การศึกษาผลกระทบของการออกแบบสถาปัตยกรรมของแอพลิเคชั่นสำหรับสตรีมวิดีโอที่แตกต่างกันต่อคุณภาพการให้บริการบนเครือข่าย LTE

A Study of Effect of Architectural Design on Quality of Service of a Live Streaming Application with Multiple Endpoints over LTE Network

นายชารีฟ ประภาวิทย์

Charlif Prapawit

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science
Prince of Songkla University

2565

การศึกษาผลกระทบของการออกแบบสถาปัตยกรรมของแอพลิเคชั่นสำหรับสตรีมวิดีโอที่แตกต่างกันต่อคุณภาพการให้บริการบนเครือข่าย LTE

A Study of Effect of Architectural Design on Quality of Service of a Live Streaming Application with Multiple Endpoints over LTE Network

นายชารีฟ ประภาวิทย์

Charlif Prapawit

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
มหาวิทยาลัยสงขลานครินทร์
A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Science
Prince of Songkla University
2565

ชื่อวิทยานิพนธ์ การศึกษาผลกระทบของการออกแบบสถาปัตยกรรมของแอพลิเคชั่นสำหรับสตรีมวิดีโอที่แตกต่างกันต่อคุณภาพการให้บริการบนเครือข่าย LTE

ผู้เขียน นายชารีฟ ประภาวิทย์

สาขาวิชา วิทยาการคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

.....

(ผู้ช่วยศาสตราจารย์ ดร.ชินพงศ์ อังสุโชติเมธี)

คณะกรรมการสอบ

.....ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร. พิศาล เศรษฐวงศ์)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร. สุภาภรณ์ กานต์สมเกียรติ)

.....กรรมการ

(รองศาสตราจารย์ ดร. สาธิต อินทจักร์)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ชินพงศ์ อังสุโชติเมธี)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

.....

(ผู้ช่วยศาสตราจารย์ ดร. เถกิง วงศ์ศิริโชติ)

รักษาการแทนคณบดีบัณฑิตวิทยาลัย

ขอรับรองว่า ผลงานวิจัยนี้มาจากการศึกษาวิจัยของนักศึกษาเอง และได้แสดงความขอบคุณบุคคลที่มีส่วนช่วยเหลือแล้ว

ลงชื่อ .....
(ผู้ช่วยศาสตราจารย์ ดร.ชินพงศ์ อังสุโชติเมธี)
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

ลงชื่อ .....
(นายชารีฟ ประภาวิทย์)
นักศึกษา

ข้าพเจ้าขอรับรองว่า ผลงานวิจัยนี้ไม่เคยเป็นส่วนหนึ่งในการอนุมัติปริญญาในระดับใดมาก่อน และไม่ได้ถูกใช้ในการยื่นขออนุมัติปริญญาในขณะนี้

ลงชื่อ .....
(นายชารีฟ ประภาวิทย์)
นักศึกษา

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จได้ด้วยความช่วยเหลือและการสนับสนุนจากบุคคลหลายฝ่าย ผู้วิจัยรู้สึกซาบซึ้งและขอกราบขอบพระคุณเป็นอย่างสูง คือ

ผู้ช่วยศาสตราจารย์ ดร.ชินพงศ์ อังสุโชติเมธี อาจารย์ที่ปรึกษาวิทยานิพนธ์ที่ได้กรุณาให้คำปรึกษาแนะนำและช่วยเหลือในการแก้ปัญหาต่างๆ ให้แก่ผู้วิจัยเสมอมา พร้อมทั้งตรวจทานแก้ไขวิทยานิพนธ์ให้แก่ผู้วิจัย

ผู้ช่วยศาสตราจารย์ ดร. พิศาล เศรษฐวงศ์ ประธานกรรมการในการสอบวิทยานิพนธ์ ที่กรุณาช่วยให้คำแนะนำที่มีคุณค่า ทำให้วิทยานิพนธ์ให้มีความสมบูรณ์

ผู้ช่วยศาสตราจารย์ ดร. สุภาภรณ์ กานต์สมเกียรติ กรรมการในการสอบวิทยานิพนธ์ที่กรุณาช่วยให้คำแนะนำในการแก้ไขวิทยานิพนธ์ให้มีความสมบูรณ์

รองศาสตราจารย์ ดร. สาธิต อินทจักร์ กรรมการในการสอบวิทยานิพนธ์ที่กรุณาช่วยให้คำแนะนำในการแก้ไขวิทยานิพนธ์ให้มีความสมบูรณ์

เจ้าหน้าที่ภาควิชาวิทยาการคำนวณและเจ้าหน้าที่บัณฑิตวิทยาลัยทุกท่านที่ให้ความช่วยเหลือและอำนวยความสะดวกเกี่ยวกับเอกสารต่างๆ

เพื่อนๆ พี่ๆ และน้องๆ สาขาวิทยาการคอมพิวเตอร์ ที่คอยให้กำลังใจและช่วยเหลือให้คำปรึกษาในการทำวิทยานิพนธ์

คุณพ่อ คุณแม่ และน้องชาย รวมถึงญาติๆ ทุกคน ที่ให้การสนับสนุนคอยเป็นห่วงสุขภาพและให้กำลังใจแก่ผู้วิจัยมาโดยตลอด

ผู้วิจัยขอขอบคุณทุกท่านเป็นอย่างสูงมา ณ โอกาสนี้

ชารีฟ ประภาวิทย์

ชื่อวิทยานิพนธ์ การศึกษาผลกระทบของการออกแบบสถาปัตยกรรมของแอพลิเคชั่นสำหรับสตรีมวิดีโอที่แตกต่างกันต่อคุณภาพการให้บริการบนเครือข่าย LTE

ผู้เขียน นายชารีฟ ประภาวิทย์

สาขาวิชา วิทยาการคอมพิวเตอร์

ปีการศึกษา 2565

บทคัดย่อ

จำนวนผู้ให้บริการสตรีมมิ่งเพิ่มขึ้นอย่างมากทุกปี ดังนั้นผู้ใช้บริการที่ต้องการเผยแพร่สตรีมของตนไปยังผู้ให้บริการหลายแห่งเพื่อเพิ่มการมองเห็นจากผู้ชม อย่างไรก็ตาม ผู้ให้บริการส่วนใหญ่ต้องการผูกขาดบริการของตน ทำให้การศึกษาการออกแบบสถาปัตยกรรมที่เหมาะสมของบริการสตรีมมิ่งที่รองรับหลายปลายทางการสตรีมจึงไม่ได้รับความสนใจมากนัก ในการศึกษานี้จะตรวจสอบผลกระทบของการนำการออกแบบสถาปัตยกรรมที่แตกต่างกันมาใช้ในการพัฒนาบริการสตรีมมิงแบบสดผ่านเครือข่าย Long Term Evolution(LTE) ซึ่งสามารถรองรับปลายทางการสตรีมหลายจุด การออกแบบที่สำคัญสองแบบซึ่งเป็นสถาปัตยกรรมแบบใช้หน่วยการส่งต่อแบบคัดเลือกและสถาปัตยกรรมแบบไม่ใช้หน่วยการส่งต่อแบบคัดเลือก ผลลัพธ์ที่ได้แสดงให้เห็นว่าสถาปัตยกรรมแบบที่ใช้หน่วยการส่งต่อแบบคัดเลือกมีข้อได้เปรียบเหนือสถาปัตยกรรมที่ไม่ใช้หน่วยส่งต่อแบบคัดเลือกโดยรักษาการหน่วงเวลาการสตรีมจากต้นทางถึงปลายทางโดยเฉลี่ยน้อยกว่า ในขณะที่การหน่วงเวลาของสถาปัตยกรรมแบบไม่ใช้หน่วยการส่งต่อแบบคัดเลือกจากต้นทางถึงปลายทางจะผันผวน ในแบบการทดลองนี้ ผลลัพธ์ การอภิปราย และข้อเสนอแนะเกี่ยวกับการศึกษาในอนาคตท้ายการศึกษานี้

Thesis Title A Study of Effect of Architectural Design on Quality of Service of a Live Streaming Application with Multiple Endpoints over LTE Network

Author Mr.Charlif Prapawit

Major Program Computer Science

Academic Year 2022

ABSTRACT

The number of streaming service providers has been increasing dramatically every year. Hence, users may prefer to publish their stream to multiple service endpoints simultaneously to increase visibility. However, most service providers prefer to monopolize their services. Hence, a study of a suitable architectural design of a streaming service that supports multiple streaming endpoints has not gained lots of attention. In this study, the effect of adopting different architectural design on developing a live streaming service over LTE network which can supports multiple streaming endpoints are investigated. Two major designs are selected which are a selective forwarding unit-based architecture, and a non-selective forwarding unit-based architecture. The results suggest that a selective forwarding unit architecture has an advantage over a non-selective forwarding unit-based architecture on keeping the overall average streaming end-to-end delay to be minimum., while a fluctuation in an end-to-end delay occurs in a non-selective forwarding unit based architecture in our experiment testbed. The results, discussions, and suggestions on future studies are given at the end of this study.

สารบัญ

หน้าที่

รายการตาราง

หน้าที่

รายการรูปภาพ

หน้าที่

บทที่ 1

บทนำ

งานวิจัยฉบับนี้ เป็นการศึกษาเพื่อเปรียบเทียบคุณภาพของการให้บริการในการสตรีมมิ่งไปยังจุดปลายทางหลายจุดพร้อมกับบนเครือข่าย LTE โดยมีรายละเอียดดังนี้

1.1 หลักการและเหตุผล ความเป็นมาและความสำคัญของปัญหา

บริการสตรีมมิงแบบสดเป็นหนึ่งในบริการยอดนิยมบนอินเทอร์เน็ตในปัจจุบัน บริการสตรีมมิงแบบสดช่วยให้ผู้ใช้สามารเผยแพร่ฟีดสื่อแบบสดไปยังผู้ชมที่ต้องการโดยตรงจากคอมพิวเตอร์หรืออุปกรณ์พกพา ผู้ให้บริการสตรีมมิ่งในปัจจุบันนั้นเพิ่มขึ้นอย่างมากบริการที่เป็นที่นิยม เช่น Facebook Live, YouTube, Dailymotion, Vimeo และ Twitch ซึ่งข้อดีที่สำคัญของการสตรีมมิ่งแบบสดไม่เพียงแต่ทำให้ผู้ใช้หรือผู้ชมโต้ตอบกับตนเองเท่านั้น แต่ยังช่วยส่งเสริมธุรกิจของพวกเขาอย่างมากจากการถูกมองเห็นโดยผู้ใช้งานในบริการนั้น ๆ ซึ่งนำไปสู่การเข้าถึงกลุ่มลูกค้าที่มากขึ้นในธุรกิจของพวกเขา [1]

ข้อจำกัดที่สำคัญอย่างหนึ่งของแอพพลิเคชั่นสตรีมมิ่งที่มีอยู่ส่วนใหญ่หรือบริการสตรีมมิ่งคือผู้ใช้งานสตรีมมิ่งของแต่ละบริการสามารถเผยแพร่สตรีมสดไปยังบริการเดียวหรือมากกว่าสองบริการพร้อมกัน ด้วยความจำเป็นเพื่อให้ผู้ใช้สตรีมมิ่งหรือผู้เผยแพร่เข้าถึงผู้ชมในวงกว้าง เนื่องจากจำนวนบริการที่มากขึ้นทำให้ผู้คนต่างกระจัดกระจายไปตามความสะดวกใจตามความชื่นชอบและความนิยมตามพื้นที่ การสตรีมมิ่งที่ช่วยให้ผู้ใช้สามารถสตรีมมิ่งไปยังหลายจุดปลายทางพร้อมกันเป็นสิ่งจำเป็นยิ่งไปกว่านั้น เนื่องจากอุปกรณ์ส่วนใหญ่ที่ผู้ใช้งานสตรีมมิ่งใช้เชื่อมต่ออินเทอร์เน็ตในปัจจุบันส่วนใหญ่เป็นอุปกรณ์พกพา[2] เช่น สมาร์ทโฟน แท็ปเล็ด ดังนั้นบริการสตรีมมิ่งดังกล่าวต้องรองรับกรณีที่ผู้ใช้เผยแพร่สตรีมจากอุปกรณ์มือถือผ่านเครือข่ายเซลลูลาร์ LTE

เนื่องจากผู้ให้บริการสตรีมมิ่งส่วนใหญ่ต้องการให้ผู้ใช้บริการใช้บริการเดียวโดยเฉพาะของตนเพื่อให้ผู้ให้บริการสามารถผูกขาดบริการได้ มีเพียงไม่กี่การศึกษาที่เน้นการออกแบบสถาปัตยกรรมของบริการสตรีมสดที่อนุญาตให้ผู้ใช้เผยแพร่สตรีมของตนสามารถเผยแพร่ไปยังจุดสิ้นสุดหลายจุด

การออกแบบสถาปัตยกรรมของบริการดังกล่าวมีการกล่าวถึงสั้น ๆ ใน [3][4] บริการสตรีมมิ่งที่มีอยู่ส่วนใหญ่อาศัยการใช้งาน Selective Forwarding Unit (SFU) สำหรับการจัดการสตรีมข้อมูลจากผู้ใช้ ประมวลผล และกำหนดเส้นทางไปยังจุดหมายปลายทางที่เหมาะสม [5] ในทางกลับกัน การสตรีมบางส่วน

บริการต้องการให้สถาปัตยกรรมของพวกเขาเรียบง่ายผ่านการใช้งานของการสตรีมวิดีโอแบบเพียร์ทูเพียร์ [6] ดังนั้น โดยทั่วไปมี 2 แบบคือหมวดหมู่หลักของการออกแบบสถาปัตยกรรมของบริการสตรีมมิ่งทุกวันนี้ซึ่งเป็น สถาปัตยกรรมที่ใช้ Selective Forwarding Unit(SFU) และ สถาปัตยกรรมที่ไม่ใช้ SFU สถาปัตยกรรมทั้งสองมีข้อดีและข้อเสีย อย่างไรก็ตาม จากการศึกษาเท่าที่ผู้วิจัยค้นหามาทราบว่าจนถึงปัจจุบัน ยังไม่มีการศึกษาที่มีอยู่มุ่งเน้นไปที่การเปรียบเทียบสถาปัตยกรรมทั้งสองในด้านการพัฒนาบริการถ่ายทอดสดที่สามารถรองรับได้ปลายทางหลายจุดพร้อมกัน

ในงานวิจัยนี้ ตรวจสอบความแตกต่างระหว่างสถาปัตยกรรมที่ใช้ SFU และสถาปัตยกรรมที่ไม่ใช้ SFU ในบริการสตรีมสดที่สามารถรองรับได้หลายรายการพร้อมกันปลายทางการสตรีมสด และเนื่องจากว่าจนถึงปัจจุบันผู้ใช้ส่วนใหญ่ชอบใช้โทรศัพท์มือถือบนพีซี ดังนั้นการศึกษานี้จึงเน้นที่กรณีที่ใช้โทรศัพท์มือถือในการเชื่อมต่ออินเทอร์เน็ตของเครือข่าย LTE การศึกษาดำเนินการโดยใช้การทดลองโดยใช้ซอฟต์แวร์จำลองเครือข่าย เพื่อผลลัพธ์และการอภิปรายว่าแบบอิง SFU หรือไม่ใช้ SFU จะดีกว่าหรือไม่ในสถานะการณ์ดังกล่าว

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อตอบคำถามว่าสถาปัตยกรรมแบบไหนเหมาะสมกับการสตรีมมิ่งไปยังหลายบริการบนเครือข่าย LTE โดยใช้ QoS เป็นเกณฑ์

1.3 ขอบเขตและวิธีการศึกษา

1.3.1 งานวิจัยนี้ทดลองบนการซอฟต์แวร์จำลอง

1.3.2 ลักษณะการสตรีมมิ่งบนการจำลองในงานวิจัยนี้กำหนดให้อุปกรณ์สตรีมต้นทางอยู่กับที่

1.4 ขั้นตอนและระยะเวลาการดำเนินการ

1.4.1 ขั้นตอนการดำเนินการ

1. ศึกษางานวิจัยและเอกสารที่เกี่ยวข้อง

2. ศึกษาเทคโนโลยีและเครื่องมือสำหรับงานวิจัย

3. กำหนดขอบเขตของปัญหาในการทำวิจัย

4. วิเคราะห์และออกแบบกระบวนการ

5. พัฒนาและทดสอบประสิทธิภาพกระบวนการที่ได้ออกแบบไว้

6. เขียนบทความวิจัยและเผยแพร่

7. จัดทำเอกสารวิทยานิพนธ์

ตารางที่ 1: แผนการดำเนินการวิจัย

ขั้นตอน	2564												2565					
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6
1																		
2																		
3																		
4																		
5																		
6																		
7																		

1.4.2 ระยะเวลาดำเนินการวิจัย

มกราคม 2564 – มิถุนายน 2565

1.5 สถานที่และเครื่องมือที่ใช้ในการวิจัย

1.5.1 สถานที่

ห้องวิจัยกลุ่มคอมพิวเตอร์และเครือข่าย Cs209 ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่

1.5.2 เครื่องมือใช้

1) ด้านฮาร์ดแวร์

- เครื่องคอมพิวเตอร์ส่วนบุคคล CPU Intel Core I5 2.4 GHz Hard Disk 1TB Ram 16 จำนวน 1 เครื่อง

2) ด้านซอฟต์แวร์

- ระบบปฏิบัติการ Window10 64 bit
- ระบบปฏิบัติการ Ubuntu 20.0.4 LTS
- โปรแกรมประยุกต์ OMNet++

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถยืนยันการเลือกใช้สถาปัตยกรรมที่เหมาะสมสำหรับการสตรีมไปยังหลายจุดบนเครือข่าย LTE ด้วยผลเปรียบเทียบจาก QoS

บทที่ 2

เทคโนโลยีและงานวิจัยที่เกี่ยวข้อง

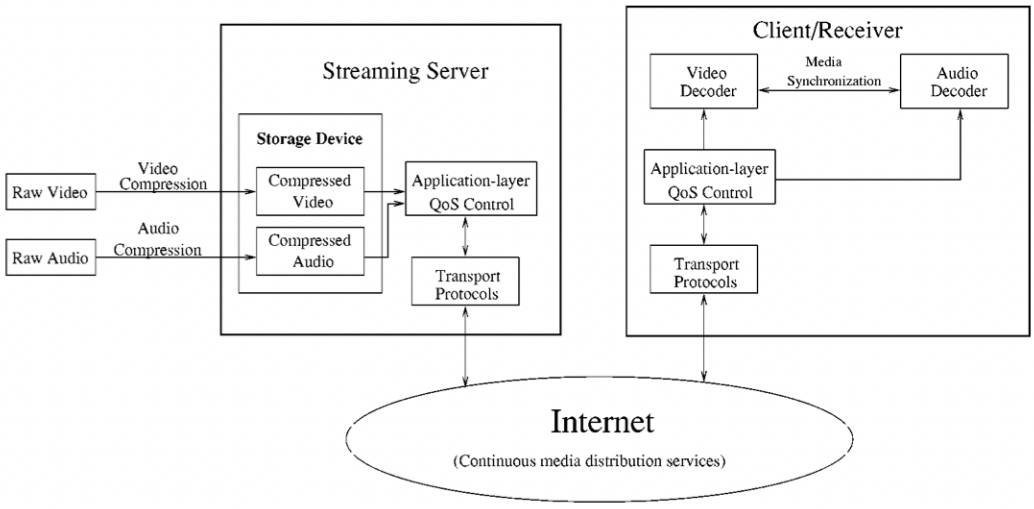
ในเนื้อหาบทนี้เป็นการกล่าวถึงทฤษฎีของการศึกษาในหัวข้อการสตรีมไปยังหลายผู้ให้บริการ โดยเนื้อหาในบทนี้จะถูกแบ่งออกเป็นสองส่วน ส่วนแรกจะกล่าวถึงเทคโนโลยีและเอกสารที่เกี่ยวข้องกับการสตรีมมิ่งที่จำเป็นต้องกล่าวถึงเพื่ออธิบายรูปแบบการทำงานของระบบ ส่วนที่สองจะเกี่ยวกับงานวิจัยเก่าที่เกี่ยวข้องซึ่งใช้ในการอ้างอิงระเบียบในงานวิจัยนี้

2.1 เทคโนโลยีและเอกสารที่เกี่ยวข้อง

เนื้อหาในส่วนนี้เป็นการกล่าวถึงเทคโนโลยีและเอกสารที่เกี่ยวข้องกับหัวข้อการสตรีมไปยังหลายผู้บริการ โดยย่อยออกเป็นสองส่วนซึ่งมีรายละเอียดคือ ส่วนย่อยที่หนึ่งเนื้อหาแสดงถึงสถาปัตยกรรมการสตรีมมิ่งที่สามารถสตรีมไปยังหลายผู้ให้บริการ ส่วนย่อยที่สองคือตัวชี้วัดคุณภาพของการให้บริการหรือ Quality Of Service (QoS) ที่ใช้ในงานวิจัยนี้

2.1.1 เทคโนโลยีการสตรีมภาพและเสียง

เทคโนโลยีการสตรีมเป็นเทคโนโลยีการส่งเนื้อหาโดยทำการส่งวิดีโอรวมกับเสียงเรียก” มัลติมีเดีย”ซึ่งจะทำการตัดแบ่งแล้วส่งไปเป็นส่วนๆ ตามช่วงเวลา ซึ่งทำให้ผู้ชมสามารถรับชมได้ถึงแม้จะไม่ได้ดาวโหลดไฟล์จนครบทั้งไฟล์

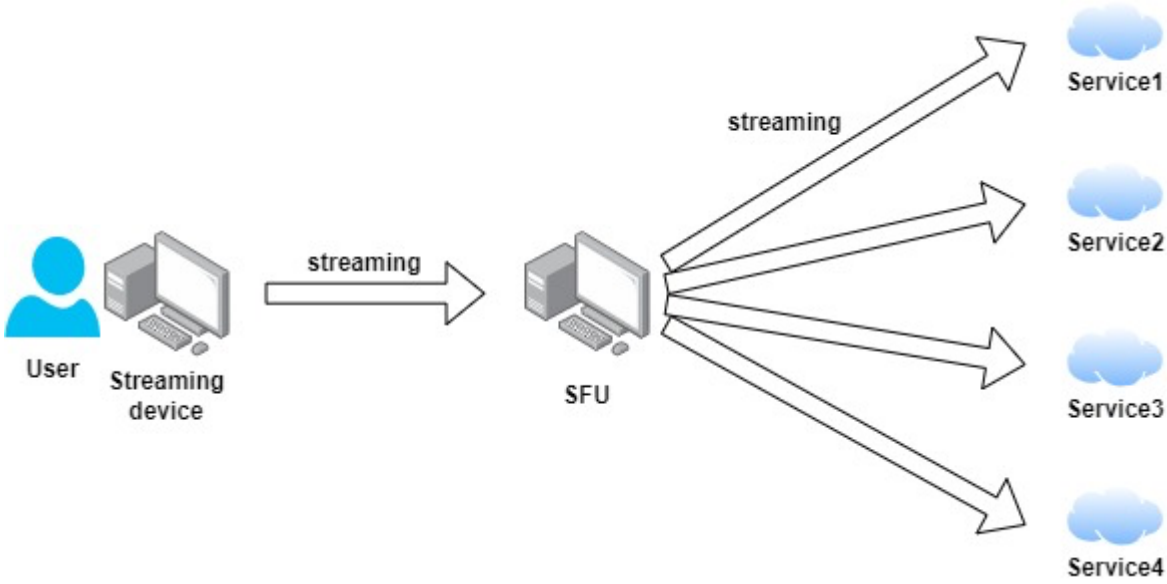


รูปที่ 1 : สถาปัตยกรรมของการสตรีมวิดีโอ[7]

ซึ่งบริการสตรีมมิ่งที่มีอยู่ส่วนใหญ่ในปัจจุบันอาศัยการเชื่อมต่อซ็อกเก็ตแบบ TCP-based [8] แบบง่ายเป็นหลักสำหรับการส่งเนื้อหาการสตรีม อย่างไรก็ตาม ปัจจุบันมีบริการสตรีมมิ่งจำนวนมาก ผู้ใช้อาจต้องการเผยแพร่เนื้อหาสดของตนไปยังบริการมากกว่าหนึ่งแห่งในครั้งเดียว (เช่น การแพร่ภาพเนื้อหาไปยัง Facebook Live และ YouTube พร้อมกันจากเครื่องเดียว) ในการทำเช่นนั้น สถาปัตยกรรมของบริการสตรีมมิ่งและแอพพลิเคชั่นที่เกี่ยวข้องจะต้องได้รับการออกแบบมาอย่างดี เนื่องจากผู้ให้บริการแต่ละที่ใช้รูปแบบของข้อมูลไม่เหมือนกันไม่ว่าจะเป็นคุณภาพของข้อมูลที่รองรับชนิดของข้อมูล โดยทั่วไปสถาปัตยกรรมบริการสตรีมมิ่งที่มีอยู่ที่สามารถให้ผู้ใช้เผยแพร่เนื้อหาของตนไปยังหลายบริการพร้อมกันนั้น สามารถแบ่งออกเป็นสองประเภท ได้แก่ (i) สถาปัตยกรรมที่ใช้ SFU (ii) โครงสร้างสถาปัตยกรรมที่ไม่ใช่ SFU โดยมีรายละเอียดดังนี้

2.1.1.1 สถาปัตยกรรมสตรีมมิ่งที่ใช้ SFU

Selective Forwarding Unit (SFU) เป็นหน่วยที่จัดการการขนส่งของข้อมูลมัลติมีเดีย บริการสตรีมที่ใช้ SFU ใช้ความสามารถในการถ่ายทอดสื่อของ SFU เพื่อส่งต่อเนื้อหาจากผู้ใช้ไปยังปลายทางที่กำหนด ในการเผยแพร่เนื้อหาสตรีมมิงแบบสดไปยังบริการต่าง ๆ พร้อมกัน SFU มีหน้าที่รับแพ็กเก็ตจากผู้ใช้ (เช่น ผู้เผยแพร่เนื้อหา) คัดลอกแพ็กเก็ตและส่งต่อไปยังบริการที่กำหนด ฟังก์ชันการทำงานของ SFU



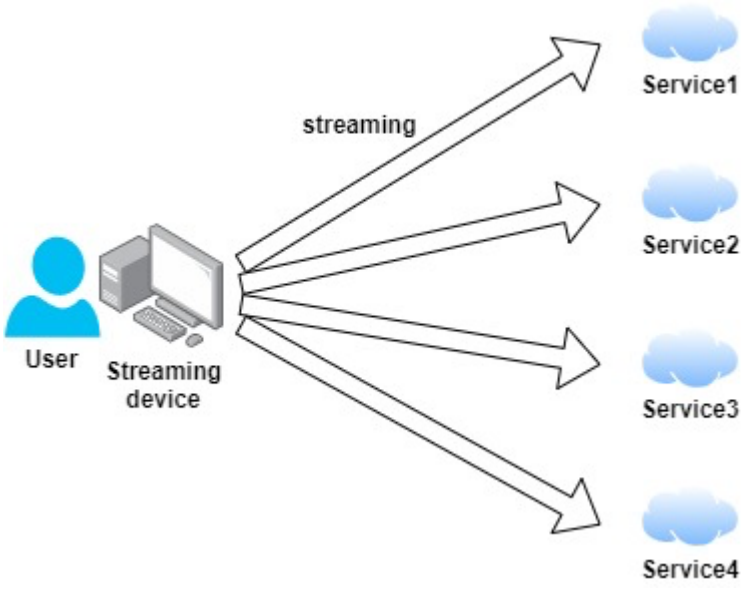
รูปที่ 2 : แผนภาพสถาปัตยกรรมสตรีมมิ่งแบบใช้ SFU

สถาปัตยกรรมดังแสดงในรูปที่ 2 ยังมีการอธิบายเพิ่มเติมไว้ใน[5] การออกแบบดังกล่าวเมื่อนำมาใช้ในงานวิจัยนี้จะถูกเสนอไว้เป็นเพียงแผนการทำงาน เนื่องจากไม่สามารถเข้ารหัสเนื้อหามัลติมีเดียและส่งไปยังจุดสิ้นสุดบริการสตรีมมิ่งหลายจุดพร้อมกันโดยไม่ต้องพึ่งพาคอมพิวเตอร์ที่มีประสิทธิภาพสูงได้ ซึ่งความท้าทายดังกล่าวทำให้สถาปัตยกรรมที่ใช้ SFU สามารถทำได้มากกว่าการเป็นตัวกลางที่ควบคุมเส้นทางการสตรีมในมุมที่ช่วยแบ่งเบาการทำงานสำหรับของคอมพิวเตอร์ผู้ใช้ที่มีประสิทธิภาพไม่เพียงพอ ดังนั้น การใช้ SFU จึงมีความจำเป็น อย่างไรก็ตาม หากประสิทธิภาพของคอมพิวเตอร์ผู้ใช้ได้รับการปรับปรุงอย่างมากเมื่อเวลาผ่านไป ความจำเป็นในการใช้ SFU ยังมีความจำเป็นอยู่หรือไม่

2.1.1.2 สถาปัตยกรรมสตรีมมิ่งที่ไม่ใช้ SFU

สถาปัตยกรรมที่ไม่ใช่ SFU ตรงกันข้ามกับสถาปัตยกรรมที่ใช้ SFU หมายถึงสถาปัตยกรรมที่ถือว่าคอมพิวเตอร์ของผู้ใช้มีประสิทธิภาพเพียงพอที่จะรองรับการสตรีมไปยังผู้ให้บริการปลายทางหลายจุด สถาปัตยกรรมดังกล่าวแสดงไว้ในรูปที่ 3 ซึ่งใช้กับอุปกรณ์เครื่องเดียวเพื่อจัดการข้อมูลทั้งหมดในการ

จัดการและส่งออก เป็นสถาปัตยกรรมการสตรีมที่ใช้บ่อยที่สุด การสตรีมที่อยู่ฝั่งผู้ใช้จะต้องดำเนินการเข้ารหัสและส่งข้อมูลสำหรับแต่ละผู้ให้บริการ



รูปที่ 3 : แผนภาพสถาปัตยกรรมสตรีมมิ่งแบบไม่ใช้ SFU

ในรูปที่ 3 สถาปัตยกรรมการสตรีมนี้ถือว่าแอปพลิเคชันการสตรีมบนเครื่องของลูกค้าสามารถรองรับการสตรีมสดไปยังจุดปลายหลายจุดได้ ข้อดีของการออกแบบนี้คือคุณสมบัติการประหยัดต้นทุนเนื่องจากไม่จำเป็นต้องใช้เซิร์ฟเวอร์ SFU แยกต่างหากอีกต่อไป อย่างไรก็ตาม จนถึงปัจจุบัน อุปกรณ์ส่วนใหญ่ที่สามารถเชื่อมต่ออินเทอร์เน็ตได้ส่วนใหญ่เป็นอุปกรณ์พกพา ดังนั้นจึงยังคงจำเป็นต้องตรวจสอบว่าอุปกรณ์มือถือสามารถรองรับการสตรีมสดไปยังปลายทางหลายจุดได้หรือไม่ ยิ่งไปกว่านั้น อุปกรณ์ส่วนใหญ่เชื่อมต่อกับอินเทอร์เน็ตผ่านเครือข่าย LTE [1] ดังนั้นจึงยังคงต้องตรวจสอบว่าจุดสิ้นสุดของการสตรีมสดหลายจุดใช้แบนด์วิดท์มากเกินไปหรือทำให้เกิดปัญหาคอขวดในเครือข่าย LTE หรือไม่

2.1.2 ตัวชี้วัดคุณภาพของการให้บริการ (Quality of Service)

Quality of Service (QoS) เป็นตัวชี้วัดคุณภาพของการสื่อสารทางอินเตอร์เน็ต [9] ที่ใช้กันอย่างสากลเพื่อบงบอกและเป็นเกณฑ์ในชั้น Network Layer QoS จะมีรายละเอียดดังนี้

2.1.2.1 Packet Delay

Packet delay เป็นหนึ่งในเครื่องมือของ QoS ที่แสดงถึงเวลาที่ใช้ในการถ่ายโอนแพ็กเก็ตจากจุดหนึ่งไปยังอีกจุดหนึ่งในเครือข่ายที่กำหนดไว้ ซึ่งมีปัจจัยที่ส่งผลต่อความล่าช้าของแพ็กเก็ต

1. ล่าช้าผ่านสื่อทางกายภาพ
2. ความล่าช้าผ่านการประมวลผล ที่เกิดจากการเข้ารหัส

อาการดีเลย์แสดงถึงความคลาดเคลื่อนของการสื่อสารที่เวลาถูกยืดออกไปดังนั้นยิ่งค่าดีเลย์น้อยแสดงถึงคุณภาพที่ดี คำถามที่น่าสนใจคือการที่สตรีมไปผู้ให้บริการหลายจุดก็จะเกิดดีเลย์ขึ้นของแต่ละบริการซึ่งคุณภาพที่ดีควรเป็นอย่างไร

2.1.2.2 Packet Loss

Packet loss เป็นหนึ่งในเครื่องมือ QoS ที่แสดงถึงการสูญหายของแพ็กเก็ตในช่วงเวลาที่กำหนดไว้ซึ่งเกิดระหว่างการขนส่งข้อมูลซึ่งอาจสูญหายจาก

1. สัญญาณรบกวน
2. ถูกสกัดออกด้วยความสามารถของอุปกรณ์

เมื่อ packet loss ทำให้ข้อมูลสูญหาย ดังนั้นค่า packet loss ที่ดีควรมีค่าน้อย ในการสตรีมมิ่งอาการที่จะเกิดขึ้นคือภาพไม่ครบถ้วน เพี้ยน หรือส่งผลให้มีการเรียกข้อมูลซ้ำเพื่อซ่อมแซมส่วนที่เสียส่งผลไปยังดีเลย์ อาการพวกนี้ขึ้นอยู่กับโปรโตคอลที่ใช้ในการส่งข้อมูล ถึงแม้ว่าปัจจุบันผู้วิจัยมักเห็นการสตรีมส่วนใหญ่จะอยู่บน UDP ด้วยความคิดที่ว่าการสตรีมแบบสดความเร็วสำคัญกว่าคุณภาพ แต่ก็มีการใช้งานบน TCP เช่น survey [10]

2.2 งานวิจัยที่เกี่ยวข้อง

ผู้วิจัยได้ค้นและศึกษางานวิจัยเก่า ซึ่งพบว่ามีการศึกษาหลายชิ้นเกี่ยวข้องกับการวัดประสิทธิภาพการสตรีม ซึ่งการศึกษาก่อนหน้านี้ที่เกี่ยวข้องกับการวิจัยของผู้วิจัยสามารถแบ่งออกเป็นสองกลุ่มคือ (i) การศึกษาประสิทธิภาพการสตรีมในสภาพแวดล้อมจำลองและ (ii) การศึกษาประสิทธิภาพการสตรีมในบนอุปกรณ์จริง รายละเอียดมีดังนี้

2.2.1 การศึกษาประสิทธิภาพการสตรีมในสภาพแวดล้อม LTE จำลอง

การศึกษากลุ่มนี้เลือกที่จะทำการทดสอบการวัดประสิทธิภาพการสตรีมผ่านเครือข่าย LTE ในสภาพแวดล้อมจำลองโดยใช้ชุดซอฟต์แวร์จำลองเครือข่าย

ในปี ค.ศ. 2015 André Santos และคณะได้ศึกษาและวิเคราะห์การพัฒนาและประสิทธิภาพ LTE ในการสตรีมมัลติมีเดียผ่านการวิเคราะห์ประสิทธิภาพของแพ็คเกจการส่งวิดีโอ นอกจากนี้ยังตรวจสอบอิทธิพลของ LTE ในการสตรีมมัลติมีเดียผ่านการใช้เครื่องจำลอง ผลลัพธ์ของผู้วิจัยแสดงถึงพฤติกรรมของเครือข่าย LTE ระหว่างการสื่อสารแบบสตรีมวิดีโอระหว่างอุปกรณ์มือถือ ดังนั้นจึงเป็นสิ่งสำคัญที่จะต้องเข้าใจคุณลักษณะเหล่านี้ เนื่องจากคุณลักษณะเหล่านี้แสดงให้เห็นถึงการปรับปรุงที่นำเสนอโดย 4G ในความพยายามที่จะปรับปรุงบริการที่มีอยู่ในปัจจุบันและในอนาคต LTE จะเพิ่มแบนด์วิดท์และอัตราการส่งข้อมูล โดยสรุปแล้วการส่งแพ็คเกจวิดีโอมีความเกี่ยวข้องกับตลาดมือถือ การวิเคราะห์เกี่ยวกับการส่งสัญญาณวิดีโอช่วยให้ผู้วิจัยเข้าใจพฤติกรรมของเครือข่าย LTE และเริ่มศึกษาวิธีใหม่ๆ ในการทำให้ระบบมีประสิทธิภาพมากขึ้น [11]

ในปี ค.ศ. 2017 H.-F. Bermudez และคณะได้ศึกษาโดยดำเนินการทดลองโดยใช้โปรแกรมจำลองเครือข่ายที่พัฒนาขึ้น NS-3 ซึ่งการศึกษานี้มุ่งเน้นไปที่การทดสอบความแตกต่างระหว่างการสตรีมวิดีโอ 720p และ 1080p ผ่านเครือข่าย LTE ทดสอบโดยใช้พารามิเตอร์ QoS บนสองโปรโตคอลซึ่งให้ผลว่า RTMP เหมาะกับการส่งข้อมูลแบบ LVS และ RTSP เหมาะสำหรับเครือข่าย LTE [12].

ในปี ค.ศ. 2018 Mauricio Iturralde และคณะได้ศึกษาโดยมุ่งเน้นไปที่แอปพลิเคชัน Voice over IP (VoIP) โดยพิจารณาว่าการใช้กลไกการถ่ายทอดแพ็กเก็ตเสียงเคลื่อนที่ ส่งผลต่อคุณภาพการบริการหรือไม่ ผลการวิจัยพบว่าการใช้กลไกการถ่ายทอดแพ็กเก็ตเสียงบนมือถือทำให้บริการมี

คุณภาพดีขึ้น แม้ว่าจะมีการใช้โอเวอร์เฮดแพ็กเก็ตแพ็กเก็ตเพิ่มเติม 40 ไบต์ในระหว่างกระบวนการส่งต่อก็ตาม [13] กลไกนี้คล้าย SFU ที่กำหนดไว้สำหรับการกำหนดเส้นทางแพ็กเก็ตเสียง

ในปี ค.ศ. 2017 Yangyang Chen และคณะได้ศึกษาประสิทธิภาพของเครือข่ายเซลลูลาร์เมื่อมีการปรับใช้รีเลย์กับโทรศัพท์มือถือ ซึ่งพิจารณาสองแบบ คือ การถ่ายทอดสัญญาณมือถือ Frequency Division Duplex (FDD) กับ Time Division Duplex / Frequency Division Duplex (TDD/FDD) โดยใช้แบบจำลองวิเคราะห์รีเลย์ด้วยวิธีการเรขาคณิตสุ่ม พบว่าการสูญเสียการเจาะเป็นปัจจัยที่กำหนดจำนวนการส่งต่อมือถือที่สามารถรับได้ ผลลัพธ์เชิงตัวเลขแสดงให้เห็นว่าเมื่ออัตราส่วน UE ยานพาหนะใน 1 เสาร์สัญญาณ เท่ากับ 0.4 และการสูญเสียการเจาะ 20 dB โหมดรีเลย์โหมดรีเลย์มือ FDD และ TDD/FDD สามารถรับอัตราต่อเสาร์สัญญาณ +16.3, +29.1% ตามลำดับเมื่อเทียบกับโหมดปกติ [14]

ในปี ค.ศ. 2017 M. Solera และคณะได้นำเสนอชุดทดสอบสำหรับการประเมิน QoE ของบริการสตรีมวิดีโอ 3 มิติ โดยได้เสนอ 3 องค์ประกอบ คือ 1. เซิฟเวอร์การสตรีม 2. อินเตอร์เน็ตโปรโตคอลของmobile network emulator 3. ไคลเอนต์ของการสตรีม โดยทำการทดสอบการดีเลย์การมาถึงของแพ็กเกจ ซึ่งผลลัพท์ทำให้ทราบว่าโหลดของเครือข่ายโดยเฉลี่ยและตำแหน่งผู้ใช้ภายในเสาร์สัญญาณมีผลกระทบอย่างมากต่อ QoS และ QoE ที่ผู้ใช้วิดีโอปลายทางรับรู้ [15]

ในปี ค.ศ. 2018 F Krasniqi และคณะได้นำเสนอการปรับใช้โปรโตคอล TCP และ UDP สำหรับการสตรีมวิดีโอบน VANET ซึ่งเป็นเครือข่ายสื่อสารสำหรับยานพาหนะที่มีการเคลื่อนด้วยความเร็วสูงเพื่อใช้ปรับปรุงปัญหาด้านความปลอดภัยและการขับขี่อย่างสะดวกสะบายยิ่งขึ้น อย่างไรก็ตามมีการท้าทายต่างๆในประเด็นเรื่องการปรับปรุงคุณภาพ ซึ่งพิจารณาเฉพาะโปรโตคอล Transport Layer โดยเปรียบเทียบ 2 โปรโตคอลคือ TCP และ UDP ซึ่งตัวชี้วัดที่พิจารณา ได้แก่

1. Throughput แสดงถึงจำนวนแพ็กเก็ตที่ส่งสำเร็จต่อหน่วยเวลา ซึ่งวัดเป็นบิตต่อวินาที (bps) มีสูตรดังนี้

$$Throughput[kbps] = \frac{Recceived\_data[kbytes]}{Simulation\_time[second]}$$

ค่าปริมาณงาน UDP (396.75 kbps) สูงกว่าค่าปริมาณงาน TCP (243.76 kbps) สาเหตุหลักของผลลัพธ์นี้คือเมื่อผู้ส่ง TCP ตรวจพบการสูญหายของแพ็กเก็ต จะทริกเกอร์การควบคุมความแออัดและลดอัตราการส่งข้อมูลเพื่อลดภาระของเครือข่ายเพื่อหลีกเลี่ยงความแออัด ดังนั้น ลักษณะการทำงานนี้จะนำไปสู่อัตราการใช้แบนด์วิดท์ต่ำ และทำให้ปริมาณงานลดลง ตรงกันข้ามกับผู้ส่ง UDP ที่ใช้แบนด์วิดท์อย่างสมบูรณ์

2. Packet Delivery Ratio (PDR) และ Packet Loss Ratio (PLR): PDR คืออัตราส่วนของแพ็กเก็ตที่ได้รับเมื่อเปรียบเทียบกับจำนวนแพ็กเก็ตที่ส่งทั้งหมด โดยมีสูตรดังนี้

$$PDR[percenyage] = \frac{Received\_Packet}{Sent\_Packets}$$

ในทางตรงกันข้าม PLR คืออัตราส่วนของแพ็กเก็ตที่สูญหายเมื่อเปรียบเทียบกับจำนวนแพ็กเก็ตที่ส่งทั้งหมด คำนวณโดยสมการต่อไปนี้

$$PLR[percentage] = \frac{Lost\_Packets}{Sent\_Packets} = 1 - PDR$$

PLR ของ TCP (31.45%) ต่ำกว่า PLR ของ UDP (43.35%) ผลลัพธ์นี้เกิดจากกลไกความน่าเชื่อถือของ TCP ซึ่งส่งแพ็กเก็ตที่สูญหายอีกครั้ง ตรงกันข้ามกับ UDP ซึ่งไม่สามารถกู้คืนแพ็กเก็ตที่สูญหายได้ รูปที่ 9 แสดงด้วยว่า PDR ของ TCP (68.55%) นั้นสูงกว่า PDR ของ UDP (56.65%) เนื่องจาก PLR ของ TCP นั้นต่ำกว่า PLR ของ UDP

3. Average End-to-End delay (E2E delay) แสดงถึงผลรวมของการหน่วงเวลาตั้งแต่ต้นทางถึงปลายทางของแพ็กเก็ตที่ส่งทั้งหมดตามจำนวนทั้งหมดของแพ็กเก็ตเหล่านี้ ในขณะที่การหน่วงเวลาแบบ end-to-end ของแพ็กเก็ตเป็นช่วงเวลาระหว่างเวลาที่ส่งและเวลาที่ได้รับของแพ็กเก็ตนี้ ค่าเฉลี่ยความล่าช้าจากต้นทางถึงปลายทางคำนวณได้ดังนี้

$$Average\_E2E = \frac{\sum_{j \in received\_packets}(Received[j] - Send[j])}{Received\_Packets}$$

TCP ให้การหน่วงเวลาจากต้นทางถึงปลายทางโดยเฉลี่ย (43.45 มิลลิวินาที) สูงกว่า UDP (30.47 มิลลิวินาที) ผลลัพธ์นี้เป็นสาเหตุของเทคนิคการส่งสัญญาณซ้ำและกลไกการควบคุมความแออัดของโปรโตคอล TCP ซึ่งเพิ่มความล่าช้าในการรับส่งข้อมูลเพิ่มเติมซึ่งเพิ่มความล่าช้าโดยเฉลี่ยจากต้นทางถึงปลายทาง ตรงกันข้ามกับ UDP ซึ่งไม่ส่งแพ็กเก็ตที่สูญหายซ้ำหรือควบคุมปัญหาความแออัด

4. Peak Signal-to-Noise Ratio (PSNR) เป็นการวัดตามวัตถุประสงค์ที่ใช้กันอย่างแพร่หลายเป็นวิธีเปรียบเทียบคุณภาพของภาพที่บีบอัดและวัดกับภาพต้นฉบับ PSNR อิงจากความคลาดเคลื่อนกำลังสองเฉลี่ย (MSE) และคำนวณได้ดังนี้

$$PSNR[dB] = 20log_{10}\frac{V_{peak}}{MSE}$$

เมื่อ $V_{peak} = 2^k - 1$ และ k คือจำนวนบิตต่อพิกเซล

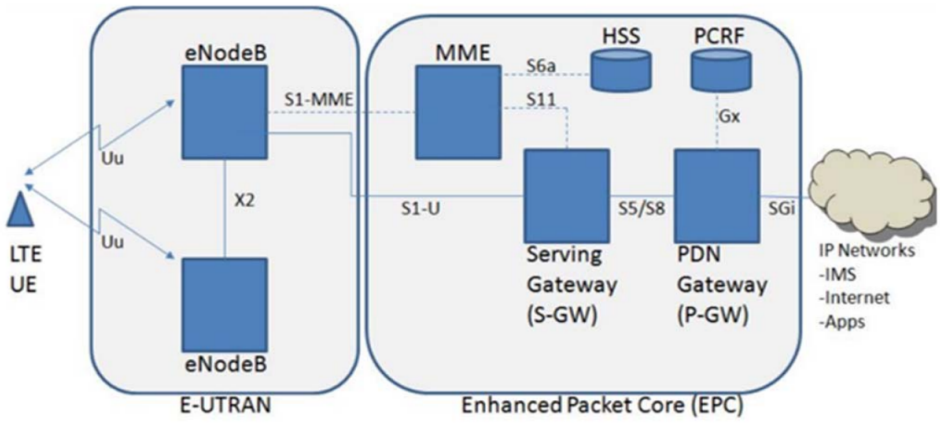
UDP และ TCP เทียบกับวิดีโอต้นฉบับ UDP (22.96 db) ให้ PSNR ที่ต่ำกว่า TCP (26.09 db) ประสิทธิภาพที่เหนือกว่าของ TCP นี้เกิดจาก PLR ที่ต่ำกว่าเมื่อเทียบกับ UDP ซึ่งมีอิทธิพลอย่างมากต่อคุณภาพของวิดีโอที่ส่ง ผลลัพธ์ที่ได้แสดงให้เห็นว่า UDP ให้การส่งข้อมูลวิดีโอได้เร็วกว่า TCP ในขณะที่ TCP ให้คุณภาพวิดีโอที่สูงกว่า UDP เพราะ TCP มีความน่าเชื่อถือมากกว่า UDP [8]

2.2.2 การศึกษาประสิทธิภาพการสตรีมในแบบการทดสอบ LTE จริง

การศึกษากลุ่มนี้เลือกที่จะทำการทดสอบการวัดประสิทธิภาพการสตรีมจริงผ่านเครือข่าย LTE โดยใช้เครือข่ายจากผู้ให้บริการเครือข่าย LTE ที่มีอยู่

ในปี ค.ศ. 2018 Fidel Krasniqi และคณะได้ศึกษาการวัดประสิทธิภาพของเครือข่าย LTE ของ Telecom of Kosovo (TK) ว่าเป็นไปตามดมาตรฐาน 3GPP หรือไม่ ผลลัพธ์แสดงให้เห็นว่าประสิทธิภาพของเครือข่ายสามารถรองรับแอพพลิเคชั่นหรือบริการต่างๆ รวมถึงบริการสตรีมมิ่ง อย่างไรก็ตาม การทิ้งแพ็คเกจและการโต้แย้งที่ไซต์เซลล์ยังคงเป็นข้อกังวลสำคัญที่ต้องระมัดระวังในการส่งมอบบริการด้วยคุณภาพการบริการที่ยอมรับได้ เพื่อเอาชนะข้อกังวลด้านคุณภาพของบริการ

การศึกษาบางชิ้นแนะนำว่าต้องใช้วิธีการจัดเตรียมบริการที่มีคุณภาพ [16] มีการกล่าวถึงสถาปัตยกรรมโดยรวมของเครือข่าย LTE นั้นคล้ายคลึงกับเครือข่าย GSM และ UMTS (ระบบโทรคมนาคมเคลื่อนที่สากล) โดยหลักการแล้ว เครือข่ายแบ่งออกเป็นส่วนเครือข่ายวิทยุและส่วนเครือข่ายกลาง ดังแสดงในรูป



รูปที่ 4 : สถาปัตยกรรม LTE ตาม 3GPP [17]

อย่างไรก็ตาม จำนวนโหนดเครือข่ายแบบลอจิคัลได้ลดลงเพื่อทำให้สถาปัตยกรรมโดยรวมง่ายขึ้น การลดความซับซ้อนของสถาปัตยกรรมจะช่วยลดค่าใช้จ่ายในการดำเนินการเช่นเดียวกับความล่าช้าโดยรวมในเครือข่าย[18][19]

ในปี ค.ศ. 2017 Yanli Xu และคณะได้ศึกษาที่เน้นที่คุณภาพของการจัดหาบริการ (QoS) วิธีการสำหรับบริการจัดส่งเนื้อหาแบบอุปกรณ์ต่ออุปกรณ์ผ่านเครือข่าย LTE ผลลัพธ์ที่ได้แสดงให้เห็นว่าการจัดเตรียมบริการที่ดีและกลไกการแคชเนื้อหาเป็นสิ่งจำเป็นเพื่อให้บริการด้วย QoS ที่ยอมรับได้ (เช่น low latency และ low packet drop) ในเครือข่าย LTE จริง [20] การศึกษาเกี่ยวกับการส่งเนื้อหาจากอุปกรณ์ต่ออุปกรณ์มีเพิ่มเติมตามนี้ [21]

ในปี ค.ศ. 2015 Song Gun Lee และคณะได้เสนอการปรับเปลี่ยนคำขอ HTTP GET อย่างง่ายบนอุปกรณ์ผู้ใช้ LTE สามารถบรรลุการประหยัดพลังงานที่เพิ่มขึ้นอย่างมาก ด้วยการทำความเข้าใจไดนามิกการใช้พลังงานของโมเด็ม LTE และใช้ประโยชน์จากฟิลด์ส่วนหัวของช่วงในคำขอ HTTP ผู้วิจัยสามารถลดการใช้พลังงานลงครึ่งหนึ่งโดยประมาณเพื่อสตรีมไฟล์วิดีโอทั่วไป ไม่จำเป็นต้องใช้ฮาร์ดแวร์พิเศษใดๆ บนเครือข่ายหรืออุปกรณ์ของผู้ใช้ แต่เป็นซอฟต์แวร์การแปล HTTP

แบบง่ายบนมือถือ ด้วยการใช้งานต้นแบบบนสมาร์ทโฟน Android และการวัดผ่านบริการ LTE เชิงพาณิชย์ ผู้วิจัยวัดได้ว่าสามารถประหยัดแบตเตอรี่ได้ 30-40% สำหรับคลิปวิดีโอที่เป็นตัวแทน ซึ่งสามารถขยายได้ถึง 70% หาก DRX พร้อมใช้งาน ด้วยการผสมผสานรูปแบบที่เสนอเป็นพร็อกซีบนอุปกรณ์ผู้ใช้ LTE หรือเป็นส่วนหนึ่งของแอปสตรีมมิ่งบนสมาร์ทโฟนในอนาคต ผู้วิจัยจะสามารถประหยัดแบตเตอรี่ได้อย่างมากในหมวดหมู่แอปพลิเคชันการสตรีมบนมือถือที่ได้รับความนิยมเพิ่มขึ้นนี้ สุดท้าย อาจจำเป็นต้องมีการใช้งานแอปในกรณีที่ขนาดก้อนที่แอปพลิเคชัน (เช่น HLS หรือ DASH) สร้างมีขนาดเล็กกว่าขนาดต่อเนื่อง [22]

ในปี ค.ศ. 2011 Mohammad A. Hoque และคณะได้เสนอแนวคิดเรื่องผลกระทบของพร็อกซีการจัดรูปแบบการรับส่งข้อมูลต่อการใช้พลังงานของการสตรีมเสียงสำหรับอุปกรณ์มือถือ เมื่อใช้ WLAN ได้ค้นพบว่าการประหยัดพลังงานที่ทำได้คือ 30% ถึง 65% ขึ้นอยู่กับระยะเวลาบัฟเฟอร์ อัตราการสตรีม ตำแหน่งของพร็อกซี และสถานการณ์การรับส่งข้อมูลข้ามในเครือข่าย WLAN ในพื้นที่ ในกรณีของ 3G การประหยัดพลังงานจะแตกต่างกันไประหว่าง รุ่นโทรศัพท์และตัวดำเนินการ ชุดค่าผสมบางอย่างช่วยประหยัดพลังงานในขณะที่ชุดค่าผสมอื่นๆ แสดงความผันแปรของความล่าช้าที่ผิดปกติและพฤติกรรมการควบคุมการไหลของ TCP ผู้วิจัยวางแผนที่จะตรวจสอบเพิ่มเติมถึงที่มาของพฤติกรรมการควบคุมการไหลที่ไม่คาดคิดของ TCP ในกรณีของ 3G ยิ่งไปกว่านั้น ในกรณีของ WLAN ผู้วิจัยตั้งใจที่จะค้นหาว่าผู้วิจัยสามารถระบุช่วงเวลาบัฟเฟอร์ที่เหมาะสมที่สุดที่พร็อกซีได้โดยอัตโนมัติเพียงใดโดยศึกษา ZWA ที่ได้รับ เซิร์ฟเวอร์บางเซิร์ฟเวอร์อาจได้ส่งทราฟฟิกสตรีมมิ่งในรูปแบบของการระเบิดแล้ว ซึ่งในกรณีนี้ไม่จำเป็นต้องให้พรอกซีเข้ามาแทรกแซง ดังนั้นผู้วิจัยจึงต้องการพัฒนาโซลูชันเพื่อตรวจจับโดยอัตโนมัติว่าเมื่อใดที่พร็อกซีควรกำหนดรูปแบบการสตรีมและเมื่อไม่เป็นเช่นนั้น สุดท้าย ผู้วิจัยวางแผนที่จะขยายพร็อกซีเซิร์ฟเวอร์ของผู้วิจัยสำหรับแอปพลิเคชันการสตรีมวิดีโอ [23] โดยงานวิจัยที่ศึกษาพลังงานของอุปกรณ์เคลื่อนที่ในลักษณะเดียวกันที่พบได้แก่ [24][25]

งานวิจัยดังที่กล่าวมาในบทนี้แสดงให้เห็นแบบวิธีการทดลองและโครงสร้างของระบบสตรีมมิ่งบนเครือข่าย LTE ซึ่งใช้เป็นมารตฐานที่ผู้วิจัยจะนำไปใช้ในงานวิจัยชิ้นนี้

บทที่ 3

ระบบ Simulation และ การทดลอง

เพื่อที่จะตอบปัญหาความเหมาะสมของสถาปัตยกรรมการสตรีมมิ่งไปยังผู้ให้บริการหลายจุดบนเครือข่าย LTE เกณฑ์ในการวัดนั้นจึงใช้เป็น QoS โดยในบทนี้ได้อธิบายประเด็นของการทดลองบนระบบจำลองทั้งการแนะนำ Simulator ที่นำมาใช้ในการทดลอง การออกแบบและการใช้งานเพื่อการทดลองสถาปัตยกรรมที่ผู้วิจัยสนใจ และการตั้งค่าการทดลองของงานวิจัยนี้

3.1 ระบบ Simulation

ตามที่ระบุไว้ก่อนหน้านี้ การศึกษานี้มุ่งเน้นไปที่การศึกษาความแตกต่างทางสถาปัตยกรรมในการนำเสนอบริการสตรีมมิงแบบสดไปยังจุดปลายหลายจุดพร้อมกัน สถาปัตยกรรมที่เลือกเพื่อการศึกษาคือ (i) สถาปัตยกรรมที่ใช้ SFU และ (ii) สถาปัตยกรรมที่ไม่ใช้ SFU เพื่อลดผลกระทบจากการรบกวนทางกายภาพ ผู้วิจัยทำการทดลองในสภาพแวดล้อมจำลองเท่านั้น

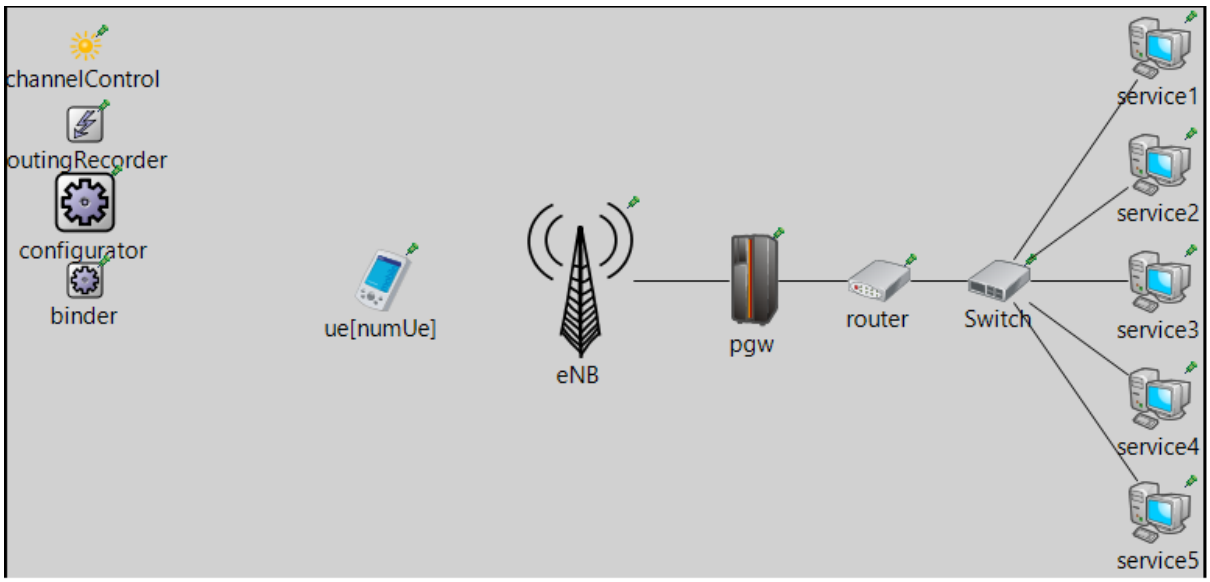
ซอฟต์แวร์ที่ใช้สำหรับทำการทดลองในการศึกษานี้คือ OMNet++ v5.6.2 (https://omnetpp.org/)เป็นซอฟท์แวร์จำลองการทำงานทางเน็ตเวิคที่มีความน่าเชื่อถือในสากล ซึ่งอุปกรณ์เครือข่ายและเครือข่าย LTE ถูกจำลองโดยใช้ปลั๊กอินสำหรับ OMNeT++ ชื่อ INET Framework (https://inet.omnetpp.org) และเนื่องจาก INET Framework ไม่ครอบคลุมในเงื่อนไขเคลือข่าย LTE ผู้วิจัยจึงขยายโมดูลเพื่อให้เหมาะสมกับงานของผู้วิจัยโดยใช้ SimuLTE มาเพิ่มในจุดนี้

การศึกษานี้ดำเนินการทดลองสองชุด ได้แก่ (i) การทดลองสถาปัตยกรรมแบบไม่ใช้ SFU และ (ii) การทดลองสถาปัตยกรรมที่ใช้ SFU รายละเอียดมีดังนี้

3.1.1 การทดลองสถาปัตยกรรมสตรีมมิ่งแบบไม่ใช้ SFU

เป้าหมายในการทดลองคือวัด QoS ที่เกิดขึ้นจากสถาปัตยกรรมการสตรีม เพื่อทำการทดลองสำหรับสถาปัตยกรรมสตรีมมิ่งแบบไม่ใช้ SFU ผู้วิจัยอิงการทำงานจากแผนภาพในรูปภาพที่ 3 นำมา

ออกแบบลงบนโปรแกรมจำลอง OMNet++ ด้วยองค์ประกอบขั้นต่ำที่สุดที่ควรจะมีในระบบของการสตรีมมิ่งบนเครือข่าย LTE เมื่ออ้างอิงจากการศึกษาแล้ว ผู้วิจัยจะได้ผลลัพธ์ดังนี้



รูปที่ 5 : สถาปัตยกรรมการสตรีมแบบไม่ใช้ SFU บน OMNet++

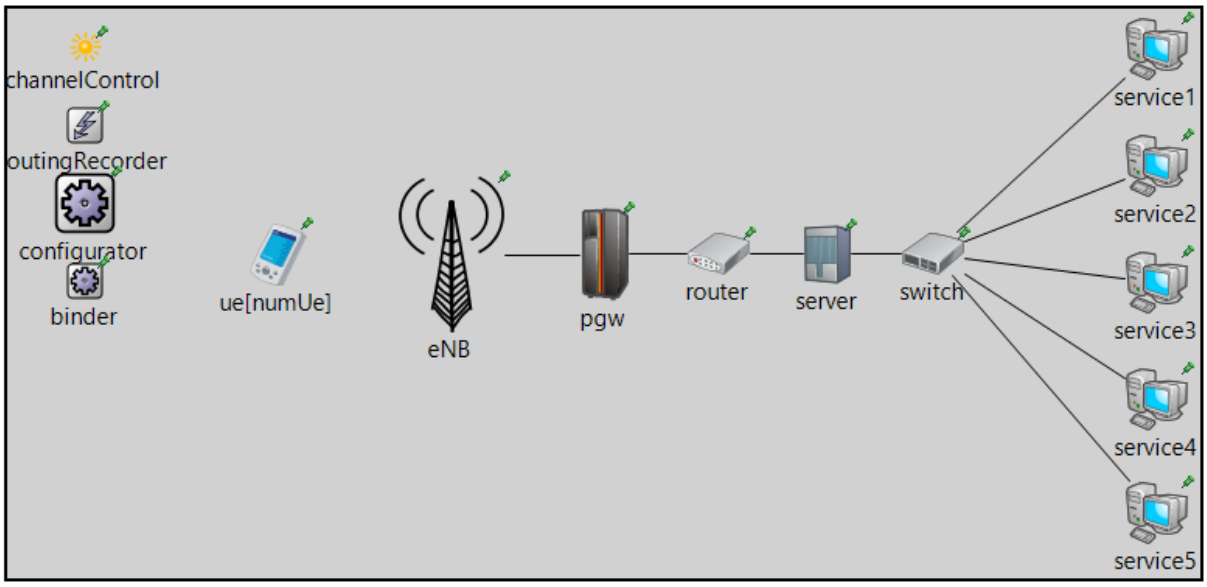
ในรูปที่ 5 การกำหนดค่าเครือข่าย LTE อย่างง่ายนั้นสร้างแบบจำลองภายใน OMNet++ ซึ่งออกแบบให้น้อยชิ้นที่สุดเท่าที่เป็นไปได้ ระบบจึงประกอบด้วย

1. โทรศัพท์หรืออุปกรณ์พกพา (Ue) แทนอุปกรณ์ที่ผู้ใช้งานฝั่งผู้สตรีม
2. เสาร์สัญญาณ (eNB) สำหรับติดต่อระหว่างโทรศัพท์ผ่านสัญญาณ LTE
3. แพ็กเก็ตเกตเวย์ (Pgw) สำหรับจัดการแพ็กเก็ต
4. Router สำหรับระบุเส้นทางการส่งข้อมูล
5. Switch สำหรับเชื่อมต่อเส้นทางและความสามารถ Broadcast
6. Service แทนผู้ให้บริการสตรีมมิ่งเว็บไซต์

ซึ่งองค์ประกอบเหล่านี้เชื่อมต่อกันตามรูปที่ 4 โดยจะใช้เสาร์สัญญาณ LTE (eNB) เพียงหนึ่งจุดซึ่งจะถูกเชื่อมต่อกับบริการ LTE Packet Gateway (PGW) อย่างง่าย จากนั้น PGW เชื่อมต่อกับ Router และ Routerเชื่อมต่อกับสวิตช์อย่างง่ายที่เชื่อมต่อกับบริการห้าจุดเพื่อทำงานเป็นจุดปลายทางสำหรับเซสชันการสตรีมสดซึ่งการเชื่อมต่อดังที่กล่าวมามีตัวกลางเป็น Lan และ ue ใช้เป็นจุดเริ่มต้นในการ

เริ่มเซสชันการสตรีมที่จะส่งต่อข้อมูลสู่ eNB ผ่านตัวกลางเป็นคลื่น Lte Network โดยพารามิเตอร์ของอุปกรณ์และตัวกลางในการส่งสัญญาณทั้งหมดจะถูกเก็บไว้ในสภาวะที่เหมาะสม

3.1.2 การทดลองสถาปัตยกรรมสตรีมมิ่งแบบใช้ SFU



รูปที่ 6 : สถาปัตยกรรมการสตรีมแบบ SFU บน OMNet++

ชุดการทดลองนี้เป็นไปตามสถาปัตยกรรมดังที่อธิบายไว้ในรูปที่ 2 การทดสอบโดยจำลอง สำหรับชุดการทดลองในรูปที่ 6 มีความแตกต่างที่มีนัยสำคัญเพียงอย่างเดียว คือ SFU เซิร์ฟเวอร์ที่วางอยู่ระหว่างเราเตอร์และสวิตช์ภายในสภาพแวดล้อมจำลองเมื่ออิงตามรูปที่ 2 เซิร์ฟเวอร์นี้ได้รับการกำหนดค่าด้วยแอปพลิเคชันที่มีหน้าที่หลักสองประการ คือ (i) รับแพ็กเก็ตสตรีมมิงขาเข้าจากอุปกรณ์เคลื่อนที่ของผู้ใช้ (ii) คัดลอกแพ็กเก็ตและส่งต่อไปยังปลายทางบริการสตรีมมิ่งทั้งห้าแห่ง จำนวนอุปกรณ์เคลื่อนที่และพารามิเตอร์การวัดจะเหมือนกันกับการทดลองที่ไม่ใช้ SFU เพื่อให้สามารถเปรียบเทียบผลลัพธ์ระหว่างชุดการทดลองแต่ละชุดได้

3.2 การกำหนดค่าและการทดลอง

การจำลองสถาปัตยกรรมทั้งสองถูกทดสอบในอุปกรณ์ชุดที่มีการตั้งค่าไว้เหมือนกันเพื่อเปรียบเทียบเฉพาะผลจากการเปลี่ยนสถาปัตยกรรม ชุดการทดลองนี้เริ่มต้นโดยเริ่มต้นการทดสอบการจำลองด้วยอุปกรณ์เคลื่อนที่จำนวนเพิ่มขึ้นอย่างมีนัยยะ ผู้วิจัยได้ใช้การทดลองห้าชุดโดยปรับใช้อุปกรณ์เคลื่อนที่เป็นจำนวน 10, 20, 30, 40 และ 50 เครื่อง ซึ่งเพียงพอที่จะสามารถคาดเดาการเปลี่ยนแปลงได้

อุปกรณ์เคลื่อนที่แต่ละเครื่องจะมีการใช้งานแอปพลิเคชันสตรีมมิงจำลองที่ใช้ TCP เพื่อเริ่มเซสชันสตรีมโดยไม่สนใจขั้นตอนในการส่งข้อมูลหากันเพื่อเชื่อมต่อก่อนการส่งเนื้อหาของสตรีม แอปพลิเคชันการสตรีมในแต่ละอุปกรณ์นั้นข้อมูลไม่ว่าจะภาพ เสียง หรืออื่น ๆ จะถูกตีความเป็นกลุ่มก้อนแพ็กเก็ต ซึ่งได้ทำการกำหนดค่าให้ส่งแพ็กเก็ตขนาด 1,500 ไบต์ในทุก 20 มิลลิวินาที ตัวเลขนี้อ้างถึงการใช้งานจริงของแอปพลิเคชั่นการสตรีมที่มีขนาดมัลติมีเดียมากที่สุดเท่าที่จะเป็นไปได้และช่วงเวลาการแพ็กเก็ต 20ms สำหรับทุกแพ็กเก็ต[26] ซึ่งในกรณีของสถาปัตยกรรมแบบไม่ใช้ SFU อุปกรณ์แต่ละตัวจะต้องคัดลอกและส่งแต่ละแพ็กเก็ตแบบเดียวกันนี้ไปยังจุดปลายบริการทั้งห้าออกไปอย่างพร้อมกัน ในอีกด้านหนึ่งคือสถาปัตยกรรมแบบใช้ SFU นั้น อุปกรณ์แต่ละตัวจะส่งแพ็กเก็ตออกไปยังเซิร์ฟเวอร์เพียงสำเนาเดียวแล้วแพ็กเก็ตจะถูกคัดลอกโดยเซิร์ฟเวอร์และส่งต่อไปยังแต่ละจุดปลายทางบริการทั้งห้า ซึ่งการทำงานในลักษณะนี้ไม่ได้มีมาแต่ดั้งเดิมจึงจำเป็นต้องสร้างการทำงานนี้เพิ่มเข้าไป

ความล่าช้าจากต้นทางถึงปลายทางถูกใช้เป็นพารามิเตอร์การวัด การหน่วงเวลาแบบ end-to-end คำนวณจากความแตกต่างของเวลาระหว่างเวลาที่แพ็กเก็ตถูกส่งจากอุปกรณ์เคลื่อนที่จนกระทั่งเมื่อมาถึงจุดสิ้นสุดของบริการที่เกี่ยวข้อง การวัดนี้ทำได้โดยการใส่ Timestamp เข้าไปในแพ็กเก็ตเมื่อถูกส่งออกจากต้นทางแล้วเมื่อแพ็กเก็ตมาถึงปลายทางจะนำเวลาที่แฝงมากับแพ็กเก็ตเทียบเวลาปัจจุบันทำให้ได้ความต่างของเวลาประจำตัวของแพ็กเก็ตนั้น ๆ

3.3 การกำหนดค่าใน simulator

นอกจากโครงสร้างของการทดลองจะถูกตั้งในโปรแกรมจำลองเพื่อใช้ทดสอบแล้ว ส่วนของค่าที่กำหนดนั้นก็จำเป็นจะถูกนำไปกำหนดในโปรแกรมจำลองอย่างถูกต้องอีกด้วย ซึ่งการทดลองทั้ง 2

สถาปัตยกรรมนั้นมีส่วนที่แตกต่างกันที่ผู้วิจัยจะต้องออกแบบและเขียนโปรแกรมเพิ่มเติมเข้าไปซึ่งอธิบายได้ดังนี้

3.3.1 การปรับแต่ง Applications

Applications ที่ว่านั้นเป็นรูปแบบการทำงานที่ถูกใช้กันเป็นมาตรฐานอย่างในงานของผู้วิจัยนั้นจะเป็น tcpapp จากทาง inet เป็นผู้ออกแบบไว้ ซึ่งโดยพื้นฐานสามารถใช้วัดการสตรีมได้แต่เนื่องจากงานของผู้วิจัยมีรูปแบบที่ไม่มีในพื้นฐานผู้วิจัยจึงจำเป็นต้องปรับปรุงหรือสร้างเพิ่มดังนี้

TcpBombApp เป็นรูปแบบการทำงานการสตรีมแพ็คเก็ตโดยประยุคมาจาก TcpEchoApp ที่เป็นรูปแบบการทำงานการสตรีมแบบสะท้อนข้อความกลับแบบพื้นฐาน ซึ่งผู้วิจัยปรับแต่งเพื่อให้สอดคล้องกับโครงสร้างการทำงานของสถาปัตยกรรมแบบที่ใช้ SFU โดยการทำงานของ TcpBombApp นั้นจะทำการรับ package มาแล้วทำสำเนาเพื่อกระจายต่อไปยังปลายทางที่ระบุไว้ TcpBombApp จะถูกใช้ที่ตำแหน่ง sever ซึ่งหลักๆแล้วจะมี 2 ส่วนคือ 1.การจัดการเมื่อได้รับแพ็กเก็ต 2.การจัดการเมื่อส่งออกแพ็กเก็ค ความสามารถที่ใช้อธิบายได้ตามนี้

- เมื่อข้อมูลหรือแพ็กเก็ตถูกรับรู้ว่ามาถึงยังตำแหน่ง server จะถูกตรวจสอบเวลาโดยเช็คจากเวลาที่แพ็กเก็ตถูกสร้างขึ้นลบกับเวลาที่มาถึงเพื่อใช้สำหรับตรวจสอบความล่าช้าที่เกิดขึ้น นอกจากนั้นในระบบจะทำสำเนาแพ็กเก็ตไว้เพื่อเตรียมกระจายออกไปยังจุดหมายปลายทาง ซึ่งในการทดลองนี้ ผู้วิจัยได้ระบุตำแหน่งปลายทางไว้อย่างชัดเจนโดนการต่อ socket ไปยัง ID (ในการทดลองของผู้วิจัย ID ของปลายทางคือ 14, 15, 16, 17 และ 18)ของปลายทางทั้ง 5 ที่ตลอดเวลา โดยการอธิบายที่ผ่านมานำมาเขียนโปรแกรมได้ดังนี้

```
void TcpBombAppThread::dataArrived(Packet *rcvdPkt, bool urgent)
{
    echoAppModule->emit(packetReceivedSignal, rcvdPkt);
    int64_t rcvdBytes = rcvdPkt->getByteLength();
    echoAppModule->bytesRcvd += rcvdBytes;
    auto data = rcvdPkt->peekData();
      auto regions = data->getAllTags<CreationTimeTag>();
      for (auto& region : regions) {
        auto creationTime = region.getTag()->getCreationTime();
        auto delay = simTime() - creationTime;
        cout << region.getOffset() << region.getLength() << delay;
      }
    if (echoAppModule->echoFactor > 0 && sock->getState() ==
TcpSocket::CONNECTED ) {
       Packet *outPkt = new Packet(rcvdPkt->getName(), TCP_C_SEND);
```

```
        Packet *outPkt1 = new Packet(rcvdPkt->getName(), TCP_C_SEND);
        Packet *outPkt2 = new Packet(rcvdPkt->getName(), TCP_C_SEND);
        Packet *outPkt3 = new Packet(rcvdPkt->getName(), TCP_C_SEND);
        Packet *outPkt4 = new Packet(rcvdPkt->getName(), TCP_C_SEND);

        outPkt->addTag<SocketReq>()->setSocketId(14);
        outPkt1->addTag<SocketReq>()->setSocketId(15);
        outPkt2->addTag<SocketReq>()->setSocketId(16);
        outPkt3->addTag<SocketReq>()->setSocketId(17);
        outPkt4->addTag<SocketReq>()->setSocketId(18);

        long outByteLen = rcvdBytes * echoAppModule->echoFactor;

        if (outByteLen < 1)
                outByteLen = 1;

       int64_t len = 0;
       for ( ; len + rcvdBytes <= outByteLen; len += rcvdBytes) {
                outPkt->insertAtBack(rcvdPkt->peekDataAt(B(0), B(rcvdBytes)));
                outPkt1->insertAtBack(rcvdPkt->peekDataAt(B(0),
B(rcvdBytes)));
                outPkt2->insertAtBack(rcvdPkt->peekDataAt(B(0),
B(rcvdBytes)));
                outPkt3->insertAtBack(rcvdPkt->peekDataAt(B(0),
B(rcvdBytes)));
                outPkt4->insertAtBack(rcvdPkt->peekDataAt(B(0),
B(rcvdBytes)));
        }
        if (len < outByteLen){
                outPkt->insertAtBack(rcvdPkt->peekDataAt(B(0), B(outByteLen -
len)));
                outPkt1->insertAtBack(rcvdPkt->peekDataAt(B(0), B(outByteLen -
len)));
                outPkt2->insertAtBack(rcvdPkt->peekDataAt(B(0), B(outByteLen -
len)));
                outPkt3->insertAtBack(rcvdPkt->peekDataAt(B(0), B(outByteLen -
len)));
                outPkt4->insertAtBack(rcvdPkt->peekDataAt(B(0), B(outByteLen -
len)));
        }
                   ASSERT(outPkt->getByteLength() == outByteLen);
                   ASSERT(outPkt1->getByteLength() == outByteLen);
                   ASSERT(outPkt2->getByteLength() == outByteLen);
                   ASSERT(outPkt3->getByteLength() == outByteLen);
                   ASSERT(outPkt4->getByteLength() == outByteLen);

                   if (echoAppModule->delay == 0){
                       echoAppModule->sendDown(outPkt);
                       echoAppModule->sendDown(outPkt1);
                       echoAppModule->sendDown(outPkt2);
                       echoAppModule->sendDown(outPkt3);
                       echoAppModule->sendDown(outPkt4);
                   }else{
                       scheduleAt(simTime() + echoAppModule->delay, outPkt);
                   }
```

```
            }
        }
    }
    delete rcvdPkt;
}
```

- อีกส่วนหนึ่งคือการส่งออกแพ็คเก็ตจะทำการส่งออกแพ็คเก็ตที่สำเนาไว้ไปยังตำแหน่ง socket ปลายทางที่ระบุ ซึ่งผู้วิจัยจะเก็บข้อมูล byte ที่ผ่านทาง Socket ที่ได้ต่อไว้เพื่อใช้ตรวจสอบการสูญหายของข้อมูลที่เกิดขึ้น นำมาเขียนโปรแกรมได้ดังนี้

```
void TcpBombApp::sendDown(Packet *msg)
{
    if (msg->isPacket()) {
        Packet *pk = static_cast<Packet *>(msg);
        bytesSent += pk->getByteLength();
        emit(packetSentSignal, pk);
    }
    msg->addTagIfAbsent<DispatchProtocolReq>()-
>setProtocol(&Protocol::tcp);
    msg->getTag<SocketReq>();

    send(msg, "socketOut");
    cout << "==================================" << endl;
}
```

3.3.2 การตั้งค่า init

นอกจากการเขียน Application ในการออกแบบการทำงานของระบบแล้ว ผู้วิจัยยังต้องกำหนดค่าคงที่ต่าง ๆ ที่ผู้วิจัยใช้ในการทดลอง การตั้งค่าเริ่มต้นสำหรับ 2 สถาปัตยกรรมได้แก่ ระยะเวลาในการทดลอง ความสามารถพื้นฐานของอุปกรณ์ต่าง ๆ ขนาดแพ็กเก็ต ระยะทางรวม จะไม่แตกต่างกัน สิ่งที่จะต่างกันก็คือลำดับการเชื่อมต่อซึ่งแสดงได้ดังนี้

3.3.2.1แบบที่ไม่ใช้ SFU

ผู้วิจัยตั้งค่าให้ sevice ทั้ง5ทำการเปิด port ประจำตัวไว้และ ue จะเชื่อมต่อกับ port ทั้ง 5 เสมอ

```
network = Node
num-rngs=1
image-path=../../images
output-scalar-file-append = false
sim-time-limit=30s
############### Statistics ##################
output-scalar-file = ${resultdir}/${configname}/${repetition}.sca
output-vector-file = ${resultdir}/${configname}/${repetition}.vec
seed-set = ${repetition}
```

```
**.vector-recording = false
################ Mobility parameters #####################
**.mobility.constraintAreaMinZ = 0m
**.mobility.constraintAreaMaxZ = 0m
**.mobility.initFromDisplayString = true
############### Number of Resource Blocks ################
**.numRbDl = 6
**.numRbUl = 6
**.binder.numBands = 6 # this value should be kept equal to the number of
RBs

############### Transmission Power ##################
**.ueTxPower = 26
**.eNodeBTxPower = 40
**.ue[*].numApps = 5
**.mobility.constraintAreaMinX = 0m
**.mobility.constraintAreaMinY = 0m
**.mobility.constraintAreaMaxX = 800m
**.mobility.constraintAreaMaxY = 400m

**.ue[*].mobility.initialX = uniform(300m,330m)
**.ue[*].mobility.initialY = uniform(250m,260m)
**.ue[*].app[*].typename = "TcpSessionNew"
**.ue[*].app[*].active = true
**.ue[*].app[*].localPort = -1
**.ue[*].app[0].connectAddress = "service1"
**.ue[*].app[0].connectPort = 3099
**.ue[*].app[0].tOpen = 0.2s
**.ue[*].app[1].tOpen = 0.21s
**.ue[*].app[2].tOpen = 0.22s
**.ue[*].app[3].tOpen = 0.23s
**.ue[*].app[4].tOpen = 0.24s
**.ue[*].app[*].tSend = 0.4s
**.ue[*].app[*].sendBytes = 819B
**.ue[*].app[*].sendScript = ""
**.ue[*].app[*].tClose = 60s
**.ue[*].app[*].dataTransferMode = "bytestream"
**.ue[*].app[1].connectAddress = "service2"
**.ue[*].app[1].connectPort = 3088
**.ue[*].app[2].connectAddress = "service3"
**.ue[*].app[2].connectPort = 3077
**.ue[*].app[3].connectAddress = "service4"
**.ue[*].app[3].connectPort = 3066
**.ue[*].app[4].connectAddress = "service5"
**.ue[*].app[4].connectPort = 3055

**.service*.numApps = 1
**.service*.app[*].typename = "TcpSinkNew"
**.service1.app[0].localPort = 3099
**.service2.app[0].localPort = 3088
**.service3.app[0].localPort = 3077
**.service4.app[0].localPort = 3066
**.service5.app[0].localPort = 3055

**.ue[*].masterId = 1
```

```
**.ue[*].macCellId = 1
**.ue[*].mobility.initFromDisplayString = false
**.ue[*].mobilityType = "StationaryMobility"
```

3.3.2.2 แบบที่ใช้ SFU

ผู้วิจัยจะให้ sever เปิด port ประจำตัวไว้ แล้วทาง sevice และ ue จะต่อเข้ามา

```
[General]
network = TcpLastTest
#*.numTarget = 5
total-stack = 70000MiB

*.visualizer.*.interfaceTableVisualizer.displayInterfaceTables = true

**.server.numPcapRecorders = 0
**.server.pcapRecorder[0].pcapFile = "results/${configname}-${runnumber}-
server.pcap"

##
*.client2.numApps = 1
**.client2.app[*].typename = "TcpSessionApp"
**.client2.app[0].active = true
**.client2.app[0].localPort = -1
**.client2.app[0].connectAddress = "server"
**.client2.app[0].connectPort = 1000
**.client2.app[0].tOpen = 0.21s
**.client2.app[0].tClose = 30s

*.client3.numApps = 1
**.client3.app[*].typename = "TcpSessionApp"
**.client3.app[0].active = true
**.client3.app[0].localPort = -1
**.client3.app[0].connectAddress = "server"
**.client3.app[0].connectPort = 1000
**.client3.app[0].tOpen = 0.21s
**.client3.app[0].tClose = 30s

*.client4.numApps = 1
**.client4.app[*].typename = "TcpSessionApp"
**.client4.app[0].active = true
**.client4.app[0].localPort = -1
**.client4.app[0].connectAddress = "server"
**.client4.app[0].connectPort = 1000
**.client4.app[0].tOpen = 0.21s
**.client4.app[0].tClose = 30s

*.client5.numApps = 1
**.client5.app[*].typename = "TcpSessionApp"
**.client5.app[0].active = true
**.client5.app[0].localPort = -1
**.client5.app[0].connectAddress = "server"
```

```
**.client5.app[0].connectPort = 1000
**.client5.app[0].tOpen = 0.21s
**.client5.app[0].tClose = 30s

*.client6.numApps = 1
**.client6.app[*].typename = "TcpSessionApp"
**.client6.app[0].active = true
**.client6.app[0].localPort = -1
**.client6.app[0].connectAddress = "server"
**.client6.app[0].connectPort = 1000
**.client6.app[0].tOpen = 0.21s
**.client6.app[0].tClose = 30s

##
**.server*.numApps = 1
**.server*.app[*].typename = "TcpBombApp"
**.server*.app[0].localPort = 1000
**.server*.app[0].echoFactor = 2.0
**.server*.app[0].echoDelay = 0s

##LTE
image-path=../../images
output-scalar-file-append = false
sim-time-limit=30s

################ Statistics ##################
output-scalar-file = ${resultdir}/${configname}/${repetition}.sca
output-vector-file = ${resultdir}/${configname}/${repetition}.vec
seed-set = ${repetition}
**.vector-recording = false

################ Mobility parameters #####################
# *
**.mobility.constraintAreaMinZ = 0m
**.mobility.constraintAreaMaxZ = 0m
**.mobility.initFromDisplayString = true

################ Number of Resource Blocks ################
**.numRbDl = 6
**.numRbUl = 6
**.binder.numBands = 6 # this value should be kept equal to the number of
RBs

################ Transmission Power ##################
**.ueTxPower = 26
**.eNodeBTxPower = 40

##1-5
*.n = 5
description = direct stream.

*.configuratorLTE.config = xmldoc("demo.xml")

**.numUe = 1 # ${numUes=1,2,5,10,20,50,100}
```

```
**.mobility.constraintAreaMinX = 300m
**.mobility.constraintAreaMinY = 200m
**.mobility.constraintAreaMaxX = 800m
**.mobility.constraintAreaMaxY = 400m

**.ue[*].masterId = 1
**.ue[*].macCellId = 1
**.ue[*].mobility.initFromDisplayString = false
**.ue[*].mobilityType = "StationaryMobility"

*.ue[*].numApps = 1
**.ue[*].app[*].typename = "TcpSessionApp"
**.ue[*].app[0].active = true
**.ue[*].app[0].localPort = -1
**.ue[*].app[0].connectAddress = "server"
**.ue[*].app[0].connectPort = 1000
**.ue[*].app[0].tOpen = 0.2s
**.ue[*].app[0].tSend = 0.4s
**.ue[*].app[0].sendBytes = 819B
**.ue[*].app[0].sendScript = ""
**.ue[*].app[0].tClose = 30s
**.ue[*].app[0].dataTransferMode = "bytestream"
```

บทที่ 4

ผลการทดลอง

ในบทนี้ผู้วิจัยจะกล่าวถึงผลการทดลองจากการทดลองในบทที่ 3 โดยทำการทดลอง 3 ครั้ง สำหรับแต่ละชุดการทดลองความล่าช้าแบบ End-to-end ของการทดสอบแต่ละรอบและทำบันทึกผลลัพธ์เป็นค่าเฉลี่ยนั้น ๆ และวิเคราะห์ผลทางสถิติเพื่อใช้เปรียบเทียบประสิทธิภาพด้าน QoS ของ 2 สถาปัตยกรรมในการสตรีมไปยังหลายจุดบนเครือข่าย LTE

4.1 ผลการทดลอง

การทดลองสตรีมแบบไม่ใช้ SFU ให้ผลลัพธ์ดังนี้

ตารางที่ 2 : ผลการทดลองสถาปัตยกรรมที่ไม่ใช้ SFU

จำนวนของอุปกรณ์เคลื่อนที่ (Ue)	ค่าเฉลี่ยดีเลย์ที่จุดปลายทางของการสตรีม (มิลลิวินาที)				
	จุดปลายทางที่ 1	จุดปลายทางที่ 2	จุดปลายทางที่ 3	จุดปลายทางที่ 4	จุดปลายทางที่ 5
10	8,313.66	161.75	449.26	1,205.23	2,911.26
20	6,954.93	711.96	1,033.98	6,290.58	451.31
30	4,801.40	2,545.29	4,996.15	675.94	16,689.47
40	1,770.92	1,706.65	2,777.08	5,958.83	18,057.17
50	9,280.92	8,117.61	4,378.80	496.75	1,064.32

จะเห็นได้ว่าจากตารางที่ 2 ผลลัพธ์การดีเลย์ของแต่ละจุดปลายทางในกรณีที่จำนวนอุปกรณ์เคลื่อนที่เท่ากันนั้น มีความแปรปรวนค่อนข้างสูง และคาดเดาแนวทางประสิทธิภาพต่อการเพิ่มขึ้นของอุปกรณ์เคลื่อนที่ได้ยาก

การทดลองสตรีมแบบใช้ SFU ให้ผลลัพธ์ดังนี้

ตารางที่ 3 : ผลการทดลองสถาปัตยกรรมที่ใช้ SFU

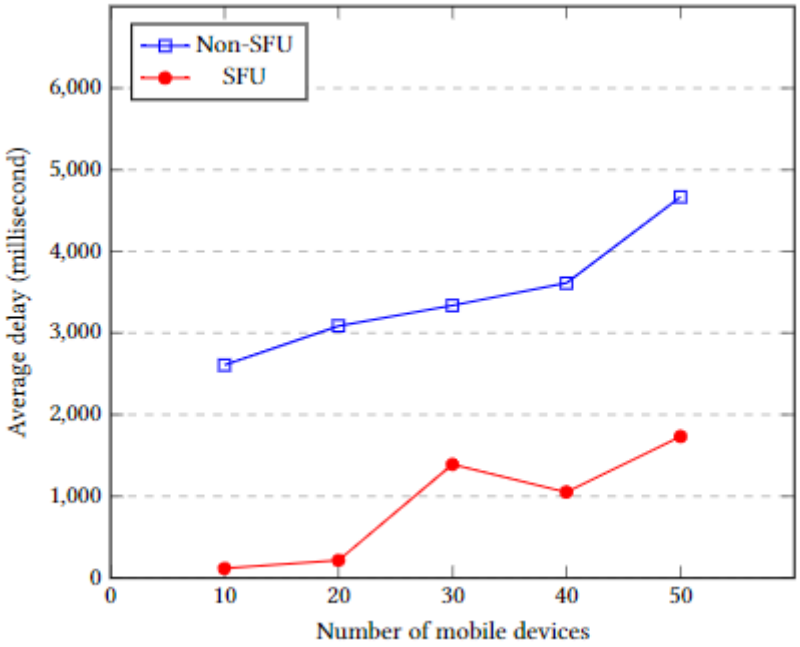
จำนวนของอุปกรณ์เคลื่อนที่ (Ue)	ค่าเฉลี่ยดีเลย์ที่จุดปลายทางของการสตรีม (มิลลิวินาที)				
	จุดปลายทางที่ 1	จุดปลายทางที่ 2	จุดปลายทางที่ 3	จุดปลายทางที่ 4	จุดปลายทางที่ 5
10	117.43	117.44	117.44	117.44	117.44
20	217.15	217.15	217.15	217.15	217.15
30	1,391.32	1,391.32	1,391.32	1,391.32	1,391.32
40	1,052.24	1,052.24	1,052.24	1,052.24	1,052.24
50	1,733.24	1,733.10	1,733.11	1,733.11	1,733.11

สามารถคำนวณค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานของแต่ละชุดการทดลองได้ดังตารางที่ 4

ตารางที่ 4 : ค่าทางสถิติของผลการทดลองทั้งสองสถาปัตยกรรม

จำนวนของอุปกรณ์เคลื่อนที่ (Ue)	แบบไม่ใช้ SFU		แบบใช้ SFU	
	ค่าเฉลี่ย	ค่าเบี่ยงเบนมาตรฐาน	ค่าเฉลี่ย	ค่าเบี่ยงเบนมาตรฐาน
10	2,608.23	3,363.72	117.44	0.0009
20	3,088.55	3,241.38	217.15	0.0008
30	3,337.89	1,783.02	1,391.32	0.0005
40	3,611.43	2,133.46	1,052.24	0.0007
50	4,667.68	3,988.99	1,733.24	0.0612
	3,462.76	**2,902.12**	**902.28**	**0.0128**

กราฟการเปรียบเทียบ



รูปที่ 7 : ความล่าช้าจากต้นทางถึงปลายทางโดยเฉลี่ย: การเปรียบเทียบระหว่างสถาปัตยกรรมที่ไม่ใช้ SFU และใช้ SFU

ผลลัพธ์เกิดตรงกันข้ามกับการทดลองสถาปัตยกรรมแบบไม่ใช้ SFU การทดสอบที่ใช้ SFU เผยให้เห็นว่าการหน่วงเวลาเฉลี่ยที่จุดสิ้นสุดแต่ละจุดมีแนวโน้มใกล้เคียงกัน สังเกตว่าค่าเฉลี่ยที่จุดสิ้นสุดแต่ละจุดในตารางที่ 3 ส่วนใหญ่จะคล้ายกัน ผลลัพธ์แสดงความแตกต่างเล็กน้อยในความล่าช้าเฉลี่ยที่ความแม่นยำของตัวเลขทศนิยม 4-5 หลัก ผู้วิจัยถือว่าความแตกต่างดังกล่าวไม่ถือว่ามีนัยสำคัญ ดังนั้นผู้วิจัยจึงปัดเศษผลลัพธ์ด้วยความแม่นยำของตัวเลขทศนิยม 2 หลักเพื่อให้ผลลัพธ์ง่ายสำหรับผู้อ่าน เพื่อเปรียบเทียบรูปแบบของผลการหน่วงเวลาแบบ end-to-end ระหว่างการทดสอบแบบไม่ใช้ SFU และการทดสอบแบบ SFU ผู้วิจัยดำเนินการวิเคราะห์เพิ่มเติมโดยคำนวณการหน่วงเวลาแบบ end-to-end โดยเฉลี่ยที่จุดสิ้นสุดแต่ละจุดสำหรับการทดสอบแต่ละครั้งและบรรยาย โดยใช้แผนภูมิเส้น ผลลัพธ์แสดงไว้ในรูปที่ 7 ผลลัพธ์แสดงให้เห็นว่าการหน่วงเวลาแบบ end-to-end สำหรับสถาปัตยกรรมที่ใช้ SFU นั้นต่ำกว่ามากในทุกกรณี

4.2 อภิปรายผล

ผลลัพธ์ของการทดลองในกรณีสถาปัตยกรรมที่ไม่ใช้ SFU แสดงให้เห็นความผันผวนของดีเลย์โดยเฉลี่ยจากต้นทางถึงปลายทางในทุกกรณีการทดลอง ซึ่งหมายความว่าแม้ว่าสถาปัตยกรรมที่ไม่ใช้ SFU อาจมีข้อได้เปรียบเหนือสถาปัตยกรรมที่ใช้ SFU ในแง่ของการประหยัดต้นทุน (กล่าวคือ ไม่จำเป็นต้องใช้เซิร์ฟเวอร์ SFU แยกต่างหาก) คุณภาพของบริการในสถาปัตยกรรมที่ไม่ใช่ SFU อาจเป็นข้อเสียที่สำคัญ สิ่งนี้มาจากข้อเท็จจริงที่ว่าในกรณีที่ไม่ใช่ SFU ดังนั้นการดีเลย์ของเวลาจากต้นทางถึงปลายทางโดยเฉลี่ยจึงผันผวนเนื่องจากอุปกรณ์เคลื่อนที่แต่ละเครื่องอาจต้องรอนานขึ้นที่จุดเสาร์วิทยุสื่อสารจะพร้อมใช้งานสำหรับการส่งแพ็กเก็ต ในทางกลับกัน การทดลองสถาปัตยกรรมที่ใช้ SFU ทำให้เกิดความผวนน้อยลงในความล่าช้าแบบ End-to-end โดยเฉลี่ยนี้มาจากข้อเท็จจริงที่ว่าการแย่งชิงเครือข่ายที่ไซต์เซลล์ LTE ในการทดลองจำลองนั้นต่ำกว่าการทดทดสอบที่ไม่ใช่ SFU ถึงห้าเท่า ซึ่งหมายความว่าการควบคุมคุณภาพโดยรวมของการบริการสำหรับบริการสตรีมมิ่งจะง่ายขึ้นหากบริการเลือกที่จะใช้สถาปัตยกรรมที่ใช้ SFU แทนที่จะเป็นแบบที่ไม่ใช้ SFU

การศึกษานี้อนุมานว่าอุปกรณ์พกพาไม่ใช่อุปกรณ์พกพา พวกเขา อยู่นิ่งตลอดการทดลองทั้งหมดเพื่อให้การจำลอง รุ่นที่เรียบง่ายและปราศจากการรบกวนข้ามเลเยอร์ นอกจากนี้ผู้วิจัยใช้เสาร์สัญญาณ LTE เดียวเท่านั้นในการทดลองจำลอง เพราะฉะนั้นผลลัพธ์อาจแตกต่างกันในกรณีที่อุปกรณ์เคลื่อนที่สามารถเคลื่อนย้ายได้และเสาร์สัญญาณ LTE มีมากกว่าหนึ่งแห่ง กรณีเหล่านี้จะเป็นงานที่ผู้วิจัยจะทดลองเพิ่มเติมในอนาคต

บทที่ 5

สรุปผลการทดลองและข้อเสนอแนะ

ในบทนี้ผู้วิจัยจะกล่าวถึงการสรุปผลการทดลองที่เกิดขึ้นจากบทที่ 4 และเสนอแนะการทดลองที่จะเกิดขึ้นในอนาคตหรือต่อยอดจากงานชิ้นนี้

5.1 สรุปผลการทดลอง

การศึกษานี้เน้นศึกษาผลกระทบของความแตกต่างทางสถาปัตยกรรมของแอปพลิเคชันสตรีมมิงแบบสดผ่านเครือข่าย LTE ในการสตรีมไปยังปลายทางบริการหลายจุดพร้อมกัน (เช่น การสตรีมสดไปยัง Facebook Live, Youtube และเวลา Twitch) กำลังดำเนินการศึกษาใช้การทดลองโดยใช้ OMNet++ กับปลั๊กอิน INET Framework และติดตั้งปลั๊กอิน SimuLTE มีการศึกษาสถาปัตยกรรมสองประเภท ได้แก่ สถาปัตยกรรมแบบ SFU และสถาปัตยกรรมที่ไม่ใช้ SFU ผลปรากฏว่าแม้ปัจจัยทางด้านค่าใช้จ่ายของ non-SFU based จะดีกว่าแบบ สถาปัตยกรรมสถาปัตยกรรมที่ใช้ SFU เนื่องจากไม่มีค่าใช้จ่ายอุปกรณ์ SFU แต่พบว่าแบบที่ใช้ SFU มีข้อดีในการจัดการหาเนื้อหาเครือข่ายน้อยลงที่ไซต์เซลล์ของเครือข่าย LTE ซึ่งทำให้การควบคุมคุณภาพการบริการ สามารถเข้าถึงได้มากกว่ากรณีที่ไม่ใช้ SFU

5.2 ข้อเสนอแนะ

จากที่กล่าวในสรุปผลการทดลอง สถานการณ์ของการศึกษานี้จำกัดเฉพาะกรณีที่อุปกรณ์เคลื่อนที่ อยู่กับที่ระหว่างการทดลอง และมีไซต์เซลล์ LTE เพียงแห่งเดียวเท่านั้น ทำให้ผลการทดลองนี้จะไม่สามารถตีความได้ในกรณีอุปกรณ์มีการเคลื่อนที่ ดังนั้นงานในอนาคตที่สามารถต่อยอดได้คือการตรวจสอบเพิ่มเติมหากมือถืออุปกรณ์สามารถเคลื่อนที่ได้และเมื่อมีการเพิ่มไซต์เซลล์ LTE มากกว่าหนึ่งแห่ง ผลของการอยู่ร่วมกับแอปพลิเคชันอื่น เช่น การดำเนินการ เซสชั่นการสตรีมสดในขณะที่ลูกค้ารายอื่นกำลังเล่นเกมออนไลน์อาจเป็นอีกแง่มุมที่น่าตื่นเต้นในการตรวจสอบในอนาคต

บรรณานุกรม

[1] L. Hu and Q. Ming, “Live Commerce: Understanding How Live Streaming Influences Sales and Reviews,” *ICIS 2020 Proc.*, Dec. 2020, Accessed: Jul. 26, 2021. [Online]. Available: https://aisel.aisnet.org/icis2020/digital_commerce/digital_commerce/6

[2] C. Montag, E. Wegmann, R. Sariyska, Z. Demetrovics, and M. Brand, “How to overcome taxonomical problems in the study of Internet use disorders and what to do with ‘smartphone addiction’?,” *J. Behav. Addict.*, vol. 9, no. 4, pp. 908–914, Jan. 2021, doi: 10.1556/2006.8.2019.59.

[3] X. Li, M. A. Salehi, and M. Bayoumi, “VLSC: Video live streaming using cloud services,” *Proc. - 2016 IEEE Int. Conf. Big Data Cloud Comput. BDCloud 2016, Soc. Comput. Networking, Soc. 2016 Sustain. Comput. Commun. Sustain. 2016*, pp. 595–600, Oct. 2016, doi: 10.1109/BDCLOUD-SOCIALCOM-SUSTAINCOM.2016.93.

[4] M. D. Schmill, L. M. Gordon, N. R. Magliocca, E. C. Ellis, and T. Oates, “GLOBE: Analytics for assessing global representativeness,” *Proc. - 5th Int. Conf. Comput. Geospatial Res. Appl. COM.Geo 2014*, pp. 25–32, Sep. 2014, doi: 10.1109/COM.GEO.2014.21.

[5] M. Siekkinen, M. A. Hoque, J. K. Nurminen, and M. Aalto, “Streaming over 3G and LTE: How to Save Smartphone Energy in Radio Access Network-Friendly Way,” *Proc. 5th Work. Mob. Video - MoVid ’13*, 2013, doi: 10.1145/2457413.

[6] BrienzaSimone, C. Efsun, M. Saeid, HlavacsHelmut, ÖzkasapÖznur, and AnastasiGiuseppe, “A Survey on Energy Efficiency in P2P Systems,” *ACM Comput. Surv.*, vol. 48, no. 3, Dec. 2015, doi: 10.1145/2835374.

[7] D. Wu, Y. T. Hou, W. Zhu, Y. Q. Zhang, and J. M. Peha, “Streaming video over the internet: Approaches and directions,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 282–300, Mar. 2001, doi: 10.1109/76.911156.

[8] Y. Sahraoui, A. Ghanam, S. Zaidi, S. Bitam, and A. Mellouk, “Performance evaluation of TCP and UDP based video streaming in vehicular ad-hoc networks,” *2018 Int. Conf. Smart Commun. Netw. Technol. SaCoNeT 2018*, pp. 67–72, Dec. 2018, doi: 10.1109/SACONET.2018.8585447.

[9] A. Jorge and S. Cardoso, “QUALITY OF SERVICE AND SEMANTIC COMPOSITION OF WORKFLOWS”.

[10] M. Wang, C. Xu, S. Jia, and G.-M. Muntean, “Video Streaming Distribution Over Mobile Internet: A Survey,” *Front.Comput.Sci*.

[11] A. Santos, E. Pinheiro, R. Mateus, A. Odeh, and E. A. Fattah, “Analysis of video streaming using LTE technology,” *2015 IEEE Long Isl. Syst. Appl. Technol. Conf. LISAT 2015*, Jul. 2015, doi: 10.1109/LISAT.2015.7160213.

[12] H. F. Bermudez, R. Sanchez-Iborra, J. L. Arciniegas, W. Y. Campo, and M. D. Cano, “Performance validation of NS3-LTE emulation for live video streaming under QoS parameters,” *Int. Conf. Wirel. Mob. Comput. Netw. Commun.*, vol. 2017-October, pp.

300–307, Nov. 2017, doi: 10.1109/WIMOB.2017.8115836.

[13] M. Iturralde, T. Galezowski, and X. Lagrange, “Performance of mobile relays in loaded conditions for railway transportation,” *Proc. 2018 16th Int. Conf. Intell. Transp. Syst. Telecommun. ITST 2018*, Dec. 2018, doi: 10.1109/ITST.2018.8566864.

[14] Y. Chen, F. Yan, and X. Lagrange, “Performance analysis of cellular networks with mobile relays under different modes,” *Telecommun. Syst. 2017 662*, vol. 66, no. 2, pp. 217–231, Feb. 2017, doi: 10.1007/S11235-017-0284-5.

[15] M. Solera, M. Toril, I. Palomo, G. Gomez, and J. Poncela, “A Testbed for Evaluating Video Streaming Services in LTE,” *Wirel. Pers. Commun.*, vol. 98, no. 3, pp. 2753–2773, Feb. 2018, doi: 10.1007/S11277-017-4999-0/TABLES/6.

[16] F. Krasniqi, A. Maraj, and E. Blaka, “Performance analysis of mobile 4G/LTE networks,” *South-East Eur. Des. Autom. Comput. Eng. Comput. Networks Soc. Media Conf. SEEDA_CECNSM 2018*, Nov. 2018, doi: 10.23919/SEEDA-CECNSM.2018.8544937.

[17] TSGS, “TS 132 450 - V9.1.0 - Universal Mobile Telecommunications System (UMTS); LTE; Telecommunication management; Key Performance Indicators (KPI) for Evolved Universal Terrestrial Radio Access Network (E-UTRAN): Definitions (3GPP TS 32.450 version 9.1.0 Release 9),” 2010, Accessed: Jun. 21, 2022. [Online]. Available: http://portal.etsi.org/chaircor/ETSI_support.asp

[18] “Methodology of Cognitive computing : LTE Network Architecture.” http://ershoeb.blogspot.com/2016/03/lte-network-architecture.html (accessed Jun. 21, 2022).

[19] N. Kryvinska, C. Strauss, B. Collini-Nocker, and P. Zinterhof, “Enterprise network maintaining mobility - architectural model of services delivery,” *Int. J. Pervasive Comput. Commun.*, vol. 7, no. 2, pp. 114–131, Jun. 2011, doi: 10.1108/17427371111146419/FULL/XML.

[20] Y. Xu and F. Liu, “QoS Provisionings for Device-to-Device Content Delivery in Cellular Networks,” *IEEE Trans. Multimed.*, vol. 19, no. 11, pp. 2597–2608, Nov. 2017, doi: 10.1109/TMM.2017.2700208.

[21] A. Asadi, Q. Wang, and V. Mancuso, “A survey on device-to-device communication in cellular networks,” *IEEE Commun. Surv. Tutorials*, vol. 16, no. 4, pp. 1801–1819, 2014, doi: 10.1109/COMST.2014.2319555.

[22] S. G. Lee, J. Park, and H. Kim, “A user-side energy-saving video streaming scheme for LTE devices,” *IEEE Commun. Lett.*, vol. 19, no. 6, pp. 965–968, Jun. 2015, doi: 10.1109/LCOMM.2015.2420626.

[23] M. A. Hoque, M. Siekkinen, and J. K. Nurminen, “On the energy efficiency of proxy-based traffic shaping for mobile audio streaming,” *2011 IEEE Consum. Commun. Netw. Conf. CCNC’2011*, pp. 891–895, 2011, doi: 10.1109/CCNC.2011.5766635.

[24] G. B. Creus and M. Kuulusa, “Optimizing mobile software with built-in power profiling,” *Mob. Phone Program. its Appl. to Wirel. Netw.*, pp. 449–462, 2007, doi: 10.1007/978-1-4020-5969-8_25.

[25] S. R. Yang, S. Y. Yan, and H. N. Hung, “Modeling UMTS power saving with bursty packet data traffic,” *IEEE Trans. Mob. Comput.*, vol. 6, no. 12, pp. 1398–1408, Dec. 2007, doi: 10.1109/TMC.2007.1072.

[26] K. Lavangnananda, C. Angsuchotmetee, and P. Bouvry, “Effect of packetization interval on number of connections in AAC audio streaming over WLAN 802.11g,” *17th Asia-Pacific Conf. Commun. APCC 2011*, pp. 530–535, 2011, doi: 10.1109/APCC.2011.6152866.

ภาคผนวก ก

Application TcpBombApp Sourcecode

TcpBombApp.h

```
#ifndef INET_APPLICATIONS_TCPAPP_TCPBOMBAPP_H_
#define INET_APPLICATIONS_TCPAPP_TCPBOMBAPP_H_

#include "inet/common/INETDefs.h"
#include "inet/applications/tcpapp/TcpServerHostApp.h"
#include "inet/common/INETMath.h"
#include "inet/transportlayer/contract/tcp/TcpSocket.h"

#include <vector>
#include "inet/common/INETDefs.h"
#include "inet/applications/tcpapp/TcpAppBase.h"
#include "inet/common/lifecycle/LifecycleOperation.h"
#include "inet/common/lifecycle/NodeStatus.h"

namespace inet {

class INET_API TcpBombApp : public TcpServerHostApp
{
  protected:
    simtime_t delay;
    double echoFactor = NaN;

    long bytesRcvd = 0;
    long bytesSent = 0;

  protected:
    virtual void sendDown(Packet *packet);

    virtual void initialize(int stage) override;
    virtual int numInitStages() const override { return NUM_INIT_STAGES; }
    virtual void finish() override;
    virtual void refreshDisplay() const override;

  public:
    TcpBombApp();
    ~TcpBombApp();

    friend class TcpBombAppThread;
};
class INET_API TcpBombAppThread : public TcpServerThreadBase
{
  protected:
    TcpBombApp *echoAppModule = nullptr;

  public:
    int i;

    virtual void established() override;

    virtual void dataArrived(Packet *msg, bool urgent) override;
```

```
    virtual void timerExpired(cMessage *timer) override;

    virtual void init(TcpServerHostApp *hostmodule, TcpSocket *socket)
override { TcpServerThreadBase::init(hostmodule, socket); echoAppModule =
check_and_cast<TcpBombApp *>(hostmod); }
};

}

#endif
```

TcpBombApp.cc

```
#include "TcpBombApp.h"
#include "inet/applications/common/SocketTag_m.h"
#include "inet/common/ModuleAccess.h"
#include "inet/common/ProtocolTag_m.h"
#include "inet/common/lifecycle/ModuleOperations.h"
#include "inet/common/packet/Packet_m.h"
#include "inet/transportlayer/contract/tcp/TcpCommand_m.h"
#include <iostream>

#include "inet/applications/base/ApplicationPacket_m.h"
#include "inet/common/lifecycle/ModuleOperations.h"
#include "inet/common/TagBase_m.h"
#include "inet/common/TimeTag_m.h"
#include "inet/common/packet/Packet.h"
#include "inet/common/packet/chunk/ByteCountChunk.h"
#include "inet/common/packet/chunk/BytesChunk.h"
#include "inet/networklayer/common/L3AddressResolver.h"
#include <iostream>
#include "inet/applications/tcpapp/QosParam.h"


using namespace std;
namespace inet {

Define_Module(TcpBombApp);

Define_Module(TcpBombAppThread);

#define MSGKIND_CONNECT    1
#define MSGKIND_SEND       2
#define MSGKIND_CLOSE      3


TcpBombApp::TcpBombApp() {
    // TODO Auto-generated constructor stub

}

TcpBombApp::~TcpBombApp() {
```

```
    // TODO Auto-generated destructor stub
}

void TcpBombApp::initialize(int stage)
{


    TcpServerHostApp::initialize(stage);
    if (stage == INITSTAGE_LOCAL) {
        delay = par("echoDelay");
        echoFactor = par("echoFactor");

        bytesRcvd = bytesSent = 0;
        WATCH(bytesRcvd);
        WATCH(bytesSent);

    }

}
//change
void TcpBombApp::sendDown(Packet *msg)
{
    if (msg->isPacket()) {
        Packet *pk = static_cast<Packet *>(msg);
        bytesSent += pk->getByteLength();
        emit(packetSentSignal, pk);
    }
    msg->addTagIfAbsent<DispatchProtocolReq>()-
>setProtocol(&Protocol::tcp);
    msg->getTag<SocketReq>();

    send(msg, "socketOut");
    cout << "==================================" << endl;

    for (auto socket : socketMap.getMap()) {
    }
}
void TcpBombApp::refreshDisplay() const
{
    ApplicationBase::refreshDisplay();

    char buf[160];
    sprintf(buf, "threads: %d\nrcvd: %ld bytes\nsent: %ld bytes",
socketMap.size(), bytesRcvd, bytesSent);

    getDisplayString().setTagArg("t", 0, buf);

}


void TcpBombApp::finish()
{
    TcpServerHostApp::finish();

    recordScalar("bytesRcvd", bytesRcvd);
```

```
    recordScalar("bytesSent", bytesSent);
}

void TcpBombAppThread::established()
{
}

void TcpBombAppThread::dataArrived(Packet *rcvdPkt, bool urgent)
{
    echoAppModule->emit(packetReceivedSignal, rcvdPkt);
    int64_t rcvdBytes = rcvdPkt->getByteLength();
    echoAppModule->bytesRcvd += rcvdBytes;
    auto data = rcvdPkt->peekData(); // get all data from the packet
      auto regions = data->getAllTags<CreationTimeTag>(); // get all tag
regions
      for (auto& region : regions) { // for each region do
        auto creationTime = region.getTag()->getCreationTime(); // original
time
        auto delay = simTime() - creationTime; // compute delay
        cout << region.getOffset() << region.getLength() << delay; // use
data
      }
    if (echoAppModule->echoFactor > 0 && sock->getState() ==
TcpSocket::CONNECTED ) {

                Packet *outPkt = new Packet(rcvdPkt->getName(),
TCP_C_SEND);
                Packet *outPkt1 = new Packet(rcvdPkt->getName(),
TCP_C_SEND);
                Packet *outPkt2 = new Packet(rcvdPkt->getName(),
TCP_C_SEND);
                Packet *outPkt3 = new Packet(rcvdPkt->getName(),
TCP_C_SEND);
                Packet *outPkt4 = new Packet(rcvdPkt->getName(),
TCP_C_SEND);

                outPkt->addTag<SocketReq>()->setSocketId(14);
                outPkt1->addTag<SocketReq>()->setSocketId(15);
                outPkt2->addTag<SocketReq>()->setSocketId(16);
                outPkt3->addTag<SocketReq>()->setSocketId(17);
                outPkt4->addTag<SocketReq>()->setSocketId(18);

                long outByteLen = rcvdBytes * echoAppModule->echoFactor;

                if (outByteLen < 1)
                    outByteLen = 1;

                int64_t len = 0;
                for ( ; len + rcvdBytes <= outByteLen; len += rcvdBytes) {
                    outPkt->insertAtBack(rcvdPkt->peekDataAt(B(0),
B(rcvdBytes)));
                    outPkt1->insertAtBack(rcvdPkt->peekDataAt(B(0),
B(rcvdBytes)));
                    outPkt2->insertAtBack(rcvdPkt->peekDataAt(B(0),
B(rcvdBytes)));
```

```
                    outPkt3->insertAtBack(rcvdPkt->peekDataAt(B(0),
B(rcvdBytes)));
                    outPkt4->insertAtBack(rcvdPkt->peekDataAt(B(0),
B(rcvdBytes)));
                }
                if (len < outByteLen){
                    outPkt->insertAtBack(rcvdPkt->peekDataAt(B(0),
B(outByteLen - len)));
                    outPkt1->insertAtBack(rcvdPkt->peekDataAt(B(0),
B(outByteLen - len)));
                    outPkt2->insertAtBack(rcvdPkt->peekDataAt(B(0),
B(outByteLen - len)));
                    outPkt3->insertAtBack(rcvdPkt->peekDataAt(B(0),
B(outByteLen - len)));
                    outPkt4->insertAtBack(rcvdPkt->peekDataAt(B(0),
B(outByteLen - len)));
                }

                ASSERT(outPkt->getByteLength() == outByteLen);
                ASSERT(outPkt1->getByteLength() == outByteLen);
                ASSERT(outPkt2->getByteLength() == outByteLen);
                ASSERT(outPkt3->getByteLength() == outByteLen);
                ASSERT(outPkt4->getByteLength() == outByteLen);

                if (echoAppModule->delay == 0){
                    echoAppModule->sendDown(outPkt);
                    echoAppModule->sendDown(outPkt1);
                    echoAppModule->sendDown(outPkt2);
                    echoAppModule->sendDown(outPkt3);
                    echoAppModule->sendDown(outPkt4);
                }else{
                    scheduleAt(simTime() + echoAppModule->delay, outPkt);
                }
    }
    delete rcvdPkt;
}

void TcpBombAppThread::timerExpired(cMessage *timer)
{
    Packet *pkt = check_and_cast<Packet *>(timer);
    pkt->setContextPointer(nullptr);
    echoAppModule->sendDown(pkt);
}

}
```

ภาคผนวก ข

Init Sourcecode

แบบ non-sfu

```
[General]
network = Node

num-rngs=1

image-path=../../images
output-scalar-file-append = false
sim-time-limit=30s

################ Statistics ##################
output-scalar-file = ${resultdir}/${configname}/${repetition}.sca
output-vector-file = ${resultdir}/${configname}/${repetition}.vec
seed-set = ${repetition}
**.vector-recording = false

################# Mobility parameters #####################
# *
**.mobility.constraintAreaMinZ = 0m
**.mobility.constraintAreaMaxZ = 0m
**.mobility.initFromDisplayString = true

################ Number of Resource Blocks ################
**.numRbDl = 6
**.numRbUl = 6
**.binder.numBands = 6 # this value should be kept equal to the number of
RBs

################ Transmission Power ##################
**.ueTxPower = 26
**.eNodeBTxPower = 40

**.ue[*].numApps = 5

**.mobility.constraintAreaMinX = 0m
**.mobility.constraintAreaMinY = 0m
**.mobility.constraintAreaMaxX = 800m
**.mobility.constraintAreaMaxY = 400m

**.ue[*].mobility.initialX = uniform(300m,330m)
**.ue[*].mobility.initialY = uniform(250m,260m)
**.ue[*].app[*].typename = "TcpSessionNew"
**.ue[*].app[*].active = true
**.ue[*].app[*].localPort = -1
**.ue[*].app[0].connectAddress = "service1"
**.ue[*].app[0].connectPort = 3099
**.ue[*].app[0].tOpen = 0.2s
**.ue[*].app[1].tOpen = 0.21s
**.ue[*].app[2].tOpen = 0.22s
**.ue[*].app[3].tOpen = 0.23s
**.ue[*].app[4].tOpen = 0.24s
**.ue[*].app[*].tSend = 0.4s
**.ue[*].app[*].sendBytes = 819B
```

```
**.ue[*].app[*].sendScript = ""
**.ue[*].app[*].tClose = 60s
**.ue[*].app[*].dataTransferMode = "bytestream"
**.ue[*].app[1].connectAddress = "service2"
**.ue[*].app[1].connectPort = 3088
**.ue[*].app[2].connectAddress = "service3"
**.ue[*].app[2].connectPort = 3077
**.ue[*].app[3].connectAddress = "service4"
**.ue[*].app[3].connectPort = 3066
**.ue[*].app[4].connectAddress = "service5"
**.ue[*].app[4].connectPort = 3055

**.service*.numApps = 1
**.service*.app[*].typename = "TcpSinkNew"
**.service1.app[0].localPort = 3099
**.service2.app[0].localPort = 3088
**.service3.app[0].localPort = 3077
**.service4.app[0].localPort = 3066
**.service5.app[0].localPort = 3055

**.ue[*].masterId = 1
**.ue[*].macCellId = 1
**.ue[*].mobility.initFromDisplayString = false
**.ue[*].mobilityType = "StationaryMobility"
```

แบบ SFU

```
[General]
network = TcpLastTest
#*.numTarget = 5
total-stack = 70000MiB

*.visualizer.*.interfaceTableVisualizer.displayInterfaceTables = true

**.server.numPcapRecorders = 0
**.server.pcapRecorder[0].pcapFile = "results/${configname}-${runnumber}-
server.pcap"

##
*.client2.numApps = 1
**.client2.app[*].typename = "TcpSessionApp"
**.client2.app[0].active = true
**.client2.app[0].localPort = -1
**.client2.app[0].connectAddress = "server"
**.client2.app[0].connectPort = 1000
**.client2.app[0].tOpen = 0.21s
**.client2.app[0].tClose = 30s

*.client3.numApps = 1
**.client3.app[*].typename = "TcpSessionApp"
**.client3.app[0].active = true
**.client3.app[0].localPort = -1
```

```
**.client3.app[0].connectAddress = "server"
**.client3.app[0].connectPort = 1000
**.client3.app[0].tOpen = 0.21s
**.client3.app[0].tClose = 30s

*.client4.numApps = 1
**.client4.app[*].typename = "TcpSessionApp"
**.client4.app[0].active = true
**.client4.app[0].localPort = -1
**.client4.app[0].connectAddress = "server"
**.client4.app[0].connectPort = 1000
**.client4.app[0].tOpen = 0.21s
**.client4.app[0].tClose = 30s

*.client5.numApps = 1
**.client5.app[*].typename = "TcpSessionApp"
**.client5.app[0].active = true
**.client5.app[0].localPort = -1
**.client5.app[0].connectAddress = "server"
**.client5.app[0].connectPort = 1000
**.client5.app[0].tOpen = 0.21s
**.client5.app[0].tClose = 30s

*.client6.numApps = 1
**.client6.app[*].typename = "TcpSessionApp"
**.client6.app[0].active = true
**.client6.app[0].localPort = -1
**.client6.app[0].connectAddress = "server"
**.client6.app[0].connectPort = 1000
**.client6.app[0].tOpen = 0.21s
**.client6.app[0].tClose = 30s

##
**.server*.numApps = 1
**.server*.app[*].typename = "TcpBombApp"
**.server*.app[0].localPort = 1000
**.server*.app[0].echoFactor = 2.0
**.server*.app[0].echoDelay = 0s

##LTE
image-path=../../images
output-scalar-file-append = false
sim-time-limit=30s

################ Statistics ##################
output-scalar-file = ${resultdir}/${configname}/${repetition}.sca
output-vector-file = ${resultdir}/${configname}/${repetition}.vec
seed-set = ${repetition}
**.vector-recording = false

################ Mobility parameters #####################
# *
**.mobility.constraintAreaMinZ = 0m
**.mobility.constraintAreaMaxZ = 0m
**.mobility.initFromDisplayString = true
```

```
################ Number of Resource Blocks ################
**.numRbDl = 6
**.numRbUl = 6
**.binder.numBands = 6 # this value should be kept equal to the number of
RBs

################ Transmission Power ##################
**.ueTxPower = 26
**.eNodeBTxPower = 40

##1-5
*.n = 5
description = direct stream.

*.configuratorLTE.config = xmldoc("demo.xml")



**.numUe = 1 # ${numUes=1,2,5,10,20,50,100}

**.mobility.constraintAreaMinX = 300m
**.mobility.constraintAreaMinY = 200m
**.mobility.constraintAreaMaxX = 800m
**.mobility.constraintAreaMaxY = 400m


**.ue[*].masterId = 1
**.ue[*].macCellId = 1
**.ue[*].mobility.initFromDisplayString = false
**.ue[*].mobilityType = "StationaryMobility"

*.ue[*].numApps = 1
**.ue[*].app[*].typename = "TcpSessionApp"
**.ue[*].app[0].active = true
**.ue[*].app[0].localPort = -1
**.ue[*].app[0].connectAddress = "server"
**.ue[*].app[0].connectPort = 1000
**.ue[*].app[0].tOpen = 0.2s
**.ue[*].app[0].tSend = 0.4s
**.ue[*].app[0].sendBytes = 819B
**.ue[*].app[0].sendScript = ""
**.ue[*].app[0].tClose = 30s
**.ue[*].app[0].dataTransferMode = "bytestream"
```

ภาคผนวก ค

Ned Sourcecode

แบบไม่ใช้ SFU

```
package one_to_five;

import inet.networklayer.ipv4.RoutingTableRecorder;
import inet.node.ethernet.Eth10G;
import inet.node.ethernet.Eth10M;
import inet.node.ethernet.EtherSwitch;
import inet.node.inet.Router;
import inet.node.inet.StandardHost;
import lte.corenetwork.binder.LteBinder;
import lte.corenetwork.nodes.Ue;
import lte.corenetwork.nodes.eNodeB;
import lte.world.radio.LteChannelControl;
import lte.epc.PgwStandardSimplified;
import lte.common.LteNetworkConfigurator;

network Node
{
    parameters:
        int numUe = 30;
        //@display("i=block/network2;bgb=991,558;bgi=background/budapest");
    submodules:
        channelControl: LteChannelControl {
            @display("p=50,25;is=s");
        }
        routingRecorder: RoutingTableRecorder {
            @display("p=50,75;is=s");
        }
        configurator: LteNetworkConfigurator {
            @display("p=50,125");
        }
        binder: LteBinder {
            @display("p=50,175;is=s");
        }
        router: Router {
            @display("p=804,274;i=device/smallrouter");
        }
        pgw: PgwStandardSimplified {
            nodeType = "PGW";
            @display("p=629,275;is=l");
        }
        eNB: eNodeB {
            @display("p=472,275;is=vl");
        }
        ue[numUe]: Ue {
            @display("p=260,305");
        }
        Switch: EtherSwitch {
            @display("p=928,275");
        }
        service1: StandardHost {
            @display("p=1031,126");
        }
```

```
        service2: StandardHost {
            @display("p=1031,201");
        }
        service3: StandardHost {
            @display("p=1031,276");
        }
        service4: StandardHost {
            @display("p=1031,351");
        }
        service5: StandardHost {
            @display("p=1031,424");
        }
    connections:
        service1.ethg++ <--> Eth10G <--> Switch.ethg++;
        service2.ethg++ <--> Eth10G <--> Switch.ethg++;
        service3.ethg++ <--> Eth10G <--> Switch.ethg++;
        service4.ethg++ <--> Eth10G <--> Switch.ethg++;
        service5.ethg++ <--> Eth10G <--> Switch.ethg++;

        Switch.ethg++ <--> Eth10G <--> router.ethg++;
        pgw.pppg++ <--> Eth10G <--> eNB.ppp;
        router.pppg++ <--> Eth10G <--> pgw.filterGate;
}
```

แบบ SFU

```
package onetoonetofive;


import inet.node.ethernet.Eth10G;
import inet.common.misc.NetAnimTrace;
import inet.common.scenario.ScenarioManager;
import inet.networklayer.configurator.ipv4.Ipv4NetworkConfigurator;
import inet.node.inet.StandardHost;
import inet.visualizer.integrated.IntegratedVisualizer;
import ned.DatarateChannel;

import inet.networklayer.ipv4.RoutingTableRecorder;
import inet.node.ethernet.EtherSwitch;
import inet.node.inet.Router;
import inet.node.inet.StandardHost;
import lte.corenetwork.binder.LteBinder;
import lte.corenetwork.nodes.Ue;
import lte.corenetwork.nodes.eNodeB;
import lte.world.radio.LteChannelControl;
import lte.epc.PgwStandardSimplified;
import lte.common.LteNetworkConfigurator;


//
// TODO auto-generated type
//
network TcpLastTest
```

```
{
    parameters:
        int n;
        int numUe = default(1);
        //int numTarget;
    types:
        channel C extends DatarateChannel
        {
            datarate = 10Mbps;
            delay = 0.1us;
        }
    submodules:
        visualizer: IntegratedVisualizer {
            @display("p=50,50");
        }
        server: StandardHost {
            parameters:
                @display("p=475,200;i=device/pc2");
        }
        configurator: Ipv4NetworkConfigurator {
            parameters:
                @display("p=100,110;is=s");
        }
        netAnimTrace: NetAnimTrace {
            @display("p=100,208;is=s");
        }
        scenarioManager: ScenarioManager {
            @display("p=100,256;is=s");
        }
        client2: StandardHost {
            @display("p=557,50");
        }
        client3: StandardHost {
            @display("p=557,125");
        }
        client4: StandardHost {
            @display("p=557,200");
        }
        client5: StandardHost {
            @display("p=557,275");
        }
        client6: StandardHost {
            @display("p=557,348");
        }
        channelControl: LteChannelControl {
            @display("p=147,33;is=s");
        }
        routingRecorder: RoutingTableRecorder {
            @display("p=277,31;is=s");
        }
        configuratorLTE: LteNetworkConfigurator {
            @display("p=410,33");
        }
        binder: LteBinder {
            @display("p=100,156;is=s");
```

```
        }
        router: Router {
            @display("p=400,199;i=device/smallrouter");
        }
        pgw: PgwStandardSimplified {
            nodeType = "PGW";
            @display("p=331,200;is=l");
        }
        eNB: eNodeB {
            @display("p=500,300;is=vl");
        }
        ue[numUe]: Ue {
            @display("p=354,339");
        }


    connections:

        //client1.ethg++ <--> C <--> server.ethg++;
        client2.ethg++ <--> C <--> server.ethg++;
        client3.ethg++ <--> C <--> server.ethg++;
        client4.ethg++ <--> C <--> server.ethg++;
        client5.ethg++ <--> C <--> server.ethg++;
        client6.ethg++ <--> C <--> server.ethg++;

        pgw.pppg++ <--> Eth10G <--> eNB.ppp;
        router.pppg++ <--> Eth10G <--> pgw.filterGate;
        server.ethg++ <--> Eth10G <--> router.pppg++;
}
```

ภาคผนวก ง.

ผลงานตีพิมพ์

ผลงานตีพิมพ์

เรื่อง	A Study of Effect of Architectural Design on Quality of Service of a Live Streaming Application with Multiple Endpoints over LTE Network
งานประชุมวิชาการ	The 12th International Conference on Advances in Information Technology
สถานที่	**มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี**
วันที่	**30 มิถุนายน พุทธศักราช 2564**

A Study of Effect of Architectural Design on Quality of Service of a Live Streaming Application with Multiple Endpoints over LTE Network

Charlif Prapawit
6210220095@email.psu.ac.th
Faculty of Science, Prince of Songkla University
Songkhla, Thailand

Chinnapong Angsuchotmetee*
chinnapong.a@psu.ac.th
Faculty of Science, Prince of Songkla University
Songkhla, Thailand

ABSTRACT

The number of streaming service providers has been increasing dramatically every year. Hence, users may prefer to publish their stream to multiple service endpoints simultaneously to increase visibility. However, most service providers prefer to monopolize their services. Hence, a study of a suitable architectural design of a streaming service that supports multiple streaming endpoints has not gained lots of attention. In this study, the effect of adopting different architectural design on developing a live streaming service over LTE network which can supports multiple streaming endpoints are investigated. Two major designs are selected which are a selective forwarding unit based architecture, and a non-selective forwarding unit based architecture. The results suggest that a selective forwarding unit architecture has an advantage over a non-selective forwarding unit based architecture on keeping overall average streaming end-to-end delay to be minimum., while a fluctuation in an end-to-end delay occurs in a non-selective forwarding unit based architecture in our experiment testbed. The results, discussions, and suggestions on future studies are given at the end of this study.

KEYWORDS

Streaming Architecture, Multiple Endpoints Live Streaming, Selective Forwarding Unit, LTE Network

ACM Reference Format:
Charlif Prapawit and Chinnapong Angsuchotmetee. 2021. A Study of Effect of Architectural Design on Quality of Service of a Live Streaming Application with Multiple Endpoints over LTE Network. In *The 12th International Conference on Advances in Information Technology (IAIT2021), June 29-July 1, 2021, Bangkok, Thailand.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3468784.3469855

1 INTRODUCTION

A live streaming service is one of the most popular services on the Internet nowadays. A live streaming service allows users to publish their live media feed to their desired audiences directly from their computer or mobile device. Significant streaming service providers nowadays include Facebook Live, Youtube, Dailymotion, Vimeo, and Twitch. The significant advantages of adopting live streaming services do not only allow users to interact with their own designated audiences, but it also significantly boost their businesses invisibility which leads to more customers to their businesses [4].

One major limitation of most existing streaming applications or streaming services is that a streaming client of each service can publish a live stream to only one service or now more than two simultaneous services. In order to allow users (i.e., streaming publishers) to reach their audiences on a broader scale, a streaming service that allows users to multiple streaming service end-point simultaneously is required. On top of that, since most of the devices that users are used for connecting to the Internet nowadays are mostly mobile devices[9] (e.g., smartphone), it is preferable that such a streaming service must support a case where users publish their stream from their mobile devices over LTE cellular network.

Since most streaming service providers prefer to allow users to use a single service exclusively, such that the provider can monopolize the service, only a few studies focus on the architecture design of a live streaming service that allows users to publish their streams to multiple end-points. The architecture design of such a service is briefly discussed in [8] [6]. Lots of existing streaming services rely on the usage of a *Selective Forwarding Unit* (SFU) for handling streaming data from users, processing, and routing them to their appropriate destinations [12]. On the other hand, some streaming service prefers to keep their architecture simple through the usage of peer-to-peer video streaming [3]. Hence, in general, there are two main categories of an architectural design of a streaming service nowadays which are (i) SFU-based architecture and (ii) non-SFU-based architecture. Both architectures have their advantages and disadvantages. However, to the best of our knowledge so far, none of the existing studies focus on comparing both architectures in an aspect of developing a live streaming service capable of supporting multiple simultaneous endpoints.

In this study, we investigate the difference between adopting an SFU-based architecture and a non-SFU-based architecture in a live streaming service capable of supporting multiple simultaneous endpoints live streaming. Because, to date, most users prefer using mobile phones over PC hence, this study focuses mainly on a case in which clients use their mobile phones for connecting the Internet of an LTE network. The study is conducted using experimentation using a software network simulator. The results and discussions on whether SFU-based or non-SFU-based are more preferable are given at the end of the study.

*Division of Computational Science

IAIT2021, June 29-July 1, 2021, Bangkok, Thailand

ACM ISBN 978-1-4503-9012-5/21/06…$15.00
https://doi.org/10.1145/3468784.3469855

The outline of this paper is as follows. Section 2 describes in detail the SFU-based and non-SFU-based streaming architecture, their major differences, advantages, disadvantages, and what is needed to be investigated further. This section follows Section 3 which describes existing studies related to our work. The experiment setup and experiment methodology are given in Section 4. The results and discussions are given in Section 5 and Section 6. Section 7 concludes this study and describes our future work direction.

2 STREAMING SERVICE ARCHITECTURE OVERVIEW: SFU BASED VS NON-SFU BASED

Most existing streaming services nowadays rely primarily on a simple TCP-based[11] socket connection for delivering streaming content. However, due to the fact that there are lots of streaming services nowadays, users may want to publish their live content to more than one service simultaneously (e.g., broadcasting content to Facebook Live and Youtube at the same time from one machine). To do so, the architecture of a streaming service and related applications have to be well designed. In general, existing streaming service architecture, which allows users to publish their contents to multiple services at once, can be categorized into two categories which are (i) SFU-Based architecture and (ii) non-SFU-based architecture. Their details aere as follows.

2.1 SFU-based Streaming Architecture

A selective forwarding unit (SFU) is a unit that manages the transport of multimedia data. A streaming service that adopts an SFU uses the media relaying capability of an SFU to relay content from users to a designated destination. In publishing live streaming content to multiple services simultaneously, an SFU is responsible for receiving packets from a user (i.e., content publisher), copying packets, and relaying them to designated services. This functionality of an SFU is depicted in Figure 1.

The architecture, as depicted in Figure 1, is also described in [12]. Such a design is proposed mainly because it cannot encode multimedia contents and send them to multiple streaming service end-points simultaneously without relying on a high-performance computer. Such a challenge makes an SFU-based architecture preferable when we deem that the client's computer performance is not enough. Thus, the usage of SFU is necessary. However, to the best of our knowledge so far, it is still arguable to date that if the client's computer performance were improved dramatically over time, the need to use an SFU still be necessary.

2.2 Non-SFU based Streaming Architecture

A non-SFU architecture, in contrast to an SFU-based architecture, refers to an architecture that assumes that a client's computer has enough performance to support multiple endpoints live streaming. Such an architecture is depicted in Figure 2.

It is used with a single device to handle all data handling and output. It is the most commonly used streaming architecture, with it being straightforward that one streaming on the present side will have to pre-process, encode, transmission for one service. We can repeat for send to multiple services.

As depicted in Figure 2, this streaming architecture assumes that the streaming application on the client's machine can support live streaming to multiple end-points. The advantage of this design is the cost-saving feature because the usage of a separate SFU server is no longer necessary. However, to date, most devices that can connect to the Internet are primarily mobile devices. Hence, it is still needed to verify whether mobile devices can support live streaming to multiple end-points. On top of that, most of the devices are mostly connected to the Internet through LTE networks[1]. Hence, it is still needed to be verified whether multiple end-points of live streaming consume too much bandwidth or causing too much contention of the LTE network to or not.

3 RELATED STUDIES

Several studies are related to streaming performance measurement. Previous studies which are related to our research can be categorized into two groups which are (i) studies of streaming performance in a simulated environment and (ii) studies of streaming performance in a physical test-bed. Studies discussed in this section are limited to only studies that conduct experiments using the LTE network only. The details are as follows.

3.1 Studies of streaming performance in a simulated LTE environment

This group of studies chooses to conduct a streaming performance measurement experiment over an LTE network in a simulated environment using a network simulator software suite. One of the significant studies in [2]. This study conducts experiments using NS-3 network simulator[1]. This study focuses on testing the difference between streaming 720p and 1080p video over an LTE network. Another critical study is in [5]. This study focuses on a Voice over IP (VoIP) application. This study focuses on determining whether the usage of a mobile *voice packet relay* mechanism [2] affects the quality of service or not. The result reveals that adopting a mobile voice packet relaying mechanism leads to a better quality of services despite the additional 40 bytes packet overhead used during the relaying process.

3.2 Studies of streaming performance in a physical LTE test-bed.

This group of studies chooses to conduct an actual physical streaming performance measurement experiment over an LTE network using a network from an existing LTE network provider. A significant study is in [7]. This study measures the performance of the LTE network of Telecom of Kosovo (TK) whether it has complied with the 3GPP standard or not. The result shows that the network performance can support any applications or services, including a streaming service. However, package drop and contention at a cell site is still a significant concern that has to be careful on delivering service with an acceptable quality of service. To overcome the quality of service concern, some study suggests that quality of service provisioning method has to be used. One of the significant studies is in [13] This study focuses on the quality of service provisioning

[1] https://www.nsnam.org/
[2] an SFU-like mechanism designated for routing audio packets

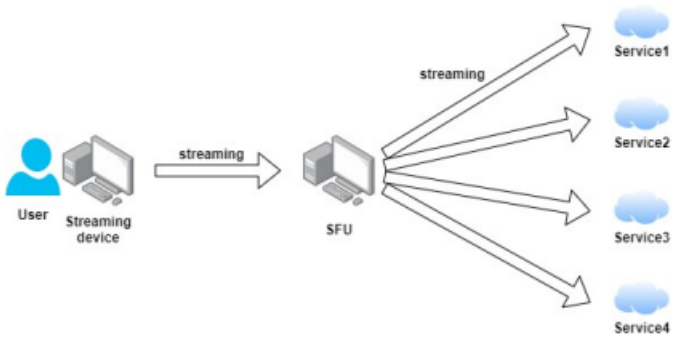


Figure 1: An SFU-based Streaming Architecture

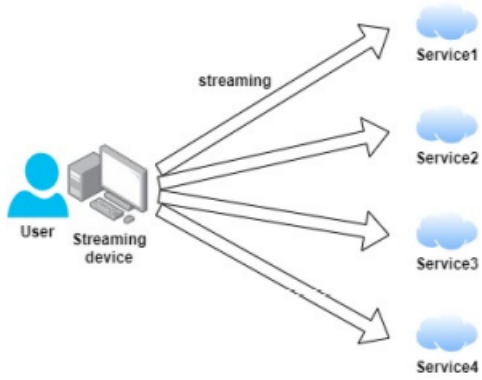


Figure 2: A non-SFU based Streaming Architecture

methodology for a device-to-device content delivery service over an LTE network. The result suggests that good quality of service provisioning and content caching mechanism is mandatory to provide a service with an acceptable quality of service (i.e., low latency and low packet drop) in an actual physical LTE network.

4 EXPERIMENTS : SFU-BASED VS NON-SFU BASED IN A LIVE STREAMING APPLICATION WITH MULTIPLE ENDPOINTS

As stated earlier, this study focuses on studying an architectural difference in delivering a live streaming service to multiple endpoints simultaneously. Selected architecture for studying is (i) SFU-based architecture and (ii) non-SFU-based architecture. In order to minimize the effect of physical interference, we conduct experiments solely in a simulated environment. The software used for conducting experiments in this study is OMNeT++ v5.6.2 [3]. Network equipment and LTE networks are simulated using plugins for OMNeT++ named INET Framework[4], and SimuLTE[10]. This study conducts two sets of experiments which are (i) non-SFU-based architecture experiment and (ii) SFU-based architecture experiment. The details are as follows.

4.1 Non-SFU based Streaming Architecture Experiment

In this set of experiments, a non-SFU-based live streaming architecture as depicted in Figure 2 are modeled within OMNeT++. The simulated environment used is given in Figure 3.

[3]https://omnetpp.org/
[4]https://inet.omnetpp.org/

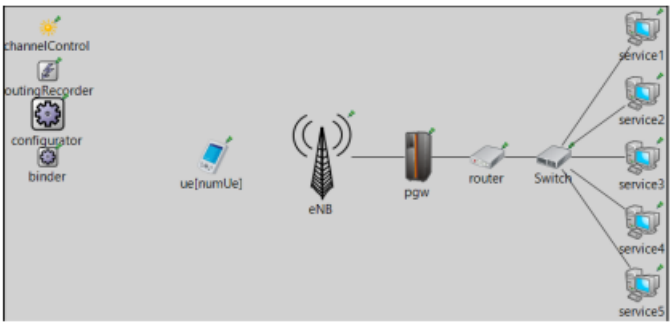


Figure 3: Non-SFU based live streaming architecture: OMNeT++ experiment testbed

In Figure 3, a simple LTE network configuration is modeled within OMNeT++. One LTE cell site is deployed and connected to a simple LTE Packet Gateway (PGW) service. The PGW is connected with a simple router. A router is connected with a simple switch with five service terminals attached to work as endpoints for live streaming sessions. The *ue[numUe]* within Figure 3 refers to mobile devices of users. These devices are used as origins for initiating streaming sessions. All device and transmission medium parameters are kept in an ideal condition such that signal deterioration from other interference factors is kept minimal.

The simulation of this set of experiments begins by initializing a simulation testbed with specific numbers of mobile devices. Five sets of experiments are conducts using 10, 20, 30, 40, and 50 devices consecutively. Each device is implemented with a simulated live streaming application that used TCP to initiate a streaming session. The streaming application in each device is configured for sending 1500 bytes of a package for every 20ms. This reflects the actual usage of a streaming application with the largest possible multimedia size and 20ms of packetization interval duration. For every packet, each device must copy and send each packet to all five service endpoints simultaneously. An end-to-end delay is used as a measurement parameter. The end-to-end delay is calculated as a time difference between when a packet is transmitted from a mobile device and when it has arrived at its corresponding service endpoint.

4.2 SFU-based Streaming Architecture Experiment

This set of experiments follows an architecture as described in Figure 1. The simulated testbed for this set of experiments is given in Figure 4.

The only significant difference of this experiment, as depicted in Figure 4, in comparison with Figure 2, is the SFU server placed between the router and the switch within the simulated environment. This server is configured with an application that has two primary responsibilities, which are (i) accepting incoming streaming packets from users' mobile devices, and (ii) copying packets and relaying them to all five streaming services endpoints. The number of mobile devices and measurement parameters are kept the same as the non-SFU-based experiment such that the result can be compared between each set of experiments.

5 EXPERIMENT RESULT

The experiments as described in Section 4 were conducted in this study. The experiments were conducted five times for each set of experiments. The end-to-end delay of each round of experiments is averaged and recorded. The non-SFU-based streaming experiment results are given in Table 1 as follows.

Table 1 shows average end-to-end delay at each streaming endpoint. It can be seen in the Table that the delay between each endpoint in each set of experiments has heavily fluctuated. The result of the SFU-based streaming experiment is given in Table 2.

Table 2 shows the average end-to-end delay at each streaming endpoint in the SFU-based streaming architecture experiment. In contrast with the non-SFU-based architecture experiment, the SFU-based experiment reveals that the average delay at each endpoint tends to be similar. It is to be noted that the average value at each endpoint within Table 2 are mostly similar. The results reveal a minor difference in average delay at 4-5 floating point digits precision. We deemed that such a difference is not considered significant. Thus, we round the result two up to 2 floating point digits precision to keep the result simple for readers.

To compare the pattern of end-to-end delay results between non-SFU-based and SFU-based experiments, we conduct an additional analysis by calculating the average end-to-end delay at each endpoint for each experiment and depicted it using a line chart. The result is depicted in Figure 5. The result shows that the end-to-end delay for SFU-based architecture is considerably lower in all cases.

6 DISCUSSION

The result in a non-SFU-based architecture case reveals the fluctuation in average end-to-end delay in all experiment cases. This means that despite the fact a non-SFU-based architecture may have an

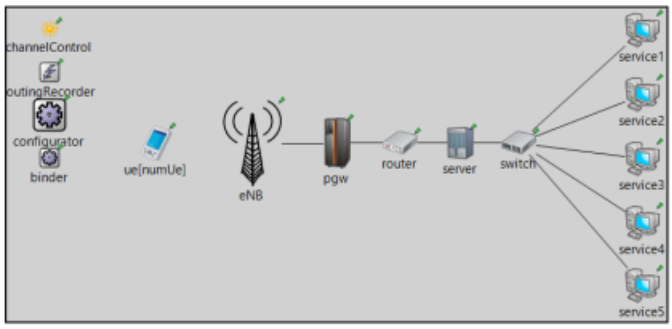


Figure 4: SFU based live streaming architecture: OMNeT++ experiment testbed

Table 1: Experiment Result: non-SFU-based Architecture

Number of Mobile Devices	Average Delay at End Points(ms)				
	End Point 1	End Point 2	End Point 3	End Point 4	End Point 5
10	5,513.81	114.77	98.72	112.01	124.97
20	584.56	163.93	2,875.11	18,590.94	731.14
30	581.77	7,304.57	182.12	3,982.69	66.59
40	5,161.31	4,356.85	4,724.22	3,553.40	6,257.58
50	23,781.00	1,456.37	50.97	65.76	80.66

Table 2: Experiment Result: SFU-based Architecture

Number of Mobile Devices	Average Delay at End Points(ms)				
	End Point 1	End Point 2	End Point 3	End Point 4	End Point 5
10	117.43	117.44	117.44	117.44	117.44
20	217.15	217.15	217.15	217.15	217.15
30	1,391.32	1,391.32	1,391.32	1,391.32	1,391.32
40	1,052.24	1,052.24	1,052.24	1,052.24	1,052.24
50	1,733.24	1,733.10	1,733.11	1,733.11	1,733.11

advantage over an SFU-based architecture in terms of a cost-saving aspect (i.e., a separate SFU server is not required), the quality of service in a non-SFU based architecture may be a major disadvantage. This comes from the fact that, in a non-SFU-based case, the number of packets sent to an LTE cell site is considerably higher than in a non-SFU cased. Thus, the average end-to-end delay fluctuates because each mobile device may need to wait longer before the radio medium can be ready to be used for transmitting packets.

On the other hand, the SFU-based architecture experiment reveals less fluctuation in an average end-to-end delay. This comes from the fact that the network contention at the LTE cell site in the simulated experiment is five times lower than a non-SFU-based case. This means that controlling the overall quality of service for a streaming service can be easier if a service chooses to adopt an SFU-based architecture over a non-SFU-based one.

This study assumes that mobile devices are not mobile. They stay stationary during the whole experiment to keep the simulation model simple and free from cross-layer interference. Furthermore, only one LTE cell site is used in the simulated experiment. Hence, the result may differ in a case in which mobile devices can move, and more than one LTE cell site is used. These aspects will be investigated further in our future study.

7 CONCLUSION AND FUTURE WORKS

This study focuses on studying the effect of architectural differences of a live streaming application over an LTE network in streaming to multiple service endpoints simultaneously (e.g., live streaming to Facebook Live, Youtube, and Twitch time). The study is conducted using experiments using OMNeT++ with INET Framework plugins and SimuLTE plugins installed. Two kinds of architectures are

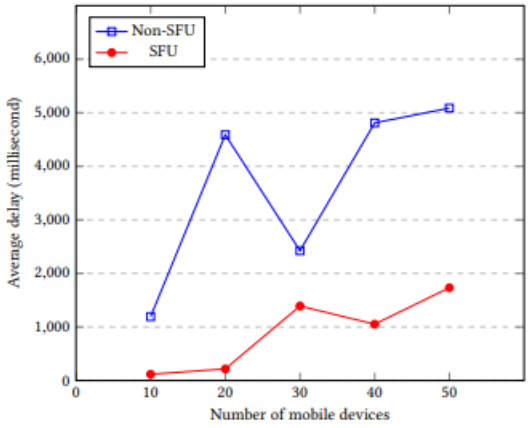


Figure 5: Average End-to-End Delay: Comparison between non-SFU and SFU-based architecture

studied, which are SFU-based and non-SFU-based architecture. The result reveals that despite the economic factor of a non-SFU based architecture, an SFU-based architecture reveals that it has more advantages in providing less network content at a cell site of an LTE network, which in turn, making the quality of service control can be done more accessible than a non-SFU based case.

The scenario of this study is limit to a case where mobile devices stay stationary during experiments, and only one LTE cell site is used. Hence, our future work will investigate further if mobile devices can move and more than one LTE cell site is added. The effect of co-existence with other applications, such as conducting a live streaming session while another client is playing online games, may be another exciting aspect to investigate in our future work.

ACKNOWLEDGMENTS

The author would like to acknowledge the Division of Computational Science, Faculty of Science, Prince of Songkla University for providing all equipments, materials, and funding required for conducting this research.

REFERENCES

[1] M Junaid Arshad, Amjad Farooq, and Abad Shah. 2010. Evolution and development towards 4th generation (4G) mobile communication systems. *Journal of American Science* 6, 12 (2010), 63–68.

[2] H.-F. Bermudez, R. Sanchez-Iborra, J.L. Arciniegas, W. Y. Campo, and M.-D. Cano. 2017. Performance validation of NS3-LTE emulation for live video streaming under QoS parameters. In *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. 300–307. https://doi.org/10.1109/WiMOB.2017.8115836

[3] Simone Brienza, Sena Efsun Cebeci, Seyed Saeid Masoumzadeh, Helmut Hlavacs, Öznur Özkasap, and Giuseppe Anastasi. 2015. A Survey on Energy Efficiency in P2P Systems: File Distribution, Content Streaming, and Epidemics. *ACM Comput. Surv.* 48, 3, Article 36 (Dec. 2015), 37 pages. https://doi.org/10.1145/2835374

[4] Lixia Hu and Qingfei Ming. 2020. Live Commerce: Understanding How Live Streaming Influences Sales and Reviews. (2020).

[5] Mauricio Iturralde, Thomas Galezowski, and Xavier Lagrange. 2018. Performance of Mobile Relays in Loaded Conditions for Railway Transportation. In *2018 16th International Conference on Intelligent Transportation Systems Telecommunications (ITST)*. 1–7. https://doi.org/10.1109/ITST.2018.8566864

[6] Xiantao Jiang, F. Richard Yu, Tian Song, and Victor C.M. Leung. 2021. A Survey on Multi-Access Edge Computing Applied to Video Streaming: Some Research Issues and Challenges. *IEEE Communications Surveys Tutorials* (2021), 1–1. https://doi.org/10.1109/COMST.2021.3065237

[7] Fidel Krasniqi, Arianit Maraj, and Emin Blaka. 2018. Performance analysis of mobile 4G/LTE networks. In *2018 South-Eastern European Design Automation, Computer Engineering, Computer Networks and Society Media Conference (SEEDA_CECNSM)*. 1–5. https://doi.org/10.23919/SEEDA-CECNSM.2018.8544937

[8] Xiangbo Li, Mohsen Amini Salehi, and Magdy Bayoumi. 2016. VLSC: Video Live Streaming Using Cloud Services. In *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*. 595–600. https://doi.org/10.1109/BDCloud-SocialCom-SustainCom.2016.93

[9] Christian Montag, Elisa Wegmann, Rayna Sariyska, Zsolt Demetrovics, and Matthias Brand. 15 Jan. 2021. How to overcome taxonomical problems in the study of Internet use disorders and what to do with "smartphone addiction"? *Journal of Behavioral Addictions JBA* 9, 4 (15 Jan. 2021), 908 – 914. https://doi.org/10.1556/2006.8.2019.59

[10] Virdis A. Stea G. Buono A. Nardini, G. 2018. SimuLTE-MEC: extending SimuLTE for Multiaccess Edge Computing. , 35–42 pages.

[11] Yesin Sahraoui, Atef Ghanam, Sofiane Zaidi, Salim Bitam, and Abdelhamid Mellouk. 2018. Performance evaluation of TCP and UDP based video streaming in vehicular ad-hoc networks. In *2018 International Conference on Smart Communications in Network Technologies (SaCoNeT)*. 67–72. https://doi.org/10.1109/SaCoNeT.2018.8585447

[12] Matti Siekkinen, Mohammad Ashraful Hoque, Jukka K. Nurminen, and Mika Aalto. 2013. Streaming over 3G and LTE: How to Save Smartphone Energy in Radio Access Network-Friendly Way. In *Proceedings of the 5th Workshop on Mobile Video* (Oslo, Norway) *(MoVid '13)*. Association for Computing Machinery, New York, NY, USA, 13–18. https://doi.org/10.1145/2457413.2457417

[13] Yanli Xu and Feng Liu. 2017. QoS Provisionings for Device-to-Device Content Delivery in Cellular Networks. *IEEE Transactions on Multimedia* 19, 11 (2017), 2597–2608. https://doi.org/10.1109/TMM.2017.2700208

ประวัติผู้เขียน

ชื่อ สกุล นายชารีฟ ประภาวิทย์

รหัสประจำตัวนักศึกษา 6210220095

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิทยาศาสตรบัณฑิต (คณิตศาสตร์)	มหาวิทยาลัยสงขลานครินทร์	2560

การตีพิมพ์เผยแพร่ผลงาน

Prapawit, C., & Angsuchotmetee, C. (2021, June). A Study of Effect of Architectural Design on Quality of Service of a Live Streaming Application with Multiple Endpoints over LTE Network. In *The 12th International Conference on Advances in Information Technology* (pp. 1-6).