู้บริหารโครงข่าย (administrator) สามารถจัดการความคับคั่งที่เกิดขึ้นในโครงข่ายได้โดยตรงผ่านระนาบควบคุม โดยงานวิจัยนี้เลือกศึกษาเปรียบเทียบการสตรีมวีดิทัศน์เป็นกลุ่มก้อน (chunked video streaming) ผ่านวิถีหนึ่งวิถี (single path) และพหุวิถี (multi-path) ดังนั้นวัตถุประสงค์ของงานวิจัยชิ้นนี้คือ เพื่อศึกษาเปรียบเทียบสมรรถนะของโครงข่ายที่ถูกควบคุมด้วยตัวควบคุมเอสดีเอ็น ในการรับ-ส่งลำดับของแพ็กเกตวีดิทัศน์ เมื่อทำการสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบหนึ่งวิถีและพหุวิถีบนระบบทดสอบโอเพนโฟลว์ขนาดเล็ก ในห้องปฏิบัติการวิจัยโทรคมนาคม อาคาร 4 ชั้น 13 ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

การสตรีมวีดิทัศน์ (video streaming) เป็นเทคโนโลยีที่ได้รับความนิยมเป็นอย่างมากในปัจจุบัน เว็บไซต์ เช่น YouTube [7], Ustream [8] ได้นำเทคโนโลยีนี้มาใช้ เพื่อให้บริการวีดิทัศน์บนอินเทอร์เน็ตแก่ผู้ใช้บริการ ถึงแม้ว่าผู้ใช้บริการจะสามารถรับชมวีดีทัศน์ผ่านทางอินเทอร์เน็ตได้ง่าย แต่ก็ยังคงไม่พึงพอใจกับคุณภาพของวีดิทัศน์ที่ได้รับชม เช่น วีดิทัศน์ไม่มีความต่อเนื่อง เกิดการหน่วง ภาพไม่คมชัด เป็นต้น ซึ่งปัญหาเหล่านี้อาจมีสาเหตุมาจากสมรรถนะของโครงข่าย รวมทั้งวิธีที่ใช้ในการสตรีมวีดิทัศน์ โดยทั่วไปแล้ว ผู้ให้บริการจะส่งวีดิทัศน์ไปยังผู้ใช้บริการปลายทางผ่านวิถีเพียงหนึ่งวิถี แม้ว่าระหว่างผู้ให้บริการและผู้ใช้บริการจะเชื่อมต่อกันผ่านวิถีหลายวิถีก็ตาม หากวิถีที่

กำลังถูกใช้เกิดความคับคั่ง หรือขัดข้อง ข้อมูลที่ถูกส่งสูญหายไประหว่างทาง ทำให้ผู้ใช้บริการได้รับข้อมูลล่าช้า หรือได้รับข้อมูลไม่ครบ ดังนั้นวีดิทัศน์ที่ผู้ใช้บริการได้รับจึงมีคุณภาพต่ำ ไม่เป็นไปอย่างที่ต้องการ จะเห็นได้ว่าการสตรีมวีดิทัศน์ไปบนวิถีหนึ่งวิถี เป็นการใช้ประโยชน์จากโครงข่ายได้ไม่เต็มสมรรถนะ จึงมีนักวิจัยหลายท่านเสนอแนวทางในการปรับปรุงสมรรถนะโครงข่ายและพัฒนาคุณภาพวีดิทัศน์ โดยการสตรีมวีดิทัศน์แบบพหุวิถี (multi-path video streaming) [9]-[12] และอีกวิธีหนึ่งคือ การแบ่งวีดิทัศน์ให้เป็นกลุ่มก้อน (chunk) [13]-[15] ก่อนส่งไปให้ผู้ใช้บริการ

งานวิจัยที่เกี่ยวกับการสตรีมวีดิทัศน์แบบพหุวิถี เช่น [9] ได้ใช้การแก้ปัญหาแบบศึกษาสำนึก (heuristic approach) ที่อ้างอิงการสมดุลโหลด (load balancing) เพื่อจัดสรรแบนด์วิดท์ให้วิถีแต่ละวิถี และจัดกำหนดการของแพ็กเกตวีดิทัศน์ (video packet scheduling) บนโครงข่ายที่มีวิถีหลายวิถีระหว่างเครื่องแม่ข่าย (server) และเครื่องลูกข่าย (client) โดยวิธีดังกล่าวให้ผลคือ ลดจำนวนข้อมูลวีดิทัศน์ที่สูญหาย ทำให้เครื่องลูกข่ายได้รับวีดิทัศน์คุณภาพสูงสุด นอกจากนี้งานวิจัย [10] ได้พัฒนาแผนการที่ใช้สำหรับการถ่ายทอดสด (live streaming) ด้วยทีซีพี (TCP) โดยเปรียบเทียบการส่งวีดิทัศน์แบบพหุวิถี และหนึ่งวิถี พบว่า การถ่ายทอดสดแบบพหุวิถีสามารถเพิ่มปริมาณงานทีซีพี (TCP throughput) และสามารถรองรับอัตราเร็วบิตของสื่อ (media bit rate) ได้มากกว่าการส่งแบบหนึ่งวิถี ส่วน [11] เสนอวิธีการสตรีมวีดิทัศน์แบบพหุวิถีไปยังผู้ใช้บริการในโครงข่ายเคลื่อนที่ (mobile network) เพื่อเพิ่มคุณภาพวีดิทัศน์ให้แก่ผู้ใช้บริการ โดยมีขั้นตอนคือ เลือกวิถีที่ใช้ส่ง และจัดกำหนดการข้อมูลวีดิทัศน์ จากนั้นวัดคุณภาพวีดิทัศน์ที่ผู้ใช้ได้รับเทียบกับวิธีใน [9] ซึ่งวัดจาก PSNR (peak signal to noise ratio) ผลที่ได้คือผู้ใช้บริการได้รับวีดิทัศน์ที่มีค่า PSNR มากกว่าใช้วิธี [9] คือได้รับวีดิทัศน์ที่มีคุณภาพดีกว่านั่นเอง และ [12] ได้เสนอหลักการ CMT-QA (quality-aware adaptive concurrent multipath transfer) สำหรับการส่งข้อมูล และการส่งวีดิทัศน์แบบเวลาจริง (real time) ผ่านวิถีหลายวิถี ซึ่งมีวิธีการคือ กระจายข้อมูลที่ถูกแบ่งเป็นกลุ่มก้อนไปยังวิถีหลายวิถี และควบคุมอัตราทราฟฟิกข้อมูลของวิถีแต่ละวิถี เพื่อลดจำนวนข้อมูลที่ไม่เป็นไปตามลำดับ (out-of-order) และลดการส่งข้อมูลใหม่โดยไม่จำเป็น ทำให้การส่งข้อมูลมีประสิทธิภาพมากยิ่งขึ้น

นอกจากนี้ยังมีงานวิจัยที่ศึกษาเรื่องกลุ่มก้อนวีดิทัศน์ (chunked video) เช่น [13] ได้เสนอวิธีการเข้ารหัสก้อน HTTP (HTTP chunked encoding) เพื่อลดเวลาแฝง (latency) ในการถ่ายทอดสดวีดิทัศน์ ใน [14] เสนอวิธีแก้ปัญหาการจัดกำหนดการของก้อนวีดิทัศน์ และการจัดสรรแบนด์วิดท์สำหรับระบบวีดิทัศน์ตามคำขอ (video on demand : VoD) ในโครงข่ายระดับเดียวกัน (peer-to-peer : P2P) ซึ่งวิธีที่นำเสนอทำให้เพียร์สามารถใช้ประโยชน์จากแบนด์วิดท์ได้อย่างสูงสุด และ [15] ได้ปรับปรุงความต่อเนื่องของวีดิทัศน์สำหรับระบบวีดิทัศน์ตามคำขอในโครงข่ายระดับเดียวกัน โดยใช้วิธีการ PRCP (peer ratio-based chunk regulation) ส่วน [16] แสดงการเพิ่มคุณภาพวีดิทัศน์ของการสตรีมวีดิทัศน์ในโครงข่ายระดับเดียวกัน โดยการส่งก้อนวีดิทัศน์ขนาดแปรผันได้ (variable size chunk) และพบว่าเพียร์ได้รับวีดิทัศน์ที่มีค่า PSNR สูง ขณะที่ผู้ให้บริการวีดิทัศน์สามารถลดการใช้แบนด์วิทด์ลงได้

จะเห็นได้ว่าทั้งการสตรีมวีดิทัศน์แบบพหุวิถี และการแบ่งวีดิทัศน์เป็นกลุ่มก้อน ช่วยให้สรรมถนะโครงข่าย และคุณภาพวีดิทัศน์ดีขึ้น ดังนั้น วิทยานิพนธ์นี้จึงเกิดความสนใจที่จะนำวิธีการทั้ง 2 วิธีมาประยุกต์ใช้ร่วมกัน เพื่อเพิ่มคุณภาพวีดิทัศน์บนโครงข่ายโอเพนโฟลว์ หลายปีที่ผ่านมีงานวิจัยที่ศึกษาเกี่ยวกับวีดิทัศน์บนโครงข่ายโอเพนโฟลว์ เช่น [17]-[20] ซึ่งงานเหล่านี้พิจารณาการส่งวีดิทัศน์จากผู้ให้บริการไปยังผู้ใช้บริการแบบหนึ่งวิถี วิทยานิพนธ์นี้จึงต้องการศึกษาการสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถีบนโครงข่ายโอเพนโฟลว์ เพื่อให้สามารถใช้ประโยชน์จากโครงข่ายได้อย่างเต็มสมรรถนะ พร้อมทั้งเพิ่มคุณภาพของวีดิทัศน์ ณ ผู้ใช้ปลายทาง

1.2 วัตถุประสงค์ของวิทยานิพนธ์

เพื่อศึกษาเปรียบเทียบสมรรถนะของโครงข่ายที่ถูกควบคุมด้วยตัวควบคุมเอสดีเอ็น ในการรับ-ส่งลำดับของแพ็กเกตวีดิทัศน์ เมื่อทำการสตรีมวีดิทัศน์แบบหนึ่งวิถีและสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถี บนระบบทดสอบโอเพนโฟลว์ขนาดเล็กในห้องปฏิบัติการวิจัยโทรคมนาคม อาคาร 4 ชั้น 13 ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

1.3 ขอบเขตวิทยานิพนธ์

1. สตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถี บนระบบทดสอบโอเพนโฟลว์ในห้องปฏิบัติการวิจัยโทรคมนาคม อาคาร 4 ชั้น 13 ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย และจะพิจารณาการสตรีมวีดิทัศน์บนวิถี 2 วิถีเท่านั้น
2. พิจารณากลุ่มก้อนวีดิทัศน์ที่มีขนาดคงที่ (fixed-chunk size) บนวิถีแต่ละวิถี ไม่แปรตามเวลาตลอดช่วงเวลาการรับ-ส่งวีดิทัศน์ แต่ทั้งนี้ขนาดของกลุ่มก้อนวีดิทัศน์อาจขึ้นอยู่กับความจุ (capacity) ของวิถีแต่ละวิถี
3. สตรีมและเข้ารหัส (encode) วีดิทัศน์ด้วยอัตราบิตคงที่ ที่ 500 กิโลบิตต่อวินาที โดยพิจารณาการสตรีมวีดิทัศน์บนโพรโตคอลอาร์ทีพี (RTP) เท่านั้น
4. ไม่พิจารณาความเรียบเนียน (smooth) ของวีดิทัศน์ที่แสดง ณ ผู้รับปลายทาง เนื่องจากไม่พิจารณาการทำงานของชั้นโปรแกรมยุกต์ (application layer) และโปรแกรมที่ใช้แสดงวีดิทัศน์ ซึ่งความเรียบเนียนของวีดิทัศน์จะแตกต่างกันไปตามโปรแกรมที่เลือกใช้ แต่จะพิจารณาตัวชี้วัดเป็นการสูญเสียแพ็กเกตที่เกิดขึ้น ณ ตัวเล่นวีดิทัศน์

1.4 ประโยชน์ที่คาดว่าจะได้รับ

องค์ความรู้ในการเพิ่มสมรรถนะของโครงข่ายโอเพนโฟลว์ ที่ถูกควบคุมด้วยตัวควบคุมเอสดีเอ็น เพื่อให้สตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถีได้อย่างมีประสิทธิภาพ ส่งผลให้โปรแกรมประยุกต์ที่ใช้แสดงวีดิทัศน์ ณ ปลายทางได้รับแพ็กเกตวีดิทัศน์ที่มีการไหว (jitter) โดยเฉลี่ยอย่างเท่า ๆ กัน

1.5 ประมวลวิทยานิพนธ์

บทที่ 1 บทนำ: กล่าวถึงความเป็นมาของเอสดีเอ็นและระบบทดสอบโอเพนโฟลว์ ความสำคัญของการสตรีมวีดิทัศน์ รวมถึงงานวิจัยในอดีตที่เกี่ยวข้องและประโยชน์ของการสตรีมวีดิทัศน์แบบพหุวิถีและการแบ่งวีดิทัศน์เป็นกลุ่มก้อน

บทที่ 2 เอสดีเอ็นและการสตรีมวีดิทัศน์แบบพหุวิถี: กล่าวถึงลักษณะของสถาปัตยกรรมโครงข่ายเอสดีเอ็น โพรโตคอลโอเพนโฟลว์ การทำงานของสวิตช์โอเพนโฟลว์ และวิธีการสตรีมวีดิทัศน์แบบพหุวิถี

บทที่ 3 การทดลองบนโครงข่ายโอเพนโฟลว์ด้วยโปรแกรมมินิเน็ต: กล่าวถึงการทดลองเบื้องต้นเมื่อทำการจำลองสร้างโครงข่ายโอเพนโฟลว์โดยใช้โปรแกรมมินิเน็ต แล้วทดลองการรับ-ส่งแพ็กเกต

และสตรีมวีดิทัศน์ผ่านโครงข่ายโอเพนโฟลว์เสมือน และแสดงการเปรียนเทียบสมรรถนะระหว่างโครงข่ายโอเพนโฟลว์เสมือน กับโครงข่ายโอเพนโฟลว์จริงบนระบบทดสอบในห้องปฏิบัติการ เมื่อทดลองสตรีมวีดิทัศน์ไปบนวิถีแต่ละวิถี

บทที่ 4 การสตรีมกลุ่มก้อนวีดิทัศน์แบบพหุวิถีบนระบบทดสอบโอเพนโฟลว์ในห้องปฏิบัติการ: อธิบายโครงสร้างของระบบทดสอบโอเพนโฟลว์ขนาดเล็กที่พิจารณา การทำงานของอุปกรณ์แต่ละตัวที่อยู่ในระบบทดสอบ และแสดงผลการทดลองเมื่อสตรีมกลุ่มก้อนวีดิทัศน์แบบพหุวิถี บนระบบทดสอบโอเพนโฟลว์ที่สร้างขึ้นมาภายในห้องปฏิบัติการ

บทที่ 5 บทสรุปและข้อเสนอแนะ: สรุปงานวิจัยทั้งหมดในวิทยานิพนธ์ฉบับนี้และเสนอแนวทางในการพัฒนางานวิจัยต่อไป

บทที่ 2

เอสดีเอ็นและการสตรีมวีดิทัศน์แบบพหุวิถี

2.1 เอสดีเอ็น

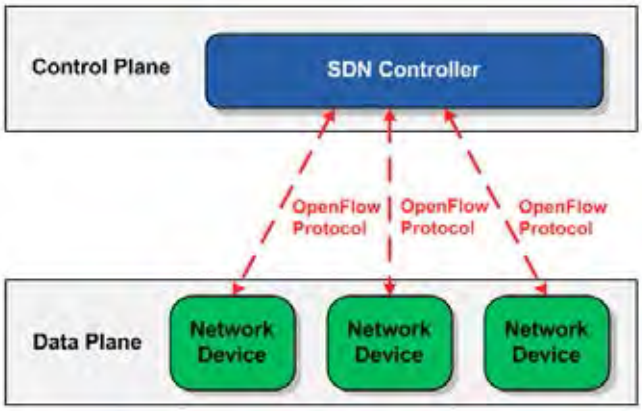
ในสถาปัตยกรรมโครงข่ายแบบดั้งเดิมที่ใช้งานอยู่ในปัจจุบัน พบว่า มีความซับซ้อนและยุ่งยากในการบริหารจัดการการให้บริการด้วยโครงข่าย เนื่องจากในโครงข่ายมีการใช้โพรโตคอลหลายชนิด ซึ่งโพรโตคอลแต่ละชนิดมีหน้าที่และการทำงานแตกต่างกันไป นอกจากนี้ยังมีการใช้งานอุปกรณ์โครงข่าย (network device) จากผู้ผลิตหลายราย ซึ่งแต่ละรายย่อมไม่ต้องการเปิดเผยการทำงานภายในอุปกรณ์ของตนเอง ทำให้ไม่สามารถเชื่อมต่ออุปกรณ์โครงข่ายจากผู้ผลิตต่าง ๆ เข้าด้วยกันได้ ดังนั้นเอสดีเอ็นจึงเป็นสถาปัตยกรรมโครงข่ายรูปแบบใหม่ที่ถูกพัฒนาขึ้นมา เพื่อแก้ปัญหาที่มีในสถาปัตยกรรมโครงข่ายแบบดั้งเดิมข้างต้น โดยมีหลักการดังนี้

อุปกรณ์โครงข่าย เช่น สวิตช์ (switch) หรืออุปกรณ์จัดเส้นทาง (router) ในสถาปัตยกรรมโครงข่ายแบบดั้งเดิม ได้ถูกรวมระนาบควบคุมและระนาบข้อมูลไว้ในอุปกรณ์ตัวเดียวกัน ดังนั้นระนาบควบคุมจะควบคุมการทำงานเฉพาะอุปกรณ์ตัวนั้นเพียงตัวเดียว ไม่สามารถควบคุมอุปกรณ์ตัวอื่นได้ แต่ในสถาปัตยกรรมเอสดีเอ็นระนาบควบคุมจะถูกแยกออกจากระนาบข้อมูลดังแสดงในรูปที่ 2.1 ทำให้อุปกรณ์โครงข่ายมีเพียงระนาบข้อมูลที่ใช้ในการส่งต่อ (forwarding) ข้อมูลเท่านั้น ส่วนระนาบควบคุมจะมีตัวควบคุมเอสดีเอ็น (SDN controller) ที่ทำหน้าที่เป็นศูนย์กลางควบคุมการทำงานของโครงข่ายทั้งหมด ผู้บริหารโครงข่ายสามารถเขียนโปรแกรมเพื่อควบคุมโครงข่ายผ่านตัวควบคุมเอสดีเอ็นได้โดยตรง ซึ่งตัวควบคุมเอสดีเอ็นจะติดต่อสื่อสารกับอุปกรณ์โครงข่ายทุกตัวผ่านโพรโตคอลโอเพนโฟลว์ จะเห็นได้ว่า การรวมการควบคุมโครงข่ายทั้งหมดไว้ที่ตัวควบคุมเอสดีเอ็นนั้นทำให้ผู้บริหารโครงข่ายสามารถมองภาพรวมของโครงข่ายได้ชัดเจน และสามารถบริหารจัดการโครงข่ายได้ง่ายขึ้น

ปัจจุบันผู้ผลิตอุปกรณ์โครงข่ายหลายรายได้เพิ่มความสามารถในการสื่อสารผ่านมาตรฐานเปิดของโอเพนโฟลว์ให้กับอุปกรณ์ของตน ทำให้สามารถนำอุปกรณ์โครงข่ายที่รองรับโอเพนโฟลว์นี้มาใช้งานในสถาปัตยกรรมเอสดีเอ็นได้ และยังสามารถเชื่อมต่อกับอุปกรณ์โครงข่ายที่รองรับโอเพนโฟลว์จากผู้ผลิตต่าง ๆ ได้อีกด้วย

2.2 โอเพนโฟลว์

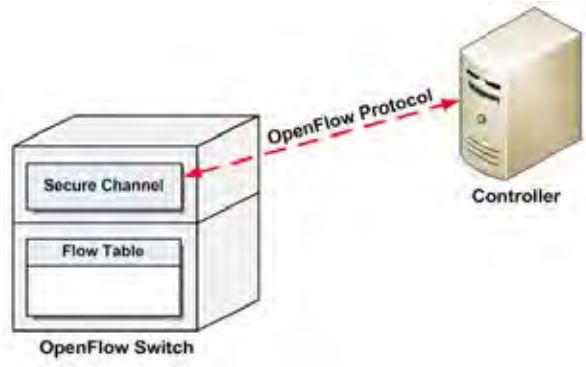
โอเพนโฟลว์เป็นโพรโตคอลแรกที่ถูกพัฒนาขึ้น เพื่อใช้ในการติดต่อสื่อสารระหว่างระนาบควบคุมและระนาบข้อมูลของสถาปัตยกรรมเอสดีเอ็น แนวคิดของโอเพนโฟลว์ คือ ใช้โฟลว์ (flow) กำหนดทราฟฟิกของโครงข่าย โดยอาศัยหลักเกณฑ์การจับคู่ (match rule) ระหว่างส่วนหัว (header) ของแพ็กเกตกับตารางโฟลว์ (flow table) ที่ถูกนิยามไว้ก่อนแล้วโดยซอฟต์แวร์ที่อยู่ในตัวควบคุมเอสดีเอ็น ผู้บริหารโครงข่ายสามารถกำหนดโฟลว์ของทราฟฟิกที่ผ่านอุปกรณ์โครงข่าย เช่น สวิตช์ ได้ ซึ่งสวิตช์ที่รองรับโอเพนโฟลว์ จะเรียกว่า "สวิตช์โอเพนโฟลว์ (OpenFlow switch)"



รูปที่ 2.1: สถาปัตยกรรมเอสดีเอ็น [2]

2.2.1 สวิตช์โอเพนโฟลว์

สวิตช์โอเพนโฟลว์ประกอบด้วย ตารางโฟลว์ และช่องสัญญาณแบบปลอดภัย (secure channel) [21] โดยตารางโฟลว์จะใช้สำหรับตรวจดู (lookup) แพ็กเกตเพื่อจับคู่ ส่วนช่องสัญญาณแบบปลอดภัยมีหน้าที่ในการติดต่อสื่อสารระหว่างตัวควบคุมภายนอกและสวิตช์ โดยใช้โพรโตคอลโอเพนโฟลว์ ดังแสดงในรูปที่ 2.2



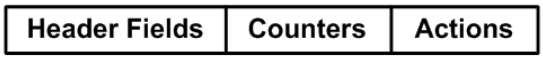
รูปที่ 2.2: การติดต่อระหว่างสวิตช์โอเพนโฟลว์กับเครื่องควบคุมผ่านช่องสัญญาณแบบปลอดภัย

ภายในตารางโฟลว์จะบรรจุเซตของโฟลว์เอนทรี (flow entry) ตัวนับกิจกรรม (activity counter) และเซตของการกระทำ (action) ที่ใช้สำหรับจับคู่ให้แพ็กเกต (matching packet) สวิตช์จะประมวลผลแพ็กเกตทั้งหมด โดยนำไปเปรียบเทียบกับตารางโฟลว์ หากพบว่าส่วนหัว (header) ของแพ็กเกตตรงกับโฟลว์เอนทรีในตาราง (แพ็กเกตถูกจับคู่) แพ็กเกตนั้นก็จะถูกกระทำตามที่ระบุอยู่ในโฟลว์เอนทรี เช่น ส่งแพ็กเกตออกไปยังช่องทาง (port) ที่กำหนดไว้ แต่หากส่วนหัวของแพ็กเกตไม่ตรงกับโฟลว์เอนทรีใด ๆ หรือแพ็กเกตไม่ถูกจับคู่ ในกรณีนี้ข้อมูลส่วนหัวของแพ็กเกตนั้นจะถูกส่งไปยังตัวควบคุม แต่หากสวิตช์ไม่มีบัฟเฟอร์ หรือบัฟเฟอร์เต็ม สวิตช์จะส่งทั้งแพ็กเกตไปยังตัวควบคุมผ่านช่องสัญญาณแบบปลอดภัยแทน ตัวควบคุมจะกำหนดว่าต้องจัดการกับแพ็กเกตที่ไม่ถูกจับคู่อย่างไร นอกจากนี้ตัวควบคุมยังสามารถสั่งการให้สวิตช์เพิ่มหรือลบโฟลว์เอนทรีออกจากตารางโฟลว์ได้อีกด้วย

2.2.2 ตารางโฟลว์

โฟลว์เอนทรีแต่ละแถวที่อยู่ในตารางโฟลว์จะประกอบด้วย 3 ส่วน [21] ดังรูปที่ 2.3 ได้แก่

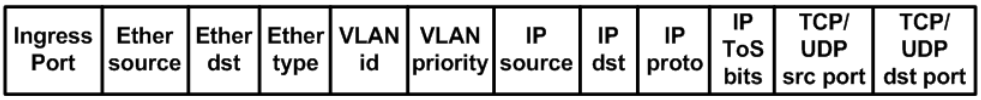
- **เขตข้อมูลส่วนหัว (header field)** : ใช้เพื่อเปรียบเทียบกับส่วนหัวของแพ็กเกตที่วิ่งเข้ามายังสวิตช์
- **ตัวนับ** : ใช้ปรับ (update) ข้อมูลเชิงสถิติ เช่น จำนวนแพ็กเกตที่ถูกรับ-ส่ง หรือเวลา เป็นต้น
- **การกระทำ** : ใช้ระบุการกระทำต่อแพ็กเกตที่ถูกจับคู่



รูปที่ 2.3: โฟลว์เอนทรี

เขตข้อมูลส่วนหัว

ในโฟลว์เอนทรีแต่ละเอนทรี เราสามารถเลือกชนิดเขตข้อมูลเพื่อใช้ในการจับคู่กับส่วนหัวของแพ็กเกตได้ ยกตัวอย่างเช่น หากสวิตช์รองรับตัวพรางเครือข่ายย่อย (subnet mask) ในเขตข้อมูลไอพี (IP) ต้นทาง และ/หรือ ไอพีปลายทาง เราก็สามารถเลือกใช้เขตข้อมูลไอพีกำหนดการจับคู่ได้ โดยเขตข้อมูลที่สามารถใช้ได้ในโอเพนโฟลว์มีทั้งหมด 12 ประเภท [21] ดังแสดงในรูปที่ 2.4



รูปที่ 2.4: เขตข้อมูลที่สามารถใช้เพื่อจับคู่แพ็กเกต

ตัวนับ

ตัวนับจะนับค่าต่าง ๆ เช่น จำนวนแพ็กเกตที่ถูกจับด้วยเงื่อนไขในแถวแต่ละแถวของตารางการจับคู่ จำนวนแพ็กเกตที่ถูกรับ (received packet) เข้ามาในช่องทาง (port) แต่ละช่องทาง หรือจำนวนไบต์ที่ส่งออก (transmitted byte) ผ่านช่องทางแต่ละช่องทาง เป็นต้น

การกระทำ

โฟลว์เอนทรีแต่ละเอนทรี อาจจะไม่มีการกระทำ (ไม่ได้ระบุการกระทำไว้ในโฟลว์เอนทรี) หรือสามารถมีการกระทำมากกว่า 1 การกระทำได้ โดยการกระทำนี้จะเป็นตัวบอกสวิตช์ว่าต้องจัดการกับแพ็กเกตที่ถูกจับคู่อย่างไร หากไม่มีการกระทำระบุไว้ แพ็กเกตนั้นจะถูกทิ้งไป (drop) ตัวอย่างการกระทำของสวิตช์ต่อแพ็กเกตที่เข้ามา เช่น

- **ส่งต่อ (forward)** : สวิตช์โอเพนโฟลว์จะส่งต่อแพ็กเกตออกไปยังช่องทางกายภาพ (physical port) และช่องทางเสมือน ในลักษณะต่าง ๆ ได้แก่
  - **ALL** : ส่งแพ็กเกตออกไปยังอินเตอร์เฟซ (interface) ทุกอัน ยกเว้นอินเตอร์เฟซที่แพ็กเกตเข้ามา
  - **CONTROLLER** : ห่อ (encapsulate) แล้วส่งแพ็กเกตไปยังตัวควบคุม
  - **IN_PORT** : ส่งแพ็กเกตออกไปยังช่องทางเข้า (input port)
  - **FLOOD** : กระจายแพ็กเกตออกไปทุกทิศทางด้วยต้นไม้แบบทอดข้ามน้อยที่สุด (minimum spanning tree) ยกเว้นอินเตอร์เฟซที่แพ็กเกตเข้ามา

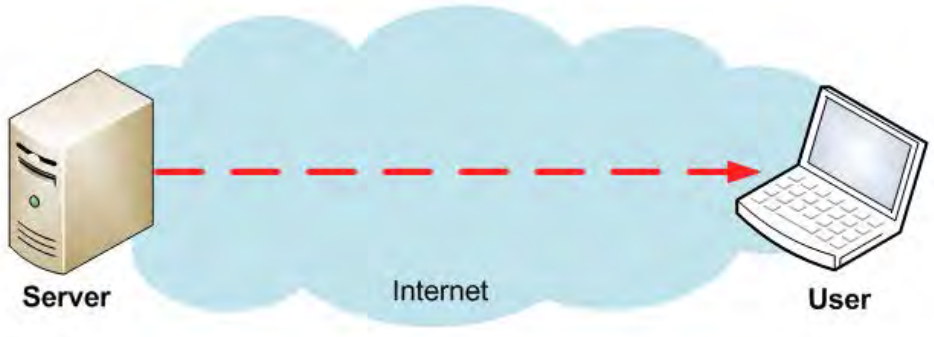
- **เข้าคิว (enqueue)** : ส่งแพ็กเกตออกไปยังช่องทางออก พร้อมทั้งตั้งค่าคิวให้แพ็กเกต เพื่อกำหนดคุณภาพบริการให้คิวแต่ละคิว
- **ทิ้ง (drop)** : ทุกแพ็กเกตที่ถูกจับคู่กับโฟลว์เอนทรีที่ไม่มีการกระทำกำหนดไว้ รวมถึงแพ็กเกตที่ไม่เป็นไปตามเงื่อนไข เช่น ความมั่นคง (security) หรือ การควบคุมความคับคั่ง (congestion control) ที่กำหนดไว้ จะถูกทิ้งทั้งหมด
- **แก้ไขเขตข้อมูล (modify field)** : แก้ไขเขตข้อมูลส่วนหัวของแพ็กเกต

2.2.3 การจับคู่ (matching)

เมื่อแพ็กเกตเข้ามายังสวิตช์ผ่านช่องทางเข้า สวิตช์จะนำเขตข้อมูลส่วนหัวของแพ็กเกตไปตรวจเทียบว่าตรงกับเอนทรีในตารางหรือไม่ หากตรงกับเอนทรีใด แพ็กเกตก็จะถูกกระทำตามการกระทำที่ระบุในเอนทรีนั้น ถ้าแพ็กเกตสามารถจับคู่ได้กับโฟลว์เอนทรีหลายเอนทรี ซึ่งการจับคู่จะอ้างอิงจากลำดับความสำคัญ โดยโฟลว์เอนทรีที่มีลำดับความสำคัญสูงย่อมถูกจับคู่ก่อนเสมอ แต่หากโฟลว์เอนทรีมีลำดับความสำคัญเท่ากัน สวิตช์สามารถเลือกได้เองว่าจะจับคู่แพ็กเกตกับโฟลว์เอนทรีใด เมื่อแพ็กเกตที่ถูกจับคู่กับโฟลว์เอนทรี ตัวนับที่เกี่ยวข้องกับเอนทรีนั้นจะถูกปรับ แต่ถ้าแพ็กเกตไม่ถูกจับคู่กับโฟลว์เอนทรีใดเลย แพ็กเกตนั้นจะถูกส่งไปยังตัวควบคุมผ่านช่องสัญญาณปลอดภัย

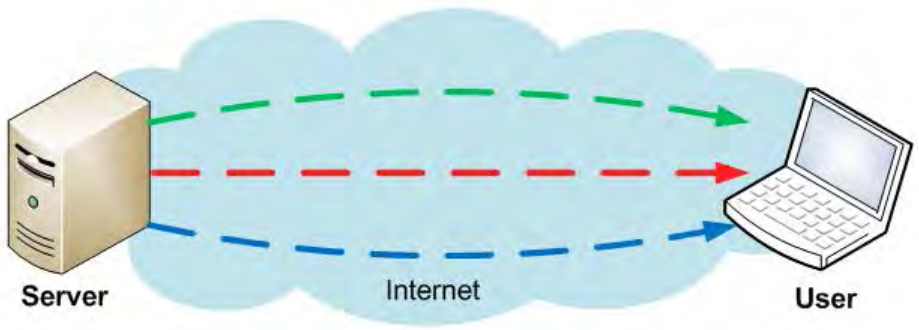
2.3 การสตรีมวีดิทัศน์แบบพหุวิถี

การสตรีมวีดิทัศน์บนอินเทอร์เน็ตแบบหนึ่งวิถี (single path) ดังแสดงในรูปที่ 2.5 เป็นวิธีที่ผู้ให้บริการส่วนใหญ่ใช้เพื่อสตรีมวีดิทัศน์ไปยังผู้ใช้บริการ แม้ว่าผู้ใช้บริการจะเชื่อมต่อกับผู้ให้บริการผ่านวิถีหลายวิถีก็ตาม ซึ่งการสตรีมวีดิทัศน์แบบหนึ่งวิถีย่อมพบกับความคับคั่ง อันนำไปสู่ปัญหาต่าง ๆ เช่น การสูญหายของแพ็กเกต การหน่วง (delay) ของข้อมูล ส่งผลให้ผู้ใช้บริการได้รับวีดิทัศน์ที่มีคุณภาพต่ำอย่างหลีกเลี่ยงไม่ได้ จึงมีงานวิจัยหลายชิ้น [9]-[12] นำเสนอวิธีการสตรีมวีดิทัศน์แบบพหุวิถีเพื่อแก้ปัญหาข้างต้น โดยผู้ให้บริการจะสตรีมวีดิทัศน์ไปยังผู้ใช้บริการผ่านวิถีหลายวิถีที่สามารถใช้งานได้ในขณะนั้น และหลีกเลี่ยงวิถีที่มีความคับคั่ง ทำให้จำนวนแพ็กเกตที่สูญหายระหว่างการส่งวีดิทัศน์ และความหน่วงลดลง ผู้ใช้บริการจึงได้รับวีดิทัศน์ที่มีคุณภาพดีขึ้น



รูปที่ 2.5: การสตรีมวีดิทัศน์แบบหนึ่งวิถี

นอกจากนี้ การสตรีมวีดิทัศน์แบบพหุวิถียังมีข้อดีอีกประการหนึ่ง คือ เพิ่มแบนด์วิดท์ที่ใช้ในการส่งวีดิทัศน์ [22] เนื่องจากการสตรีมวีดิทัศน์แบบหนึ่งวิถีเป็นการจำกัดแบนด์วิดท์ที่ในการสตรีมวีดิทัศน์ ผู้ให้บริการจึงต้องบีบอัดวีดิทัศน์ให้มีขนาดเล็กลง เพื่อให้สามารถส่งวีดิทัศน์ไปยังผู้ใช้บริการได้ ทำให้ผู้ใช้บริการได้รับวีดิทัศน์ที่มีคุณภาพต่ำ ซึ่งการสตรีมวีดิทัศน์แบบพหุวิถีทำให้สามารถใช้แบนด์วิดท์ได้มากขึ้น การสตรีมวีดิทัศน์จึงมีประสิทธิภาพมากขึ้น ซึ่งทำให้วีดิทัศน์มีคุณภาพดีขึ้น โดยตัวอย่างการสตรีมวีดิทัศน์แบบพหุวิถีสามารถแสดงได้ดังรูปที่ 2.6

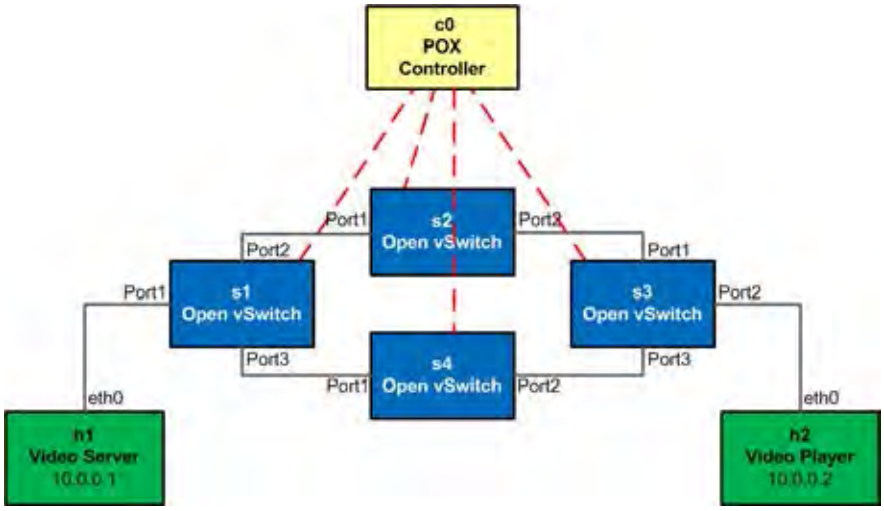


รูปที่ 2.6: การสตรีมวีดิทัศน์แบบพหุวิถี

บทที่ 3

การทดลองบนโครงข่ายโอเพนโฟลว์ด้วยโปรแกรมมินิเน็ต

งานวิจัยนี้ได้ทำการทดลองเบื้องต้นบนโครงข่ายโอเพนโฟลว์ ที่ถูกจำลองโดยใช้โปรแกรมมินิเน็ต (Mininet) [23] ซึ่งเป็นโปรแกรมที่ใช้สร้างโครงข่ายโอเพนโฟลว์เสมือน อันประกอบไปด้วยตัวควบคุม สวิตช์ และแม่ข่าย (host) อีกทั้งโปรแกรมมินิเน็ตยังรองรับโอเพนโฟลว์ และเอสดีเอ็น โดยตัวควบคุม เอสดีเอ็นและสวิตช์โอเพนโฟลว์ที่อยู่ในโปรแกรมมินิเน็ต ได้แก่ ตัวควบคุมพอกซ์ (POX controller) [25] และโอเพนวีสวิตช์ (Open vSwitch) [26] ตามลำดับ ผู้วิจัยได้จำลองโครงข่ายโอเพนโฟลว์ที่มีลักษณะทอพอโลยี (topology) ดังรูปที่ 3.1 เนื่องจากงานวิจัยนี้ต้องการศึกษาการสตรีมวีดิทัศน์แบบพหุวิถี ดังนั้นผู้วิจัยจึงออกแบบทอพอโลยีที่ใช้ทดลองให้มีวิถีระหว่างต้นทางและปลายทาง 2 วิถี อย่างสมมาตรกัน โดยโครงข่ายโอเพนโฟลว์เสมือนที่ถูกจำลองขึ้นมานี้ประกอบไปด้วย ตัวบริการวีดิทัศน์ (video server) ตัวเล่นวีดิทัศน์ (video player) ตัวควบคุมเอสดีเอ็น (พอกซ์) และโอเพนวีสวิตช์ จำนวน 4 ตัว



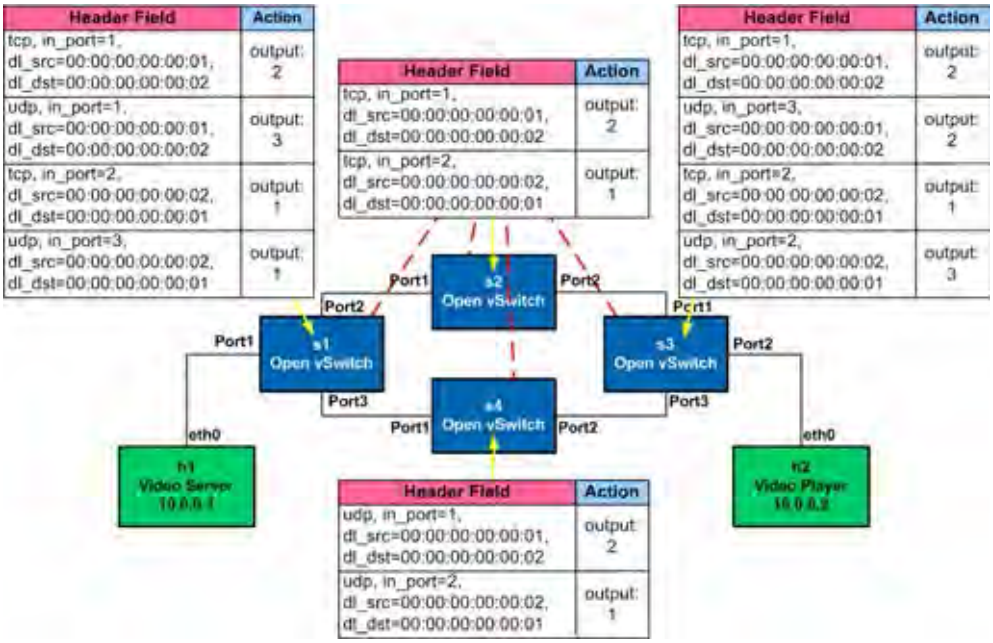
รูปที่ 3.1: ทอพอโลยีโครงข่ายโอเพนโฟลว์เสมือน

จากแผนภาพจะเห็นว่า ตัวบริการวีดิทัศน์ (h1) และตัวเล่นวีดิทัศน์ (h2) สามารถติดต่อกันได้ผ่านวิถี 2 วิถี โดยวิถีแรกติดต่อกันผ่านโอเพนวีสวิชต์ 1, 2 และ 3 (s1, s2 และ s3) และวิถีที่ 2 ผ่านโอเพนวีสวิชต์ 1, 4 และ 3 (s1, s4 และ s3) ผู้วิจัยได้เขียนโปรแกรมภาษาไพทอน (python) เพื่อให้ตัวควบคุมพอกซ์ใช้ควบคุมการส่งข้อมูลภายในโครงข่าย จากนั้นจึงทดลองส่งแพ็กเกตระหว่าง h1 และ h2 และทดลองสตรีมวีดิทัศน์จาก h1 ไป h2 ซึ่งผลการทดลองเป็นไปดังนี้

3.1 การทดลองที่ 1: การรับ-ส่งแพ็กเกต

ในการทดลองที่ 1 ผู้วิจัยได้ออกแบบการทดลอง ให้ h1 และ h2 สามารถรับ-ส่งแพ็กเกตได้ ซึ่งแพ็กเกตประเภททีซีพี จะถูกส่งผ่านวิถีที่ 1 (s1-s2-s3) ส่วนแพ็กเกตประเภทยูดีพี (UDP) จะถูกส่งผ่านวิถีที่ 2 (s1-s4-s3) ซึ่งตารางโฟลว์สำหรับโอเพนวีสวิตช์แต่ละตัว

สามารถแสดงได้ดังรูปที่ 3.2 ในการทดลองนี้จะใช้ทีซีพีและยูดีพีเป็นเพียงไอดีของแพ็กเกตเท่านั้น เพื่อให้โอเพนวีสวิตช์สามารถจำแนกชนิดแพ็กเกต แล้วส่งออกไปยังช่องทางที่ต่างกันได้ โดยไม่ได้ดำเนินงาน (run) โพรโตคอลทีซีพีและยูดีพีที่ h1 และ h2 การทดลองนี้ใช้โปรแกรมสกาปี (scapy) [27] สร้างแพ็กเกตทั้ง 2 ประเภทขึ้นมา โดยโปรแกรมที่ 3.1 เป็นโปรแกรมสร้างแพ็กเกตทีซีพี และโปรแกรมที่ 3.2 เป็นโปรแกรมสร้างแพ็กเกตยูดีพี



รูปที่ 3.2: ตารางโฟลว์ของโอเพนวี สวิตช์ ในการทดลองที่ 1

```
# send_packet.py by Parichat Pannwaree.
# This file is for creating and sending tcp packet.

#! /usr/bin/env python
from scapy.all import *

def packet():
        infile = open('payload','r')
        vpayload = infile.readlines()
        infile.close
        p = Ether()/IP()
        p.dst = "00:00:00:00:00:02"
        p.src = "00:00:00:00:00:01"
        p[IP].proto = 6   #tcp
        p[IP].payload = str(vpayload)
        return p

if __name__ == '__main__':
        p = packet()
        print p.show
        sendp(p, iface = 'h1-eth0')
```

โปรแกรมที่ 3.1: โปรแกรมสร้างแพ็กเกตทีซีพี

```
# send_packet2.py by Parichat Pannwaree.
# This file is for creating and sending udp packet.

#! /usr/bin/env python
from scapy.all import *

def packet():
        infile = open('payload2','r')
        vpayload = infile.readlines()
        infile.close
        p = Ether()/IP()
        p.dst = "00:00:00:00:00:02"
        p.src = "00:00:00:00:00:01"
        p[IP].proto = 17  #udp
        p[IP].payload = str(vpayload)
        return p

if __name__ == '__main__':
        p = packet()
        print p.show
        sendp(p, iface = 'h1-eth0')
```

โปรแกรมที่ 3.2: โปรแกรมสร้างแพ็กเกตยูดีพี

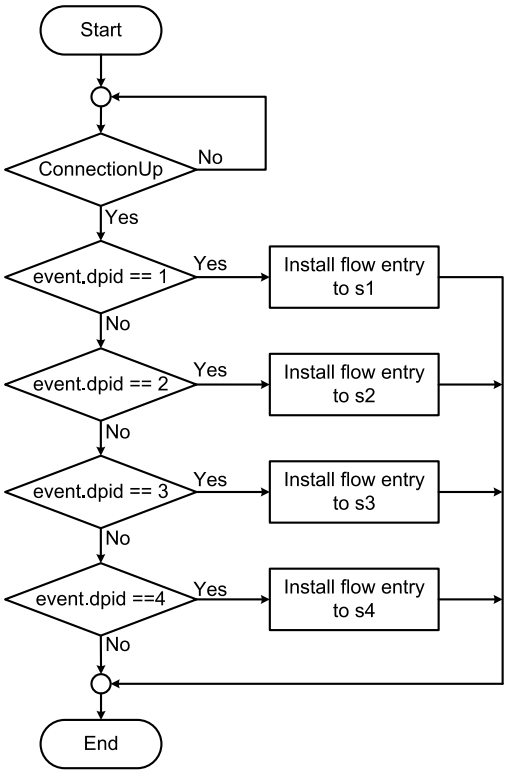
หลังจากที่สร้างโครงข่ายโอเพนโฟลว์ด้วยมินิเน็ตแล้ว h1 และ h2 จะยังไม่สามารถติดต่อกันได้ เนื่องจากตัวควบคุมพอกซ์ยังไม่เริ่มทำงาน และโอเพนวีสวิตช์ไม่มีโฟลว์เอนทรีใด ๆ ในตารางโฟลว์ ทำให้โอเพนวีสวิตช์ต้องทิ้งแพ็กเกตไป ดังนั้นในการทดลองนี้ ผู้วิจัยจึงได้เขียนโปรแกรมเพื่อให้ตัวควบคุมพอกซ์ติดตั้งโฟลว์เอนทรีให้แก่โอเพนวีสวิตช์ทุกตัว โดยโปรแกรมเริ่มทำงานด้วยการลงทะเบียนส่วนโปรแกรม MyComponent (core.registerNew(MyComponent)) เพื่อให้คลาส (class) MyComponent ทำงาน โดยคลาส MyComponent มีฟังก์ชัน (function) การทำงานคือ รอฟังเหตุการณ์ (event) ที่จะเกิดขึ้น (core.openflow.addListeners(self)) เมื่อโอเพนวีสวิตช์เชื่อมต่อกับตัวควบคุมพอกซ์สำเร็จ จะเกิดเหตุการณ์ ConnectionUp ขึ้น ตัวควบคุมพอกซ์จะจัดการกับเหตุการณ์ ConnectionUp (_handle_ConnectionUp) โดยตรวจสอบเหตุการณ์นั้นว่ามาจากโอเพนวีสวิตช์ตัวใด โดยพิจารณาจากไอดีวิถีข้อมูล (datapath id : dpid) เช่น หากเหตุการณ์ที่เกิดขึ้นมี dpid เท่ากับ 1 (event.dpid == 1) แสดงว่าโอเพนวีสวิตช์ 1 (s1) ได้เชื่อมต่อกับตัวควบคุมพอกซ์แล้ว เป็นต้น จากนั้นตัวควบคุมพอกซ์จะติดตั้งโฟลว์เอนทรีให้แก่โอเพนวีสวิตช์โดยใช้คำสั่ง event.connection.send เพื่อส่งรายละเอียดของโฟลว์เอนทรี ซึ่งสามารถกำหนดโฟลว์เอนทรีได้โดยใช้คำสั่ง of.ofp_flow_mod โดยโปรแกรมของตัวควบคุมพอกซ์สามารถแสดงได้ดังโปรแกรมที่ 3.3 และการทำงานของตัวควบคุมพอกซ์สามารถแสดงได้ดังรูปที่ 3.3

```
# proto_mac.py by Parichat Pannwaree.
# This file is for adding flow entries to all switches.
# Also matching by using protocol no., in_port, mac_src and mac_dst.

from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.util import dpid_to_str
from pox.lib.util import str_to_bool
from pox.lib.addresses import IPAddr, EthAddr
import pox.lib.packet as pkt
```

```
11
12 log = core.getLogger()
13
14 class MyComponent (object):
15
16     def __init__ (self):
17         core.openflow.addListeners(self)
18
19     def _handle_ConnectionUp (self, event):
20         print "Switch", event.dpid, "has come up.", dpid_to_str(event.dpid)
21
22         if event.dpid == 1:
23         print "Install flow entry to switch" ,event.dpid
24         event.connection.send( of.ofp_flow_mod( action=of.ofp_action_output(port=2),
25                     match=of.ofp_match(in_port=1, dl_type=0x800, nw_proto=6,
26                     dl_src=EthAddr("00:00:00:00:00:01"),
27                     dl_dst=EthAddr("00:00:00:00:00:02")) ))
28         event.connection.send( of.ofp_flow_mod( action=of.ofp_action_output(port=1),
29                     match=of.ofp_match(in_port=2, dl_type=0x800, nw_proto=6,
30                     dl_src=EthAddr("00:00:00:00:00:02"),
31                     dl_dst=EthAddr("00:00:00:00:00:01")) ))
32         event.connection.send( of.ofp_flow_mod( action=of.ofp_action_output(port=3),
33                     match=of.ofp_match(in_port=1, dl_type=0x800, nw_proto=17,
34                     dl_src=EthAddr("00:00:00:00:00:01"),
35                     dl_dst=EthAddr("00:00:00:00:00:02")) ))
36         event.connection.send( of.ofp_flow_mod( action=of.ofp_action_output(port=1),
37                     match=of.ofp_match(in_port=3, dl_type=0x800, nw_proto=17,
38                     dl_src=EthAddr("00:00:00:00:00:02"),
39                     dl_dst=EthAddr("00:00:00:00:00:01")) ))
40
41         elif event.dpid == 2:
42         print "Install flow entry to switch" ,event.dpid
43         event.connection.send( of.ofp_flow_mod( action=of.ofp_action_output(port=2),
44                     match=of.ofp_match(in_port=1, dl_type=0x800, nw_proto=6,
45                     dl_src=EthAddr("00:00:00:00:00:01"),
46                     dl_dst=EthAddr("00:00:00:00:00:02")) ))
47         event.connection.send( of.ofp_flow_mod( action=of.ofp_action_output(port=1),
48                     match=of.ofp_match(in_port=2, dl_type=0x800, nw_proto=6,
49                     dl_src=EthAddr("00:00:00:00:00:02"),
50                     dl_dst=EthAddr("00:00:00:00:00:01")) ))
51
52         elif event.dpid == 3:
53         print "Install flow entry to switch" ,event.dpid
54         event.connection.send( of.ofp_flow_mod( action=of.ofp_action_output(port=2),
55                     match=of.ofp_match(in_port=1, dl_type=0x800, nw_proto=6,
56                     dl_src=EthAddr("00:00:00:00:00:01"),
57                     dl_dst=EthAddr("00:00:00:00:00:02")) ))
58         event.connection.send( of.ofp_flow_mod( action=of.ofp_action_output(port=1),
59                     match=of.ofp_match(in_port=2, dl_type=0x800, nw_proto=6,
60                     dl_src=EthAddr("00:00:00:00:00:02"),
61                     dl_dst=EthAddr("00:00:00:00:00:01")) ))
62         event.connection.send( of.ofp_flow_mod( action=of.ofp_action_output(port=2),
63                     match=of.ofp_match(in_port=3, dl_type=0x800, nw_proto=17,
64                     dl_src=EthAddr("00:00:00:00:00:01"),
65                     dl_dst=EthAddr("00:00:00:00:00:02")) ))
66         event.connection.send( of.ofp_flow_mod( action=of.ofp_action_output(port=3),
67                     match=of.ofp_match(in_port=2, dl_type=0x800, nw_proto=17,
68                     dl_src=EthAddr("00:00:00:00:00:02"),
```

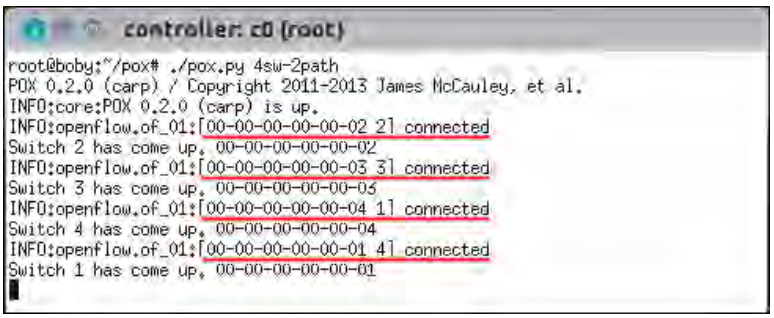
```
                    dl_dst=EthAddr("00:00:00:00:00:01")) ))

        elif event.dpid == 4:
        print "Install flow entry to switch" ,event.dpid
        event.connection.send( of.ofp_flow_mod( action=of.ofp_action_output(port=2),
                    match=of.ofp_match(in_port=1, dl_type=0x800, nw_proto=17,
                    dl_src=EthAddr("00:00:00:00:00:01"),
                    dl_dst=EthAddr("00:00:00:00:00:02")) ))
        event.connection.send( of.ofp_flow_mod( action=of.ofp_action_output(port=1),
                    match=of.ofp_match(in_port=2, dl_type=0x800, nw_proto=17,
                    dl_src=EthAddr("00:00:00:00:00:02"),
                    dl_dst=EthAddr("00:00:00:00:00:01")) ))

def launch ():
    core.registerNew(MyComponent)
```

โปรแกรมที่ 3.3: โปรแกรมของตัวควบคุมพอกซ์ในการทดลองที่ 1

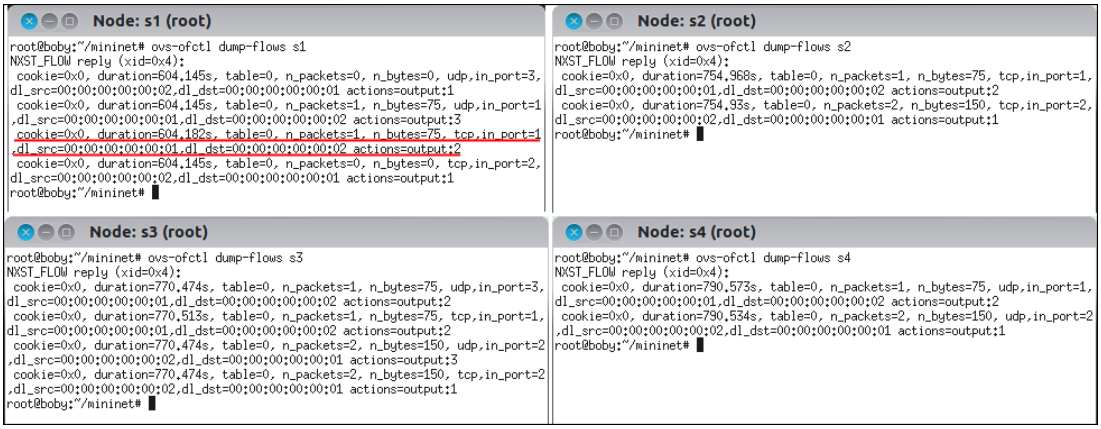


รูปที่ 3.3: ผังงานการทำงานของตัวควบคุมพอกซ์

เมื่อตัวควบคุมพอกซ์เริ่มทำงานตามโปรแกรม พอกซ์จะรอการเชื่อมต่อจากโอเพนวีสวิตช์ทุกตัว ถ้าโอเพนวีสวิตช์เชื่อมต่อกับตัวควบคุมพอกซ์สำเร็จ พอกซ์จะแสดงหมายเลข dpid ของสวิตช์ตัวนั้น ๆ พร้อมทั้งแสดงข้อความว่า connected เพื่อยืนยันว่าได้เชื่อมต่อแล้ว ดังแสดงในรูปที่ 3.4 ซึ่งจะเห็นว่าโอเพนวีสวิตช์ทุกตัวได้เชื่อมต่อกับตัวควบคุมพอกซ์ และได้ถูกติดตั้งโฟลว์เอนทรีแล้ว โดยสามารถตรวจสอบได้ว่า โอเพนวีสวิตช์มีโฟลว์เอนทรีอยู่ในตารางโฟลว์หรือไม่ โดยใช้คำสั่ง ovs-ofctl dump-flows <switch> ดังในรูปที่ 3.5 ซึ่งแสดงให้เห็นว่าโอเพนวีสวิตช์ทุกตัวมีโฟลว์เอนทรีแล้ว



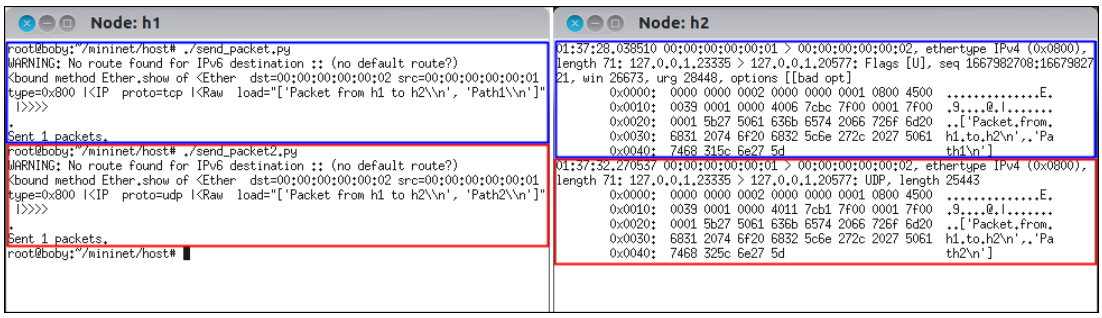
รูปที่ 3.4: ตัวควบคุมพอกซ์แสดงข้อความการเชื่อมต่อกับโอเพนวีสวิตช์



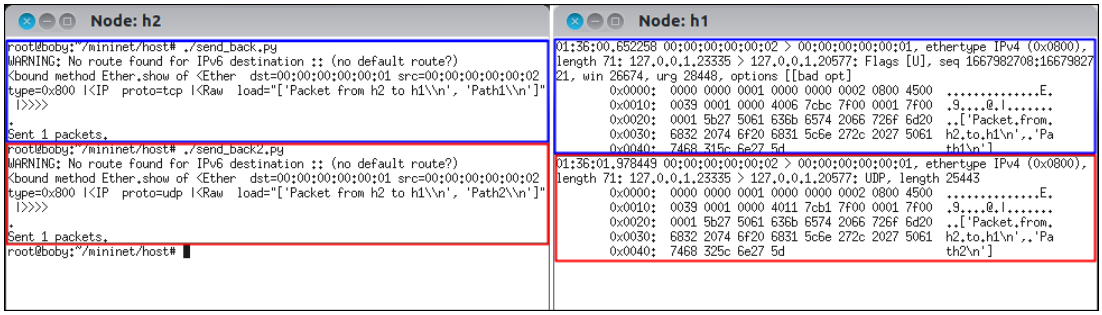
รูปที่ 3.5: โฟลว์เอนทรีของโอเพนวีสวิตช์ในการทดลองที่ 1

จากรูปที่ 3.5 สามารถอธิบายโฟลว์เอนทรีของโอเพนวีสวิตช์ 1 (s1) ที่ถูกขีดเส้นใต้ได้ว่า ตารางโฟลว์ที่ 0 (table=0) ถูกใช้งานมาแล้วเป็นเวลา 604.182 วินาที (duration=604.182s) มีแพ็กเกตผ่าน s1 มาแล้ว 1 แพ็กเกต (n_packets=1) หรือ 75 ไบต์ (n_bytes=75) ถ้าแพ็กเกตเข้ามาที่ช่องทาง 1 (in_port=1) เป็นแพ็กเกตชนิดทีซีพี และส่วนหัวของแพ็กเกตประกอบด้วย ที่อยู่ของต้นทางชั้นเส้นทางเชื่อมโยงข้อมูล (data link layer) เท่ากับ 00:00:00:00:00:01 (dl_scr=00:00:00:00:00:01) และที่อยู่ของปลายทางเท่ากับ 00:00:00:00:00:02 (dl_dst=00:00:00:00:00:02) ให้กระทำกับแพ็กเกตนี้โดยส่งออกไปช่องทาง 2 (actions=output:2) หมายความว่าให้ส่งแพ็กเกตทีซีพี จาก h1 ออกไปยังวิถีที่ 1 นั่นเอง

หลังจากที่โอเพนวีสวิตช์ทุกตัวมีโฟลว์เอนทรีแล้ว ผู้วิจัยได้ทดลองส่งแพ็กเกตทีซีพี และ ยูดีพี จาก h1 ไป h2 และจาก h2 มา h1 ด้วย โดยครั้งแรกให้ h1 เป็นผู้ส่ง และ h2 เป็นผู้รับ ซึ่งที่ h2 จะใช้คำสั่ง tcpdump เพื่อตรวจจับแพ็กเกตที่เข้ามา ซึ่งผลการทดลองเป็นไปตามรูปที่ 3.6 โดยแพ็กเกตทีซีพีจะถูกส่งผ่านวิถีที่ 1 ส่วนแพ็กเกตยูดีพี จะถูกส่งผ่านวิถีที่ 2 จากนั้นทดลองให้ h2 ส่งแพ็กเกตกลับมายัง h1 โดยผลการทดลองแสดงได้ดังรูปที่ 3.7 ซึ่งแสดงให้เห็นว่า h1 สามารถรับแพ็กเกตทั้ง 2 ชนิดจาก h2 ได้



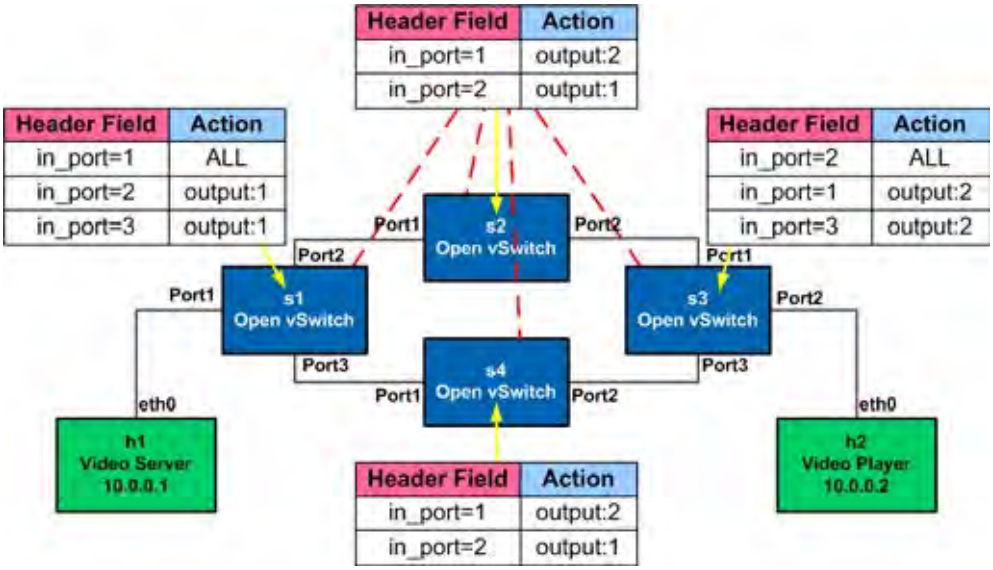
รูปที่ 3.6: h1 ส่งแพ็กเกตทีซีพี และยูดีพี ไปยัง h2



รูปที่ 3.7: h2 ส่งแพ็กเกตกลับมายัง h1

3.2 การทดลองที่ 2: การสตรีมวีดิทัศน์แบบพหุวิถี

การทดลองที่ 2 เป็นการทดลองสตรีมวีดิทัศน์แบบพหุวิถี จาก h1 ไป h2 โดยใช้โครงข่ายโอเพนโฟลว์เดียวกับการทดลองแรก ดังรูปที่ 3.1 การทดลองนี้ h1 และ h2 จะใช้โปรแกรมวีแอลซี (VLC) [24] ในการจำลองเป็นตัวบริการและตัวเล่นวีดิทัศน์ตามลำดับ ซึ่งวีแอลซีเป็นโปรแกรมเล่นสื่อประสม (multimedia player) ที่รองรับสื่อได้หลายรูปแบบ และสามารถสตรีมและรับวีดิทัศน์ได้ การทำงานของตัวควบคุมพอกซ์จะมีลักษณะเดียวกับการทดลองที่ 1 คือ จะติดตั้งโฟลว์เอนทรีให้กับโอเพนวีสวิตช์หลังจากที่โอเพนวีสวิตช์ได้เชื่อมต่อกับพอกซ์สำเร็จ ดังผังงานในรูปที่ 3.3 แต่ส่วนที่ต่างจากการทดลองที่ 1 คือ รายละเอียดของโฟลว์เอนทรี ดังในรูปที่ 3.8 ซึ่งโปรแกรมของตัวควบคุมพอกซ์ในการทดลองที่ 2 สามารถแสดงได้ดังโปรแกรมที่ 3.4



รูปที่ 3.8: ตารางโฟลว์ของโอเพนวี สวิตช์ ในการทดลองที่ 2

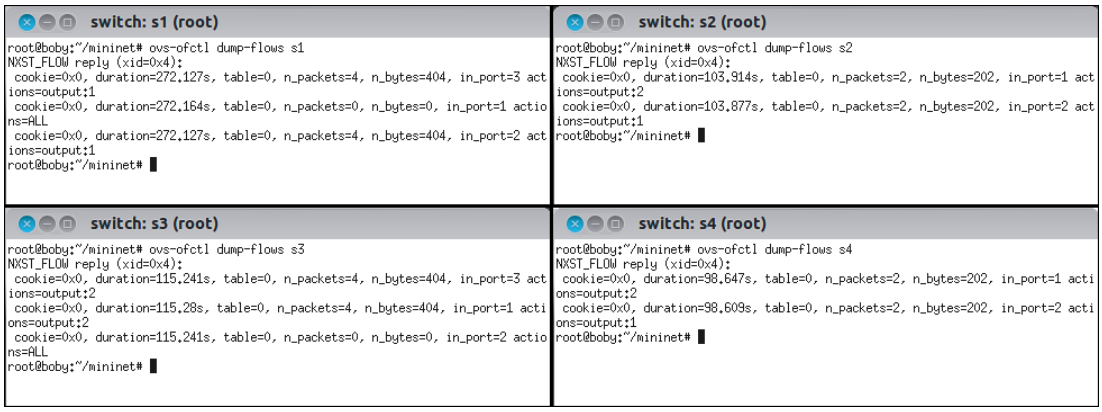
```
1  # 4sw-2path.py by Parichat Pannwaree.
2  # This file is for adding flow entries to all switches. (2 paths)
3  # Also matching by using in_port.
4
5  from pox.core import core
6  import pox.openflow.libopenflow_01 as of
7  from pox.lib.util import dpid_to_str
8  from pox.lib.util import str_to_bool
9  from pox.lib.addresses import IPAddr, EthAddr
10 from pox.lib.packet.tcp import tcp
11 import time
12
13
14 log = core.getLogger()
15
16 class MyComponent (object):
17
18     def __init__ (self):
19         core.openflow.addListeners(self)
20
21     def _handle_ConnectionUp (self, event):
22         print "Switch", event.dpid, "has come up.", dpid_to_str(event.dpid)
23
24         if event.dpid == 1:
25             event.connection.send( of.ofp_flow_mod(
26                         action=of.ofp_action_output(port=of.OFPP_ALL),
27                         match=of.ofp_match(in_port=1)))
28             event.connection.send( of.ofp_flow_mod(
29                         action=of.ofp_action_output(port=1),
30                         match=of.ofp_match(in_port=2)))
31             event.connection.send( of.ofp_flow_mod(
32                         action=of.ofp_action_output(port=1),
33                         match=of.ofp_match(in_port=3)))
34
35         elif event.dpid == 2:
```

```
36              event.connection.send( of.ofp_flow_mod(
37                          action=of.ofp_action_output(port=2),
38                          match=of.ofp_match(in_port=1)))
39              event.connection.send( of.ofp_flow_mod(
40                          action=of.ofp_action_output(port=1),
41                          match=of.ofp_match(in_port=2)))
42
43          elif event.dpid == 3:
44              event.connection.send( of.ofp_flow_mod(
45                          action=of.ofp_action_output(port=2),
46                          match=of.ofp_match(in_port=1)))
47              event.connection.send( of.ofp_flow_mod(
48                          action=of.ofp_action_output(port=of.OFPP_ALL),
49                          match=of.ofp_match(in_port=2)))
50              event.connection.send( of.ofp_flow_mod(
51                          action=of.ofp_action_output(port=2),
52                          match=of.ofp_match(in_port=3)))
53
54          elif event.dpid == 4:
55              event.connection.send( of.ofp_flow_mod(
56                          action=of.ofp_action_output(port=2),
57                          match=of.ofp_match(in_port=1)))
58              event.connection.send( of.ofp_flow_mod(
59                          action=of.ofp_action_output(port=1),
60                          match=of.ofp_match(in_port=2)))
61
62  def launch ():
63      core.registerNew(MyComponent)
```

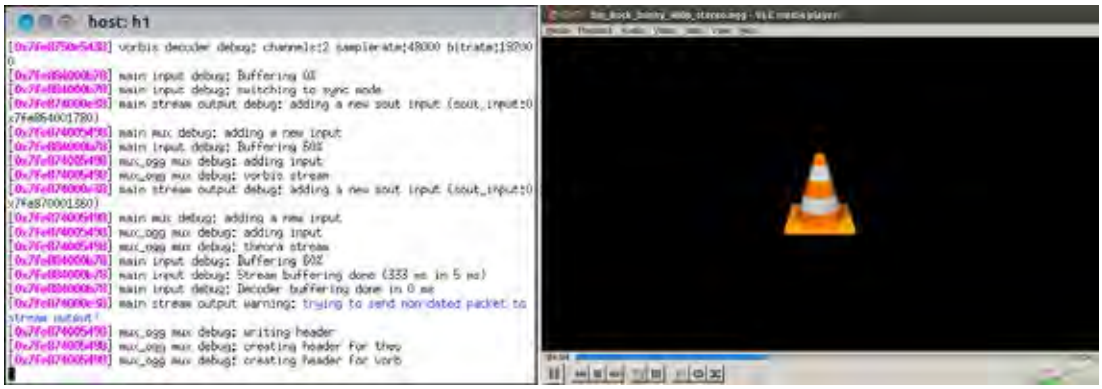
โปรแกรมที่ 3.4: โปรแกรมของตัวควบคุมพอกซ์ในการทดลองที่ 2

จากโฟลว์เอนทรีภายในโอเพนวีสวิตช์แต่ละตัว ดังในรูปที่ 3.9 สามารถอธิบายได้ดังนี้ ในโอเพนวีสวิตช์ 1 (s1) เมื่อแพ็กเกตเข้ามายังช่องทาง 1 (in_port=1) ให้ส่งแพ็กเกตออกไปยังทุกช่องทาง (actions=ALL) ยกเว้นช่องทางที่แพ็กเกตเข้ามา และเมื่อแพ็กเกตเข้ามายังช่องทาง 2 และ 3 (in_port=2, in_port=3) ให้ส่งแพ็กเกตออกไปยังช่องทาง 1 (actions=output:1) หมายความว่า หาก s1 ได้รับแพ็กเกตวีดิทัศน์จาก h1 ให้ s1 ส่งแพ็กเกตนั้นออกไปทั้งวิถีที่ 1 และ 2 และแพ็กเกตเข้ามาที่ช่องทาง 2 และ 3 ให้ส่งแพ็กเกตนั้นกลับไปยัง h1 ในโอเพนวีสวิตช์ 2 และ 4 (s2 และ s4) จะมีโฟลว์เอนทรีเหมือนกัน คือ ให้ส่งแพ็กเกตที่เข้ามายังช่องทาง 1 (in_port=1) ออกไปช่องทาง 2 (actions=output:2) และส่งแพ็กเกตที่เข้ามายังช่องทาง 2 (in_port=2) ออกไปช่องทาง 1 (actions=output:1) โดย s2 จะอยู่บนวิถีที่ 1 ส่วน s4 อยู่บนวิถีที่ 2 และในโอเพนวีสวิตช์ 3 (s3) จะส่งแพ็กเกตที่ได้รับจากทั้ง s2 (in_port=1) และ s4 (in_port=3) หรือจากวิถีที่ 1 และ 2 ตามลำดับ ออกไปยังช่องทาง 2 (actions=output:2) เพื่อส่งต่อไปยัง h2 และส่งแพ็กเกตที่เข้ามายังช่องทาง 2 (in_port=2) ออกไปทั้งช่องทาง 1 และ 3 (actions=ALL)

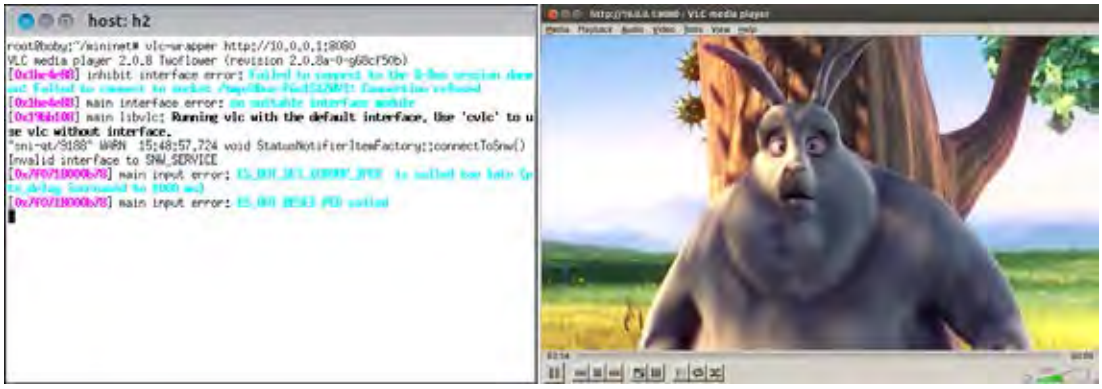
จากนั้นทดลองสตรีมวีดิทัศน์แบบพหุวิถีจาก h1 ไป h2 โดยใช้โปรแกรมวีแอลซี ซึ่งการทดลองนี้ได้ใช้วีดิทัศน์เรื่อง Big Buck Bunny [28] ที่มีความละเอียด 854x480 และความยาวเท่ากับ 9.56 นาที ซึ่งผลการทดลองคือ h1 สามารถสตรีมวีดิทัศน์ได้ ดังรูปที่ 3.10 และ h2 สามารถรับและเล่นวีดิทัศน์ได้อย่างต่อเนื่อง ดังรูปที่ 3.11



รูปที่ 3.9: โฟลว์เอนทรีของโอเพนวีสวิตช์ในการทดลองที่ 2

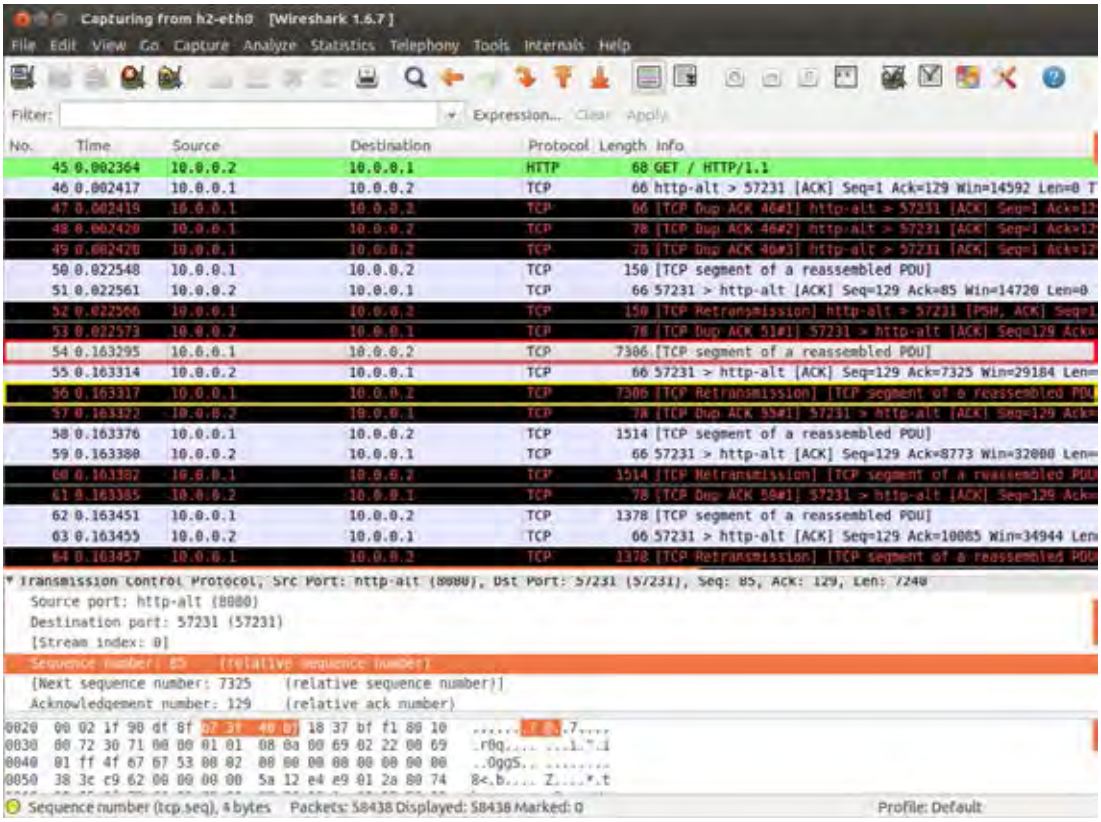


รูปที่ 3.10: h1 สตรีมวีดิทัศน์ไปให้ h2



รูปที่ 3.11: h2 รับและเล่นวีดิทัศน์อย่างต่อเนื่องจาก h1

เมื่อใช้โปรแกรมไวร์ชาร์ก (Wireshark) ตรวจจับแพ็กเกตที่เข้ามายัง h2 พบว่า h2 ได้รับแพ็กเกตที่เหมือนกัน 2 แพ็กเกต ดังในรูปที่ 3.12 แสดงว่า h2 ได้รับแพ็กเกตจากทั้งวิถีที่ 1 และวิถีที่ 2 ซึ่งโปรแกรมวีแอลซีจะจัดการกับแพ็กเกตทั้งสอง โดยทิ้งแพ็กเกตที่ซ้ำไป และนำเพียงแพ็กเกตแรกที่ได้รับมาประมวลผลเพื่อเล่นวีดิทัศน์

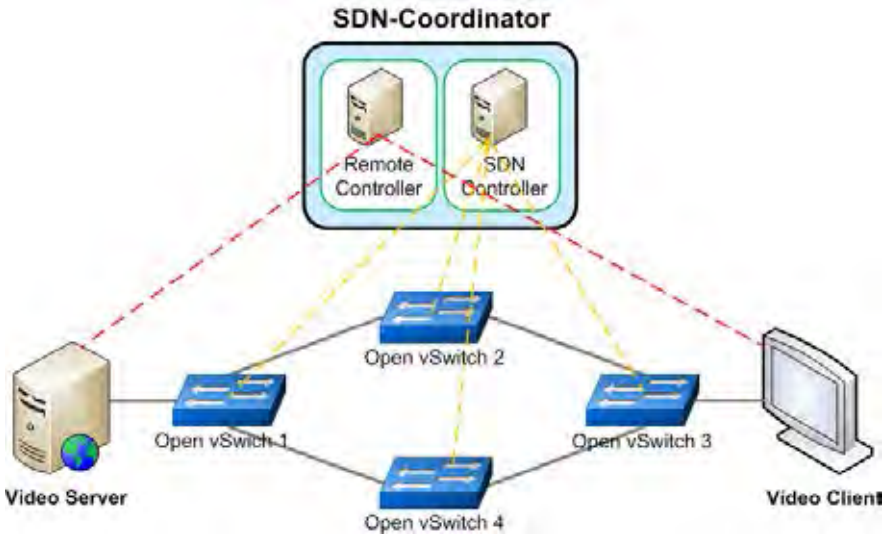


รูปที่ 3.12: แพ็กเกตที่ h2 ได้รับ

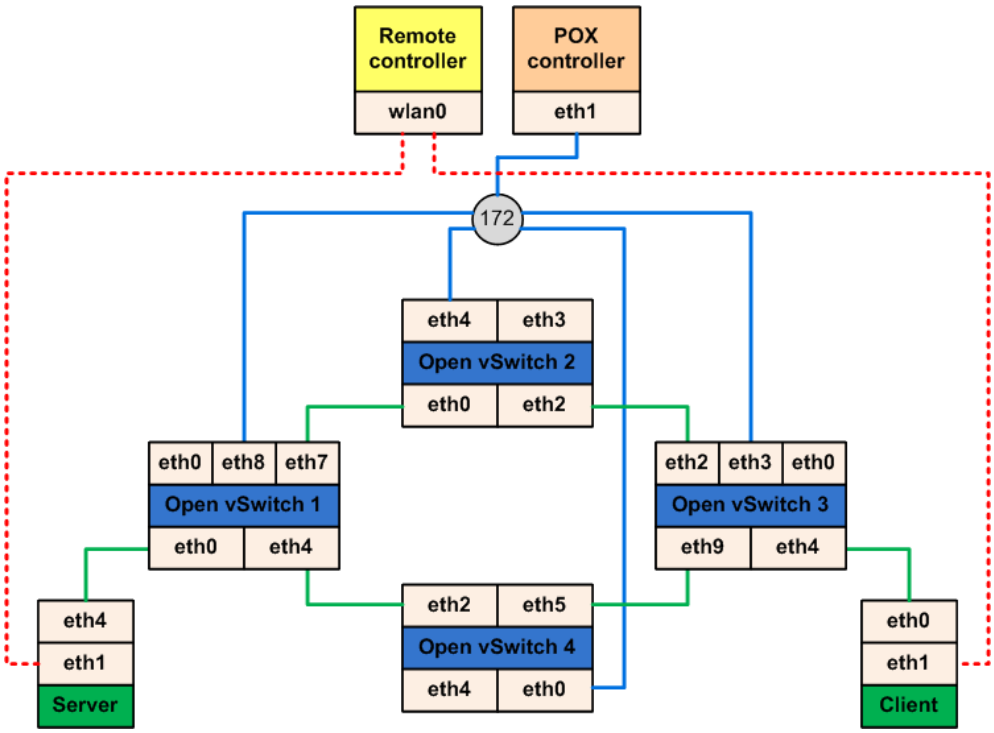
3.3 การเปรียบเทียบสมรรถนะของการสตรีมวีดิทัศน์บนโครงข่ายโอเพนโฟลว์เสมือนและระบบทดสอบโอเพนโฟลว์จริง

จากการทดลองทั้ง 2 การทดลองที่ได้เสนอไปข้างต้น เป็นการแสดงให้เห็นว่าโครงข่ายโอเพนโฟลว์เสมือนที่ถูกจำลองด้วยโปรแกรมมินิเน็ตสามารถทำงานได้ ทั้งการรับ-ส่งแพ็กเกต และการสตรีมวีดิทัศน์ ในหัวข้อนี้ เป็นการศึกษาสมรรถนะของการสตรีมวีดิทัศน์บนโครงข่ายโอเพนโฟลว์เสมือนเทียบกับโครงข่ายโอเพนโฟลว์จริง โดยผู้วิจัยได้ใช้ระบบทดสอบโอเพนโฟลว์ที่มีทอพอโลยีคล้ายกับโครงข่ายโอเพนโฟลว์ที่ใช้ในการทดลอง 1 และ 2 ซึ่งในระบบทดสอบประกอบไปด้วย ตัวบริการวีดิทัศน์ ตัวเล่นวีดิทัศน์ โอเพนวีสวิตช์จำนวน 4 ตัว และตัวประสานงานเอสดีเอ็น (SDN coordinator) โดยในตัวประสานงานเอสดีเอ็นจะประกอบด้วย ตัวควบคุมเอสดีเอ็น (งานวิจัยชิ้นนี้เลือกใช้พอกซ์เป็นตัวควบคุมเอสดีเอ็นในระบบทดสอบโอเพนโฟลว์) และตัวควบคุมระยะไกล (remote controller) ที่มีหน้าที่เฝ้าสังเกต (monitor) และสั่งการให้ตัวบริการวีดิทัศน์ และตัวเล่นวีดิทัศน์ในระบบทดสอบทำงานจากระยะไกล จะเห็นได้ว่า โครงข่ายโอเพนโฟลว์เสมือนและระบบ

ทดสอบโอเพนโฟลว์มีส่วนที่แตกต่างกันเพียงอย่างเดียวคือ ระบบทดสอบโอเพนโฟลว์มีตัวควบคุมระยะไกลเพิ่มเข้ามา ซึ่งไม่ได้ส่งผลกระทบหรือขัดขวางการทำงานของอุปกรณ์อื่นในระบบทดสอบ โดยทอพอโลยีของระบบทดสอบโอเพนโฟลว์สามารถแสดงได้ดังรูปที่ 3.13 และแผนภาพเชิงตรรกะ (logical diagram) ของระบบทดสอบสามารถแสดงได้ดังรูปที่ 3.14



รูปที่ 3.13: ทอพอโลยีของระบบทดสอบโอเพนโฟลว์



รูปที่ 3.14: แผนภาพเชิงตรรกะของระบบทดสอบโอเพนโฟลว์

ระบบทดสอบโอเพนโฟลว์ที่ใช้ในงานวิจัยนี้ ได้ถูกปรับแต่งเพิ่มมาจาก [29] ที่มีสวิตช์โอเพนโฟลว์เพียง 3 ตัว แต่ในงานวิจัยนี้ได้เพิ่มสวิตช์เข้าไปอีก 1 ตัว เพื่อให้ได้โครงข่ายโอเพนโฟลว์ที่มีวิถี 2 วิถีที่สมมาตรกันดังรูปที่ 3.13 นอกจากนี้ยังปรับสวิตช์ให้เป็นโอเพนวีสวิตช์ทั้งหมด ดังนั้น ระบบทดสอบโอเพนโฟลว์ในงานวิจัยนี้จึงถูกสร้างขึ้นด้วยคอมพิวเตอร์ที่มีระบบปฏิบัติการลินุกซ์ (Linux) Ubuntu 12.04 ทั้งหมด 8 เครื่อง โดยรูปที่ 3.15 แสดงระบบทดสอบโอเพนโฟลว์ในห้องปฏิบัติการ

รูปที่ 3.15: ระบบทดสอบโอเพนโฟลว์ในห้องปฏิบัติการ

คุณลักษณะ (specification) ของเครื่องคอมพิวเตอร์ที่ใช้ในระบบทดสอบสามารถแสดงได้ดังตารางที่ 3.1

ตารางที่ 3.1: คุณลักษณะของคอมพิวเตอร์ที่ใช้ในระบบทดสอบ

Device	Software	Processor
Video server	VLC media player 2.0.9	Core 2 Quad 2.40 GHz
Video client	VLC media player 2.0.9	Core 2 Quad 2.40 GHz
Open vSwitch 1	Open vSwitch 1.9.3	Pentium G630 2.70 GHz
Open vSwitch 2	Open vSwitch 1.9.3	Pentium G630 2.70 GHz
Open vSwitch 3	Open vSwitch 1.9.3	Core 2 Quad 2.40 GHz
Open vSwitch 4	Open vSwitch 1.9.3	Pentium G640 2.80 GHz
POX controller	POX 0.3.0	Core 2 Quad 2.40 GHz
Remote controller	-	Core i5-3475s 2.90 GHz

หลังจากที่สร้างระบบทดสอบโอเพนโฟลว์ที่มีวิถี 2 วิถีแล้ว ผู้วิจัยได้วัดแบนด์วิดท์และความหน่วงระหว่างตัวบริการวีดิทัศน์ และตัวเล่นวีดิทัศน์บนวิถีทั้ง 2 วิถี โดยใช้โปรแกรมไอเพิฟ (iperf) และปิง (ping) ตามลำดับ ซึ่งบนวิถีบน (ผ่านโอเพนวีสวิตช์ 1, 2 และ 3) มีแบนด์วิดท์ 6.5 เมกะบิตต่อวินาที และความหน่วง (ครึ่งหนึ่งของความหน่วงที่ได้จากการปิง) เท่ากับ 1.5 มิลลิวินาที ส่วนวิถีล่าง (ผ่านโอเพนวีสวิตช์ 1, 4 และ 3) วัดแบนด์วิดท์ได้ 2.5 เมกะบิตต่อวินาที และความหน่วงเท่ากับ 2 มิลลิวินาที จะเห็นได้ว่าวิถีบนมีคุณภาพดีกว่าวิถีล่าง คือ สามารถรองรับข้อมูลได้มากกว่า และส่งข้อมูลได้เร็วกว่าวิถีล่าง

เพื่อให้การเปรียบเทียบสมรรถนะระหว่างโครงข่ายเสมือนกับระบบทดสอบจริงมีความยุติธรรมมากที่สุด ผู้วิจัยได้ทำการตั้งค่าแบนด์วิดท์และความหน่วงของเส้นเชื่อมโยง (link) แต่ละเส้นของโครงข่ายโอเพนโฟลว์เสมือน ให้มีค่าแบนด์วิดท์และความหน่วงระหว่างตัวบริการวีดิทัศน์ และตัวเล่นวีดิทัศน์บนวิถีทั้ง 2 วิถี ใกล้เคียงกับค่าที่ได้จากระบบทดสอบโอเพนโฟลว์มากที่สุด โดยเส้นเชื่อมโยงทุกเส้นในรูปที่ 3.1 ถูกตั้งให้มีแบนด์วิดท์ 6.5 เมกะบิตต่อวินาที และความหน่วงเท่ากับ 0.35 มิลลิวินาที ยกเว้นเส้นเชื่อมโยงระหว่าง s1-s4 และ s4-s3 ที่ถูกตั้งค่าให้มีแบนด์วิดท์ 2.3 เมกะบิตต่อวินาที และความหน่วงเท่ากับ 0.61 มิลลิวินาที ทั้งนี้ เครื่องคอมพิวเตอร์ที่ใช้สำหรับจำลองโครงข่ายโอเพนโฟลว์เสมือนด้วยโปรแกรมมินิเน็ต เป็นคอมพิวเตอร์ระบบปฏิบัติการลินุกซ์ Ubuntu 12.04 และมีหน่วยประมวลผลคือ Core i5-3475s 2.90 GHz

ในการเปรียบเทียบสมรรถนะในการสตรีมวีดิทัศน์ของโครงข่ายทั้ง 2 โครงข่าย งานวิจัยนี้ได้เลือกใช้วีดิทัศน์เรื่อง Big Buck Bunny ที่มีความละเอียด 854x480 โคเดก (codec) แบบ H.264 และความยาวเท่ากับ 9.56 นาที เช่นเดียวกับการทดลองที่ 2 ข้างต้นมาใช้ทดสอบ เนื่องจากวิถีล่างของโครงข่ายโอเพนโฟลว์มีแบนด์วิดท์ 2.5 เมกะบิตต่อวินาที ดังนั้นในการทดลองนี้จึงเลือกสตรีมวีดิทัศน์ที่มีอัตราบิตการเข้ารหัส (encoding bit rate) เท่ากับ 2.5 เมกะบิตต่อวินาที และสตรีมวีดิทัศน์โดยใช้โพรโตคอลทีซีพี และยูดีพี ตามลำดับ ข้อมูลของวีดิทัศน์และรายละเอียดการสตรีมวีดิทัศน์สามารถแสดงได้ดังตารางที่ 3.2

ตารางที่ 3.2: ข้อมูลของวีดิทัศน์และรายละเอียดการสตรีมวีดิทัศน์บนโครงข่ายโอเพนโฟลว์

Video	Big Buck Bunny
Resolution	854x480
Codec	H.264
Duration	9.56 minutes
Size	249.2 MB
Encoding bit rate	2.5 Mbits/s
Protocol	TCP, UDP

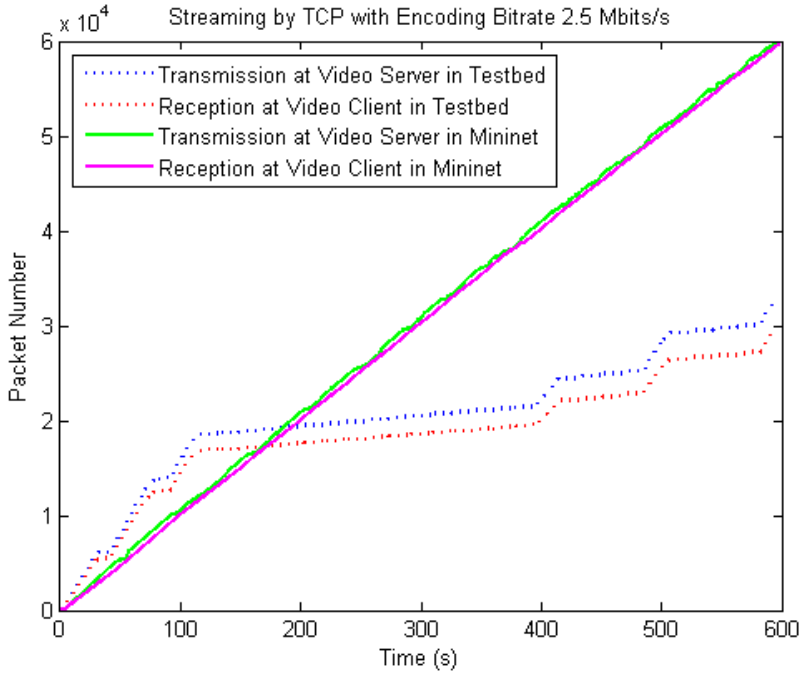
การทดลองนี้ใช้โปรแกรมไวร์ชาร์กจับแพ็กเกตที่ช่องทางอีเทอร์เน็ต (ethernet) ขาออกของตัวบริการวีดิทัศน์ และช่องทางอีเทอร์เน็ตขาเข้าของตัวเล่นวีดิทัศน์ เพื่อหาแพ็กเกตที่สูญเสียไประหว่างการสตรีมวีดิทัศน์

3.3.1 การสตรีมวีดิทัศน์ผ่านวิถีบน

วีดิทัศน์ถูกสตรีมผ่านวิถีบน โดยเริ่มจากตัวบริการวีดิทัศน์ผ่านโอเพนวีสวิตช์ 1, 2 และ 3 ของระบบทดสอบ หรือ s1, s2 และ s3 ของโครงข่ายเสมือนจนมาถึงตัวเล่นวีดิทัศน์ โดยตัวบริการวีดิทัศน์ใช้โปรแกรมวีแอลซีสตรีมวีดิทัศน์ด้วยโพรโตคอลทีซีพี และยูดีพี และตัวเล่นวีดิทัศน์ใช้โปรแกรมวีแอลซีเพื่อรับและเล่นวีดิทัศน์ที่ถูกสตรีมมาเช่นกัน ซึ่งจากการทดลองพบว่า ไม่มีแพ็กเกตสูญเสียเกิดขึ้นที่ตัวเล่นวีดิทัศน์ เมื่อสตรีมวีดิทัศน์โดยใช้โพรโตคอลทีซีพีและยูดีพี ทั้งบนโครงข่ายโอเพนโฟลว์เสมือนและระบบทดสอบโอเพนโฟลว์ เนื่องจากวิถีบนมีแบนด์วิดท์มากเพียงพอจึงสามารถรองรับแพ็กเกตวีดิทัศน์ได้ทั้งหมด

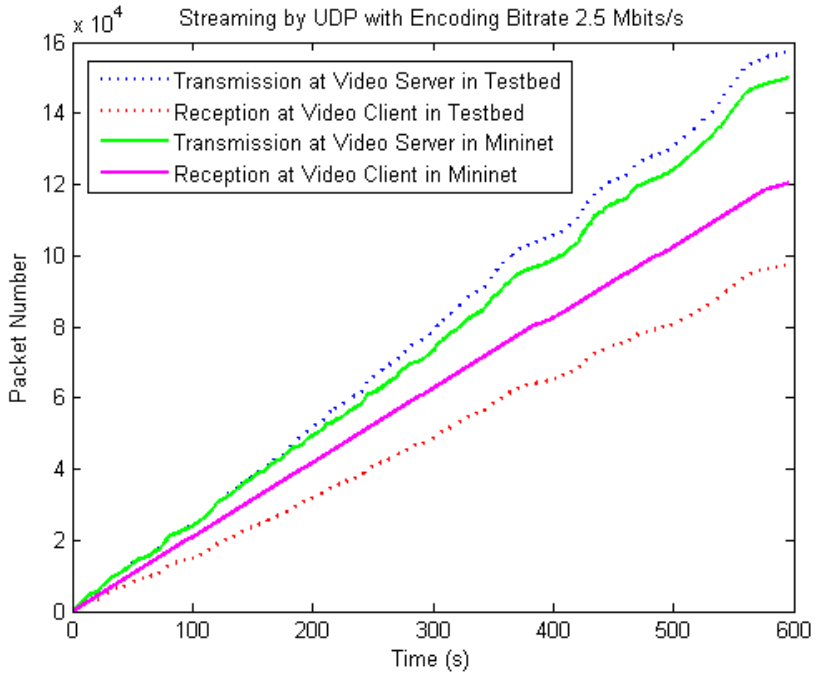
3.3.2 การสตรีมวีดิทัศน์ผ่านวิถีล่าง

เมื่อสตรีมวีดิทัศน์ผ่านวิถีล่าง จากตัวบริการวีดิทัศน์ผ่านโอเพนวีสวิตช์ 1, 4 และ 3 ของระบบทดสอบ หรือ s1, s4 และ s3 ของโครงข่ายเสมือนจนถึงตัวเล่นวีดิทัศน์ พบว่าการสตรีมวีดิทัศน์โดยใช้โพรโตคอลทีซีพีให้ผลต่างจากโพรโตคอลยูดีพี ซึ่งเมื่อใช้โพรโตคอลทีซีพีจะไม่มีแพ็กเกตสูญเสียเกิดขึ้นทั้งในโครงข่ายโอเพนโฟลว์เสมือนและระบบทดสอบจริง แต่เมื่อนำแพ็กเกตที่จับได้จากโปรแกรมไวร์ชาร์กของทั้งตัวบริการวีดิทัศน์และตัวเล่นวีดิทัศน์ มาเขียนเป็นกราฟเทียบกับเวลา ดังแสดงในรูปที่ 3.16 สามารถอธิบายได้ว่า บนโครงข่ายเสมือนที่สร้างจากโปรแกรมมินิเน็ตไม่มีแพ็กเกตสูญหายระหว่างทาง เนื่องจากเส้นกราฟที่ได้เป็นเส้นตรงที่มีความต่อเนื่อง แต่บนระบบทดสอบจะเห็นว่าเส้นกราฟไม่เป็นเส้นตรงต่อเนื่อง กราฟมีลักษณะคล้ายขั้นบันได เพราะมีแพ็กเกตสูญหายไป แต่เนื่องจากใช้โพรโตคอลทีซีพี เมื่อเกิดแพ็กเกตสูญหาย ตัวบริการวีดิทัศน์จึงส่งแพ็กเกตที่สูญหายนั้นไปให้ตัวเล่นวีดิทัศน์ใหม่อีกครั้ง (retransmit) ตัวเล่นวีดิทัศน์จึงได้รับแพ็กเกตครบ



รูปที่ 3.16: กราฟแสดงจำนวนแพ็กเกตเทียบกับเวลาเมื่อสตรีมวีดิทัศน์โดยใช้โพรโตคอลทีซีพี

เมื่อสตรีมวีดิทัศน์โดยใช้โพรโตคอลยูดีพีซึ่งเป็นโพรโตคอลที่ไม่มีความสามารถในการส่งแพ็กเกตที่หายไปใหม่ พบว่า มีการสูญเสียแพ็กเกตเกิดขึ้นที่ตัวเล่นวีดิทัศน์ของทั้งโครงข่ายโอเพนโฟลว์เสมือนและระบบทดสอบโอเพนโฟลว์ โดยโครงข่ายเสมือนเกิดแพ็กเกตสูญเสีย 19% ส่วนระบบทดสอบเกิดแพ็กเกตสูญเสีย 38% ตามลำดับ โดยรูปที่ 3.17 แสดงให้เห็นว่า ระบบทดสอบมีแพ็กเกตสูญเสียเกิดขึ้นมากกว่าโครงข่ายเสมือน การสูญเสียแพ็กเกตบนวิถีล่างในโครงข่ายโอเพนโฟลว์เสมือนและระบบทดสอบเกิดจากความคับคั่ง ทำให้วิถีล่างไม่สามารถรองรับแพ็กเกตวีดิทัศน์ทั้งหมดได้



รูปที่ 3.17: กราฟแสดงจำนวนแพ็กเกตเทียบกับเวลาเมื่อสตรีมวีดิทัศน์โดยใช้โพรโตคอลยูดีพี

จากการเปรียบเทียบสมรรถนะระหว่างการสตรีมวีดิทัศน์บนโครงข่ายโอเพนโฟลว์เสมือนกับระบบทดสอบโอเพนโฟลว์จริงนั้น แม้ว่าโครงข่ายทั้ง 2 โครงข่ายจะมีแบนด์วิดท์และความหน่วงระหว่างตัวบริการวีดิทัศน์และตัวเล่นวีดิทัศน์เท่ากัน แต่ผลการทดลองที่ได้มีความแตกต่างกัน เนื่องจากในสภาพแวดล้อมจริงหรือในระบบทดสอบ มักจะมีตัวแปรหรือพารามิเตอร์ (parameter) ที่ไม่สามารถควบคุมได้ เช่น สายเคเบิล (cable) ที่ใช้เชื่อมต่อคอมพิวเตอร์เข้าด้วยกัน อุปกรณ์ USB-to-Ethernet ที่ใช้เป็นช่องทางอีเทอร์เน็ตของเครื่องคอมพิวเตอร์ หรือแม้กระทั่งหน่วยประมวลผลของคอมพิวเตอร์ จึงเป็นผลให้สมรรถนะของระบบทดสอบจริงต่ำกว่าโครงข่ายเสมือนที่ถูกจำลองขึ้นอยู่ภายในคอมพิวเตอร์เพียงเครื่องเดียว

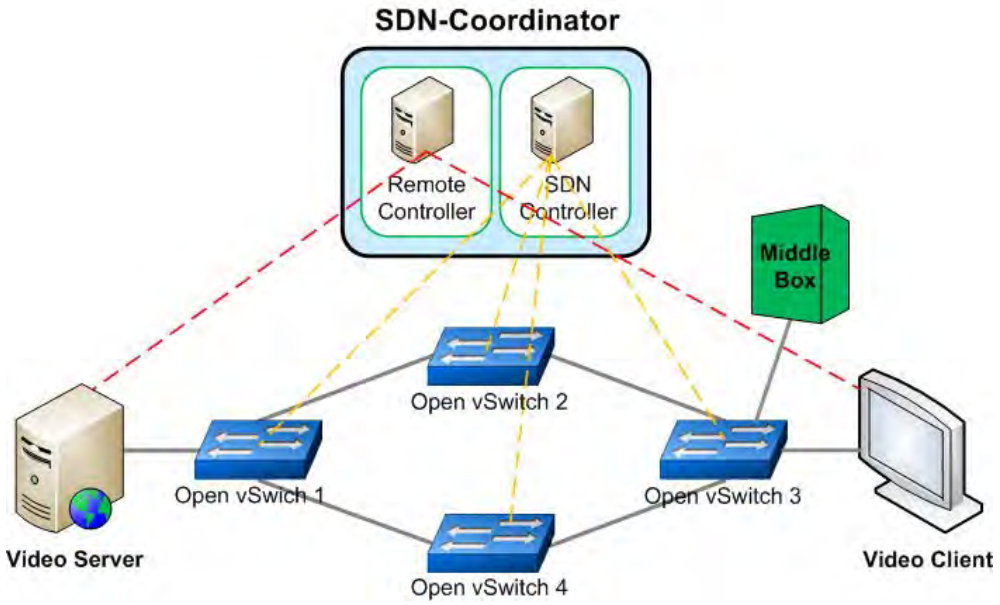
บทที่ 4

การสตรีมกลุ่มก้อนวีดิทัศน์แบบพหุวิถี บนระบบทดสอบโอเพนโฟลว์ในห้องปฏิบัติการ

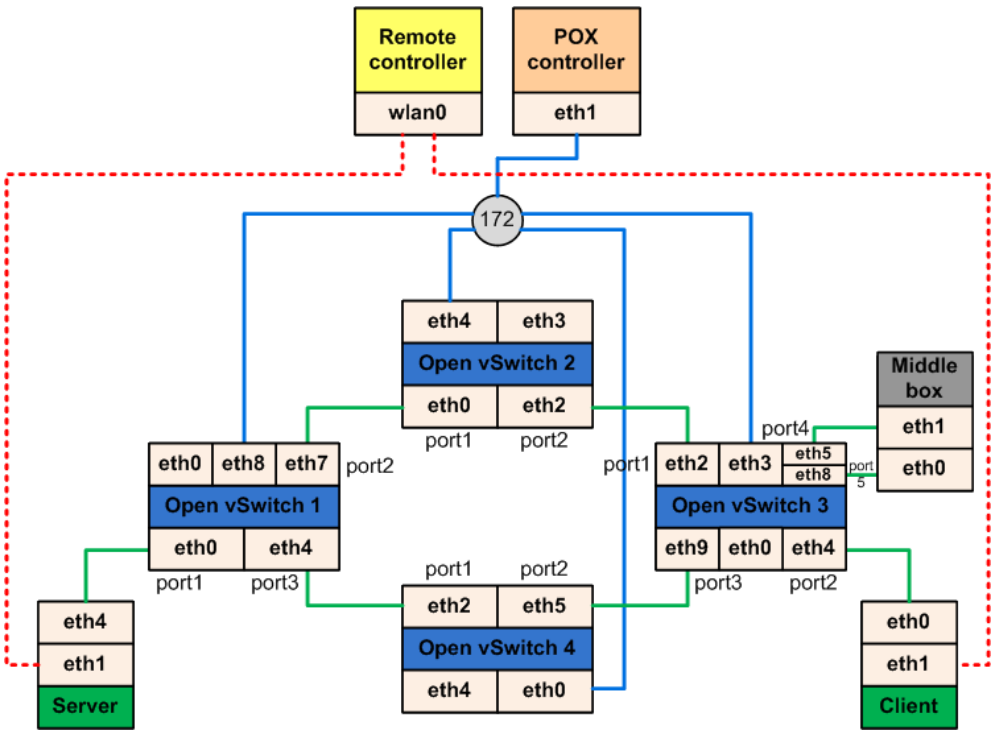
ในบทที่ผ่านมา ได้กล่าวถึงการทดลองเบื้องต้นบนโครงข่ายโอเพนโฟลว์เสมือนที่ถูกจำลองด้วยโปรแกรมมินิเน็ต และทำการเปรียบเทียบการสตรีมวีดิทัศน์บนโครงข่ายโอเพนโฟลว์เสมือนและระบบทดสอบโอเพนโฟลว์จริงในห้องปฏิบัติการ เพื่อแสดงให้เห็นถึงความแตกต่างทางด้านสมรรถนะของโครงข่ายทั้ง 2 โครงข่ายที่มีแบนด์วิดท์และความหน่วงระหว่างตัวบริการวีดิทัศน์และตัวเล่นวีดิทัศน์เท่ากัน ซึ่งผลลัพธ์ที่ได้จากการสตรีมวีดิทัศน์ผ่านวิถี 2 วิถีของโครงข่ายเสมือนและระบบทดสอบจริงมีแนวโน้มไปในทิศทางเดียวกัน แต่ผลการทดลองที่ได้จากระบบทดสอบจริงนั้นให้ผลที่แย่กว่าโครงข่ายเสมือน เนื่องจากถูกจำกัดด้วยสภาพแวดล้อมจริง ในบทนี้เป็นการทดลองบนระบบทดสอบโอเพนโฟลว์จริง โดยพิจารณาการสตรีมกลุ่มก้อนวีดิทัศน์แบบพหุวิถี ที่มีการควบคุมทิศทางของทราฟฟิกวีดิทัศน์ด้วยตัวควบคุมเอสดีเอ็น และรวมแพ็กเกตวีดิทัศน์จากวิถี 2 วิถีด้วยมิดเดิลบอกซ์ (middlebox) ก่อนส่งวีดิทัศน์ต่อไปยังตัวเล่นวีดิทัศน์

4.1 โครงสร้างของระบบทดสอบโอเพนโฟลว์ในห้องปฏิบัติการ

โครงสร้างและแผนภาพเชิงตรรกะของระบบทดสอบโอเพนโฟลว์ขนาดเล็กในห้องปฏิบัติการ ที่ใช้ในการทดลองการสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถี สามารถแสดงได้ดังรูปที่ 4.1 และ 4.2 ตามลำดับ



รูปที่ 4.1: โครงสร้างของระบบทดสอบโอเพนโฟลว์



รูปที่ 4.2: แผนภาพเชิงตรรกะของระบบทดสอบโอเพนโฟลว์

ในระบบทดสอบโอเพนโฟลว์ที่ใช้ทดลองการสตรีมกลุ่มก้อนวีดิทัศน์แบบพหุวิถี ประกอบด้วยอุปกรณ์ต่าง ๆ ได้แก่ ตัวเล่นวีดิทัศน์ ตัวรับวีดิทัศน์ ตัวประสานงานเอสดีเอ็น โอเพนวีสวิตช์ 4 ตัว และมิดเดิลบอกซ์ ซึ่งจะเห็นได้ว่าระบบทดสอบที่ใช้ทดลองในบทนี้ เป็นระบบทดสอบเดียวกันกับที่ใช้ในการทดลองในบทที่ผ่านมา แต่มีความแตกต่างกันคือ ในบทนี้มีมิดเดิลบอกซ์เพิ่มเข้ามาเพื่อใช้รวมแพ็กเกตวีดิทัศน์ที่มาจากวิถี 2 วิถี แล้วเรียงลำดับให้ถูกต้องก่อนส่งไปยังตัวเล่นวีดิทัศน์ โดยรายละเอียดการทำงานของอุปกรณ์แต่ละตัวมีดังต่อไปนี้

4.1.1 ตัวบริการวีดิทัศน์

ใช้โปรแกรมวีแอลซีจำลองเครื่องคอมพิวเตอร์ให้เป็นตัวบริการวีดิทัศน์ เพื่อให้สตรีมวีดิทัศน์เป็นกลุ่มก้อนโดยใช้โพรโตคอลอาร์ทีพี (RTP) ไปยังปลายทาง โดยตัวบริการวีดิทัศน์จะถูกเชื่อมต่อเข้ากับโอเพนวีสวิตช์ 1 และตัวควบคุมระยะไกล วิทยานิพนธ์นี้เลือกใช้โพรโตคอลอาร์ทีพีเพราะไม่ต้องการควบคุมการตอบกลับแพ็กเกตของตัวเล่นวีดิทัศน์ ซึ่งในโพรโตคอลอาร์ทีพีจะไม่มีการตอบกลับข้อมูลเช่นเดียวกับโพรโตคอลยูดีพี และภายในส่วนหัวของแพ็กเกตอาร์ทีพีจะมีตราเวลา (timestamp) และหมายเลขลำดับ (sequence number) ของแพ็กเกตคล้ายกับโพรโตคอลทีซีพี ซึ่งสามารถนำมาใช้เรียงลำดับของแพ็กเกตได้นั่นเอง

4.1.2 ตัวเล่นวีดิทัศน์

ตัวเล่นวีดิทัศน์ใช้โปรแกรมวีแอลซีเช่นเดียวกับตัวบริการวีดิทัศน์ เพื่อใช้รับและเล่นวีดิทัศน์ที่ถูกส่งมาจากโอเพนวีสวิตช์ 3 และถูกควบคุมการทำงานโดยตัวควบคุมระยะไกล

4.1.3 ตัวประสานงานเอสดีเอ็น

ประกอบด้วยตัวควบคุม 2 ชนิด คือ

4.1.3.1 ตัวควบคุมระยะไกล

ตัวควบคุมระยะไกลถูกเชื่อมต่อเข้ากับตัวบริการวีดิทัศน์และตัวเล่นวีดิทัศน์ โดยมีหน้าที่เฝ้าสังเกตพร้อมทั้งสั่งการให้อุปกรณ์ทั้ง 2 ตัวทำงาน

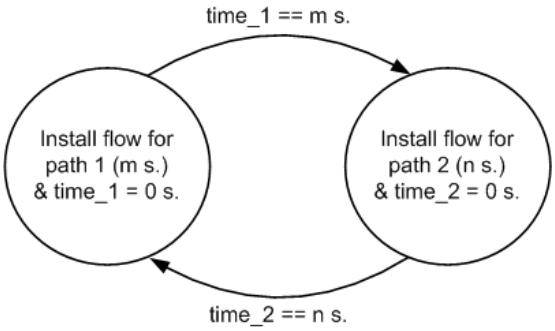
4.1.3.2 ตัวควบคุมเอสดีเอ็น

ในการทดลองนี้จะใช้ตัวควบคุมพอกซ์เป็นตัวควบคุมเอสดีเอ็น โดยพอกซ์จะมีหน้าที่ในการติดตั้งโฟลว์เอนทรีให้แก่โอเพนวีสวิตช์ทั้ง 4 ตัวในระบบทดสอบเช่นเดียวกับการทดลองในบทที่ 3 คือ เมื่อตัวควบคุมพอกซ์เริ่มทำงานแล้วได้รับการเชื่อมต่อจากโอเพนวีสวิตช์ พอกซ์จะติดตั้งโฟลว์เอนทรีให้แก่โอเพนวีสวิตช์นั้น ๆ ที่มีการเชื่อมต่อกับพอกซ์ ซึ่งพอกซ์มีการเชื่อมต่อทางกายภาพกับโอเพนวีสวิตช์ทุกตัวผ่านฮับ (hub) ดังแสดงในรูปที่ 4.2 โดยเลขที่อยู่ไอพีของพอกซ์และโอเพนวีสวิตช์ทุกตัวสามารถแสดงได้ดังตารางที่ 4.1

ตารางที่ 4.1: เลขที่อยู่ไอพีของพอกซ์และโอเพนวีสวิตช์

Device	Ethernet	IP address
POX	eth1	172.16.0.4
Open vSwitch 1	eth8	172.16.0.1
Open vSwitch 2	eth4	172.16.0.2
Open vSwitch 3	eth3	172.16.0.3
Open vSwitch 4	eth0	172.16.0.10

การทดลองนี้ จะใช้พอกซ์เป็นตัวแบ่งวีดิทัศน์ที่มาจากตัวบริการวีดิทัศน์ให้เป็นกลุ่มก้อน เพื่อส่งไปยังวิถีแต่ละวิถีผ่านโฟลว์เอนทรีของโอเพนวีสวิตช์ 1 ดังนั้นการติดตั้งโฟลว์เอนทรีให้แก่โอเพนวีสวิตช์ 1 จึงแตกต่างจากตัวอื่น ๆ โดยสามารถแสดงแผนภาพการติดตั้งโฟลว์เอนทรีให้แก่โอเพนวีสวิตช์ 1 ได้ดังรูปที่ 4.3



รูปที่ 4.3: แผนภาพการติดตั้งโฟลว์เอนทรีให้แก่โอเพนวีสวิตช์ 1

เมื่อโอเพนวีสวตช์ 1 เชื่อมต่อกับพอกซ์สำเร็จ พอกซ์จะติดตั้งโฟลว์เอนทรีที่มีเวลาหมดอายุเท่ากับ m วินาที สำหรับส่งวีดิทัศน์ไปบนวิถีที่ 1 หรือวิถีบนก่อน (ผ่านโอเพนวีสวิตช์ 1 2 และ 3) และจับเวลาเพื่อรอติดตั้งโฟลว์เอนทรีใหม่ เมื่อเวลาครบ m วินาที โฟลว์เอนทรีในโอเพนวีสวิตช์ 1 สำหรับวิถีบนจะหมดเวลาและถูกลบทิ้ง พร้อมกับพอกซ์ที่จับเวลาได้ครบ m วินาทีเช่นกัน พอกซ์จะทำการติดตั้งโฟลว์เอนทรีที่มีเวลาหมดอายุเท่ากับ n วินาที สำหรับส่งวีดิทัศน์ไปบนวิถีที่ 2 หรือวิถีล่าง (ผ่านโอเพนวีสวิตช์ 1 4 และ 3) ให้แก่โอเพนวีสวิตช์ 1 พร้อมกับจับเวลา เมื่อเวลาครบ n วินาที โฟลว์เอนทรีจะถูกลบทิ้ง พอกซ์จะติดตั้งโฟลว์เอนทรีสำหรับวิถีบนให้แก่โอเพนวีสวิตช์ พร้อมกับจับเวลาอีกครั้ง โดยโฟลว์เอนทรีของโอเพนวีสวิตช์ 1 จะถูกติดตั้งสลับวิถีบน-ล่าง ไปตลอดการสตรีมวีดิทัศน์ เพื่อเป็นการแบ่งกลุ่มก้อนวีดิทัศน์และเปลี่ยนวิถีของการสตรีมวีดิทัศน์นั่นเอง ซึ่งขนาดของกลุ่มก้อนวีดิทัศน์บนวิถีแต่ละวิถีจะมีขนาดคงที่ ขึ้นอยู่กับความจุของวิถีแต่ละวิถี และไม่แปรตามเวลาตลอดช่วงการรับ-ส่งวีดิทัศน์

4.1.4 โอเพนวีสวิตช์

ภายในระบบทดสอบโอเพนโฟลว์ ประกอบด้วยโอเพนวีสวิตช์ทั้งหมด 4 ตัว โดยทุกตัวจะรับการติดตั้งโฟลว์เอนทรีจากตัวควบคุมพอกซ์ และทำการส่งแพ็กเกตวีดิทัศน์ตามเงื่อนไขที่ระบุในโฟลว์เอนทรี ซึ่งตารางโฟลว์ของโอเพนวีสวิตช์แต่ละตัวสามารถแสดงได้ดังนี้

ตารางที่ 4.2: ตารางโฟลว์ของโอเพนวีสวิตช์ 1 สำหรับวิถีบน

Header field	Action	Timeout
in_port = 1	output:2	hard_timeout = m
in_port = 2	output:1	hard_timeout = m

ตารางที่ 4.3: ตารางโฟลว์ของโอเพนวีสวิตช์ 1 สำหรับวิถีล่าง

Header field	Action	Timeout
in_port = 1	output:3	hard_timeout = n
in_port = 3	output:1	hard_timeout = n

ตารางที่ 4.2 เป็นตารางโฟลว์ของโอเพนวีสวิตช์ 1 ที่ใช้สำหรับส่งแพ็กเกตวีดิทัศน์ไปยังวิถีบน โดยโฟลว์เอนทรีสำหรับวิถีบนจะมีอายุเท่ากับ m วินาที (hard_timeout=m) เมื่อโฟลว์เอนทรีอยู่ในตารางโฟลว์เป็นเวลา m วินาที โฟลว์เอนทรีจะถูกลบทิ้งออกจากตารางโฟลว์ทันที เมื่อแพ็กเกตวีดิทัศน์จากตัวบริการวีดิทัศน์เข้ามายังช่องทาง 1 (in_port=1) หรือ eth0 ในรูปที่ 4.2 โอเพนวีสวิตช์ 1 จะส่งแพ็กเกตออกไปยังช่องทาง 2 (actions=output:2) หรือ eth7 เพื่อส่งไปยังโอเพนวีสวิตช์ 2 เป็นเวลา m วินาที และส่งแพ็กเกตที่เข้ามาที่ช่องทาง 2 (in_port=2) ออกไปยังช่องทาง 1 (actions=output:1) เป็นเวลา m วินาทีเช่นกัน และเมื่อโฟลว์เอนทรีสำหรับวิถีบนถูกลบทิ้ง พอกซ์จะติดตั้งโฟลว์เอนทรีสำหรับวิถีล่างให้แก่โอเพนวีสวิตช์ 1 ดังตารางที่ 4.3 ซึ่งมีอายุเท่ากับ n วินาที (hard_timeout=n) โดยโอเพนวิสวิตช์จะเปลี่ยนการส่งออกแพ็กเกตวีดิทัศน์ที่รับเข้ามาจากช่องทาง 1 ให้ส่งออกไปยังช่องทาง 3 (actions=output:3) หรือ eth4 เพื่อส่งแพ็กเกต

ไปยังโอเพนวีสวิตช์ 4 และส่งแพ็กเกตที่มายังช่องทาง 3 (in_port=3) ออกไปยังช่องทาง 1 (actions=output:1) เป็นเวลา n วินาที จากนั้นโฟลว์เอนทรีสำหรับวิถีล่างจะถูกลบทิ้ง และพอกซ์จะติดตั้งโฟลว์เอนทรีสำหรับวิถีบนให้โอเพนวีสวิตช์ใหม่ สลับกันไปเรื่อย ๆ ตลอดช่วงการสตรีมวีดิทัศน์ ซึ่งระยะเวลาหรืออายุของโฟลว์เอนทรีสำหรับวิถีแต่ละวิถีจะใช้เป็นตัวกำหนดขนาดของกลุ่มก้อนวีดิทัศน์ที่ถูกส่งผ่านบนวิถีนั้น ๆ

ตารางที่ 4.4: ตารางโฟลว์ของโอเพนวีสวิตช์ 2

Header field	Action
in_port = 1	output:2
in_port = 2	output:1

ตารางโฟลว์ของโอเพนวีสวิตช์ 2 แสดงได้ดังตารางที่ 4.4 เมื่อโอเพนวีสวิตช์ 2 ได้รับแพ็กเกตวีดิทัศน์จากโอเพนวีสวิตช์ 1 ที่เข้ามายังช่องทาง 1 (in_port=1) หรือ eth0 โอเพนวีสวิตช์ 2 จะส่งแพ็กเกตไปให้โอเพนวีสวิตช์ 3 ผ่านช่องทาง 2 (actions=output:2) และส่งแพ็กเกตที่เข้ามาที่ช่องทาง 2 (in_port=2) ออกไปยังช่องทาง 1 (actions=output:1)

ตารางที่ 4.5: ตารางโฟลว์ของโอเพนวีสวิตช์ 3

Header field	Action
in_port = 1	mod_dl_src:78:cd:8e:81:86:58, output:4
in_port = 2	ALL
in_port = 3	mod_dl_src:88:17:20:06:21:58, output:4
in_port = 4	output:drop
in_port = 5	output:2

ตารางที่ 4.5 แสดงตารางโฟลว์ของโอเพนวีสวิตช์ 3 โดยโอเพนวีสวิตช์ 3 จะรับแพ็กเกตวีดิทัศน์จากวิถีบนและวิถีล่าง เมื่อแพ็กเกตวีดิทัศน์จากวิถีบนเข้ามายังช่องทาง 1 (in_port=1) หรือ eth2 โอเพนวีสวิตช์ 3 จะเปลี่ยนที่อยู่ต้นทางของชั้นเส้นทางเชื่อมโยงข้อมูลของแพ็กเกตนั้นให้มีค่าเท่ากับ 78:cd:8e:81:86:58 เพื่อเป็นการระบุว่าแพ็กเกตนี้มาจากวิถีบน แล้วส่งแพ็กเกตต่อไปยังมิดเดิลบอกซ์ผ่านช่องทาง 4 หรือ eth5 (actions=mod_dl_src:78:cd:8e:81:86:58, output:4) หากแพ็กเกตวีดิทัศน์จากวิถีล่างเข้ามายังช่องทาง 3 (in_port=3) หรือ eth9 โอเพนวีสวิตช์จะเปลี่ยนที่อยู่ต้นทางของแพ็กเกตให้เท่ากับ 88:17:20:06:21:58 เพื่อเป็นการระบุว่าแพ็กเกตนี้มาจากวิถีล่าง แล้วส่งออกไปยังมิดเดิลบอกซ์ผ่านช่องทาง 4 เช่นกัน (actions=mod_dl_src:88:17:20:06:21:58, output:4) สาเหตุที่ต้องให้โอเพนวีสวิตช์ 3 เปลี่ยนที่อยู่ต้นทางของชั้นเส้นทางเชื่อมโยงข้อมูลของแพ็กเกตเพราะเพื่อให้มิดเดิลบอกซ์สามารถแยกแยะได้ว่าแพ็กเกตนั้นมาจากวิถีใด ซึ่งจะทำให้เก็บแพ็กเกตเข้าบัฟเฟอร์ได้อย่างถูกต้องแล้วนำไปเรียงลำดับต่อนั่นเอง เมื่อแพ็กเกตวีดิทัศน์ที่ถูกเรียงลำดับแล้วจากมิดเดิลบอกซ์เข้ามายังช่องทาง 5 (in_port=5) หรือ eth8 โอเพนวีสวิตช์ 3 จะส่งแพ็กเกตนั้นต่อไปยังตัวเล่นวีดิทัศน์ผ่านช่องทาง 2 (actions=output:2) หรือ eth4 หากมีแพ็กเกตเข้ามายังช่องทาง 2 (in_port=2) แพ็กเกตจะถูกส่งออกไปทุกช่องทาง (actions=ALL) แต่หากมีแพ็กเกตเข้ามายังช่องทาง 4 โอเพนวีสวิตช์จะทิ้งแพ็กเกตนั้นไป (actions=output:drop)

ตารางที่ 4.6: ตารางโฟลว์ของโอเพนวีสวิตช์ 4

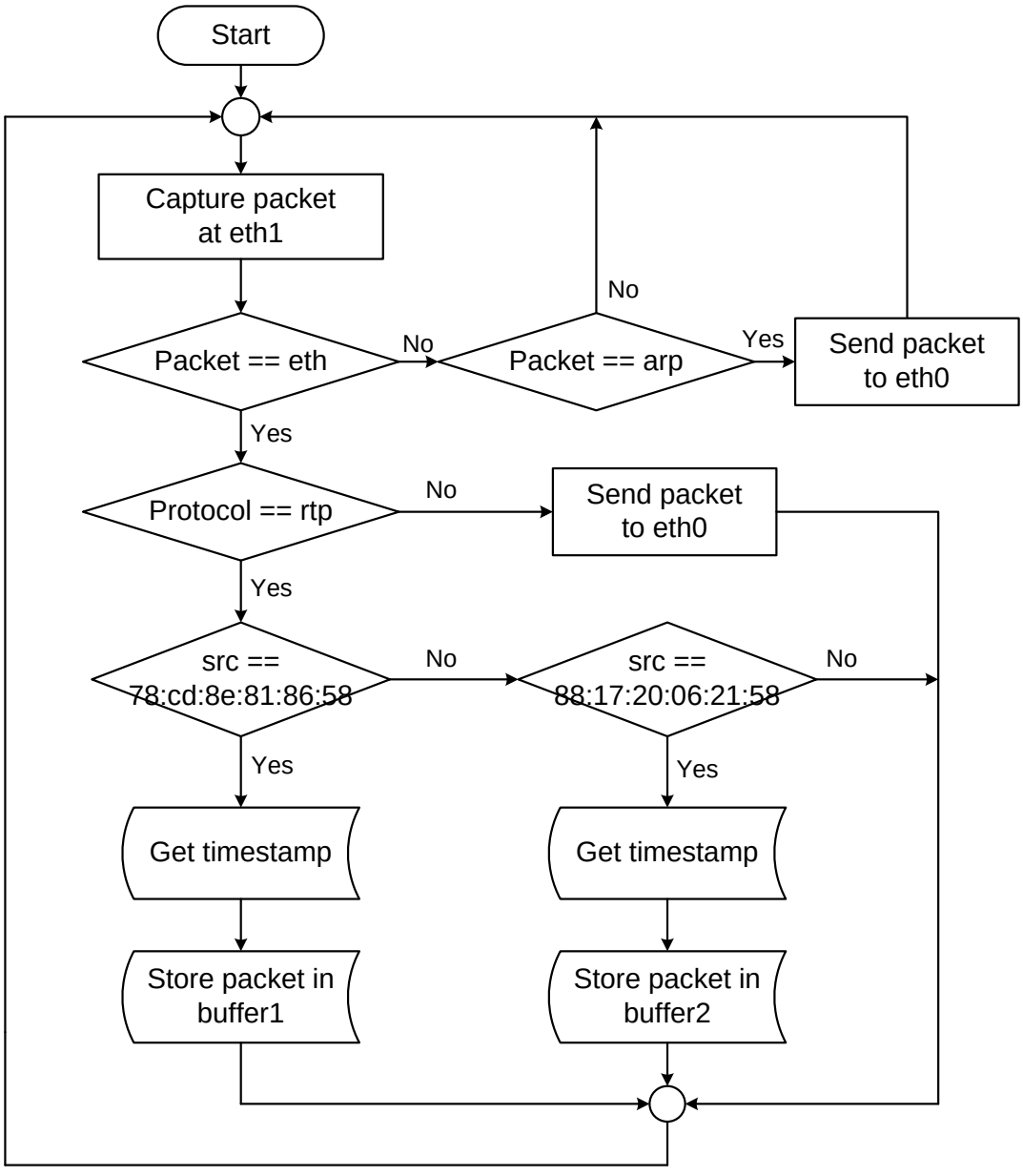
Header field	Action
in_port = 1	output:2
in_port = 2	output:1

ตารางโฟลว์ของโอเพนวีสวิตช์ 4 แสดงได้ดังตารางที่ 4.6 เมื่อแพ็กเกตวีดิทัศน์ถูกส่งจากโอเพนวีสวิตช์ 1 เข้ามายังช่องทาง 1 (in_port=1) หรือ eth2 ของโอเพนวีสวิตช์ 4 แพ็กเกตนั้นจะถูกส่งต่อไปยังโอเพนวีสวิตช์ 3 ผ่านช่องทาง 2 (actions=output:2) หรือ eth5 และส่งแพ็กเกตที่เข้ามายังช่องทาง 2 (in_port=2) ออกไปยังช่องทาง 1 (actions=output:1)

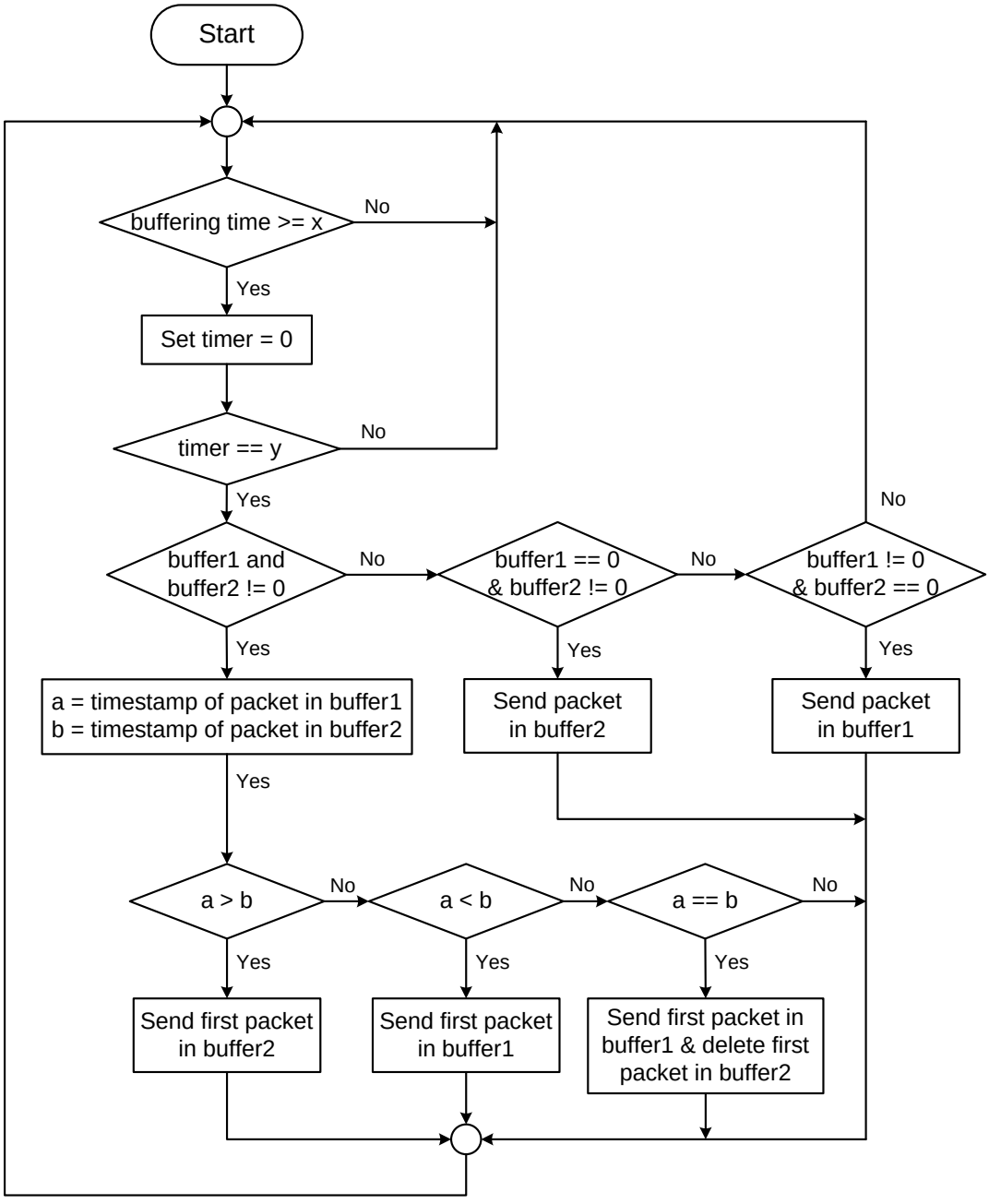
4.1.5 มิดเดิลบอกซ์

มิดเดิลบอกซ์มีหน้าที่รวมแพ็กเกตวีดิทัศน์จากวิถีบนและวิถีล่างเข้าด้วยกัน ก่อนจะเรียงลำดับให้ถูกต้อง และส่งออกไปให้โอเพนวีสวิตช์ 3 เพื่อส่งต่อไปยังตัวเล่นวีดิทัศน์ จากรูปที่ 4.2 จะเห็นว่ามิดเดิลบอกซ์เชื่อมต่อกับโอเพนวีสวิตช์ 3 ผ่านอินเตอร์เฟซ 2 อินเตอร์เฟซ คือ eth0 และ eth1 โดย eth1 จะรับแพ็กเกตทุกแพ็กเกตที่ถูกส่งมาจาก eth5 ของโอเพนวีสวิตช์ 3 แล้วจะส่งแพ็กเกตที่ถูกเรียงลำดับแล้วออกไปทาง eth0 ให้แก่ eth8 ของโอเพนวีสวิตช์ 3 สาเหตุที่เชื่อมต่อมิดเดิลบอกซ์ผ่านอินเตอร์เฟซ 2 อินเตอร์เฟซ คือ ต้องการแยกอินเตอร์เฟซที่ใช้รับ-ส่งแพ็กเกตออกจากกัน ในงานวิจัยนี้ใช้เครื่องคอมพิวเตอร์ระบบปฏิบัติการลินุกซ์ Ubuntu 12.04 หน่วยประมวลผล Core 2 Quad 2.40 GHz ที่ดำเนินงานโปรแกรมไพทอนจำลองเป็นมิดเดิลบอกซ์ โดยภายในโปรแกรมไพทอนประกอบด้วยมอดูล (module) pcapy ที่ใช้จับแพ็กเกตที่เข้ายัง eth1 มอดูล dpkt ที่ใช้แจง (parse) แพ็กเกต และตรวจดูข้อมูลส่วนหัวของแพ็กเกตเพื่อแยกแพ็กเกตว่ามาจากวิถีใด มอดูล scapy สำหรับห่อแพ็กเกตที่ถูกแจงเพื่อส่งออกผ่านทาง eth0 การทำงานของมิดเดิลบอกซ์แบ่งออกเป็น 2 ส่วน คือ การตรวจจับแพ็กเกต และการจัดเรียงและส่งออกแพ็กเกต โดยผังการทำงานของมิดเดิลบอกซ์สามารถแสดงได้ดังรูปที่ 4.4 และ 4.5

เมื่อมิดเดิลบอกซ์เริ่มทำงานโปรแกรมทั้ง 2 ส่วนจะทำงานพร้อม ๆ กัน โดยโปรแกรมส่วนการตรวจจับแพ็กเกตจะรอจับแพ็กเกตทุกแพ็กเกตที่เข้ามายังมิดเดิลบอกซ์ ณ eth1 จากนั้นโปรแกรมตรวจสอบส่วนหัวของแพ็กเกตว่าแพ็กเกตนั้นเป็นแพ็กเกตอีเทอร์เน็ตหรือไม่ หากไม่ใช่จะตรวจสอบต่อว่าเป็นแพ็กเกตเออาร์พี (ARP) หรือไม่ หากไม่ใช่โปรแกรมจะไม่ทำอะไรกับแพ็กเกตนั้นและกลับไปรอตรวจจับแพ็กเกตใหม่ แต่หากแพ็กเกตเป็นแพ็กเกตเออาร์พี แพ็กเกตจะถูกส่งออกที่ eth0 ในกรณีที่แพ็กเกตเป็นแพ็กเกตอีเทอร์เน็ต โปรแกรมจะตรวจสอบต่อว่าโพรโตคอลของแพ็กเกตนั้นเป็นโพรโตคอลอาร์ทีพีใช่หรือไม่ โดยแพ็กเกตที่มีโพรโตรคอลนอกเหนือจากอาร์ทีพีจะถูกส่งออกไปที่ eth0 แต่หากโพรโตคอลของแพ็กเกตคืออาร์ทีพี โปรแกรมจะตรวจสอบต่อว่าที่อยู่ต้นทางของชั้นเส้นทางเชื่อมโยงข้อมูลมีค่าเท่ากับเท่าไร โดยหากที่อยู่ต้นทางของแพ็กเกตมีค่าเท่ากับ 78:cd:8e:81:86:58 หรือมาจากวิถีบน โปรแกรมจะเก็บค่าตราเวลาของแพ็กเกตไว้ และเก็บแพ็กเกตไว้ในบัฟเฟอร์ 1 (buffer1) แต่ถ้าที่อยู่ต้นทางของแพ็กเกตมีค่าเป็น 88:17:20:06:21:58 หรือมาจากวิถีล่าง โปรแกรมจะเก็บตราเวลา และเก็บแพ็กเกตนั้นลงในบัฟเฟอร์ 2 เพื่อนำแพ็กเกตไปจัดเรียงและส่งออกแพ็กเกตต่อไป



รูปที่ 4.4: ผังการทำงานของมิดเดิลบอกซ์ส่วนการตรวจจับแพ็กเกต



รูปที่ 4.5: ผังการทำงานของมิดเดิลบอกซ์ส่วนการจัดเรียงและส่งออกแพ็กเกต

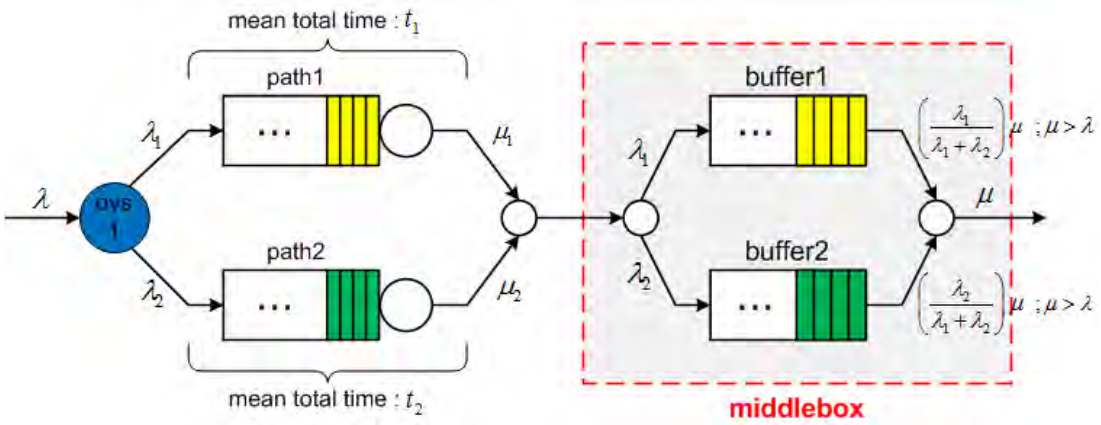
โปรแกรมส่วนการจัดเรียงและส่งออกแพ็กเกต เป็นการนำแพ็กเกตในบัฟเฟอร์ 1 และ 2 ที่ได้จากโปรแกรมส่วนการตรวจจับแพ็กเกตมาเรียงลำดับให้มีความถูกต้อง ก่อนจะส่งออกที่ eth0 ให้แก่โอเพนวีสวิตช์ 3 เพื่อส่งแพ็กเกตต่อไปยังตัวเล่นวีดิทัศน์ เมื่อเริ่มทำงานโปรแกรมจะรอให้ส่วนตรวจจับแพ็กเกตเก็บแพ็กเกตเข้าไว้ในบัฟเฟอร์ 1 และ 2 นานเป็นระยะเวลา x จากนั้นจึงจะเริ่มตั้งเวลาไว้เพื่อใช้เป็นอัตราการส่งออกแพ็กเกต หากจับเวลาได้ครบตามกำหนดคาบเวลาการส่งแพ็กเกต หรือเท่ากับ y โปรแกรมจะเริ่มตรวจสอบว่ามีแพ็กเกตอยู่ในบัฟเฟอร์ทั้ง 2 บัฟเฟอร์ หรือมีแพ็กเกตอยู่เพียงแค่ในบัฟเฟอร์ใดบัฟเฟอร์หนึ่ง หากมีแพ็กเกตอยู่ในทั้งบัฟเฟอร์ 1 และ 2 แพ็กเกตที่ถูกเก็บเข้าบัฟเฟอร์ก่อนจะถูกนำมาเปรียบเทียบกัน โดยพิจารณาจากตราเวลาของแพ็กเกต ซึ่งแพ็กเกตจะถูกกำหนดตราเวลาที่อยู่ในส่วนหัวโดยตัวบริการวีดิทัศน์ หากตราเวลาของแพ็กเกตในบัฟเฟอร์ใดมีค่าน้อยกว่า โปรแกรมจะส่งออกแพ็กเกตที่มีตราเวลาน้อยกว่านั้นไปยังโอเพนวีสวิตช์ 3 ผ่าน eth0 แต่เมื่อตรวจสอบพบว่าตราเวลาของแพ็กเกตจากบัฟเฟอร์ 1 และ 2 มีค่าเท่ากัน หรือเป็นแพ็กเกตเดียวกันที่ถูกส่งซ้ำซ้อนมาบนวิถีทั้ง 2 วิถี แพ็กเกตที่อยู่ในบัฟเฟอร์ 1 จะถูกส่งออกและแพ็กเกตในบัฟเฟอร์ 2 จะถูกลบทิ้งเพื่อป้องกันไม่ให้ส่งแพ็กเกตซ้ำออกไป ในกรณีที่มีแพ็กเกตอยู่เพียงแค่บัฟเฟอร์ใดบัฟเฟอร์หนึ่งเท่านั้น โปรแกรมจะส่งแพ็กเกตที่อยู่ในบัฟเฟอร์นั้นออกไปทันที โดยไม่พิจารณาตราเวลาของแพ็กเกต

4.2 การทดลองเพื่อตั้งค่าพารามิเตอร์การสตรีมกลุ่มก้อนวีดิทัศน์แบบพหุวิถีโดยอาศัยการแบ่งขนาดกลุ่มก้อนวีดิทัศน์ด้วยตัวควบคุมพอกซ์

วิทยานิพนธ์นี้ต้องการศึกษาผลกระทบของระยะเวลาที่มิดเดิลบอกซ์ใช้เพื่อเก็บแพ็กเกตที่ถูกส่งมาจากตัวบริการวีดิทัศน์เข้าบัฟเฟอร์ก่อนจะจัดเรียงแล้วส่งออกไปให้ตัวเล่นวีดิทัศน์ (หรือ x ในรูปที่ 4.5) และขนาดกลุ่มก้อนวีดิทัศน์ที่ถูกส่งผ่านวิถีแต่ละวิถี ที่มีผลต่อสมรรถนะในการสตรีมวีดิทัศน์บนระบบทดทดสอบโอเพนโฟลว์ โดยอาศัยการแบ่งกลุ่มก้อนวีดิทัศน์ด้วยโปรแกรมภายในตัวควบคุมพอกซ์

4.2.1 อัตราแพ็กเกตที่ผ่านวิถีแต่ละวิถี

เพื่อให้ง่ายต่อการคำนวณ งานวิจัยนี้จึงพิจารณาทราฟฟิกวีดิทัศน์บนวิถีแต่ละวิถีของระบบทดสอบโดยอ้างอิงระบบแถวคอยแบบ M/M/1 ในการหาอัตราแพ็กเกตที่วิ่งผ่านวิถีแต่ละวิถีของระบบทดสอบ ซึ่งในระบบทดสอบประกอบด้วยระบบแถวคอย M/M/1 2 ระบบ ดังแสดงในรูปที่ 4.6 โดยระบบแถวคอยแรกจะพิจารณาทราฟฟิกที่อยู่บนวิถีบน ซึ่งแพ็กเกตจะออกมาจากโอเพนวีสวิตช์ 1 ผ่านโอเพนวีสวิตช์ 2 และ 3 เข้าสู่บัฟเฟอร์ 1 ของมิดเดิลบอกซ์แล้วถูกส่งออกไป ส่วนระบบแถวคอยที่ 2 จะพิจารณาทราฟฟิกที่ผ่านวิถีล่างจากโอเพนวีสวิตช์ 1 ผ่านโอเพนวีสวิตช์ 2 และ 4 เข้าสู่บัฟเฟอร์ 2 แล้วถูกส่งออกจากมิดเดิลบอกซ์



รูปที่ 4.6: รูปจำลองระบบแถวคอยในระบบทดสอบ

พารามิเตอร์ต่าง ๆ ที่แสดงในรูปที่ 4.6 สามารถอธิบายได้ดังนี้

- λ : อัตราแพ็กเกตที่เข้าสู่ระบบทดสอบ หรืออัตราแพ็กเกตที่ออกจากตัวบริการวีดิทัศน์เข้าสู่โอเพนวีสวิตช์ 1 (แพ็กเกตต่อวินาที)
- λ_1 : อัตราแพ็กเกตที่ผ่านวิถีบนเข้าสู่บัฟเฟอร์ 1 ของมิดเดิลบอกซ์ (แพ็กเกตต่อวินาที)
- λ_2 : อัตราแพ็กเกตที่ผ่านวิถีล่างเข้าสู่บัฟเฟอร์ 2 ของมิดเดิลบอกซ์ (แพ็กเกตต่อวินาที)
- μ : อัตราแพ็กเกตที่ออกจากมิดเดิลบอกซ์ (แพ็กเกตต่อวินาที)
- μ_1 : แบนด์วิดท์ของวิถีบน (แพ็กเกตต่อวินาที)
- μ_2 : แบนด์วิดท์ของวิถีล่าง (แพ็กเกตต่อวินาที)

พิจารณาสูตรเวลารวมเฉลี่ย (mean total time) ของระบบแถวคอยแบบ M/M/1 กำหนดให้เวลารวมเฉลี่ยที่แพ็กเกต 1 แพ็กเกตถูกส่งไปบนวิถีแต่ละวิถีมีค่าเท่ากัน เพื่อเป็นการสมดุลค่าเวลาเฉลี่ยที่แพ็กเกตต้องใช้ในการเดินทางผ่านวิถีบนและวิถีล่าง เพื่อให้มิดเดิลบอกซ์เกิดภาระน้อยที่สุดในการเรียงลำดับแพ็กเกตที่เข้ามาจากวิถีทั้ง 2 วิถี จะได้ว่า

$$\begin{aligned} t_1 &= t_2 \\ \frac{1}{\mu_1 - \lambda_1} &= \frac{1}{\mu_2 - \lambda_2} \\ \mu_1 - \lambda_1 &= \mu_2 - \lambda_2 \end{aligned} \tag{4.1}$$

แพ็กเกตวีดิทัศน์ถูกส่งจากตัวบริการวีดิทัศน์ด้วยอัตรา λ เมกะบิตต่อวินาที ถูกส่งไปบนวิถีบนและวิถีล่างด้วยอัตรา λ_1 และ λ_2 แพ็กเกตต่อวินาที ตามลำดับ จะได้

$$\begin{aligned} \lambda &= \lambda_1 + \lambda_2 \\ \lambda_1 &= \lambda - \lambda_2 \end{aligned} \tag{4.2}$$

แทนสมการ (4.2) ลงในสมการ (4.1) จะได้

$$\lambda_2 = (\frac{\mu_2 - \mu_1}{2}) + (\frac{\lambda}{2}) \tag{4.3}$$

และแทนสมการ (4.3) ลงในสมการ (4.2) จะได้

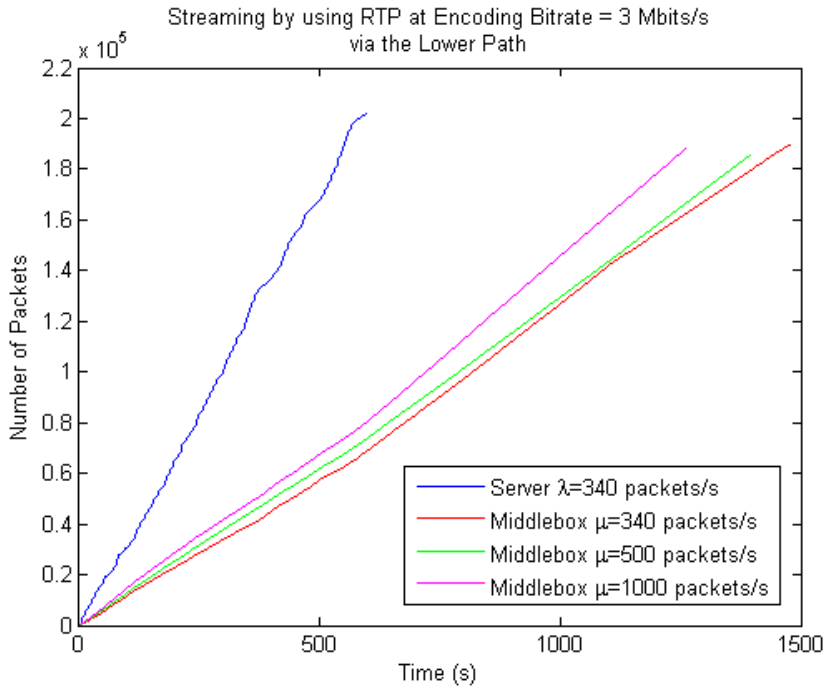
$$\lambda_1 = (\frac{\lambda}{2}) + (\frac{\mu_1 - \mu_2}{2}) \tag{4.4}$$

4.2.2 อัตราแพ็กเกตที่เข้าสู่ระบบทดสอบและอัตราแพ็กเกตที่ออกจากมิดเดิลบอกซ์

เมื่อใช้โปรแกรมไอเพิฟวัดค่าแบนด์วิดท์ระหว่างตัวบริการวีดิทัศน์และตัวเล่นวีดิทัศน์ของทั้งวิถีบนและวิถีล่าง พบว่าวิถีบนมีแบนด์วิดท์เท่ากับ 6.71 เมกะบิตต่อวินาที และวิถีล่างมีแบนด์วิดท์เท่ากับ 3.93 เมกกะบิตต่อวินาที ตามลำดับ ผู้วิจัยจึงได้ทดลองสตรีมวีดิทัศน์จากตัวบริการวีดิทัศน์ไปยังตัวเล่นวีดิทัศน์ผ่านวิถีล่างด้วยอัตราการเข้ารหัสเท่ากับ 3 เมกะบิตต่อวินาทีโดยใช้โพรโตคอลอาร์ทีพี เพื่อหาอัตราแพ็กเกตที่เข้าสู่ระบบทดสอบ โดยจะสตรีมวีดิทัศน์ไปยังมิดเดิลบอกซ์ก่อนส่งต่อไปให้แก่ตัวเล่นวีดิทัศน์ ซึ่งมิดเดิลบอกซ์จะส่งออกแพ็กเกตที่เข้ามาทันทีโดยไม่ต้องเก็บแพ็กเกตเข้าบัฟเฟอร์เพื่อเรียงลำดับเนื่องจากแพ็กเกตมาจากวิถีล่างวิถีเดียวเท่านั้นจึงไม่จำเป็นต้องรอเทียบตราเวลากับแพ็กเกตที่มาจากวิถีบน การทดลองนี้ใช้วีดิทัศน์เรื่อง Big Buck Bunny เช่นเดียวกับบทที่ 3 เมื่อใช้โปรแกรมไวร์ชาร์กจับแพ็กเกตที่ถูกสตรีมออกจากตัวบริการวีดิทัศน์ แล้วนำจำนวนแพ็กเกตมาหาความสัมพันธ์กับเวลา จะได้ว่า อัตราแพ็กเกตที่ออกจากตัวบริการวิดีทัศน์มีค่าโดยเฉลี่ยเท่ากับ 340 แพ็กเกตต่อวินาที

ในการหาอัตราแพ็กเกตที่ออกจากมิดเดิลบอกซ์นั้น ผู้วิจัยได้กำหนดอัตราการส่งแพ็กเกตในโปรแกรมไพทอนให้มิดเดิลบอกซ์ส่งแพ็กเกตโดยเริ่มจากอัตรา 340 แพ็กเกตต่อวินาที ซึ่งเป็นอัตราเดียวกับอัตราแพ็กเกตที่ถูกส่งออกจากตัวบริการวีดิทัศน์ แล้วทดลองเพิ่มค่าไปเรื่อย ๆ เป็น 500 และ 1000 แพ็กเกตต่อวินาที ตามลำดับ และเมื่อใช้โปรแกรมไวร์ชาร์กจับแพ็กเกตที่ถูกส่งออกจากมิดเดิลบอกซ์ พบว่า อัตราการส่งแพ็กเกตที่ถูกกำหนดด้วยโปรแกรมและอัตราการส่งแพ็กเกตที่วัดได้จริงมีค่าไม่เท่ากัน โดยเมื่อกำหนดให้มิดเดิลบอกซ์ส่งออกแพ็กเกตที่อัตรา 340 500 และ 1000 แพ็กเกตต่อวินาที จะวัดอัตราแพ็กเกตที่ถูกส่งออกมาจริงได้เท่ากับ 128 133 และ 149 แพ็กเกตต่อวินาที ตามลำดับ ดังแสดงในรูปที่ 4.7 ส่งผลให้วีดิทัศน์ที่แสดง ณ ตัวเล่นวีดิทัศน์มีความหน่วงเป็นอย่างมาก และไม่ว่าจะเพิ่มอัตราการส่งแพ็กเกตในโปรแกรมด้วยอัตรามากเท่าใด มิดเดิลบอกซ์ก็ไม่สามารถส่งแพ็กเกตออกไปได้ที่อัตรา 340 แพ็กเกตต่อวินาทีได้ ทั้งนี้เพราะข้อจำกัดทางฮาร์ดแวร์ (hardware) ของเครื่องคอมพิวเตอร์ที่ใช้จำลองเป็นมิดเดิลบอกซ์นั่นเอง

เนื่องจากมิดเดิลบอกซ์มีข้อจำกัดทางฮาร์ดแวร์ทำให้ไม่สามารถส่งออกแพ็กเกตด้วยอัตราสูง ๆ ได้ ดังนั้นวิทยานิพนธ์จึงเลือกสตรีมวีดิทัศน์ด้วยอัตราการเข้ารหัสเท่ากับ 500 กิโลบิตต่อวินาที ซึ่งเมื่อวัดอัตราการส่งออกแพ็กเกตของตัวบริการวีดิทัศน์จริง พบว่าตัวบริการวีดิทัศน์จะส่งแพ็กเกตออกมาที่อัตราเฉลี่ยเท่ากับ 100 แพ็กเกตต่อวินาที โดยแพ็กเกตทุกแพ็กเกตมีขนาดคงที่ตลอดการสตรีมเท่ากับ 1370 ไบต์ แต่เนื่องจากระบบทดสอบมีแบนด์วิดท์ของวิถีทั้ง 2 วิถีมากเพียงพอต่อการรองรับปริมาณทราฟฟิกเมื่อวีดิทัศน์ถูกสตรีมด้วยอัตราการเข้ารหัส 500 กิโลบิตต่อวินาที หรือแพ็กเกตที่ถูกส่งมาจากตัวบริการวีดิทัศน์ด้วยอัตรา 100 แพ็กเกตต่อวินาที ทำให้ไม่สามารถเปรียบเทียบสมรรถนะระหว่างการสตรีมวีดิทัศน์แบบหนึ่งวิถีและพหุวิถีได้ ดังนั้นผู้วิจัยจึงปรับลดค่าแบนด์วิดท์ของระบบ

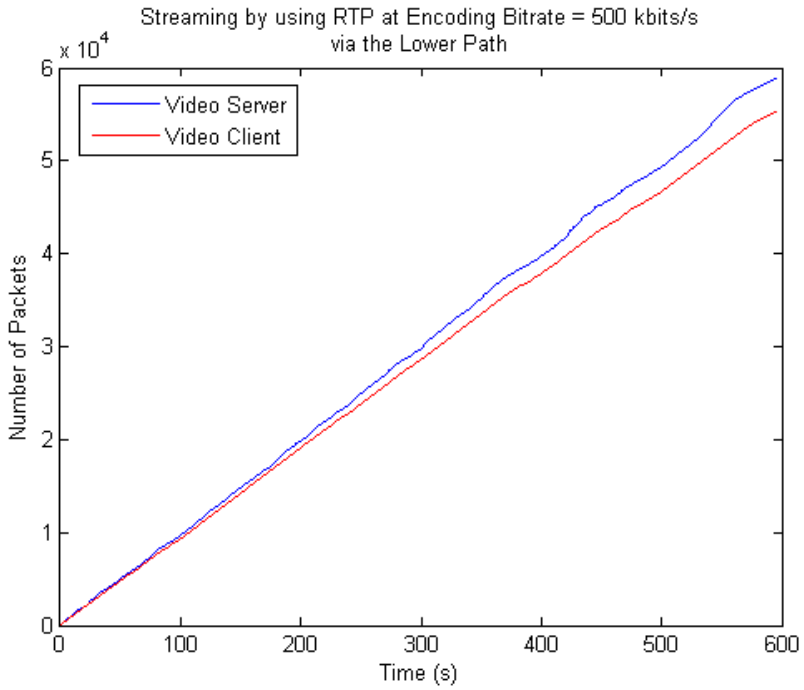


รูปที่ 4.7: กราฟแสดงจำนวนแพ็กเกตเทียบกับเวลาเมื่อสตรีมวีดิทัศน์ผ่านวิถีล่างด้วยอัตราการเข้ารหัส 3 เมกะบิตต่อวินาทีโดยใช้โพรโตคอลอาร์ทีพี

ทดสอบ โดยใช้โปรแกรม wondershaper [30] จำกัดอัตราการรับ-ส่งแพ็กเกตของโอเพนวีสวิตช์ 1 ซึ่งจะกำหนดให้วิถีบนและวิถีล่างมีแบนด์วิดท์เป็น 2.07 และ 1.05 เมกะบิตต่อวินาที หรือ μ_1 = 188 และ μ_2 = 95 แพ็กเกตต่อวินาที ตามลำดับ

เมื่อสตรีมวีดิทัศน์ผ่านวิถีล่างด้วยอัตราการเข้ารหัสเท่ากับ 500 กิโลบิตต่อวินาที และปรับค่าอัตราการส่งออกแพ็กเกตในโปรแกรมของมิดเดิลบอกซ์ให้มีค่าเป็น 100 แพ็กเกตต่อวินาที เพื่อให้เท่ากับอัตราแพ็กเกตที่ถูกส่งออกมาจากตัวบริการวีดิทัศน์ พบว่า อัตราแพ็กเกตที่เข้าสู่ตัวเล่นวีดิทัศน์จริงมีค่าเฉลี่ยเท่ากับ 93 แพ็กเกตต่อวินาที ดังรูปที่ 4.8 ทั้งนี้ เนื่องจากเกิดการสูญเสียแพ็กเกตระหว่างการเดินทางจากตัวบริการวีดิทัศน์ผ่านวิถีล่างมายังมิดเดิลบอกซ์ และสูญเสียแพ็กแกต ณ มิดเดิลเดิลบอกซ์ กล่าวคือ มิดเดิลบอกซ์ไม่สามารถนำแพ็กเกตทุกแพ็กเกตที่ได้รับเข้ามา มาทำการประมวลผลเพื่อเตรียมส่งออกไปได้ทุกแพ็กเกต เพราะข้อกำจัดทางฮาร์ดแวร์ของมิดเดิลบอกซ์ โดยแพ็กเกตที่สูญเสียทั้งหมดคิดเป็น 6.15% ของแพ็กเกตที่ถูกส่งมาจากตัวบริการวีดิทัศน์

ดังนั้นจะได้ว่า เมื่อสตรีมวีดิทัศน์ผ่านวิถีล่างด้วยอัตราการเข้ารหัสเท่ากับ 500 กิโลบิตต่อวินาที อัตราแพ็กเกตที่ออกจากตัวบริการวีดิทัศน์ที่วัดได้จริง หรืออัตราแพ็กเกตที่เข้าสู่ระบบทดสอบ (λ) มีค่าเฉลี่ยเท่ากับ 100 แพ็กเกตต่อวินาที และอัตราแพ็กเกตที่ออกจากมิดเดิลบอกซ์ซึ่งถูกกำหนดโดยโปรแกรมไพทอนมีค่าเท่ากับ 100 แพ็กเกตต่อวินาที ดังนั้นจึงได้ $\lambda = \mu = 100$ แพ็กเกตต่อวินาที และเมื่อแทนค่า λ = 100, μ_1 = 188 และ μ_2 = 95 แพ็กเกตต่อวินาที ลงในสมการที่ (4.3) และ (4.4) เพื่อหาค่า λ_1 และ λ_2 จะได้ อัตราแพ็กเกตที่ผ่านวิถีบนเข้าสู่บัฟเฟอร์ 1 λ_1 = 96 แพ็กเกตต่อวินาที และอัตราแพ็กเกตที่ผ่านวิถีล่างเข้าสู่บัฟเฟอร์ 2 λ_2 = 4 แพ็กเกตต่อวินาที ซึ่ง λ_1 คิดเป็น 24 เท่าของ λ_2



รูปที่ 4.8: กราฟแสดงจำนวนแพ็กเกตเทียบกับเวลาเมื่อสตรีมวีดิทัศน์ผ่านวิถีล่างด้วยอัตราการเข้ารหัส 500 กิโลบิตต่อวินาทีโดยใช้โพรโตคอลอาร์ทีพี

4.2.3 การแบ่งกลุ่มก้อนวีดิทัศน์

งานวิจัยนี้จะทำการทดลองแบ่งวีดิทัศน์ให้เป็นกลุ่มก้อน โดยอาศัยอัตราแพ็กเกตที่วิ่งผ่านวิถีแต่ละวิถี กล่าวคือ ระยะเวลาโฟลว์เอนทรีของโอเพนวีสวิตช์ 1 สำหรับส่งแพ็กเกตไปบนวิถีแต่ละวิถีที่ถูกกำหนดโดยพอกซ์จะแปรผันตรงกับอัตราแพ็กเกตที่วิ่งผ่านวิถีแต่ละวิถี ซึ่งจากการทดลองข้างต้นทำให้ทราบว่า อัตราแพ็กเกตที่วิ่งผ่านวิถีบนมีค่ามากกว่าอัตราแพ็กเกตที่วิ่งผ่านวิถีล่างอยู่ 24 เท่า ดังนั้นจึงพิจารณาระยะเวลาในการส่งแพ็กเกตไปยังวิถีบนและวิถีล่างที่ 10 และ 240 วินาที

4.3 ผลการทดลอง

4.3.1 การทดลองที่ 1: μ = 100 แพ็กเกตต่อวินาที, x = 0 วินาที

ในการทดลองนี้จะศึกษาผลกระทบของขนาดกลุ่มก้อนวีดิทัศน์ที่ถูกส่งผ่านวิถีทั้ง 2 วิถีของระบบทดสอบโอเพนโฟลว์ เมื่อระบบทดสอบมีอัตราส่วนเวลาที่แพ็กเกตวิ่งผ่านวิถีบนต่อเวลาที่แพ็กเกตวิ่งผ่านวิถีล่างเป็นแบบต่าง ๆ ได้แก่ 10 ต่อ 10 วินาที 10 ต่อ 240 วินาที และ 240 ต่อ 10 วินาที โดยการทดลองนี้มิดเดิลบอกซ์ยังจะไม่เก็บแพ็กเกตเข้าบัฟเฟอร์ แต่จะส่งออกไปทันทีเมื่อได้รับแพ็กเกต ซึ่งค่าพารามิเตอร์อื่น ๆ ที่ใช้ทดลองสามารถแสดงได้ในตารางที่ 4.7

จากการทดลองสตรีมวีดิทัศน์แล้วจับแพ็กเกตที่เข้ามายังตัวเล่นวีดิทัศน์ พบว่า อัตราที่แพ็กเกตเข้ามามีค่าใกล้เคียงกับอัตราแพ็กเกตที่ถูกส่งออกมาจากตัวบริการวีดิทัศน์ทั้ง 3 กรณี ดังแสดงในรูปที่ 4.9 โดยที่อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 10 ต่อ 10 วินาที 10 ต่อ 240 วินาที และ 240 ต่อ 10 วินาที จะวัดอัตราแพ็กเกตที่เข้าสู่ตัววีดิทัศน์จริงเป็น 94 93 และ 98 แพ็กเกตต่อวินาที ตามลำดับ และ

เมื่อพิจารณาการสูญเสียแพ็กเกตที่ตัวเล่นวีดิทัศน์เทียบกับตัวบริการวีดิทัศน์ พบว่า ที่อัตราส่วน 10 ต่อ 10 วินาที เกิดการสูญเสียแพ็กเกต 1.88% ที่อัตราส่วน 10 ต่อ 240 วินาที เกิดการสูญเสียแพ็กเกต 6.06% และที่อัตราส่วน 240 ต่อ 10 วินาที เกิดการสูญเสียแพ็กเกต 0.04%

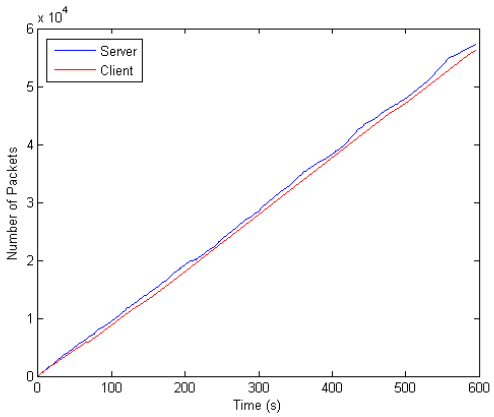
ตารางที่ 4.7: ค่าพารามิเตอร์ในการทดลองที่ 1

Parameter	Value
λ	100 packets/s
λ_1	96 packets/s
λ_2	4 packets/s
μ_1	188 packets/s
μ_2	95 packets/s
μ (python script)	100 packets/s
x	0 s
y	0.01 s
Chunk size ratio (s)	10:10, 10:240, 240:10

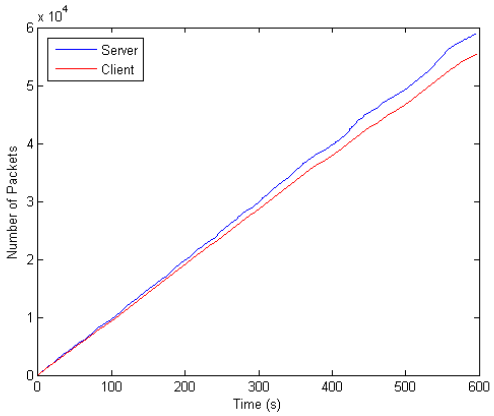
4.3.2 การทดลองที่ 2: μ = 150 แพ็กเกตต่อวินาที, x = 10 วินาที

ในการทดลองที่ 2 นี้ได้มีการเปลี่ยนอัตราการส่งแพ็กเกตในโปรแกรมไพทอนของมิดเดิลบอกซ์ให้ส่งออกไปที่ 150 แพ็กเกตต่อวินาที (μ = 150 แพ็กเกตต่อวินาที) และกำหนดให้มิดเดิลบอกซ์เก็บแพ็กเกตเข้าบัฟเฟอร์นานเป็นเวลา 10 วินาทีก่อนที่จะส่งแพ็กเกตออกไป (x = 10 วินาที) โดยรูปที่ 4.10 แสดงความสัมพันธ์ระหว่างจำนวนแพ็กเกตและเวลา เมื่อสตรีมวีดิทัศน์ผ่านระบบทดสอบที่มีอัตราส่วนเวลาของวิถีบนและวิถีล่างเช่นเดียวกับการทดลองที่ 1

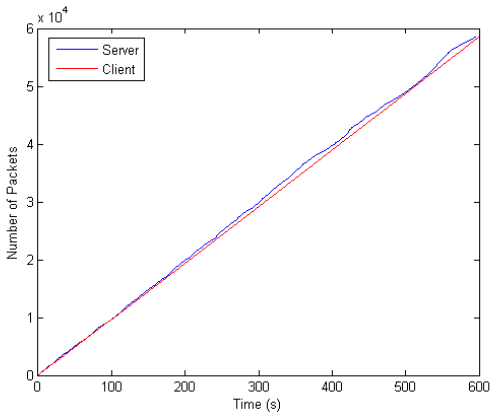
จากรูปที่ 4.10 จะเห็นได้ว่าอัตราแพ็กเกตที่เข้าสู่ตัวเล่นวีดิทัศน์มีค่าใกล้เคียงกับอัตราแพ็กเกตที่ส่งออกมาจากตัวบริการวีดิทัศน์ โดยเมื่อใช้อัตราส่วนเวลาในส่งแพ็กเกตผ่านวิถีบนต่อวิถีล่างเป็น 10 ต่อ 10 วินาที 10 ต่อ 240 วินาที และ 240 ต่อ 10 วินาที จะวัดอัตราแพ็กเกตที่เข้าสู่ตัวเล่นวีดิทัศน์โดยเฉลี่ยได้ 97 94 และ 99 แพ็กเกตต่อวินาทีตามลำดับ และมีการสูญเสียแพ็กเกตเท่ากับ 3.43% 6.54% และ 1.27% ตามลำดับ ในการทดลองนี้วีดิทัศน์ที่ถูกแสดงที่หน้าจอแสดงผลของตัวเล่นวีดิทัศน์มีความเรียบเนียนกว่าการทดลองที่ 1 แต่จะเกิดความผิดพลาด (error) ดังรูปที่ 4.11 ติดต่อกันเป็นระยะเวลาหนึ่งในช่วงแรกของการเล่นวีดิทัศน์



(a) อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 10:10 วินาที

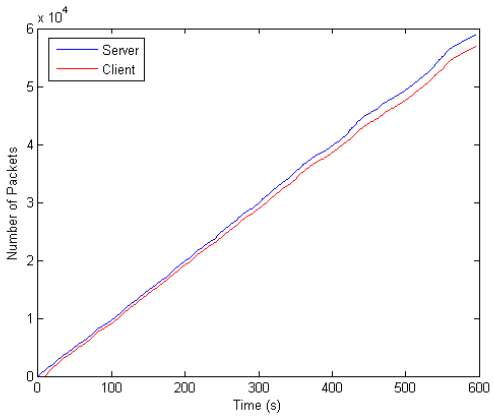


(b) อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 10:240 วินาที

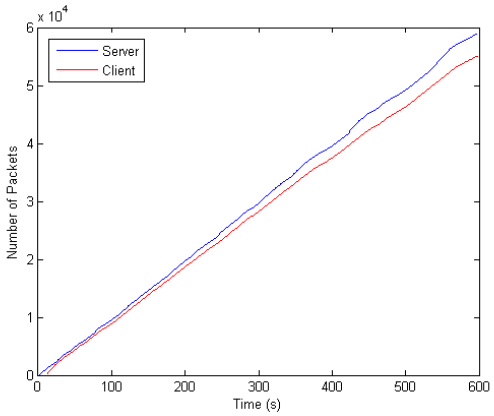


(c) อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 240:10 วินาที

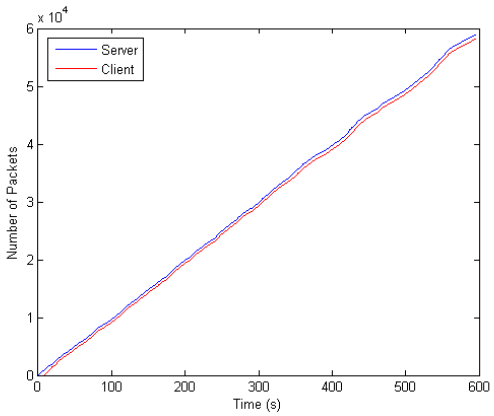
รูปที่ 4.9: ความสัมพันธ์ระหว่างจำนวนแพ็กเกตและเวลาในการทดลองที่ 1 (μ = 100 แพ็กเกตต่อวินาที, x = 0 วินาที)



(a) อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 10:10 วินาที

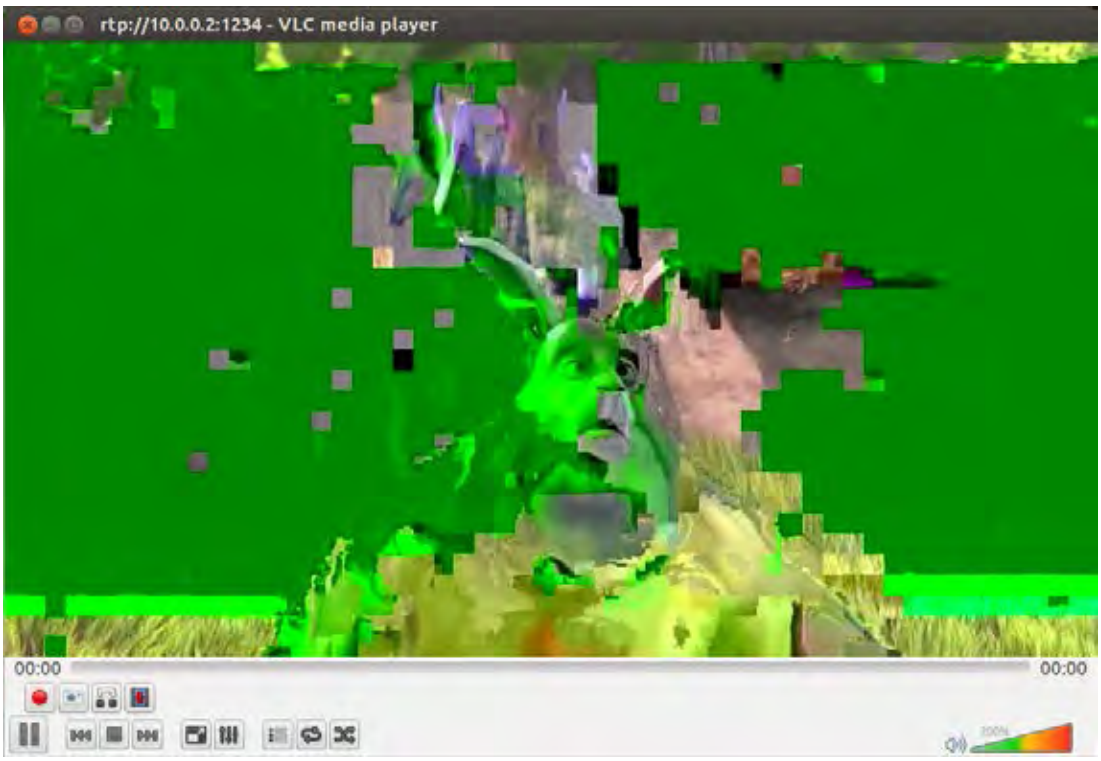


(b) อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 10:240 วินาที



(c) อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 240:10 วินาที

รูปที่ 4.10: ความสัมพันธ์ระหว่างจำนวนแพ็กเกตและเวลาในการทดลองที่ 2 (μ = 150 แพ็กเกตต่อวินาที, x = 10 วินาที)

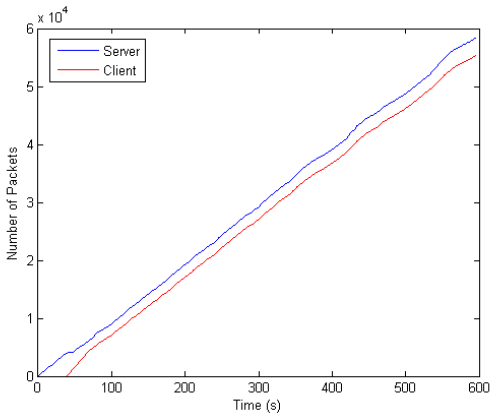


รูปที่ 4.11: ความผิดพลาดที่เกิดขึ้นในช่วงแรกของวีดิทัศน์ที่ถูกแสดง ณ ตัวเล่นวีดิทัศน์

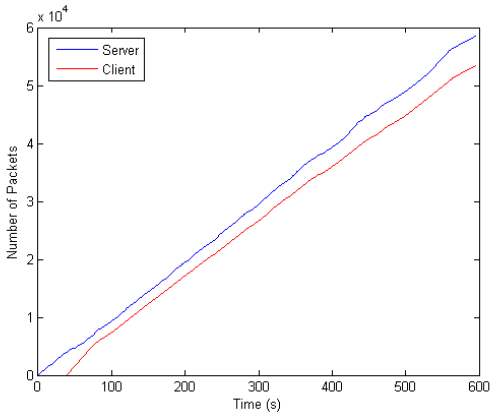
4.3.3 การทดลองที่ 3: μ = 150 แพ็กเกตต่อวินาที, x = 40 วินาที

ในการทดลองนี้จะกำหนดให้มิดเดิลบอกซ์เก็บแพ็กเกตเข้าบัฟเฟอร์เป็นเวลานานขึ้นเท่ากับ 40 วินาที (x = 40 วินาที) แต่ยังคงให้ส่งแพ็กเกตออกที่อัตราเท่ากับ 150 แพ็กเกตต่อวินาที (μ = 150 แพ็กเกตต่อวินาที) และเมื่อวัดอัตราแพ็กเกตที่ตัวเล่นวีดิทัศน์รับได้จริง พบว่า อัตราการรับแพ็กเกตของตัวเล่นวีดิทัศน์มีค่าเท่ากับ 99 96 และ 100 แพ็กเกตต่อวินาที และแพ็กเกตสูญเสียที่ตัวเล่นวีดิทัศน์มีค่าเท่ากับ 5.25% 8.79% และ 3.71% เมื่อใช้อัตราส่วนเวลาของแพ็กเกตที่ผ่านวิถีบนต่อวิถีล่างเป็น 10 ต่อ 10 วินาที 10 ต่อ 240 วินาที และ 240 ต่อ 10 วินาที ตามลำดับ และพบว่าวีดิทัศน์ที่ถูกแสดงที่ตัวเล่นวีดิทัศน์มีความผิดพลาดในช่วงแรกนานกว่าการทดลองที่ 2 โดยรูปที่ 4.12 แสดงความสัมพันธ์ระหว่างจำนวนแพ็กเกตและเวลาในแต่ละกรณี

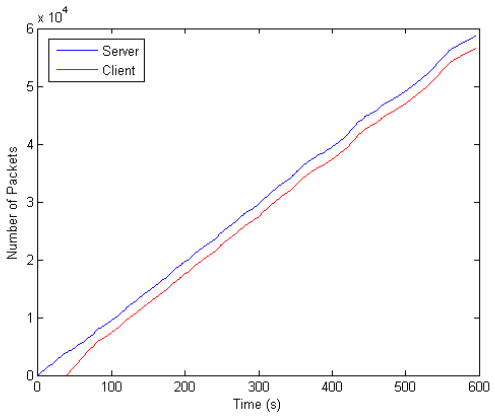
จากการทดลองทั้ง 3 การทดลองพบว่าข้างต้นพบว่า เมื่อใช้อัตราส่วนเวลาในการส่งแพ็กเกตผ่านวิถีบนต่อวิถีล่างเป็น 10 ต่อ 240 วินาที จะมีการสูญเสียแพ็กเกตที่ตัวเล่นวีดิทัศน์มากสุด เนื่องจากแพ็กเกตส่วนใหญ่ถูกส่งผ่านวิถีล่างซึ่งเป็นวิถีที่มีแบนด์วิดท์ต่ำ ทำให้แพ็กเกตสูญเสียไปเป็นจำนวนมาก ต่างจากกรณีที่ใช้อัตราส่วนเวลาเป็น 240 ต่อ 10 วินาที กรณีนี้แพ็กเกตเกือบทั้งหมดถูกส่งไปบนวิถีบนซึ่งสอดคล้องกับการคำนวณหาค่าอัตราแพ็กเกตที่ผ่านวิถีแต่ละวิถีจากสมการที่ (4.3) และ (4.4) คือ λ_1 = 96 แพ็กเกตต่อวินาที และ λ_2 = 4 แพ็กเกตต่อวินาที ซึ่งอัตราแพ็กเกตที่มาจากตัวบริการวีดิทัศน์เท่ากับ 100 แพ็กเกตต่อวินาที ดังนั้นจึงควรส่งแพ็กเกตไปบนวิถีบน ส่งผลให้กรณีนี้มีการสูญเสียแพ็กเกตน้อยที่สุดนั่นเอง โดยการสูญเสียแพ็กเกตที่เกิดขึ้นจากการทดลองแต่ละการทดลองสามารถแสดงได้ดังตารางที่ 4.10



(a) อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 10:10 วินาที



(b) อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 10:240 วินาที



(c) อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 240:10 วินาที

รูปที่ 4.12: ความสัมพันธ์ระหว่างจำนวนแพ็กเกตและเวลาในการทดลองที่ 3 (μ = 150 แพ็กเกตต่อวินาที, x = 40 วินาที)

ตารางที่ 4.8: การสูญเสียแพ็กเกตที่เกิดขึ้นในการทดลองที่ 1 2 และ 3

Packet loss (%)			
Experiment	Chunk size ratio		
	10:10	10:240	240:10
1	1.88	6.06	0.04
2	3.43	6.54	1.27
3	5.25	8.79	3.71

เมื่อวิเคราะห์ผลกระทบที่เกิดจากเวลาที่ใช้ในการเก็บแพ็กเกตเข้าบัฟเฟอร์ของมิดเดิลบอกซ์ พบว่า ยิ่งมิดเดิลบอกซ์ใช้เวลาเก็บแพ็กเกตนานเท่าใด วีดิทัศน์ที่ถูกแสดงในช่วงแรก ณ ตัวเล่นวีดิทัศน์จะยิ่งมีความผิดพลาดต่อเนื่องนานขึ้นไปด้วย ซึ่งในการทดลองที่ 1 มิดเดิลบอกซ์ส่งแพ็กเกตที่รับเข้ามาออกไปทันทีโดยไม่เก็บเข้าบัฟเฟอร์ พบว่า วีดิทัศน์ที่ถูกแสดงจะไม่มีความผิดพลาดในช่วงแรก แต่เมื่อให้มิดเดิลบอกซ์เพิ่มเวลาในการเก็บแพ็กเกตเข้าบัฟเฟอร์เป็น 10 และ 40 วินาที ดังการทดลองที่ 2 และ 3 วีดิทัศน์ที่ถูกแสดงจะมีความผิดพลาดต่อเนื่อง โดยการทดลองที่ 3 จะผิดพลาดนานกว่าการทดลองที่ 2 และเมื่อพิจารณาการสูญเสียที่เกิดขึ้นจะเห็นได้ว่า การสูญเสียจะเพิ่มขึ้นเมื่อมิดเดิลบอกซ์ใช้เวลาเก็บแพ็กเกตนานขึ้น เพราะก่อนที่ตัวบริการวีดิทัศน์จะเริ่มสตรีมวีดิทัศน์หรือส่งแพ็กเกตวีดิทัศน์แพ็กเกตแรกออกมานั้น ตัวบริการวีดิทัศน์จะส่งแพ็กเกตเออาร์พีไปยังตัวเล่นวีดิทัศน์ และตัวเล่นวีดิทัศน์จะตอบแพ็กเกตเออาร์พีกลับไปยังตัวบริการวีดิทัศน์เพื่อเป็นสัญญาณให้เริ่มส่งแพ็กเกตมาได้ และจะใช้แพ็กเกตเออาร์พีนั้นเป็นตัวประสานเวลา (synchronize) ระหว่างตัวบริการและตัวเล่นวีดิทัศน์ เมื่อตัวบริการวีดิทัศน์ส่งวีดิทัศน์ออกมา มิดเดิลบอกซ์จะเก็บแพ็กเกตไว้ในบัฟเฟอร์และรอจนกว่าจะครบเวลาจึงจะเริ่มส่งไปให้ตัวเล่นวีดิทัศน์อย่างต่อเนื่อง และเมื่อตัวเล่นวีดิทัศน์ได้รับแพ็กเกตวีดิทัศน์จากมิดเดิลบอกซ์ ตัวเล่นวีดิทัศน์จะนำแพ็กเกตที่ได้รับไปเปรียบเทียบกับแพ็กเกตเออาร์พี ซึ่งพบว่าแพ็กเกตที่ได้รับเข้ามามีเวลาไม่สอดคล้องกับแพ็กเกตเออาร์พีจึงทิ้งแพ็กเกตส่วนแรกบางส่วนไป ทำให้เกิดการสูญเสียและเกิดความผิดพลาดในช่วงแรกของการเล่นวีดิทัศน์ โดยการประมวลผลแพ็กเกตเพื่อใช้แสดงนั้นอยู่ในชั้นโปรแกรมประยุกต์และขึ้นอยู่กับการทำงานของโปรแกรมประยุกต์ที่ใช้ด้วย ซึ่งจะอยู่นอกเหนือขอบเขตของวิทยานิพนธ์นี้

4.3.4 การทดลองที่ 4: $\lambda_1 = 2\lambda_2$

จากการคำนวณหาอัตราแพ็กเกตที่ผ่านวิถีแต่ละวิถีโดยใช้สมการที่ (4.3) และ (4.4) พบว่าอัตราแพ็กเกตที่วิ่งผ่านวิถีบน (λ_1) มีค่ามากกว่าอัตราแพ็กเกตที่ผ่านวิถีล่าง (λ_2) มาก ๆ ซึ่งเมื่อนำอัตราส่วนของ λ_1 ต่อ λ_2 มาใช้เป็นเกณฑ์ในการแบ่งขนาดกลุ่มก้อนวีดิทัศน์หรือเวลาที่แพ็กเกตต้องวิ่งผ่านวิถีแต่ละวิถี จะพบว่า ที่อัตราส่วน 10 ต่อ 240 หรือ 240 ต่อ 10 วินาที จะมีแพ็กเกตวิ่งอยู่บนวิถีใดวิถีหนึ่งเป็นจำนวนมาก เสมือนกับเป็นการสตรีมวีดิทัศน์ผ่านวิถีเพียงวิถีเดียว ซึ่งไม่สามารถเห็นผลจากการแบ่งขนาดกลุ่มก้อนวีดิทัศน์อย่างชัดเจน ดังนั้นในการทดลองนี้จึงต้องการศึกษาผลกระทบของขนาดกลุ่มก้อนวีดิทัศน์ ที่มีต่อจำนวนแพ็กเกตในบัฟเฟอร์ของมิดเดิลบอกซ์และการสูญเสียแพ็กเกต เมื่อระบบทดสอบมีแบนด์วิดท์และอัตราแพ็กเกตที่ผ่านวิถีแต่ละวิถีใกล้เคียงกัน โดยได้ใช้โปรแกรม wondershaper ลดแบนด์วิดท์ของวิถีบน (μ_1) ให้เหลือ 1.42 เมกะบิตต่อวินาที ส่วนวิถีล่างยังคงมีแบนด์วิดท์ (μ_2) อยู่เท่ากับ 1.05 เมกะบิตต่อวินาที และจากสมการที่ (4.3) และ

(4.4) จะได้ λ_1 = 67 และ λ_2 = 33 แพ็กเกตต่อวินาที ซึ่ง $\lambda_1 = 2\lambda_2$ การทดลองนี้จะกำหนดให้มิดเดิลบอกซ์เก็บแพ็กเกตเป็นเวลานาน 60 วินาที และส่งแพ็กเกตออกไปด้วยอัตราเท่ากับ 150 แพ็กเกตต่อวินาที และพิจารณาการสตรีมวีดิทัศน์บนระบบทดสอบที่มีขนาดกลุ่มก้อนวีดิทัศน์ของวิถีบนต่อวิถีล่าง หรืออัตราส่วนเวลาในการส่งแพ็กเกตผ่านวิถีบนต่อวิถีล่างเป็น 30 ต่อ 10, 20 ต่อ 10, 10 ต่อ 10, 10 ต่อ 20 และ 10 ต่อ 30 วินาที ตามลำดับ ซึ่งพารามิเตอร์ที่ใช้ในการทดลองนี้สามารถแสดงได้ดังตารางที่ 4.9

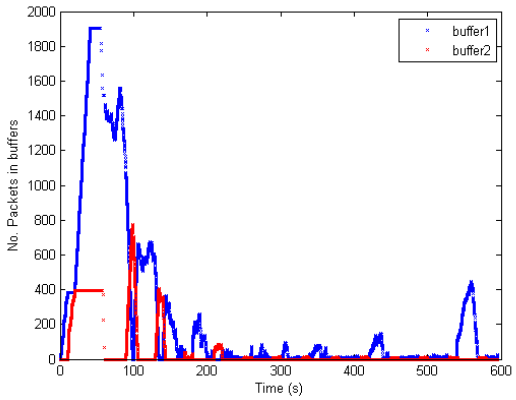
ตารางที่ 4.9: ค่าพารามิเตอร์ในการทดลองที่ 4

Parameter	Value
λ	100 packets/s
λ_1	67 packets/s
λ_2	33 packets/s
μ_1	130 packets/s
μ_2	95 packets/s
μ (python script)	150 packets/s
x	60 s
y	1/150 s
Chunk size ratio (s)	30:10, 20:10, 10:10, 10:20, 10:30

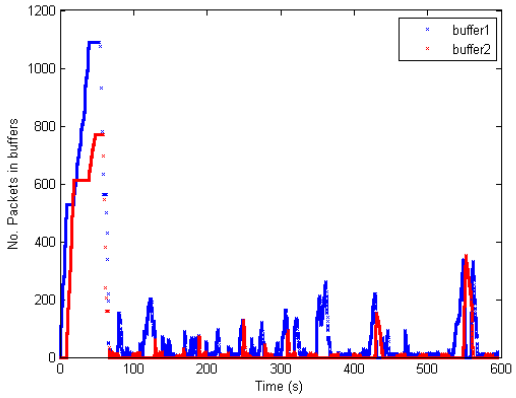
จำนวนแพ็กเกตที่อยู่ในบัฟเฟอร์ 1 และ 2 ของมิดเดิลบอกซ์เทียบกับเวลา เมื่อใช้อัตราส่วนขนาดกลุ่มก้อนวีดิทัศน์หรืออัตราส่วนเวลาในการส่งแพ็กเกตผ่านวิถีบนต่อวิถีล่างแบบต่าง ๆ สามารถแสดงได้ดังรูปที่ 4.13 และ 4.14 ซึ่งจะสังเกตได้ว่าทุกกรณีมีแนวโน้มเหมือนกันคือ เมื่อตัวบริการวีดิทัศน์เริ่มส่งแพ็กเกตออกมา มิดเดิลบอกซ์จะเก็บแพ็กเกตเข้าในบัฟเฟอร์ 1 และ 2 เป็นระยะเวลา 60 วินาที ทำให้จำนวนแพ็กเก็ตในบัฟเฟอร์เพิ่มขึ้นเรื่อย ๆ ซึ่งแพ็กเกตจะเริ่มถูกส่งมาจากวิถีใดนั้นจะขึ้นอยู่กับโฟลว์เอนทรีของโอเพนวีสวิตช์ 1 ที่ถูกกำหนดโดยพอกซ์ ณ เวลานั้น ๆ และหลังจาก 60 วินาที มิดเดิลบอกซ์จึงจะเริ่มปล่อยแพ็กเกตออกจากบัฟเฟอร์อย่างรวดเร็ว ตามอัตราการส่งออกแพ็กเกตที่ถูกกำหนดไว้ภายในโปรแกรมไพทอน จนกระทั่งแพ็กเกตในบัฟเฟอร์ใกล้หมดแพ็กเกตที่เข้ามาในบัฟเฟอร์ใหม่จะถูกส่งออกไปด้วยอัตราใกล้เคียงกับที่เข้ามา หรืออัตราที่ถูกส่งมาจากตัวบริการวีดิทัศน์ แต่ในบางช่วงหลังจาก 60 วินาทีแรกจะพบว่ามีแพ็กเกตในบัฟเฟอร์เป็นจำนวนมากทำให้เกิดเป็นช่วงกราฟที่กระโดดขึ้นมา ทั้งนี้เพราะในขณะนั้นการทำงานซีพียู (cpu utilization) ของมิดเดิลบอกซ์มีค่าสูง หรือแพ็กเกตเข้ามาในบัฟเฟอร์ด้วยอัตราที่สูงกว่าอัตราเฉลี่ย จึงทำให้มิดเดิลบอกซ์ส่งแพ็กเกตออกไปไม่ทัน แพ็กเกตจึงเหลืออยู่ในบัฟเฟอร์เป็นจำนวนมาก

วิทยานิพนธ์นี้ได้กำหนดอัตราการส่งออกแพ็กเกตของมิดเดิลบอกซ์ให้มีค่าสูงกว่าอัตราแพ็กเกตที่ถูกส่งออกจากตัวบริการวีดิทัศน์ เนื่องจากอัตราแพ็กเกตที่ถูกส่งออกมาจากตัวบริการวีดิทัศน์เป็นค่าโดยเฉลี่ยทั้งหมดจากแพ็กเกตแรกถึงแพ็กเกตสุดท้าย ซึ่งในทางปฏิบัติจริงตัวบริการวีดิทัศน์จะส่งวีดิทัศน์ออกมาเป็นเฟรม (frame) โดยในแต่ละเฟรมจะมีจำนวนแพ็กเกตไม่เท่ากันและอัตราที่ใช้ส่งเฟรมแต่ละเฟรมไม่เท่ากัน ทำให้เฟรมถูกส่งออกมาด้วยอัตราที่สูงกว่าและต่ำกว่าค่าเฉลี่ย ทั้งนี้ขึ้นอยู่กับการทำงานของโปรแกรมประยุกต์ที่ใช้เป็นตัวบริการวีดิทัศน์นั่นเอง อีกหนึ่งสาเหตุที่ต้องกำหนดให้อัตราการส่งออกแพ็กเกตของมิดเดิลบอกซ์มีค่าสูง เพราะหากใช้อัตราส่งออกแพ็กเกตที่ต่ำไปจะทำ

แพ็กเกตถูกส่งออกไปช้า ทำให้ตัวเล่นวีดิทัศน์ได้รับแพ็กเกตช้า ส่งผลให้วีดิทัศน์ที่ถูกแสดงมีความหน่วงมากและใช้เวลาเล่นนานกว่าระยะเวลาจริงของวีดิทัศน์

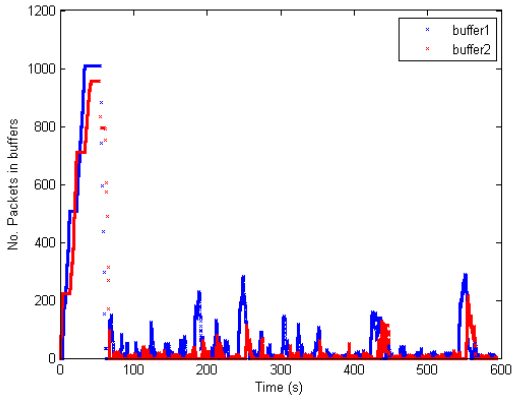


(a) อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 30:10 วินาที

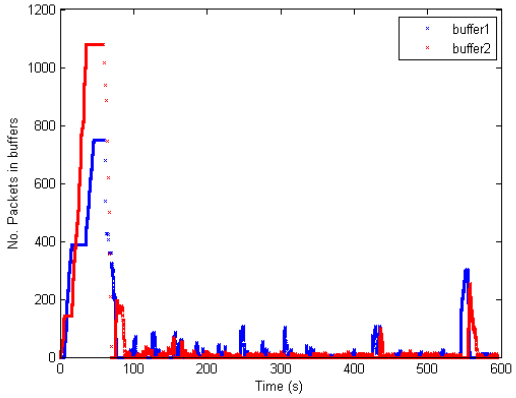


(b) อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 20:10 วินาที

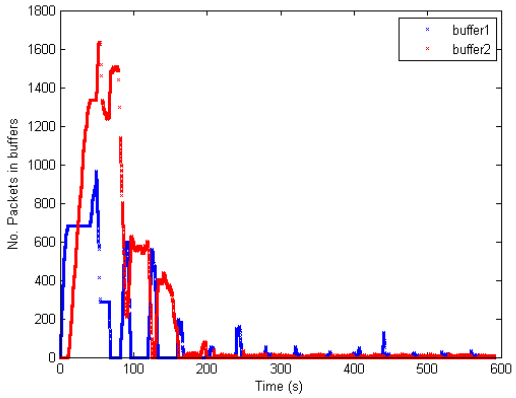
รูปที่ 4.13: ความสัมพันธ์ระหว่างจำนวนแพ็กเกตในบัฟเฟอร์ของมิดเดิลบอกซ์และเวลา



(a) อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 10:10 วินาที



(b) อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 10:20 วินาที



(c) อัตราส่วนเวลาของวิถีบนต่อวิถีล่างเท่ากับ 10:30 วินาที

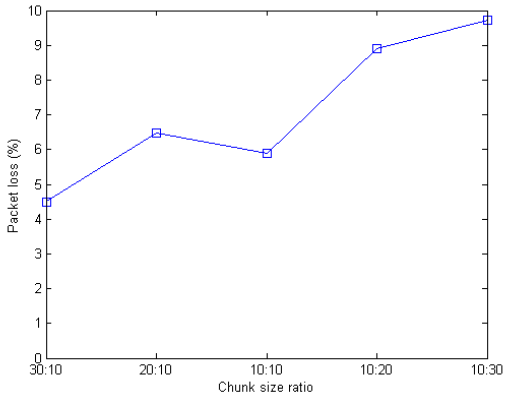
รูปที่ 4.14: ความสัมพันธ์ระหว่างจำนวนแพ็กเกตในบัฟเฟอร์ของมิดเดิลบอกซ์และเวลา (ต่อ)

การสูญเสียเสียงและวีดิทัศน์ที่ได้จากค่าสถิติของโปรแกรมวีแอลซี สามารถแสดงได้ดังรูปที่ 4.15 ซึ่งในวิทยานิพนธ์นี้จะคำนวณการสูญเสียเสียงจากเสียงที่สูญหาย (audio lost buffers) เทียบกับเสียงที่ถูกเล่น (audio played buffers) และคำนวณการสูญเสียวีดิทัศน์จากวีดิทัศน์ที่สูญหาย (video lost frames) เทียบกับวีดิทัศน์ที่ถูกเล่น (video displayed frames) และเมื่อเปรียบเทียบกับการสูญเสียแพ็กเกตที่เกิดขึ้น ณ ตัวเล่นวีดิทัศน์ จะแสดงได้ในตารางที่ 4.15 และรูปที่ 4.16

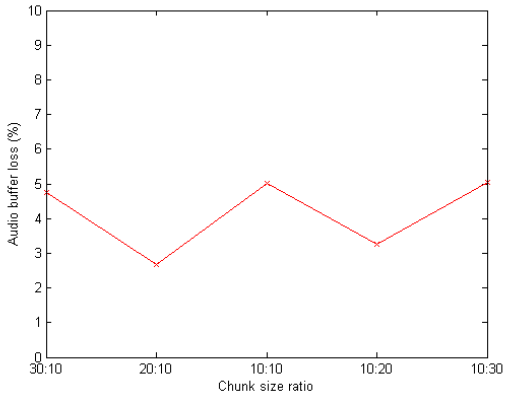
รูปที่ 4.15: ค่าสถิติแสดงการสูญเสียเสียงและวีดิทัศน์ของโปรแกรมวีแอลซี

ตารางที่ 4.10: การสูญเสียแพ็กเกต การสูญเสียเสียง และการสูญเสียวีดิทัศน์ในการทดลองที่ 4

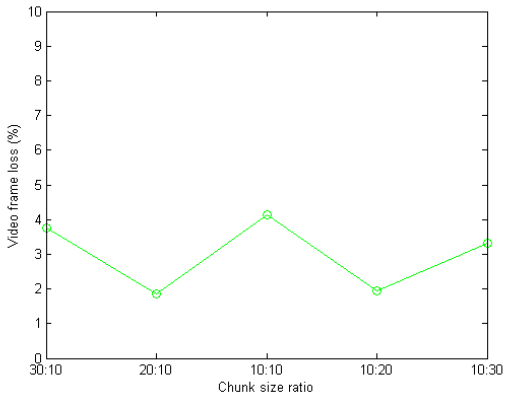
Chunk size ratio	Loss (%)		
	Packet	Audio	Video
30:10	4.50	4.76	3.75
20:10	6.48	2.67	1.87
10:10	5.90	5.00	4.15
10:20	8.89	3.25	1.95
10:30	9.71	5.04	3.71



(a) การสูญเสียแพ็กเกต



(b) การสูญเสียเสียง



(c) การสูญเสียวีดิทัศน์

รูปที่ 4.16: การสูญเสียแพ็กเกต การสูญเสียเสียงและการสูญเสียวีดิทัศน์ เมื่อใช้อัตราส่วนเวลาแบบต่าง ๆ

จากรูปที่ 4.16(a) จะเห็นว่า เมื่อใช้อัตราส่วนเวลาเป็น 30 ต่อ 10, 20 ต่อ 10 และ 10 ต่อ 10 วินาที การสูญเสียแพ็กเกตมีค่าใกล้เคียงกัน เพราะแพ็กเกตถูกส่งไปยังวิถีแต่ละวิถีด้วยขนาดที่เหมาะสม แต่เมื่อใช้อัตราส่วนเวลาเป็น 10 ต่อ 20 และ 10 ต่อ 30 วินาที อัตราการสูญเสียแพ็กเกตจะเพิ่มขึ้น เนื่องจากแพ็กเกตถูกส่งไปบนวิถีล่างมากกว่าวิถีบน อีกทั้งขนาดกลุ่มก้อนวีดิทัศน์ที่ใช้ไม่สอดคล้องกับอัตราส่วนของอัตราแพ็กเกตที่ถูกส่งผ่านวิถีแต่ละวิถี ($\lambda_1 = 2\lambda_2$) แต่เมื่อพิจารณาการสูญเสียเสียงและวีดิทัศน์ ดังรูปที่ 4.16(b) และ 4.16(c) พบว่ามีแนวโน้มไปทางเดียวกัน ทั้งนี้การคำนวณการสูญเสียเสียงและวีดิทัศน์จะขึ้นอยู่กับการทำงานของโปรแกรมวีแอลซี

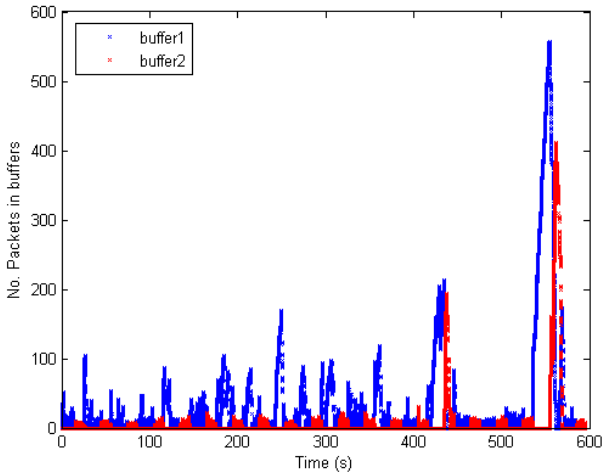
เมื่อทำการทดลองสตรีมวีดิทัศน์ โดยใช้อัตราส่วนขนาดกลุ่มก้อนวีดิทัศน์หรืออัตราส่วนเวลาเป็น 20 ต่อ 10 วินาที ให้สอดคล้องกับอัตราแพ็กเกตที่ผ่านวิถีแต่ละวิถี $\lambda_1 = 2\lambda_2$ โดยพิจารณาเวลาที่มิดเดิลบอกซ์ใช้เก็บแพ็กเกตเข้าบัฟเฟอร์ (x) เท่ากับ 0 30 60 และ 90 วินาทีตามลำดับ จะสามารถแสดงความสัมพันธ์ของจำนวนแพ็กเกตในบัฟเฟอร์และเวลาได้ดังรูปที่ 4.17 และ 4.18

จากรูปที่ 4.17(b) 4.18(a) และ 4.18(b) จะเห็นได้ว่ามีแนวโน้มเหมือนรูปที่ 4.13 คือแพ็กเกตจะถูกเก็บเข้าบัฟเฟอร์ของมิดเดิลบอกซ์เป็นจำนวนมาก จนกระทั่งครบเวลาที่กำหนดมิดเดิลบอกซ์จะส่งแพ็กเกตออกไปอย่างรวดเร็ว และเมื่อแพ็กเกตในบัฟเฟอร์ใกล้จะหมด มิดเดิลบอกซ์จะส่งแพ็กเกตออกไปด้วยอัตราที่ใกล้เคียงกับอัตราแพ็กเกตที่เข้ามา แต่ในรูปที่ 4.17(a) เป็นกรณีที่มิดเดิลบอกซ์ไม่เก็บแพ็กเกตเข้าบัฟเฟอร์ แต่จะปล่อยแพ็กเกตออกไปทันที จึงให้แพ็กเกตที่อยู่ในบัฟเฟอร์มีจำนวนใกล้เคียงกันตลอดระยะเวลาการสตรีม แต่จะมีบางช่วงที่มีแพ็กเกตเป็นจำนวนมาก ทั้งนี้เนื่องจากแพ็กเกตถูกส่งออกมาด้วยอัตราที่สูง และมิดเดิลเดิลบอกซ์ไม่สามารถประมวลผลทันเพราะมีการทำงานซีพียูสูงดังที่ได้กล่าวไว้ข้างต้นนั่นเอง และเมื่อพิจารณาการสูญเสียแพ็กเกต การสูญเสียเสียงและวีดิทัศน์จากโปรแกรมวีแอลซี พบว่า ยิ่งมิดเดิลบอกซ์ใช้เวลาในการเก็บแพ็กเกตเข้าบัฟเฟอร์มากเท่าใด การสูญเสียก็จะเพิ่มขึ้นตามไปด้วย โดยการสูญเสียที่เกิดขึ้นสามารถแสดงได้ดังตารางที่ 4.11 และรูปที่ 4.18

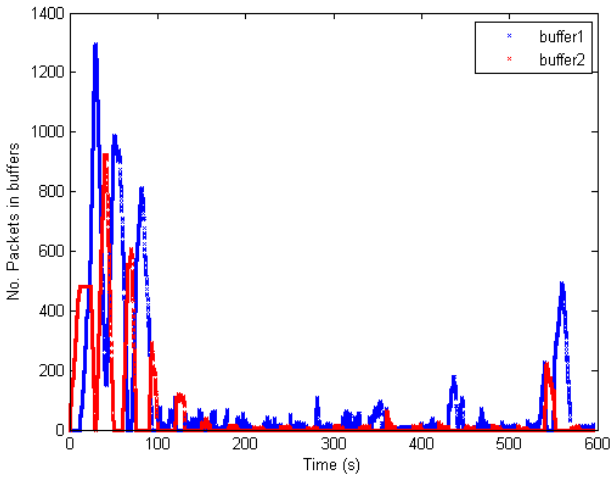
เนื่องจากในการทดลองนี้ยังไม่สามารถควบคุมการแสดงวีดิทัศน์ของตัวเล่นวีดิทัศน์ได้ แต่หากในอนาคตสามารถควบคุมจังหวะการเล่นวีดิทัศน์ของโปรแกรมประยุกต์ที่ใช้ ณ ตัวเล่นวีดิทัศน์ได้แล้ว การเพิ่มค่า x หรือเวลาในการเก็บแพ็กเกตเข้าบัฟเฟอร์ของมิดเดิลบอกซ์ น่าจะส่งผลในทางบวกต่อสมรรถนะของระบบทดสอบ ซึ่งเป็นงานที่น่าสนใจศึกษาต่อในอนาคต

ตารางที่ 4.11: การสูญเสียแพ็กเกต การสูญเสียเสียง และการสูญเสียวีดิทัศน์ เมื่อใช้อัตราส่วนเวลา 20 ต่อ 10 วินาที

x (s)	Loss (%)		
	Packet	Audio	Video
0	1.44	1.60	1.27
30	4.71	2.67	1.87
60	6.48	3.51	2.45
90	5.90	3.91	2.67

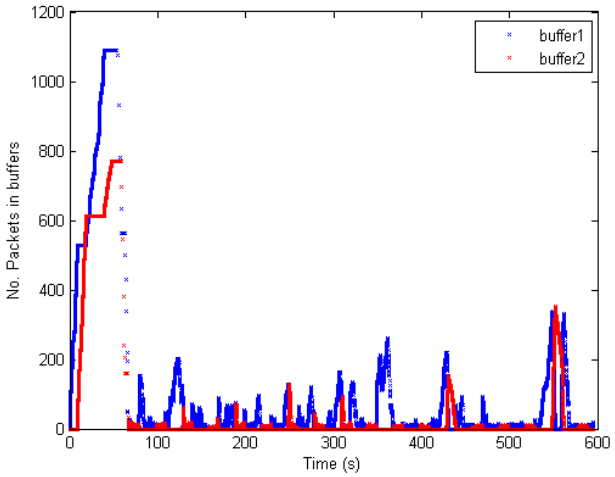


(a) x = 0 วินาที

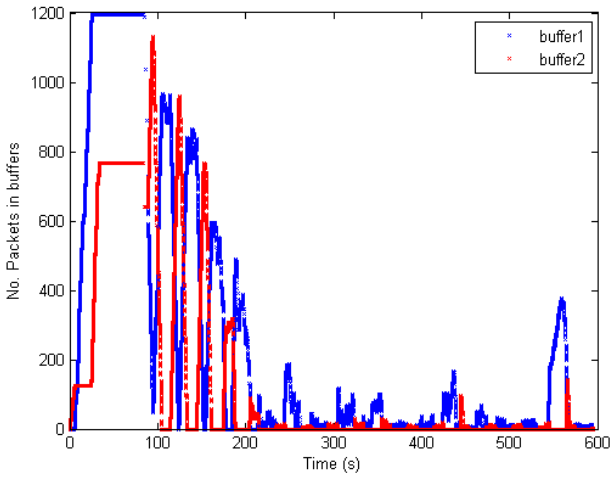


(b) x = 30 วินาที

รูปที่ 4.17: ความสัมพันธ์ระหว่างจำนวนแพ็กเกตในบัฟเฟอร์ของมิดเดิลบอกซ์และเวลา เมื่อใช้อัตราส่วนเวลาเป็น 20 ต่อ 10 วินาที

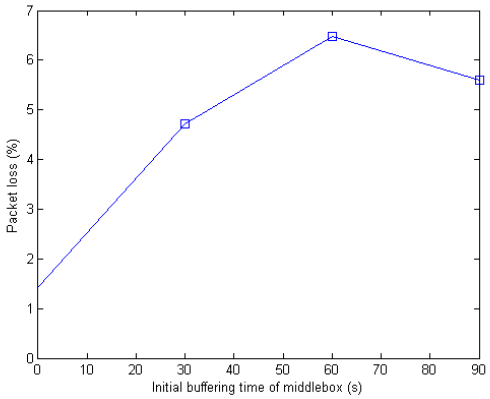


(a) x = 60 วินาที

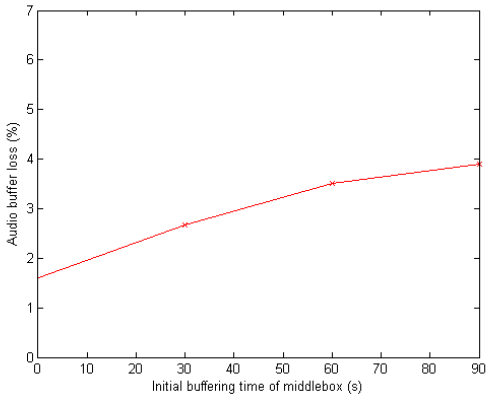


(b) x = 90 วินาที

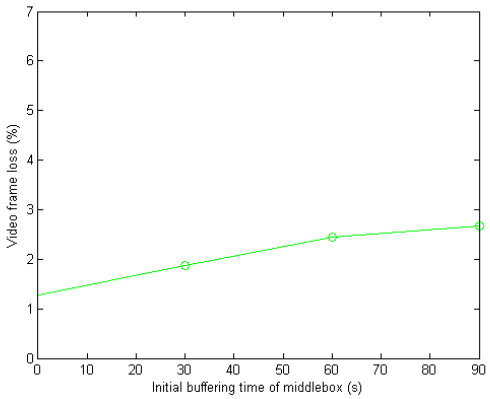
รูปที่ 4.18: ความสัมพันธ์ระหว่างจำนวนแพ็กเกตในบัฟเฟอร์ของมิดเดิลบอกซ์และเวลา เมื่อใช้อัตราส่วนเวลาเป็น 20 ต่อ 10 วินาที (ต่อ)



(a) การสูญเสียแพ็กเกต



(b) การสูญเสียเสียง



(c) การสูญเสียวีดิทัศน์

รูปที่ 4.18: การสูญเสียแพ็กเกต การสูญเสียเสียงและการสูญเสียวีดิทัศน์ เมื่อใช้อัตราส่วนเวลา 20 ต่อ 10 วินาที

บทที่ 5

บทสรุปและข้อเสนอแนะ

5.1 บทสรุป

วิทยานิพนธ์นี้ได้ศึกษาเปรียบเทียบสมรรถนะของโครงข่ายโอเพนโฟลว์ที่ถูกควบคุมด้วยตัวควบคุมเอสดีเอ็นในการรับ-ส่งลำดับของแพ็กเกตวีดิทัศน์ เมื่อสตรีมวีดิทัศน์แบบหนึ่งวิถีและสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถีบนระบบทดสอบโอเพนโฟลว์ขนาดเล็ก ในห้องปฏิบัติการวิจัยโทรคมนาคม อาคาร 4 ชั้น 13 ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย และเพื่อทดสอบการทำงานของระบบทดสอบโอเพนโฟลว์ ดังนั้นวิทยานิพนธ์นี้จึงได้ทำการทดลองบนโครงข่ายโอเพนโฟลว์เสมือนและระบบทดสอบโอเพนโฟลว์จริง โดยจากผลการทดลองสามารถสรุปได้ดังนี้

- **บทที่ 3 การทดลองบนโครงข่ายโอเพนโฟลว์ด้วยโปรแกรมมินิเน็ต**

ในบทที่ 3 เป็นการทดลองบนโครงข่ายโอเพนโฟลว์เสมือน วิทยานิพนธ์ฉบับนี้ได้จำลองโครงข่ายโอเพนโฟลว์เสมือนด้วยโปรแกรมมินิเน็ตรุ่น 2.1.0 ซึ่งในโครงข่ายเสมือนจะประกอบด้วย ตัวบริการวีดิทัศน์ ตัวเล่นวีดิทัศน์ โอเพนวีสวิตช์ 4 ตัว และตัวควบคุมพอกซ์ ซึ่งโครงข่ายเสมือนนี้ใช้เพื่อศึกษาการรับ-ส่งแพ็กเกตทีซีพีและยูดีพี และการสตรีมวีดีทัศน์แบบพหุวิถี โดยที่แพ็กเกตวีดิทัศน์ทั้งหมดจะถูกส่งผ่านวิถี 2 ทั้งวิถีอย่างพร้อม ๆ กัน นอกจากนี้ยังแสดงการเปรียบเทียบสมรรถนะของการสตรีมวีดิทัศน์บนโครงข่ายโอเพนโฟลว์เสมือนและระบบทดสอบโอเพนโฟลว์จริง ระบบทดสอบจริงนั้นถูกสร้างขึ้นจากเครื่องคอมพิวเตอร์ระบบปฏิบัติการลินุกซ์ Ubuntu 12.04 จำนวน 8 เครื่อง โดยเครื่องคอมพิวเตอร์ 2 เครื่องที่มีโปรแกรมวีแอลซีรุ่น 2.0.9 จะถูกใช้เป็นตัวบริการวีดิทัศน์ และตัวเล่นวีดิทัศน์ คอมพิวเตอร์จำนวน 4 ตัวที่จำลองเป็นโอเพนวีสวิตช์จะใช้โอเพนวีสวิตช์รุ่น 1.9.3 คอมพิวเตอร์ 1 ตัวที่ถูกจำลองเป็นคัวควบคุมเอสดีเอ็นจะใช้พอกซ์รุ่น 0.3.0 และได้เพิ่มเครื่องคอมพิวเตอร์อีก 1 ตัวเพื่อใช้เป็นตัวควบคุมระยะไกลเพื่อใช้สำหรับเฝ้าสังเกตการทำงานและสั่งการตัวบริการและตัวเล่นวีดิทัศน์ โครงข่ายโอเพนโฟลว์เสมือนและระบบทดสอบจริงจะมีทอพอโลยีเหมือนกัน แต่โครงข่ายเสมือนจะไม่มีตัวควบคุมระยะไกล ในการทดลองเปรียบเทียบสมรรถนะของโครงข่ายเสมือนและระบบทดสอบจริง วิทยานิพนธ์ นี้ได้ใช้คำสั่งในโปรแกรมมินิเน็ตปรับตั้งค่าแบนด์วิดท์และความหน่วงระหว่างตัวบริการวีดิทัศน์และตัวเล่นวีดิทัศน์ของวิถีทั้ง 2 วิถีบนโครงข่ายเสมือนให้มีค่าเท่ากับระบบทดสอบจริง โดยวิถีบนจะมีแบนด์วิดท์และความหน่วงเท่ากับ 6.5 เมกะบิตต่อวินาที และ 1.5 มิลลิวินาที ตามลำดับ และวิถีล่างมีแบนด์วิดท์และความหน่วงเท่ากับ 2.5 เมกะบิตต่อวินาที และ 2 มิลลิวินาที ตามลำดับ เมื่อสตรีมวีดิทัศน์โดยใช้โพรโตคอลทีซีพีจะพบว่าไม่มีการสูญเสียแพ็กเกตเกิด ขึ้นทั้งบนโครงข่ายเสมือนและระบบทดสอบจริง เนื่องจากในโพรโตคอลทีซีพีจะมีการส่งแพ็กเกตที่สูญหายใหม่ เมื่อมีการสูญเสียแพ็กเกตเกิดขึ้นตัวเล่นวีดิทัศน์จะไม่ ได้รับแพ็กเกตนั้นจึงไม่ส่งการตอบกลับ (acknowledgement) ไปยังตัวบริการวีดิทัศน์ เมื่อตัวบริการวีดิทัศน์ไม่ได้รับการตอบกลับมาภายในเวลาที่ กำหนดจึงส่งแพ็กเกตนั้นไปให้ตัวเล่นวีดิทัศน์ใหม่ ซึ่งเมื่อตรวจดูแพ็กเกตที่ถูกจับไว้ด้วยโปรแกรมไวร์ชาร์กจะพบว่าตัวบริการวีดิทัศน์ ในระบบทดสอบส่งแพ็กเกตซ้ำไปยังตัวบริการวีดิทัศน์เป็นจำนวนมากกว่าตัวบริการวีดิทัศน์ในโครงข่ายเสมือน แสดงให้เห็นว่าในระบบทดสอบเกิดการสูญเสียแพ็กเกตมากกว่านั่นเอง และเมื่อสตรีมวีดิทัศน์โดยใช้โพรโตคอลยูดีพีซึ่งเป็นโพรโตคอลที่ไม่รองรับการควบคุมการไหล จะเห็นได้ชัดเจน ว่าบนระบบทดสอบจริงเกิดการสูญเสียแพ็กเกตมาก

กว่าโครงข่ายเสมือน ทั้งนี้เนื่องจากข้อจำกัดทางฮาร์ดแวร์ของอุปกรณ์ที่ใช้ในระบบทดสอบที่ส่งผลให้สมรรถนะของระบบทดสอบจริงต่ำกว่าสมรรถนะที่วัดได้ในโครงข่ายโอเพนโฟลว์เสมือนที่ถูกจำลองด้วยโปรแกรมมินิเน็ต

- **บทที่ 4 การสตรีมกลุ่มก้อนวีดิทัศน์แบบพหุวิถีบนระบบทดสอบโอเพนโฟลว์ในห้องปฏิบัติการ**

วิทยานิพนธ์ฉบับนี้ ได้สร้างระบบทดสอบโอเพนโฟลว์เพื่อใช้ทดสอบการสตรีมวีดิทัศน์เป็นกลุ่มก้อนบนวีถี 2 วิถี จากตัวบริการวีดิทัศน์ 1 ตัว ไปยังตัวเล่นวีดิทัศน์ 1 ตัว โดยระบบทดสอบที่สร้างขึ้นในบทที่ 4 เป็นระบบทดสอบเดียวกับระบบทดสอบในบทที่ 3 ซึ่งประกอบไปด้วย ตัวบริการวีดิทัศน์ ตัวเล่นวีดิทัศน์ โอเพนวีสวิตช์ 4 ตัว และตัวประสานเอสดีเอ็น (ประกอบด้วยตัวควบคุมเอสดีเอ็น หรือพอกซ์ และตัวควบคุมระยะไกล) แต่ระบบทดสอบที่ใช้ทดลองในบทที่ 4 ได้ถูกเพิ่มเครื่องคอมพิวเตอร์ที่จำลองเป็นมิดเดิลบอกซ์เข้าไปเพื่อใช้ในการรวมแพ็กเกตจากวิถี 2 วิถีเข้าด้วยกัน

ในระบบทดสอบนี้ ตัวควบคุมพอกซ์จะทำงานตามโปรแกรมไพทอนที่ถูกพัฒนาขึ้นมา เพื่อสั่งการผ่านโพรโตคอลโอเพนโฟลว์ไปยังโอเพนวีสวิตช์ ให้ติดตั้งโฟลว์เอนทรีสำหรับใช้แบ่งขนาดกลุ่มก้อนวีดิทัศน์ตามเวลา โดยการแบ่งกลุ่มก้อนวีดิทัศน์จะมีความสัมพันธ์กับอัตราแพ็กเกตที่วิ่งผ่านวิถีแต่ละวิถี ซึ่งวิทยานิพนธ์นี้ได้อ้างอิงระบบแถวคอย M/M/1 เพื่อใช้ในการจัดสมดุลความหน่วงให้กับทราฟฟิกวีดิทัศน์บนวิถีแต่ละวิถีของระบบทดสอบ ซึ่งจากผลการทดลองแสดงได้ว่าพอกซ์สามารถสั่งการให้โอเพนวีสวิตช์แบ่งวีดิทัศน์ให้เป็นกลุ่มก้อนตามเวลาแล้วส่งไปยังวิถีแต่ละวิถีได้

นอกจากนี้ยังได้พัฒนามิดเดิลบอกซ์เพื่อใช้ในการรวมแพ็กเกตจากวิถีทั้ง 2 วิถีเข้าด้วยกัน ซึ่งมิดเดิลบอกซ์จะเชื่อมต่อกับโอเพนวีสวิตช์ปลายทาง (โอเพนวีสวิตช์ 3) ที่รับแพ็กเกตวีดิทัศน์มาจากวิถี 2 วิถี มิดเดิลบอกซ์ถูกพัฒนาขึ้นโดยภาษาไพทอนและใช้มอดูล pcapy เพื่อรับแพ็กเกตที่เข้ามาและใช้มอดูล dpkt ในการแยกแพ็กเกตจากวิถี 2 วิถีออกจากกัน นอกจากนี้ยังมีการพัฒนาโปรแกรมจัดกำหนดการอย่างง่ายในการส่งแพ็กเกตออกไปให้ตัวเล่นวีดิทัศน์ ซึ่งโปรแกรมจะเลือกส่งแพ็กเกตส่วนต้นของบัฟเฟอร์โดยพิจารณาจากตราเวลาของแพ็กเกตที่ถูกกำหนดโดยตัวบริการวีดิทัศน์ และใช้มอดูล scapy สำหรับห่อและส่งแพ็กเกตกลับไปยังโอเพนวีสวิตช์ 3 เพื่อให้ส่งแพ็กเกตต่อไปยังตัวบริการวีดิทัศน์

เมื่อทดสอบสมรรถนะโดยรวมของระบบทดสอบโอเพนโฟลว์ในบทที่ 4 พบว่าวิถีบนและวิถีล่างมีแบนด์วิดท์เป็น 6.71 และ 3.93 เมกะบิตต่อวินาทีซึ่งเพียงพอต่อการรองรับการสตรีมวีดิทัศน์ที่อัตราการเข้ารหัส 3 เมกะบิตต่อวินาที แต่เนื่องจากข้อจำกัดทางฮาร์ดแวร์ของมิดเดิลบอกซ์ทำให้ไม่สามารถประมวลผลแพ็กเกตจำนวนมากได้ทัน วิทยานิพนธ์นี้จึงทดสอบการสตรีมวีดิทัศน์ด้วยอัตราการเข้ารหัส 500 กิโลบิตต่อวินาที ซึ่งจะวัดอัตราที่ตัวบริการวีดิทัศน์ส่งแพ็กเกตออกมาจริงได้เท่ากับ 1.10 เมกะบิตต่อวินาที ดังนั้นจึงได้ใช้โปรแกรม wondershaper เพื่อปรับลดแบนด์วิดท์ของวิถีบนให้เหลือ 2.07 เมกะบิตต่อวินาที และลดแบนด์วิดท์ของวิถีล่างเป็น 1.05 เมกะบิตต่อวินาที เมื่อพิจารณาการสตรีมวีดิทัศน์จากตัวบริการวีดิทัศน์ไปยังวิถีล่างซึ่งเป็นวิถีที่มีแบนด์วิดท์ต่ำ เปรียบเทียบกับการสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถีที่มีขนาดกลุ่มก้อนวีดิทัศน์หรืออัตราส่วนเวลาของแพ็กเกตที่ผ่านวิถีแต่ละวิถีแบบต่าง ๆ พบว่า การสตรีมวีดิทัศน์แบบพหุวิถีมีสมรรถนะดีกว่าการสตรีมแบบ 1 วิถี โดยการเลือกใช้ขนาดกลุ่มก้อนวีดิทัศน์ที่เหมาะสม คือ กลุ่มก้อนวีดิทัศน์ที่ถูกส่งไปวิถีบนมีขนาดใหญ่กว่ากลุ่มก้อนวีดิทัศน์ที่ถูกส่งไปวิถีล่างจะทำให้การสูญเสียแพ็กเกตที่เกิดขึ้น ณ ตัวเล่นวีดิทัศน์มีค่าลดลง และเมื่อพิจารณาเวลาที่มิดเดิลบอกซ์ใช้เก็บแพ็กเกตเข้าบัฟเฟอร์ก่อนส่งไปให้ตัวเล่นวีดิทัศน์บนระบบทดสอบที่มีแบนด์วิดท์ของวิถี 2 วิถีใกล้เคียงกัน นั่นคือวิถีบนและวิถีล่างมีแบนด์วิดท์เป็น 1.42 และ 1.05 เมกะบิตต่อวินาทีตามลำดับ พบว่า ยิ่งเวลาที่ใช้เก็บแพ็กเกตมากขึ้นการสูญเสียแพ็กเกต การ

สูญเสียเสียงและการสูญเวียวีดิทัศน์ก็จะเพิ่มขึ้นตามไปด้วย

วิทยานิพนธ์ฉบับนี้ ได้นำข้อดีของการสตรีมวีดิทัศน์แบบพหุวิถีและการแบ่งกลุ่มก้อนวีดิทัศน์มาปรับใช้ร่วมกัน โดยขนาดของกลุ่มก้อนวีดิทัศน์จะขึ้นอยู่กับเวลาที่ใช้ในการส่งแพ็กเกตวีดิทัศน์ไปบนวิถีแต่ละวิถีซึ่งถูกกำหนดด้วยตัวควบคุมเอสดีเอ็น ขนาดกลุ่มก้อนวีดิทัศน์ที่เหมาะสม ย่อมส่งผลต่อการสตรีมวีดิทัศน์แบบพหุวิถีที่มีประสิทธิภาพ อีกทั้งยังมีปัจจัยอื่น ๆ อันได้แก่ อัตราแพ็กเกตที่เข้าสู่ระบบทดสอบ อัตราแพ็กเกตที่เข้าสู่ตัวเล่นวีดิทัศน์ รวมถึงข้อจำกัดของอุปกรณ์ที่ใช้ในระบบทดสอบ ที่ส่งผลกระทบต่อการสตรีมวีดิทัศน์ ดังนั้นจึงจำเป็นอย่างยิ่งที่จะต้องพิจารณาพารามิเตอร์เหล่านี้ในการทดลองบนระบบทดสอบโอเพนโฟลว์จริง

5.2 ข้อเสนอแนะ

หัวข้อที่ควรศึกษาและวิจัยต่อไปในอนาคตคือ

1. การทดลองบนระบบทดสอบ OF@TEIN

   ในวิทยานิพนธ์ฉบับนี้ ได้นำเสนอการสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถีบนระบบทดสอบโอเพนโฟลว์ขนาดเล็กในห้องปฏิบัติการ ซึ่งถือว่าเป็นระบบทดสอบที่เล็กมากเมื่อเทียบระบบทดสอบ OF@TEIN ที่เป็นระบบทดสอบโอเพนโฟลว์ขนาดใหญ่ระหว่างประเทศ ดังนั้นจึงควรทำการทดลองกับระบบทดสอบ OF@TEIN เพื่อเป็นการทดสอบการใช้งานโครงข่ายโอเพนโฟลว์จริง ซึ่งจะนำไปสู่การใช้งานโครงข่ายโอเพนโฟลว์ในอนาคตได้ โดยรายละเอียดของระบบทดสอบ OF@TEIN และแนวทางการทดลองมีดังนี้

   TEIN (trans-Eurasia information network) เป็นโครงข่ายเพื่อการศึกษาและการวิจัยในกลุ่มประเทศเอเชียตะวันออกเฉียงใต้ โดย OF@TEIN เป็นระบบทดสอบโอเพนโฟลว์ที่อยู่บนโครงข่าย TEIN มีจุดประสงค์เพื่อทดลองใช้โครงข่ายโอเพนโฟลว์ระหว่างสาธารณรัฐเกาหลีใต้และกลุ่มประเทศเอเชียตะวันออกเฉียงใต้ สำหรับในประเทศไทยนั้น ระบบทดสอบ OF@TEIN เป็นโครงการความร่วมมือของกลุ่มวิจัยโครงข่าย ห้องปฏิบัติการวิจัยโทรคมนาคม ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย และเครือข่ายการวิจัยโอเพนโฟลว์ระหว่างประเทศ

   ในการทดลองนี้จะทดสอบการสตรีมวีดิทัศน์เป็นกลุ่มก้อนจากสาธารณรัฐเกาหลีใต้ มายังประเทศไทย ผ่านระบบทดสอบ OF@TEIN ซึ่งตัวบริการวีดิทัศน์จะอยู่ที่สถาบันวิทยาศาสตร์และเทคโนโลยีเมืองกวางจู (Gwangju institute of science and technology : GIST) ส่วนตัวเล่นวีดิทัศน์จะอยู่ที่ห้องโครงข่าย อาคารจามจุรี 9 จุฬาลงกรณ์มหาวิทยาลัย ดังรูปที่ 5.1 และตัวบริการวีดิทัศน์และตัวเล่นวีดิทัศน์จะเชื่อมต่อกันผ่านระบบทดสอบ OF@TEIN ดังแสดงในรูปที่ 5.2

   ประเทศที่อยู่บนระบบทดสอบ OF@TEIN ทุกประเทศ ได้แก่ สาธารณรัฐเกาหลีใต้ ไทย มาเลเซีย เวียดนาม อินโดนีเซีย และฟิลิปปินส์ จะเสมือนเป็นสถานีเชื่อมโยงโอเพนโฟลว์ (OpenFlow node) ซึ่งทุกประเทศได้ติดตั้งสมาร์ทเอ็กซ์ แร็ก (SmartX rack) ดังรูปที่ 5.3 โดยประกอบด้วยสวิตช์โอเพนโฟลว์ที่เชื่อมต่อกับเครื่องเสมือน (virtual machine) เพื่อใช้สำหรับทดลองงานวิจัยในระบบทดสอบ OF@TEIN ซึ่งการทดลองนี้จะใช้เครื่องเสมือนจำลองเป็นตัวบริการวีดิทัศน์และตัวเล่นวีดิทัศน์ โดยตัวบริการวีดิทัศน์ ณ สาธารณรัฐเกาหลีใต้

รูปที่ 5.1: ห้องโครงข่าย อาคารจามจุรี 9 จุฬาลงกรณ์มหาวิทยาลัย

รูปที่ 5.2: การทดลองบนระบบทดสอบ OF@TEIN

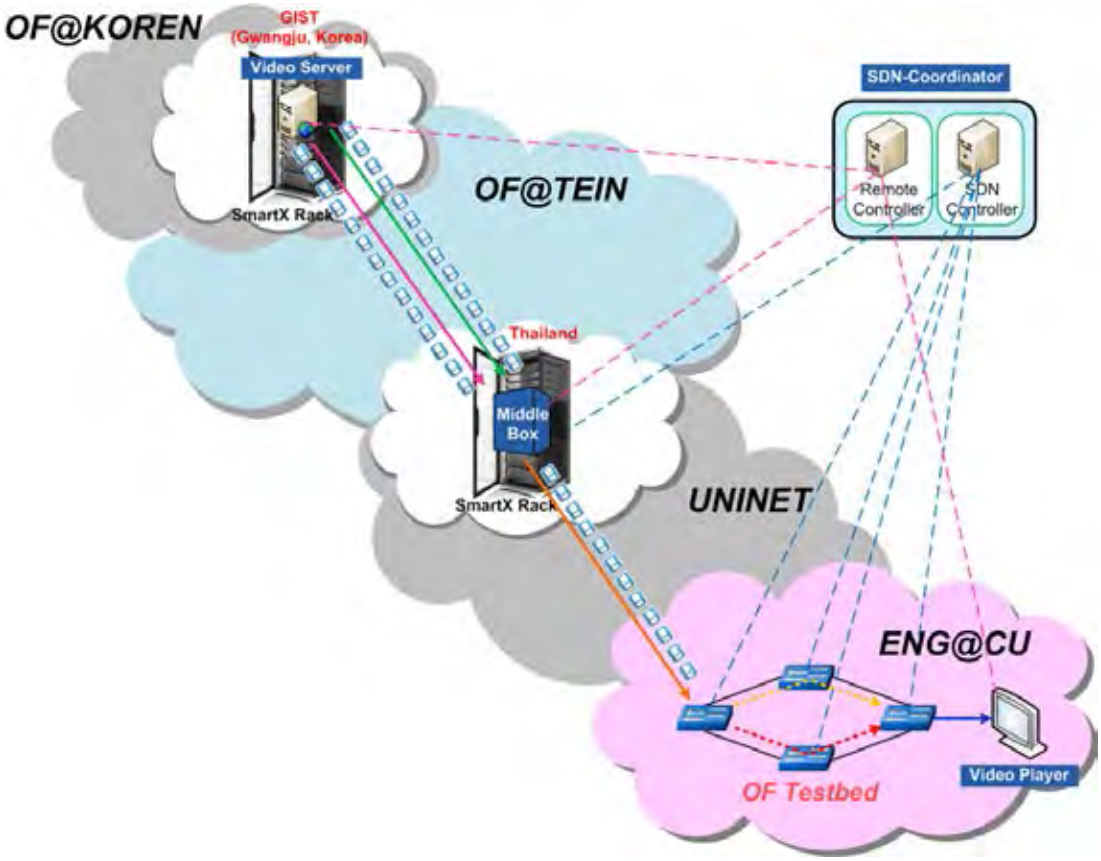
รูปที่ 5.3: สมาร์ทเอ็กซ์ แร็ก

จะสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถี ผ่านระบบทดสอบ OF@TEIN มาให้ตัวเล่นวีดิทัศน์ที่อยู่ในประเทศไทย ซึ่งการทดลองนี้ต้องการทดสอบการสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถีบนโครงข่ายโอเพนโฟลว์ขนาดใหญ่ข้ามประเทศ และเปรียบเทียบคุณภาพวีดิทัศน์ระหว่างการสตรีมแบบหนึ่งวิถีและพหุวิถี

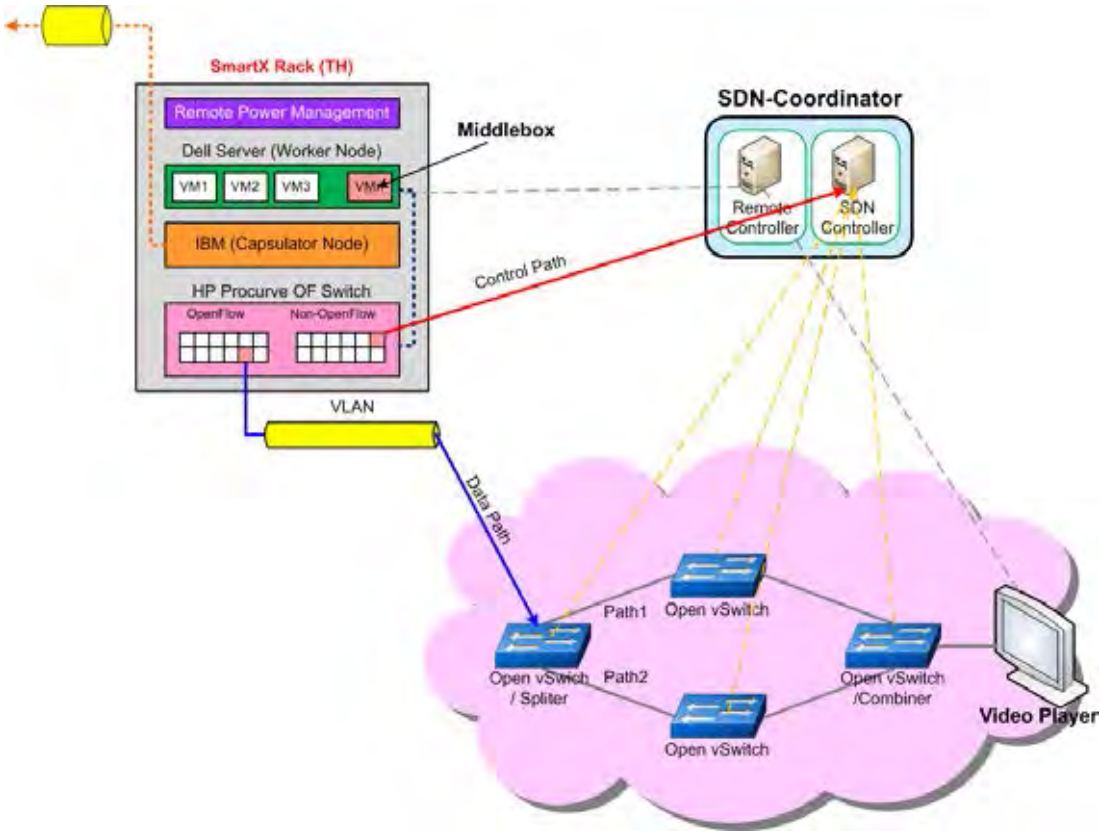
2. การทดลองบนระบบทดสอบโอเพนโฟลว์ขนาดเล็กในห้องปฏิบัติการ และระบบทดสอบ OF@TEIN

   นอกจากทดลองสตรีมวีดิทัศน์บนระบบทดสอบ OF@TEIN แล้ว แนวทางการทดลองที่แนะนำต่อมาคือ เชื่อมต่อระบบทดสอบโอเพนโฟลว์ขนาดเล็กในห้องปฏิบัติการ เข้ากับระบบทดสอบ OF@TEIN ดังแสดงในรูปที่ 5.4 แล้วทำการสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถีจากตัวบริการวิดิทัศน์ที่อยู่ในสาธารณรัฐเกาหลีใต้ มายังตัวเล่นวีดิทัศน์ที่อยู่ในระบบทดสอบโอเพนโฟลว์ขนาดเล็กในห้องปฏิบัติการวิจัยโทรคมนาคม อาคาร 4 ชั้น 13 ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ ในการทดลองนี้ตัวบริการวีดิทัศน์จะสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถีมายังมิดเดิลบอกซ์ที่อยู่ในห้องโครงข่าย อาคารจามจุรี 9 ผ่านระบบทดสอบ OF@TEIN โดยจะใช้เครื่องเสมือนจำลองเป็นตัวบริการวีดิทัศน์และมิดเดิลบอกซ์ มิดเดิลบอกซ์จะเก็บกลุ่มก้อนวีดิทัศน์และนำมาเรียงลำดับก่อนส่งไปยังระบบทดสอบโอเพนโฟลว์ขนาดเล็กในห้องปฏิบัติการผ่านโครงข่ายยูนิเน็ท (UniNet) [31] ซึ่งเป็นโครงข่ายเพื่อการศึกษาวิจัยสำหรับสถาบันศึกษาในประเทศไทย เมื่อสวิตช์โอเพนโฟลว์ต้นทางในระบบทดสอบขนาดเล็กได้รับกลุ่มก้อนวีดิทัศน์ที่ส่งมาจากมิดเดิลบอกซ์ สวิตช์ต้นทางจะสตรีมวีดิทัศน์เป็นกลุ่มก้อนแบบพหุวิถีไปยังตัวเล่นวีดิทัศน์ต่อไป

การเชื่อมต่อระหว่างสมาร์ทเอ็กซ์ แร็ก ณ ห้องโครงข่าย อาคารจามจุรี 9 และระบบทดสอบโอเพนโฟลว์ขนาดเล็กในห้องวิจัย สามารถแสดงได้ดังรูปที่ 5.5 โดยภายในสมาร์ทเอ็กซ์ แร็ก เครื่องเสมือนจะถูกเชื่อมต่อกับสวิตช์โอเพนโฟลว์ และตัวควบคุมระยะไกล เพื่อให้ตัวควบคุมระยะไกลสามารถสั่งการเครื่องเสมือนได้ สวิตช์โอเพนโฟลว์ในสมาร์ทเอ็กซ์ แร็ก จะเชื่อมต่อกับตัวควบคุมเอสดีเอ็น และสวิตช์โอเพนโฟลว์ต้นทางของระบบทดสอบโอเพนโฟลว์ขนาดเล็กในห้องปฏิบัติการ โดยตัวควบคุมเอสดีเอ็นจะควบคุมการทำงานของสวิตช์โอเพนโฟลว์ในสมาร์ทเอ็กซ์ แร็ก ควบคู่กับสวิตช์โอเพนโฟลว์ในระบบทดสอบ ขณะที่สวิตช์โอเพนโฟลว์ต้นทางของระบบทดสอบ จะทำหน้าที่รับวีดิทัศน์ที่ถูกส่งมาจากสวิตช์โอเพนโฟลว์ในสมาร์ทเอ็กซ์ แร็ก ผ่านแลนเสมือน (VLAN) เพื่อส่งวีดิทัศน์ต่อไปยังตัวเล่นวีดิทัศน์ในระบบทดสอบ



รูปที่ 5.4: การทดลองบนระบบทดสอบโอเพนโฟลว์ขนาดเล็กในห้องปฏิบัติการ และระบบทดสอบ OF@TEIN



รูปที่ 5.5: การเชื่อมต่อระหว่างสมาร์ทเอ็กซ์ แร็ก และระบบทดสอบโอเพนโฟลว์ขนาดเล็กในห้องปฏิบัติการ

3. การปรับปรุงสมรรถนะของระบบทดสอบให้สามารถรับรองการสตรีมวีดิทัศน์ที่อัตราการเข้ารหัสสูงขึ้น

   เนื่องจากเครื่องคอมพิวเตอร์ที่ใช้จำลองเป็นมิดเดิลบอกซ์มีข้อจำกัดทางฮาร์ดแวร์ ไม่สามารถประมวลแพ็กเกตที่เข้ามาได้ทั้งหมด ทำให้วิทยานิพนธ์นี้ต้องสตรีมวีดิทัศน์ที่อัตรการเข้ารหัสเท่ากับ 500 กิโลบิตต่อวินาที แต่หากต้องการสตรีมวีดิทัศน์บนระบบทดสอบด้วยอัตราการเข้ารหัสที่สูงขึ้นในอนาคตอาจทำได้ด้วยวิธี เช่น เพิ่มหน่วยประมวลผลของเครื่องคอมพิวเตอร์ที่ใช้จำลองเป็นมิดเดิลบอกซ์ เพื่อให้ประมวลแพ็กเกตที่มีจำนวนมากได้อย่างรวดเร็ว หรือปรับปรุงโปรแกรมภายในมิดเดิลบอกซ์ให้สามารถทำงานประสานกับโปรแกรมประยุกต์ ณ ตัวเล่นวีดิทัศน์ เพื่อให้มีการแคช (caching) และการจัดการกับแพ็กเกตที่เข้ามาล่าช้าได้ดีขึ้น ซึ่งจะส่งผลให้วีดิทัศน์ที่ถูกแสดงมีความผิดพลาดน้อยลงหรือมีคุณภาพดีขึ้น นอกจากนี้ควรทดสอบคุณภาพของวีดิทัศน์โดยใช้ PSNR เมื่อสตรีมวีดิทัศน์บนระบบทดสอบภายใต้ทราฟฟิกพื้นหลัง (background traffic) ที่เกิดขึ้นแบบสุ่มด้วย

รายการอ้างอิง

[1] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. Openflow: enabling innovation in campus networks. SIGCOMM Computer Communication Review 38 (April 2008): 69–74.

[2] ONF Market Education Committee, and others. Software-defined networking: The new norm for networks. ONF White Paper. Palo Alto, US: Open Networking Foundation (April 2012).

[3] OpenFlow in Europe: linking infrastructure and applications [Online]. Available from : http://www.fp7-ofelia.eu [2014, March]

[4] Elliott, C. Geni-global environment for network innovations. in Proc. of LCN, 2008 : 8.

[5] Aoyama, T. New generation network (NWGN) beyond ngn in japan [Online]. 2007. Available from : http://akariproject.nict.go.jp/document/INFOCOM2007.pdf [2014, March]

[6] Cisco Visual Networking Index. Forecast and methodology, 2012–2017 cisco systems, usa [Online]. 2013. Available from : http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf [2014, March]

[7] YouTube [Online]. Available from : http://www.youtube.com [2014, March]

[8] Ustream [Online]. Available from : http://www.ustream.tv [2014, March]

[9] Jurca, D., and Frossard, P. Video packet selection and scheduling for multipath streaming. Multimedia, IEEE Transactions 9 (April 2007): 629-641.

[10] Wang, B., Wei, W., Guo, Z., and Towsley, D. Multipath live streaming via tcp: scheme, performance and benefits. Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP) 5 (2009).

[11] Nightingale, J., Wang, Q., and Grecos, C. Optimised transmission of h. 264 scalable video streams over multiple paths in mobile networks. Consumer Electronics, IEEE Transactions on 56 (2010): 2161–2169.

[12] Xu, C., Liu, T., Guan, J., Zhang, H., and Muntean, G.-M. Cmt-qa: Quality aware adaptive concurrent multipath data transfer in heterogeneous wireless networks. Mobile Computing, IEEE Transactions on 12 (2013): 2193–2205.

[13] Swaminathan V., and Wei, S. Low latency live video streaming using http chunked encoding. in Proc. of IEEE MMSP, 2011 : 1-6.

[14] Abbasi, U., Simo, G., and Ahmed, T. Differentiated chunk scheduling for p2p video-on-demand system. in Proc. of IEEE CCNC, 2011 : 622–626.

[15] Shayeji, M. H. A., Dias, D. N., and Samrajesh, M. Effective regulation of chunks for improved video continuity in p2p video-on-demand. in Proc. of IEEE ICON, 2012 : 191–196.

[16] Gurler, C. G., Savas, S. S., and Tekalp, A. M. Variable chunk size and adaptive scheduling window for p2p streaming of scalable video. in Proc. of IEEE ICIP, 2012 : 2253–2256.

[17] Egilmez, H. E., Civanlar, S., and Tekalp, A. M. A distributed qos routing architecture for scalable video streaming over multi-domain openflow networks. in Proc. of IEEE ICIP, 2012 : 2237–2240.

[18] de Oliveira Silva, F., Goncalves, M. A., de Souza Pereira, J. H., Pasquini, R., Rosa, P. F., and Kofuji, S. T. On the analysis of multicast traffic over the entity title architecture. in Proc. of IEEE ICON, 2012 : 30–35.

[19] Marcondes, C. A., Santos, T. P., Godoy, A. P., Viel, C. C., and Teixeira, C. A. Castflow: Clean-slate multicast approach using in-advance path processing in programmable networks. in Proc. of IEEE ISCC, 2012 : 94–101.

[20] Egilmez, H. E., Civanlar, S., and Tekalp, A. M. An optimization framework for qos-enabled adaptive video streaming over openflow networks. Multimedia, IEEE Transactions on 15 (2013): 710–715.

[21] OpenFlow Switch Specification. Version 1.0.0 (wire protocol 0x01) [Online]. Available from : https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf [2014, March]

[22] Ghareeb, M. About multiple paths video-streaming: state of the art [Online]. 2008. Available from : http://hal.archives-ouvertes.fr/docs/00/33/63/06/PDF/PI-1905.pdf [2014, March]

[23] Lantz, B., Heller, B. and McKeown, N. A network in a laptop: rapid prototyping for software-defined networks. in Proc. of ACM SIGCOMM Workshop on Hot Topics in Networks, 2010.

[24] The VideoLAN project [Online]. Available from : http://www.videolan.org/vlc [2014, March]

[25] POX [Online]. Available from : http://www.noxrepo.org/pox/about-pox [2014, March]

[26] Pfaff, B., Pettit, J., Amidon, K., Casado, M., Koponen, T., and Shenker, S. Extending networking into the virtualization layer. in Proc. of Hotnets, 2009.

[27] Scapy [Online]. Available from : http://www.secdev.org/projects/scapy [2014, March]

[28] Big Buck Bunny Movie [Online]. Available from : http://www.bigbuckbunny.org [2014, March]

[29] Trinh, T., Esaki, H. and Aswakul, C. Dynamic Virtual Network Allocation for OpenFlow Based Cloud Resident Data Center. Communications, IEICE Transactions on, 96 (January 2013): 56-64.

[30] Wondershaper [Online]. Available from : http://lartc.org/wondershaper/ [2014, November]

[31] UniNet [Online]. Available from : http://www.uni.net.th [2014, March]

ภาคผนวก

ภาคผนวก ก

การจำลองโครงข่ายโอเพนโฟลว์ด้วยโปรแกรมมินิเน็ต

โครงข่ายโอเพนโฟลว์เสมือนที่ใช้ในการทดลองที่ 1 และ 2 ในบทที่ 3 สามารถสร้างได้โดยใช้โปรแกรมมินิเน็ตและโปรแกรมไพทอน 2path_topo.py และการสร้างโครงข่ายเสมือนพร้อมทั้งตั้งค่าความหน่วงและแบนด์วิดท์ให้แก่เส้นเชื่อมโยงสามารถสร้างได้โดยใช้โปรแกรม testbed_topo.py

```
"""2path_topo.py by Parichat Panwaree.

4 switches plus 2 host :

   host1 --- switch1 --- switch2 --- switch3 --- host2
                     |            |
                     -- switch4 --

This file is for creating topology with 2 path (chapter 3 ex.1&2)."""

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import CPULimitedHost
from mininet.link import TCLink
from mininet.util import dumpNodeConnections
from mininet.log import setLogLevel

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        host1 = self.addHost( 'h1' )
        host2 = self.addHost( 'h2' )
        switch1 = self.addSwitch( 's1' )
        switch2 = self.addSwitch( 's2' )
        switch3 = self.addSwitch( 's3' )
        switch4 = self.addSwitch( 's4' )

        # Add links
        self.addLink( host1, switch1)
        self.addLink( switch1, switch2)
        self.addLink( switch2, switch3)
        self.addLink( switch3, host2)
        self.addLink( switch1, switch4)
        self.addLink( switch4, switch3)

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

โปรแกรมที่ ก.1: โปรแกรม 2path_topo.py

```
"""testbed_topo.py by Parichat Panwaree.

4 switches plus 2 host :

   host1 --- switch1 --- switch2 --- switch3 --- host2
                     |            |
                     -- switch4 --

This file is for creating topology with 2 path
& setting link bandwidth and delay (chapter 3)."""

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import CPULimitedHost
from mininet.link import TCLink
from mininet.util import dumpNodeConnections
from mininet.log import setLogLevel

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        host1 = self.addHost( 'h1' )
        host2 = self.addHost( 'h2' )
        switch1 = self.addSwitch( 's1' )
        switch2 = self.addSwitch( 's2' )
        switch3 = self.addSwitch( 's3' )
        switch4 = self.addSwitch( 's4' )

        # Add links
        linkopts1 = dict(bw=6.5, delay='0.35ms')
        linkopts2 = dict(bw=2.3, delay='0.61ms')
        self.addLink( host1, switch1, **linkopts1 )
        self.addLink( switch1, switch2, **linkopts1 )
        self.addLink( switch2, switch3, **linkopts1 )
        self.addLink( switch3, host2, **linkopts1 )
        self.addLink( switch1, switch4, **linkopts2 )
        self.addLink( switch4, switch3, **linkopts2 )

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

โปรแกรมที่ ก.2: โปรแกรม testbed_topo.py

การสร้างโครงข่ายเสมือนสามารถทำได้โดยใช้คำสั่งดังต่อไปนี้

```
sudo mn --custom ~/mininet/custom/2path_topo.py --topo mytopo --link tc --mac
    --controller remote
```

ภาคผนวก ข

โปรแกรมสร้างแพ็กเกตทีซีพีและยูดีพี

โปรแกรม send_back2.py และ send_packet2.py เป็นโปรแกรมสร้างและส่งออกแพ็กเกตทีซีพีและยูดีพีตามลำดับ โดยใช้มอดูล scapy

```
# send_back2.py by Parichat Pannwaree.
# This file is for creating and sending udp packet.

#! /usr/bin/env python
from scapy.all import *

def packet():
        infile = open('goback2','r')
        vpayload = infile.readlines()
        infile.close
        p = Ether()/IP()
        p.dst = "00:00:00:00:00:01"
        p.src = "00:00:00:00:00:02"
        p[IP].proto = 6 #udp
        p[IP].payload = str(vpayload)
        return p

if __name__ == '__main__':
        p = packet()
        print p.show
        sendp(p, iface = 'h2-eth0')
```

โปรแกรมที่ ข.1: โปรแกรม send_back2.py

```
# send_packet2.py by Parichat Pannwaree.
# This file is for creating and sending udp packet.

#! /usr/bin/env python
from scapy.all import *

def packet():
        infile = open('payload2','r')
        vpayload = infile.readlines()
        infile.close
        p = Ether()/IP()
        p.dst = "00:00:00:00:00:02"
        p.src = "00:00:00:00:00:01"
        p[IP].proto = 17  #udp
        p[IP].payload = str(vpayload)
        return p

if __name__ == '__main__':
        p = packet()
        print p.show
        sendp(p, iface = 'h1-eth0')
```

โปรแกรมที่ ข.2: โปรแกรม send_packet2.py

ภาคผนวก ข

โปรแกรมตัวควบคุมพอกซ์

โปรแกรมของตัวควบคุมพอกซ์ในบทที่ 4 สำหรับใช้ในการติดตั้งโฟลว์เอนทรีให้แก่โอเพนวีสวิตช์ พร้อมทั้งแบ่งขนาดกลุ่มก้อนวีดิทัศน์ให้แก่วิถีแต่ละวิถี ณ โอเพนวีสวิตช์ 1 สามารถแสดงได้ดังนี้

```
# 2path_mbox.py by Parichat Pannwaree.
# This file is for adding flow entries to all switches.
# And use to split chunk video by time at ovs1.

from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.util import dpid_to_str
from pox.lib.util import str_to_bool
from pox.lib.addresses import IPAddr, EthAddr
import pox.lib.packet as pkt
from threading import Timer
import time
import math

log = core.getLogger()

class MyComponent (object):

    def __init__ (self):
        core.openflow.addListeners(self)

    def _handle_ConnectionUp (self, event):
        log.debug("Switch %s has come up.", dpid_to_str(event.dpid))

        if event.dpid == 149632902914180:  #149632902914180 is dpid of s2
            event.connection.send(of.ofp_flow_mod(
                        action=of.ofp_action_output(port=2),
                        match=of.ofp_match(in_port=1)))
            event.connection.send(of.ofp_flow_mod(
                        action=of.ofp_action_output(port=1),
                        match=of.ofp_match(in_port=2)))

        elif event.dpid == 132824254481935:  #s3
            event.connection.send(of.ofp_flow_mod(
                        action=(of.ofp_action_dl_addr.set_src(
                        EthAddr("78:cd:8e:81:86:58")),
                        of.ofp_action_output(port=4)),
                        match=of.ofp_match(in_port=1)))
            event.connection.send(of.ofp_flow_mod(
                        action=(of.ofp_action_dl_addr.set_src(
                        EthAddr("88:17:20:06:21:58")),
                        of.ofp_action_output(port=4)),
                        match=of.ofp_match(in_port=3)))
            event.connection.send(of.ofp_flow_mod(
                        action=of.ofp_action_output(port=of.OFPP_ALL),
                        match=of.ofp_match(in_port=2)))
            event.connection.send(of.ofp_flow_mod(
```

```
                    action=of.ofp_action_output(port=2),
                    match=of.ofp_match(in_port=5)))
        event.connection.send(of.ofp_flow_mod(
                    match=of.ofp_match(in_port=4)))

    elif event.dpid == 73588229121:  #s4
        event.connection.send(of.ofp_flow_mod(
                    action=of.ofp_action_output(port=2),
                    match=of.ofp_match(in_port=1)))
        event.connection.send(of.ofp_flow_mod(
                    action=of.ofp_action_output(port=1),
                    match=of.ofp_match(in_port=2)))

    elif event.dpid == 963353199705:  #s1

        def install_path1():
            print "Install flow entry for Path 1"
            event.connection.send(of.ofp_flow_mod(
                    action=of.ofp_action_output(port=2),
                    hard_timeout=20, match=of.ofp_match(in_port=1)))
            event.connection.send(of.ofp_flow_mod(
                    action=of.ofp_action_output(port=1),
                    hard_timeout=20, match=of.ofp_match(in_port=2)))
        def install_path2():
            print "Install flow entry for Path 2"
            event.connection.send(of.ofp_flow_mod(
                    action=of.ofp_action_output(port=3),
                    hard_timeout=10, match=of.ofp_match(in_port=1)))
            event.connection.send(of.ofp_flow_mod(
                    action=of.ofp_action_output(port=1),
                    hard_timeout=10, match=of.ofp_match(in_port=3)))

    def install_flow():

            install_path1()
            time_int_path1 = 20
            time_int_path2 = 10
            prev_time = math.floor(time.time())
            current_path_installed = 1

            while True:

                if current_path_installed == 1:
                    if math.floor(time.time())-prev_time==time_int_path1:
                        install_path2()
                        print math.floor(time.time())
                        current_path_installed = 2
                        prev_time = math.floor(time.time())

                elif current_path_installed == 2:
                    if math.floor(time.time())-prev_time==time_int_path2:
                        install_path1()
                        print math.floor(time.time())
                        current_path_installed = 1
                        prev_time = math.floor(time.time())

        t = Timer(10,install_flow)
        t.start()
```

```
106
107 def launch ():
108     core.registerNew(MyComponent)
```

โปรแกรมที่ ข.1: โปรแกรม 2path_mbox.py

การสั่งงานให้พอกซ์ทำงานทำได้โดยใช้คำสั่ง

```
1 ./pox.py 2path_mbox
```

ภาคผนวก ค

โปรแกรมของมิดเดิลบอกซ์

มิดเดิลบอกซ์ที่พัฒนาขึ้นโดยโปรแกรมไพทอนเพื่อใช้รวมแพ็กเกตจากวิถี 2 วิถี พร้อมทั้งเรียงลำดับตามตราเวลาแล้วส่งออกไปให้โอเพนวีสวิตช์ 3 แสดงได้ดังนี้

```
# sniff_dpkt2.py by Parichat Pannwaree.
# This file is for middlebox to capture, parse, store and send packets.

#!/usr/bin/env python

import socket
from struct import *
import datetime
import time
import math
import pcapy
import dpkt
import sys
from scapy.all import *
import threading

global count, buffer1, buffer2, ts1, ts2, arp, pkt, fwd, rtp1, flag
count = 0
arp = 0
flag = 0
buffer1 = []
buffer2 = []
ts1 = []
ts2 = []
pkt = []
fwd = []
sys.setrecursionlimit(5000)

def main(argv) :

    #list all devices
    devices = pcapy.findalldevs()
    #print devices

    '''
    open device
    # Arguments here are:
    #   device
    #   snaplen (maximum number of bytes to capture _per_packet_)
    #   promiscious mode (1 for true)
    #   timeout (in milliseconds)
    '''
    cap = pcapy.open_live("eth1" , 65536 , 1 , 0)

    #start sniffing packets
    while(1) :
        (header, packet) = cap.next()
```

```
48          #print ('%s: captured %d bytes, truncated to %d bytes'
    %(datetime.datetime.now(), header.getlen(), header.getcaplen()))
49          parse_packet(packet)
50
51  #function to parse a packet
52  def parse_packet(packet) :
53
54      global count, buffer1, buffer2, ts1, ts2, arp, pkt, fwd, rtp1
55
56      print time.time(),',',(len(buffer1)),',',(len(buffer2))
57
58      eth = dpkt.ethernet.Ethernet(packet)
59
60      #Parse eth packets,
61      if eth.type == dpkt.ethernet.ETH_TYPE_IP:
62
63          ip = eth.data
64          udp = ip.data
65          rtp1 = udp.data
66
67          #UDP protocol
68          if ip.p == 17:
69              #RTP packet
70              rtp = dpkt.rtp.RTP(str(rtp1))
71
72              if eth.src == '\x78\xcd\x8e\x81\x86\x58' : #eth2 PC3 (Path1)
73                  count += 1
74                  buffer1.append(packet)
75                  time_stamp = rtp.ts
76                  ts1.append(time_stamp)
77
78              elif eth.src == '\x88\x17\x20\x06\x21\x58' : #eth9 PC3 (Path2)
79                  count += 1
80                  buffer2.append(packet)
81                  time_stamp = rtp.ts
82                  ts2.append(time_stamp)
83
84          #some other UDP packet like IGMP
85          else :
86              fwd.append(packet)
87              s_fwd = fwd.pop(0)
88              h_fwd = '\x78\xcd\x8e\x81\x86\x59\x00\xe0\x4c\x53\x44\x58\x08\x00'
89              f = Ether(h_fwd+s_fwd[14:])
90              sendp(f, iface = 'eth0', verbose=0)
91
92      #ARP packet :
93      elif eth.type == dpkt.ethernet.ETH_TYPE_ARP:
94          arp += 1
95          if arp == 1:
96              pkt.append(packet)
97              s_pkt = pkt[0]
98              head_pkt = '\x78\xcd\x8e\x81\x86\x59\x00\xe0\x4c\x53\x44\x58\x08\x06'
99              y = Ether(head_pkt+s_pkt[14:])
100             sendp(y, iface = 'eth0', verbose=0)
101
102     print
103
104 def sendpkt():
```

```
105
106     global rate, buffering_time, flag
107
108     prev_time = math.floor(time.time()*rate)
109     start_time = math.floor(time.time())
110
111     while True:
112
113         head = '\x78\xcd\x8e\x81\x86\x59\x00\xe0\x4c\x53\x44\x58\x08\x00'
114
115         if count == 1:
116             start_time = math.floor(time.time())
117         else:
118             pass
119
120         if (math.floor(time.time()) - start_time >= buffering_time) and
    (math.floor(time.time()*rate) != prev_time):
121
122             if (len(buffer1) != 0) and (len(buffer2) != 0):
123                 a = ts1[0]
124                 b = ts2[0]
125
126                 if a > b:       # send packet in buffer2
127                     bb = buffer2.pop(0)
128                     ts2.pop(0)
129                     x = Ether(head+bb[14:])
130                     try :
131                         sendp(x, iface = 'eth0', verbose=0)
132                         prev_time = math.floor(time.time()*rate)
133                     except socket.error :
134                         continue
135
136                 elif b > a :    # send packet in buffer1
137                     aa = buffer1.pop(0)
138                     ts1.pop(0)
139                     z = Ether(head+aa[14:])
140                     try :
141                         sendp(z, iface = 'eth0', verbose=0)
142                         prev_time = math.floor(time.time()*rate)
143                     except socket.error :
144                         continue
145
146                 elif a == b :    # send packet in buffer1
147                     aa = buffer1.pop(0)
148                     ts1.pop(0)
149                     buffer2.pop(0)  # delete pkt in buffer2
150                     ts2.pop(0)
151                     z = Ether(head+aa[14:])
152                     try :
153                         sendp(z, iface = 'eth0', verbose=0)
154                         prev_time = math.floor(time.time()*rate)
155                     except socket.error :
156                         continue
157
158             elif len(buffer1) == 0 and len(buffer2) != 0:
159                 bb = buffer2.pop(0)
160                 ts2.pop(0)
161                 x = Ether(head+bb[14:])
```

```
                try :
                    sendp(x, iface = 'eth0', verbose=0)
                    prev_time = math.floor(time.time()*rate)
                except socket.error :
                    continue

            elif len(buffer2) == 0 and len(buffer1) != 0:
                aa = buffer1.pop(0)
                ts1.pop(0)
                z = Ether(head+aa[14:])
                try :
                    sendp(z, iface = 'eth0', verbose=0)
                    prev_time = math.floor(time.time()*rate)
                except socket.error :
                    continue

            else :
                pass

        else :
            pass


if __name__ == "__main__":

    global rate, buffering_time
    rate = 150
    buffering_time = 40

    t1 = threading.Thread(target=main, args=(sys.argv))
    t2 = threading.Thread(target=sendpkt)

    t1.start()
    t2.start()

    t1.join()
    t2.join()
```

โปรแกรมที่ ค.1: โปรแกรม sniff_dpkt2.py

ประวัติผู้เขียนวิทยานิพนธ์

ปาริฉัตร ปันวารี เกิดเมื่อวันที่ 22 พฤษภาคม พ.ศ. 2533 ที่จังหวัดลำพูน สำเร็จการศึกษาชั้นมัธยมศึกษาจากโรงเรียนจักรคำคณาทรจังหวัดลำพูน ในปีการศึกษา 2550 จากนั้นได้เข้าศึกษาต่อที่คณะวิศวกรรมศาสตร์ ภาควิชาวิศวกรรมไฟฟ้า มหาวิทยาลัยเชียงใหม่ จนสำเร็จหลักสูตรวิศวกรรมศาสตรบัณฑิต เกียรตินิยมอันดับ 1 ในปีการศึกษา 2554 จากนั้นได้เข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ณ จุฬาลงกรณ์มหาวิทยาลัย จนสำเร็จการศึกษาในปีการศึกษา 2557

บทความทางวิชาการจากวิทยานิพนธ์

[1] Panwaree, P., Kim, J. and Aswakul, C. Packet Delay and Loss Performance of Streaming Video over Emulated and Real OpenFlow Networks. The 29th International Technical Conference on Circuit/Systems Computers and Communications (ITC-CSCC) 2014.