การส่งข้อมูลไลฟ์สตรีมแบบเพียร์ทูเพียร์โดยใช้วิธีการแบบผลัก และดึงด้วยความน่าจะเป็น

นางสาวขวัญจิรา นาคเดช

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

PEER-TO-PEER LIVE DATA STREAMING USING A PROBABILISTIC PUSH-PULL APPROACH

Ms. Kwanjira Narkdej

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2011

หัวข้อวิทยานิพนธ์	การส่งข้อมูลไลฟ์สตรีมแบบเพียร์ทูเพียร์โดยใช้วิธีการแบบผลัก และดึงด้วยความน่าจะเป็น
โดย	นางสาวขวัญจิรา นาคเดช
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร. เฉลิมเอก อินทนากรวิวัฒน์

คณะวิศวกรรมศาสตร์จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

…….. คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

…… ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. เกริก ภิรมย์โสภา)

……... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร. เฉลิมเอก อินทนากรวิวัฒน์)

…… กรรมการ
(อาจารย์ ดร. กุลธิดา โรจน์วิบูลย์ชัย)

…… กรรมการภายนอกมหาวิทยาลัย
(รองศาสตราจารย์ ดร. อนันต์ ผลเพิ่ม)

ขวัญจิรา นาคเดช : การส่งข้อมูลไลฟ์สตรีมแบบเพียร์ทูเพียร์โดยใช้วิธีการแบบผลักและดึงด้วยความน่าจะเป็น. (PEER-TO-PEER LIVE DATA STREAMING USING A PROBABILISTIC PUSH-PULL APPROACH) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร. เฉลิมเอก อินทนากรวิวัฒน์, 50 หน้า.

ปัจจุบัน ความต้องการในการเข้าถึงข้อมูลไลฟ์สตรีมมิ่ง โดยผ่านทางเครือข่ายคอมพิวเตอร์มีแนวโน้มที่มากขึ้น และจากข้อจำกัดของเซิร์ฟเวอร์ที่ไม่สามารถรองรับผู้ขอรับบริการที่มีจำนวนมากขึ้นได้ การนำเครือข่ายเพียร์ทูเพียร์มาช่วยในการกระจายข้อมูลไลฟ์สตรีมมิ่งจึงมีประโยชน์อย่างมาก โดยจากการศึกษาพบว่าการกระจายข้อมูลแบบผลักจะช่วยลดความหน่วงในการกระจายข้อมูลได้ และการกระจายข้อมูลบนโอเวอร์เวย์ตาข่ายมีข้อดีมากมาย อาทิ โครงสร้างของโอเวอร์เลย์ที่ไม่ซับซ้อน ง่านต่อการสร้างและดูแล คงทนต่อการเปลี่ยนแปลงของเน็ตเวิร์ค ข้อมูลที่ส่งบนโอเวอร์เลย์ตาข่ายได้คุณภาพสูงดังนั้นงานวิจัยนี้ ได้นำข้อดีของการกระจายข้อมูลแบบผลัก และโอเวอร์เลย์ตาข่ายมาใช้ แต่อย่างไรก็ตาม การผลักข้อมูลบนโอเวอร์เลย์ตาข่ายทำให้โหนดผู้รับได้ชิ้นข้อมูลซ้ำ ซึ่งเป็นปัญหาที่วิทยานิพนธ์ฉบับนี้ได้เสนอวิธีการแก้ไข

วิทยานิพนธ์ฉบับนี้ได้เสนอการออกแบบเพื่อช่วยลดข้อมูลซ้ำที่เกิดขึ้น สำหรับการส่งข้อมูลไลฟ์สตรีมมิ่งแบบผลัก ร่วมกับการดึงหากมีข้อมูลสูญหาย บนโอเวอร์เลย์ตาข่าย ผู้วิจัยได้ออกแบบ และเสนอวิธี Reinforcement-based ซึ่งทำให้ข้อมูลซ้ำที่เกิดขึ้นในระบบลดลง เป็นผลทำให้การแบนด์วิทที่สูญเสียไปกับข้อมูลที่ไม่เป็นประโยชน์ลดลงด้วย การทดลอง ได้ใช้โปรแกรมจำลอง NS-2.34 บนสถานการณ์เสมือนจริง ผลการทดลองแสดงให้เห็นว่า วิธีการที่นำเสนอทำให้เกิดข้อมูลซ้ำในระบบลดลง

ภาควิชา.....วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อนิสิต.....
สาขาวิชา.....วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก.....
ปีการศึกษา.....2554.....

5270230021 : MAJOR COMPUTER ENGINEERING
KEYWORDS : PEER-TO-PEER / LIVE STREAMING / MESH OVERLAY / PUSH-PULL / REINFORCEMENT

KWANJIRA NARKDEJ : PEER-TO-PEER LIVE DATA STREAMING USING A PROBABILISTIC PUSH-PULL APPROACH. ADVISOR : ASST. PROF. CHALERMEK INTANAGONWIWAT, Ph.D., 50 pp.

The demand of accessing live streaming data is increasing but server bandwidth can't support all users. Peer-to-peer network is useful for distributing live streaming data. The study found that pushing data can reduce dissemination delay and mesh overlay has many advantages such as, the uncomplicated overlay structure, easy to maintenance and construct, high tolerant. Data that transmitted is also high quality over mesh overlay. This thesis proposed algorithm that takes the advantage of pushing data and mesh overlay. However, pushing data over mesh overlay causes duplicate data at receiver. This is the problem that this thesis proposed the solution.

This thesis propose the algorithm for decrease the duplicate data that is occurred from pushing data over mesh. And pull the data if data loss. We design and propose Reinforcement-based that reduce the duplicated data and decrease bandwidth used in transferring unused data. We use the simulator NS-2.34 to test our protocol. The results show that the protocol can greatly reduce the duplicated data in system.

Department : Computer Engineering Student's Signature .....
Field of Study : Computer Engineering Advisor's Signature .....
Academic Year : 2011

กิตติกรรมประกาศ

ผู้วิจัยขอขอบพระคุณสำหรับข้อเสนอแนะ แนวคิด ความช่วยเหลือ และกำลังใจจากอาจารย์ที่ปรึกษา ผศ.ดร.เฉลิมเอก อินทนากรวิวัฒน์ ซึ่งเป็นอาจารย์ที่มีความสามารถทั้งด้านการวิจัย และด้านการเรียนการสอน ซึ่งได้ผลักดันให้งานวิจัยมีความก้าวหน้าจนมาเป็นวิทยานิพนธ์ฉบับนี้

ขอขอบพระคุณ ผศ.ดร. เกริก ภิรมย์โสภา, อ.ดร. กุลธิดา โรจน์วิบูลย์ชัย และ รศ.ดร. อนันต์ ผลเพิ่ม คณะกรรมการสอบวิทยานิพนธ์ฉบับนี้ ที่ได้สละเวลามาให้ข้อเสนอแนะและข้อวิจารณ์เชิงวิชาการ ซึ่งนำไปสู่การพัฒนางานวิจัยที่ดียิ่งขึ้น

ขอขอบคุณ จุฬาลงกรณ์มหาวิทยาลัย ที่ให้โอกาสสำหรับการศึกษาในระดับปริญญาโท คณาจารย์ทุกท่านที่มีความรู้ความสามารถ และให้ความรู้เป็นอย่างดี

ขอขอบคุณ พี่ปิง พี่อ็อป พี่เต้ พี่พงษ์ น้องบูม พี่ต่อ อิด ทีป ป๊อป ยุ้ย นิน น้องอั๋น น้องอั้ม สำหรับทุกข้อคิดเห็น คำถาม ข้อแนะนำที่เป็นประโยชน์ต่องานวิจัย และกำลังใจที่มีให้เสมอมา

ขอขอบคุณ ทุกคนในครอบครัว คุณพ่อ คุณแม่ พี่ชาย ที่คอยสนับสนุนเงินทุนและให้กำลังใจ เพื่อนๆ ทุกคน ที่คอยให้คำปรึกษา และเป็นกำลังใจให้กับผู้วิจัยเสมอมา

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

บทที่1

บทนำ

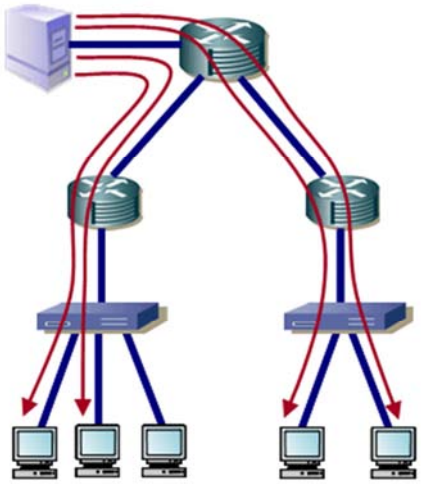
1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันความต้องการในการเข้าถึงข้อมูลไลฟ์สตรีมมิ่ง (Live Streaming Data) ผ่านทางเครือข่ายคอมพิวเตอร์ หรือที่เรียกกันว่าไอพีทีวี (IPTV) ยังคงมีแนวโน้มที่เพิ่มมากขึ้นเรื่อยๆ เนื่องจากการส่งข้อมูลแบบไลฟ์สตรีมมิ่งสามารถเป็นสื่อกลางในการเผยแพร่ข้อมูลที่มีประโยชน์ให้กับผู้รับข้อมูลที่มีจำนวนมากในช่วงเวลาเดียวกัน ไม่ว่าจะเป็นข้อมูลเพื่อการศึกษา เพื่อความบันเทิง หรือเพื่อธุรกิจ

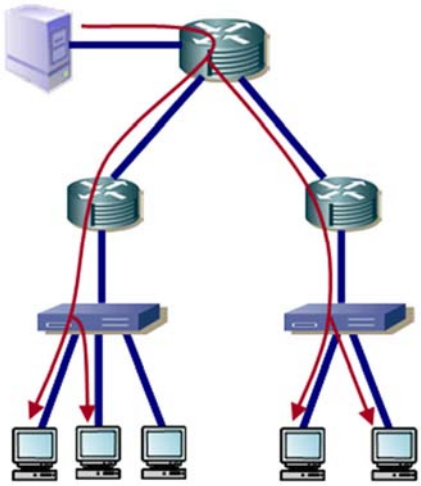
ข้อมูลไลฟ์สตรีมมิ่ง เป็นข้อมูลที่ประกอบไปด้วยภาพ และเสียง ดังนั้นในการส่งข้อมูลจึงจำเป็นต้องใช้ช่องสัญญาณ หรือแบนด์วิดท์ (Bandwidth) ที่มากกว่าการส่งข้อมูลในรูปแบบอื่นๆ และด้วยแนวโน้มความต้องการที่เพิ่มขึ้น ทำให้ผู้ให้บริการจำเป็นต้องเพิ่มขนาดของช่องสัญญาณให้กับโหนดผู้ส่ง หรือเครื่องเซิร์ฟเวอร์ (Server) เพื่อรองรับปริมาณข้อมูลที่เพิ่มมากขึ้น ทำให้สามารถส่งข้อมูลนั้นไปยังโหนดผู้รับทั้งหมดได้ โดยที่คุณภาพของข้อมูลไม่ลดลง ดังภาพที่ 1.1

การนำสำเนาข้อมูลกระจายไปตามเครื่องเซิร์ฟเวอร์ต่างๆ ที่ใกล้กับโหนดผู้รับข้อมูล (Content Distribution Network : CDN) เป็นการแก้ปัญหาวิธีหนึ่งที่สามารถรองรับผู้รับข้อมูลจำนวนมากได้ และทำให้ผู้รับข้อมูลก็จะได้ข้อมูลที่มีคุณภาพสูง แต่ก็นำมาซึ่งค่าใช้จ่าย (Cost) ที่สูงขึ้นตามไปด้วยแม้ว่าการแก้ปัญหาด้วยวิธีเพิ่มช่องสัญญาณที่เครื่องเซิร์ฟเวอร์ หรือการกระจายสำเนาไปตามเครื่องเซิร์ฟเวอร์ต่างๆ จะสามารถรองรับผู้รับข้อมูลได้เพิ่มขึ้น แต่ปัญหาเรื่องความสามารถในการรองรับการกระจายข้อมูล (Scalable) ก็ยังคงอยู่เนื่องจาก หากมีผู้รับข้อมูลจำนวนมากขึ้น ผู้ให้บริการจำเป็นต้องเพิ่มเครื่องเซิร์ฟเวอร์ให้มากขึ้นตามจำนวนผู้รับข้อมูล

จากงานวิจัยที่ผ่านมา มีการนำเทคโนโลยีไอพีมัลติคาสต์ (IP Multicast Technology) เข้ามาช่วยในการกระจายข้อมูลไปยังโหนดผู้รับหลายโหนดในเวลาเดียวกัน โดยจะมีข้อมูลเพียงชุดเดียวเท่านั้นที่จะถูกส่งไปยังกลุ่มผู้รับ หรือที่เรียกว่า กลุ่มมัลติคาสต์ (Multicast Group)โดยมีอุปกรณ์เครือข่าย อาทิ เราท์เตอร์ (Router) หรือสวิตช์ (Switch) ที่สามารถรองรับการทำงานในระบบไอพีมัลติคาสต์เป็นตัวกลางในการส่งข้อมูล วิธีนี้จะช่วยทำให้ช่วยประหยัดแบนด์วิดท์สำหรับการส่งข้อมูลไปยังทุกโหนดในได้เป็นอย่างมากดังภาพที่ 1.2



ภาพที่ 1.1 การส่งข้อมูลแบบยูนิคาสต์ (Unicast)



ภาพที่ 1.2 การส่งข้อมูลโดใช้เทคโนโลยีไอพีมัลติคาสต์ (IP Multicast)

แต่เนื่องจากข้อจำกัดบางประการ ทำให้ไม่สามารถนำเทคโนโลยีไอพีมัลติคาสต์มาประยุกต์ใช้งานบนเครือข่ายจริงได้ ด้วยเหตุผลต่างๆ อาทิ เราท์เตอร์บางรุ่นที่ใช้งานอยู่ในปัจจุบันไม่สามารถรองรับการใช้เทคโนโลยีไอพีมัลติคาสท์ได้หรือการที่จะติดตั้ง (Install) เทคโนโลยีนี้ให้กับเราท์เตอร์ทุกเครื่องเป็นเรื่องที่ยุ่งยาก หรือเหตุผลของการทำงานของเราท์เตอร์ โดยหากมีการนำเทคโนโลยีนี้มาใช้ เราท์เตอร์จะมีภาระหน้าที่มากขึ้น

จากปัญหา และข้อจำกัดที่กล่าวมา งานวิจัยภายหลัง จึงได้หันมาศึกษาการส่งข้อมูลโดยใช้แอพพลิเคชั่นเลเยอร์มัลติคาสต์แทน (Application Layer Multicast) [1]

แอพพลิเคชั่นเลเยอร์มัลติคาสต์ เป็นการส่งข้อมูล โดยที่เครื่องคอมพิวเตอร์แต่ละเครื่องจะสร้างโทโพโลยีทางตรรกะ (Logical Topology) หรือเครือข่ายโอเวอร์เลย์ (Overlay

Network) ขึ้นมาที่ชั้นแอพพลิเคชั่น (Application Layer) เพื่อเชื่อมต่อกับเครื่องอื่นๆ และส่งข้อมูลโดยที่มีเครื่องคอมพิวเตอร์ปลายทาง (End-System or End-User) คอยทำหน้าที่ส่งต่อข้อมูลไปยังเครื่องอื่นๆ ที่อยู่บนเครือข่ายโอเวอร์เลย์เดียวกัน

เครือข่ายเพียร์ทูเพียร์ (Peer-to-peer Network) ได้ใช้แนวคิดของแอพพลิเคชั่นเลเยอร์มัลติคาสต์เพื่อกระจายข้อมูลไปยังโหนดผู้รับอื่นๆ เครือข่ายเพียร์ทูเพียร์เป็นระบบกระจาย (Distributed System) ซึ่งข้อดีของเครือข่ายแบบนี้ คือ เพิ่มความคงทนให้กับระบบ (Robustness) ขจัดปัญหาการสูญเสียสัญญาณ ณ จุดใดจุดหนึ่ง (Single Point of Failure : SPOF) และทำให้โหนดในเครือข่ายสามารถแลกเปลี่ยนข้อมูลระหว่างกันได้

ข้อมูลไลฟ์สตรีมมิ่ง ที่ถูกส่งบนเครือข่ายเพียร์ทูเพียร์ จะถูกแบ่งออกเป็นชิ้นย่อยๆ โดยมีหมายเลยกำกับแต่ละชิ้น และแต่ละชิ้นจะถูกส่งจากหลายๆ โหนด ซึ่งโหนดที่ส่งจะเป็นโหนดที่มีชิ้นข้อมูลนั้นอยู่ โดยทุกโหนดบนเครือข่ายเพียร์ทูเพียร์จะสามารถเป็นทั้งผู้ส่งต่อชิ้นข้อมูล และผู้รับชิ้นข้อมูลได้ในเวลาเดียวกัน ซึ่งในปัจจุบันมีโปรแกรมประยุกต์เพียร์ทูเพียร์ไลฟ์สตรีมมิ่ง (P2P Application Live Streaming) ที่เปิดให้ดาวน์โหลด และให้บริการแล้ว อาทิ JumpTv [2], PPLive [3], SopCast [4] ซึ่งขณะนี้โปรแกรมประยุกต์ดังกล่าว ได้ให้บริการผู้รับบริการข้อมูลอยู่เป็นจำนวนมาก

งานวิจัยระบบเพียร์ทูเพียรไลฟ์สตรีมที่ผ่านมา หากจำแนกตามประเภทเครือข่ายโอเวอร์เลย์ที่ใช้ในการส่งชิ้นข้อมูล สามารถแบ่งได้ 3 ลักษณะ คือ การส่งชิ้นข้อมูลบนโอเวอร์เลย์ต้นไม้ (Tree-based Overlay) [5], [6], [7] การส่งชิ้นข้อมูลบนโอเวอร์เลย์ตาข่าย (Mesh-based Overlay) [8], [9], [10], [11] และการส่งชิ้นข้อมูลบนโอเวอร์เลย์ผสม (Hybrid Overlay)คือ ส่งชิ้นข้อมูลทั้งบนโอเวอร์เลย์ต้นไม้ ร่วมกับการส่งชิ้นข้อมูลบนโอเวอร์เลย์ตาข่าย [12]

หรือจำแนกตามาลักษณะการส่งชิ้นข้อมูล ในระบบเพียร์ทูเพียรไลฟ์สตรีมสามารถจำแนกได้ 3 รูปแบบ คือ การส่งชิ้นข้อมูลแบบดึง (Pull-based) [8], [10] การส่งชิ้นข้อมูลแบบผลัก (Push-based) [11] และการส่งชิ้นข้อมูลแบบผสม (Push-Pull based) [9], [12] การส่งชิ้นข้อมูลแบบดึง โหนดผู้รับจะทำการดึงชิ้นข้อมูลที่ต้องการจากโหนดเพื่อนบ้าน ซึ่งเป็นลักษณะของการขับเคลื่อนทางโหนดผู้รับ (Receiver-driven) การดึงชิ้นข้อมูลจะนำมาซึ่งความหน่วงในการส่ง (Delay) ที่มาก ต่างจาการกระจายข้อมูลแบบผลัก ที่จะช่วยลดความหน่วงในการส่งข้อมูลจากโหนดผู้ส่งไปยังโหนดผู้รับได้

งานวิจัย A case for End System Multicast [1] ศึกษาการกระจายข้อมูลโดยใช้แอพพลิเคชั่นมัลติคาสท์ แทนการใช้ไอพีมัลติคาสท์โดยมีผู้รับ (End System) ทำหน้าที่

กระจายข้อมูลแทนการใช้เราท์เตอร์โดยได้นำเสนอโอเวอร์เลย์ต้นไม้เดี่ยว (Single Tree) เพื่อใช้ในการกระจายข้อมูล และใช้การส่งข้อมูลลงมาตามกิ่งของต้นไม้ แต่เนื่องจากโอเวอร์เลย์ต้นไม้เดี่ยวยังมีข้อเสีย งานวิจัย SplitStream [6] จึงออกแบบโอเวอร์เลย์เพื่อแก้ไขข้อเสียของการกระจายข้อมูลบนโอเวอร์เลย์ต้นไม้เดี่ยว โดยนำเสนอโอเวอร์เลย์ต้นไม้หลายต้น (Multi-Tree Overlay) ซึ่งนอกจากจะช่วยกระจายการใช้ช่องสัญญาณให้กับโหนดทุกโหนดในระบบและยังทำให้ระบบสามารถใช้แบนด์วิดท์ได้เพิ่มขึ้น เนื่องจากได้ใช้แบนด์วิดท์เพิ่มจากโหนดใบ (Leaf Node)

CoolStreaming/DONet [8] เสนอการกระจายข้อมูลเพียร์ทูเพียรไลฟ์สตรีมแบบดึง บนโอเวอร์เลย์ตาข่าย ซึ่งมีข้อดีคือ ไม่จำเป็นต้องดูแลรักษา (Maintain) โอเวอร์เลย์มากนัก จากการทดลองพบว่าการส่งข้อมูลไลฟ์สตรีม บนโอเวอร์เลย์ตาข่ายในเครือข่ายที่มีเปลี่ยนแปลง (Dynamic Network) ไม่ทำให้คุณภาพข้อมูล (Quality) แย่ลง นอกจากนี้ยังทำให้ความต่อเนื่องของการส่ง (Continuous) ดีขึ้น เมื่อเทียบกับการกระจายข้อมูลไลฟ์สตรีมบนโอเวอร์เลย์ต้นไม้

จากความหน่วงในการส่งข้อมูลแบบดึงที่ค่อนข้างมาก GridMedia [9] จึงทำการวิเคราะห์การกระจายข้อมูลแบบดึง พบว่าในแต่ละครั้งของการกระจายข้อมูลระหว่างโหนดผู้รับและโหนดผู้ส่ง จะเกิดความหน่วงในการส่งอย่างน้อย 3 เท่าของความหน่วงระหว่างโหนด (End-to-end Delay) จึงนำเสนอรูปแบบการกระจายข้อมูลแบบผลัก และดึง โดยจะดึงข้อมูลเฉพาะชิ้นที่หายไปเท่านั้น จากผลการทดลอง พบว่า ความหน่วงในการกระจายข้อมูลลดลงเมื่อเทียบกับการกระจายข้อมูลแบบดึงทั้งในเครือข่ายที่มีการเปลี่ยนแปลง และไม่มีการเปลี่ยนแปลง (Static Network)

mTreebone [12] เสนอการส่งข้อมูลไลฟ์สตรีมบนโอเวอร์เลย์แบบผสม คือ มีการกระจายข้อมูลแบบผลักตามโอเวอร์เลย์ต้นไม้ และมีการดึงข้อมูลในกรณีที่มีข้อมูลหายจากโอเวอร์เลย์ตาข่าย โดยทำให้ระบบมีประสิทธิภาพ และทนทานต่อการเปลี่ยนแปลงของเครือข่ายมากขึ้น เนื่องจากหากเกิดการเปลี่ยนแปลงของโอเวอร์เลย์ต้นไม้ โหนดผู้รับจะยังคงสามารถรับข้อมูลได้บนโอเวอร์เลย์ตาข่าย

งานวิจัยที่กล่าวมาข้างต้น มีรูปแบบการส่งข้อมูลบนโอเวอร์เลย์ที่แตกต่างกัน ซึ่งโอเวอร์เลย์ที่น่าสนใจ ที่ง่ายต่อการดูแล สร้างไม่ซับซ้อน ทนทานต่อเครือข่ายที่มีการเปลี่ยนแปลง และรองรับระบบที่ขยายตัวได้ (Scalable) คือ โอเวอร์เลย์ตาข่าย และในส่วนการส่งข้อมูลที่ลดความหน่วงในการส่งได้ดีที่สุดคือ การกระจายข้อมูลแบบผลัก ดังนั้น งานวิจัย Epidemic Live Stream [11] จึงได้ศึกษารูปแบบการผลักข้อมูลที่แตกต่างกันบนโอเวอร์เลย์ตาข่าย ซึ่งจากการศึกษาพบว่ารูปแบบการกระจายข้อมูลแบบผลักที่ให้อัตราการรับข้อมูลสูงสุด (Optimal Delivery Rate) และมีความหน่วงในการส่งข้อมูลน้อยสุด (Optimal Delay) ได้แก่การกระจายข้อมูลแบบผลักในรูปแบบโดยเลือกโหนดผู้รับที่มีชิ้นที่ขาดแคลนมากที่สุด (Most

Deprived Peer : Dp) และเลือกชิ้นชิ้นสุดท้ายที่มีประโยชน์ (Latest Useful Chunk : Lu) ในการส่งโดยเรียกการส่งในลักษณะนี้ว่า Dp/Lu

ดังนั้นงานวิจัยนี้จึงได้นำการกระจายข้อมูลแบบผลักรูปแบบ Dp/Lu ร่วมกับการดึงข้อมูลเมื่อมีข้อมูลหายบนโอเวอร์เลย์ตาข่าย ซึ่งจะช่วยลดความหน่วงในการส่งข้อมูลและให้อัตราการในการรับข้อมูลสูงสุด และเนื่องจากการกระจายข้อมูลแบบผลัก บนโอเวอร์เลย์ตาข่ายทำให้โหนดผู้รับได้รับข้อมูลซ้ำ เนื่องจากปัญหาความหน่วงในการส่งข้อมูลแผนที่บัฟเฟอร์ (Buffer Map Delay) และปัญหาการส่งข้อมูลชิ้นเดียวกันให้โหนดผู้รับเดียวกัน (Chunk Synchronization Problem) ดังนั้นงานวิจัยนี้จึงนำเสนอวิธีการลดข้อมูลซ้ำที่เกิดขึ้นในโหนดผู้รับ โดยทำการทดลองบนโปรแกรมจำลอง (Simulator) NS-2.34 [14] และสร้างโทโพโลยีเสมือนโดยใช้เครื่องมือ GT-ITM [15]

1.2 วัตถุประสงค์ของการวิจัย

งานวิจัยนี้มีจุดประสงค์เพื่อนำเสนอรูปแบบการส่งข้อมูลไลฟ์สตรีมโดยใช้การกระจายชิ้นข้อมูลแบบผลัก และดึงบนโอเวอร์เลย์ตาข่ายโดยลดเวลาหน่วงสำหรับการกระจายข้อมูลทนทานต่อการเปลี่ยนแปลงของเครือข่าย โดยเปรียบเทียบกับการส่งข้อมูลไลฟ์สตรีมโดยใช้การกระจายข้อมูลแบบดึงบนโอเวอร์เลย์ตาข่ายและการกระจายข้อมูลแบบผลัก และดึงบนโอเวอร์เลย์ตาข่ายก่อนหน้าและงานวิจัยนี้ยังช่วยลดการสูญเสียแบนด์วิดท์ที่เกิดจากการส่งข้อมูลซ้ำ

1.3 ขอบเขตของการวิจัย

1. การพัฒนา และการทดสอบโปรโตคอลทำบนโปรแกรมจำลอง NS-2.34 โดยนำผลที่ได้จากการทดลองมาหาค่าเฉลี่ยเพื่อแสดงประสิทธิภาพการทำงานของโปรโตคอล

2. การสร้างโทโพโลยี เพื่อจำลองลักษณะเครือข่าย จะใช้เครื่องมือ GT-ITM และสร้างโทโพโลยีที่ใช้ในการทดลองในรูปแบบของTransit-stub Model

3. ในงานวิจัยนี้ ทุกโหนดในระบบจะยินยอมอัพโหลด (Upload) ข้อมูลให้กับโหนดเพื่อนบ้าน (Neighbor Node) เสมอ กล่าวคือ งานวิจัยนี้ไม่ครอบคลุมในประเด็นเรื่อง โหนดในระบบไม่ยินยอมอัพโหลดข้อมูลให้กับโหนดอื่นๆ หรือในงานวิจัยไม่มีโหนดที่เรียกว่า “Free-Rider” หรือ “Selfish Node”

4. โหนดในระบบทุกโหนด และทุกชิ้นข้อมูลไลฟ์สตรีมมิ่ง จะมีหมายเลขระบุ โดยที่หมายเลขดังกล่าวจะไม่ซ้ำกัน (Unique ID) ชิ้นข้อมูลจะถูกสร้างขึ้นที่เครื่องเซิร์ฟเวอร์ และมีโหนดในระบบเป็นผู้กระจายข้อมูล

5. การทดลองจะเน้นเรื่องการปรับปรุงประสิทธิภาพของระบบ โดยประเด็นที่เน้น คือ ลดความหน่วงเวลาในการกระจายข้อมูล และลดการส่งข้อมูลซ้ำ

1.4 ข้อจำกัดของการวิจัย

การทดสอบโดยใช้โปรแกรมจำลอง NS-2.34 จำเป็นต้องอาศัยพื้นที่หน่วยความจำที่ค่อนข้างมาก เพื่อที่จะสามารถสร้างโหนดในการทดลองให้มีจำนวนมากขึ้น และเนื่องจากพื้นที่หน่วยความจำมีอย่างจำกัด ดังนั้นการทดลองจึงใช้จำนวนโหนดมากที่สุดที่หน่วยความจำสามารถรองรับได้ และจากการทดลองบนเครือข่ายขนาดกลาง จะสะท้อนให้เห็นประสิทธิภาพของโปรโตคอล ผ่านการควบคุม และการกำหนดโทโพโลยีให้เสมือนจริงมากที่สุด

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. วิธีที่นำเสนอ สามารถลดความหน่วงในการกระจายข้อมูลไลฟ์สตรีมมิ่งซึ่งเป็นผลให้แต่ละโหนดได้รับข้อมูลที่เวลาใกล้เคียงกัน

2. วิธีที่นำเสนอ ทำให้เกิดข้อมูลซ้ำที่โหนดผู้รับน้อยลง เพื่อไม่ทำให้สูญเสียการใช้แบนด์วิดท์ที่เกิดจากการส่งข้อมูลซ้ำ

3. วิธีที่นำเสนอ ยังคงทำงานได้อย่างมีประสิทธิภาพ แม้ว่าจะมีการเปลี่ยนแปลงในเครือข่าย

1.6 วิธีดำเนินการวิจัย

1. ศึกษาทฤษฎี และงานวิจัยที่เกี่ยวข้อง สำหรับการกระจายข้อมูลไลฟ์สตรีมมิ่งบนเครือข่ายเพียร์ทูเพียร์

2. ศึกษาการใช้งานโปรแกรมจำลอง NS-2.34 และเครื่องมือสร้างโทโพโลยี GT-ITM

3. ออกแบบวิธีการกระจายข้อมูลโดยคำนึงถึงความหน่วงในการกระจายข้อมูล และจำนวนข้อมูลซ้ำ

4. สร้างโปรแกรมตามโปรโตคอลที่ได้ออกแบบไว้บนโปรแกรมจำลอง โดยใช้โทโพโลยีและสภาพแวดล้อมเสมือนจริง

5. ทำการทดสอบ และเก็บข้อมูลการทำงานของโปรโตคอล

6. วิเคราะห์ผลการทดลอง

7. ปรับปรุงแก้ไขเพื่อเพิ่มความสามารถให้กับการทำงานของโปรโตคอล

8. สรุปวิเคราะห์ผลการวิจัยและเรียบเรียงวิทยานิพนธ์

1.7 ผลงานตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของงานวิทยานิพนธ์ได้รับการตีพิมพ์เป็นบทความวิชาการในหัวข้อเรื่อง “A Reinforcement-Based Push-Pull Approach for Peer-to-peer Live Streaming Data” โดย ขวัญจิรา นาคเดช, ศุภเสฏฐ์ ชูชัยศรี และ เฉลิมเอก อิทนากรวิวัฒน์ ในบันทึกการประชุม “2011 Wireless Communication, Networking and Mobile Computing” ซึ่งจัดขึ้น ณ โรงแรม Chutian International Hotel เมืองอู่ฮั่น (Wuhan) ประเทศจีน ระหว่างวันที่ 23-25 กันยายน 2554

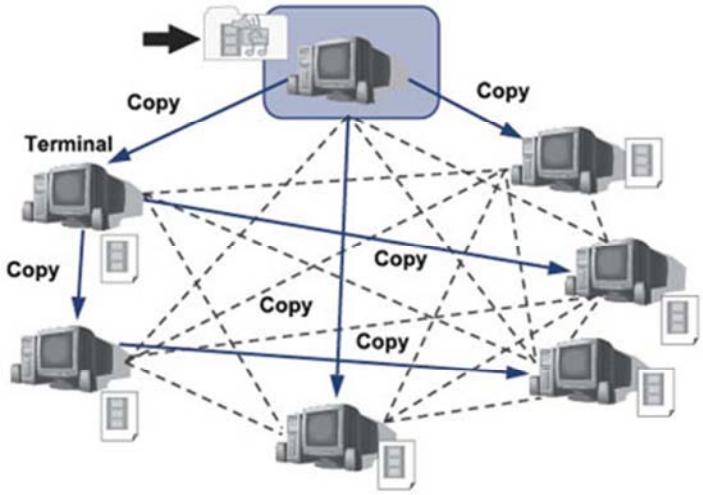
บทที่ 2

เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 เครือข่ายเพียร์ทูเพียร์ (Peer-to-peer Network)

เครือข่ายแบบเพียร์ทูเพียร์เป็นเครือข่ายที่คอมพิวเตอร์ทุก ๆ โหนดบนเครือข่ายมีความเสมอภาคเท่าเทียมกัน ทุกโหนดเชื่อมต่อกันโดยไม่จะเป็นต้องมีโหนดใดโหนดหนึ่งที่ทำหน้าที่คอยบริหารจัดการเครือข่าย ทุกโหนดสามารถเป็นได้ทั้งผู้ให้บริการ (Server) และผู้ขอรับบริการ (Client) ทุกโหนดบนเครือข่ายสามารถสื่อสารเพื่อแลกเปลี่ยนข้อมูล และทรัพยากรร่วมกันได้ โดยผ่านทางอุปกรณ์เครือข่าย ดังภาพที่ 2.1 ข้อดีของระบบนี้ คือ ช่วยกระจายภาระของระบบ ขจัดปัญหาการสูญเสียสัญญาณ ณ จุดใดจุดหนึ่ง และมีความสามารถรองรับการขยายระบบได้



ภาพที่ 2.1 เครือข่ายเพียร์ทูเพียร์ที่มีการแลกเปลี่ยนข้อมูลระหว่างกัน

โปรโตคอลที่นำรูปแบบเครือข่ายเพียร์ทูเพียร์ประยุกต์ไปใช้งาน และที่เป็นที่รู้จัก มักเป็นโปรโตคอลประเภทไฟล์แชร์ริ่ง (File Share Protocol) ได้แก่ Bittorent, WinMX, eMule, Kaza และ Napster

เครือข่ายเพียร์ทูเพียร์ สามารถจำแนกได้ 3 รูปแบบ คือ

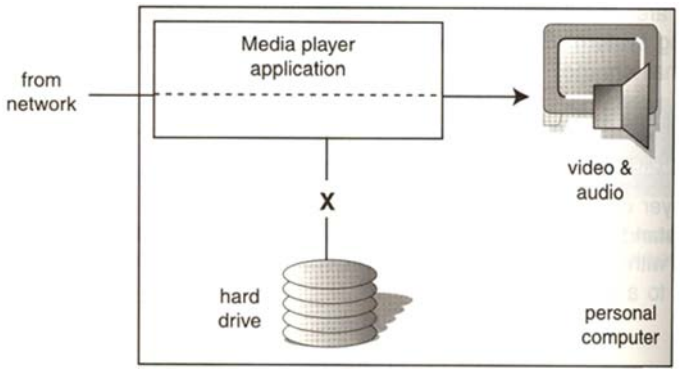
1. Centralized P2Pเป็นรูปแบบที่มีเซิร์ฟเวอร์หลักที่คอยเก็บดัชนี (Index) และตำแหน่งของไฟล์ไว้ในระบบ แต่การส่งข้อมูลจะส่งระหว่างผู้ใช้โดยตรง ซึ่งรูปแบบนี้อาจเกิดปัญหาเรื่องการสูญเสียสัญญาณ ณ จุดใดจุดหนึ่งอยู่

2. Pure P2P เป็นการเชื่อมต่อระหว่างโหนด โดยไม่มีเซิร์ฟเวอร์เก็บตำแหน่งของข้อมูล แต่จะค้นหาโหนดที่มีข้อมูลโดยตรง โดยอาจมีโปรโตคอลประเภท DHT (Distributed Hash Table Protocol) ช่วยในการค้นหา อาทิ โปรโตคอล CAN, โปรโตคอล Chord ฯลฯ

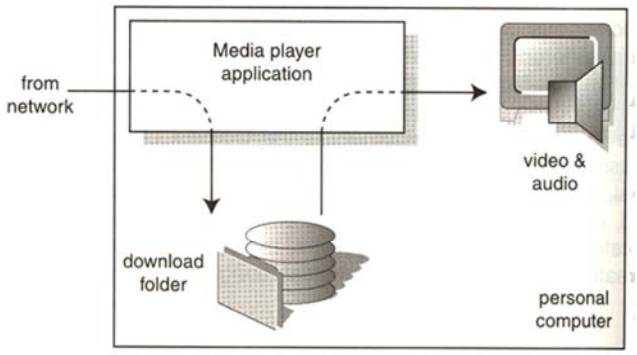
3. Hybrid P2P รูปแบบนี้จะมีโหนดกลุ่มหนึ่งที่ทำหน้าที่แทนเครื่องเซิร์ฟเวอร์ เรียกว่าโหนดกลุ่มนี้ว่า ซุปเปอร์เพียร์ (Super peers) ซึ่งจะคอยละการสื่อสารเกินความจำเป็นระหว่างโหนดผู้ใช้ลง เพราะจะมีความสามารถข้ามผ่านซุปเปอร์เพียร์แทนที่จะผ่านโหนดปกติหลายๆ ต่อจากการค้นหา

2.1.2 เทคโนโลยีสตรีมมิ่งมีเดีย (Streaming Media Technology)

เทคโนโลยีสตรีมมิ่งมีเดียเป็นเทคโนโลยีการส่งข้อมูล (ไฟล์ภาพ และเสียง) ผ่านระบบเครือข่ายอย่างต่อเนื่อง โดยข้อมูลที่ได้รับสามารถแสดงผลได้ในทันที (Real-Time) ในระหว่างการรับข้อมูล โดยไม่จำเป็นต้องรอให้ข้อมูลทั้งหมดดาวน์โหลดเสร็จสิ้นก่อน ข้อมูลที่ได้รับทั้งหมดจะถูกเก็บไว้ในบัฟเฟอร์ ก่อนที่จะนำมาแสดงผลดังภาพที่ 2.2 ซึ่งหากไม่มีเทคโนโลยีนี้แล้ว ผู้ใช้จะต้องดาวน์โหลดข้อมูลทั้งหมดมาเก็บไว้ในฮาร์ดดิส (Hard Disk) จึงจะสามารถแสดงผลได้ (Download-and-Play) ดังภาพที่ 2.3



ภาพที่ 2.2แสดงกระบวนการสตรีมมิ่ง (Streaming)



ภาพที่ 2.3 แสดงกระบวนการดาวน์โหลดแล้วจึงแสดงผล (Download-and-play)

ลักษณะการส่งข้อมูลสตรีมมิ่งมีเดียสามารถแบบได้เป็น 2 แบบ คือ

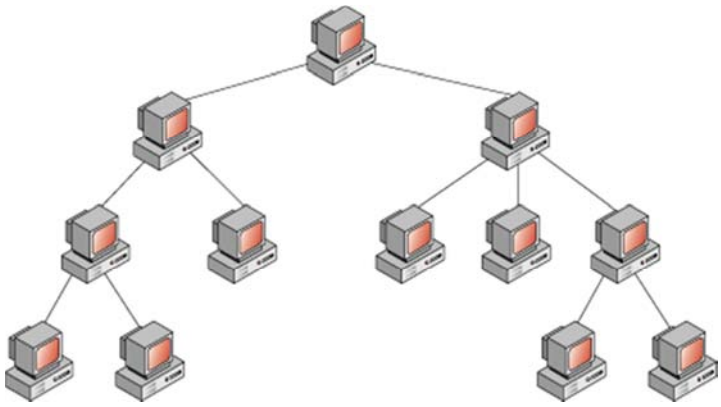
1. การถ่ายทอดสด (Live Broadcasting) เป็นการถ่ายทอดเหตุการณ์ ณ เวลานั้น โดยผู้รับชม ณ ขณะนั้นจะได้รับข้อมูลเดียวกัน

2. ไฟล์ออนดีมาน (On-Demand) ผู้ชมสามารถเข้าถึงข้อมูลได้ในทันทีที่ต้องการ โดยผู้ชมสามารถควบคุมฟังก์ชั่นการทำงานได้อย่างอิสระ อาทิ หยุดการแสดงผล (Pause) แสดงผลย้อนกลับ (Rewind) และแสดงผลซ้ำ (Replay)

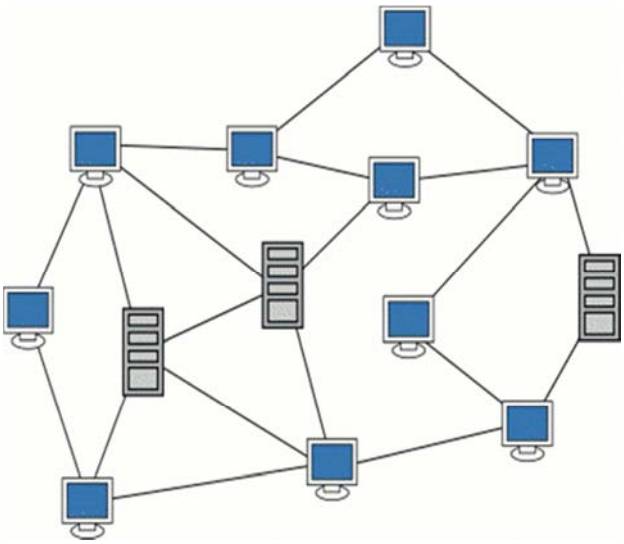
2.1.3 โทโพโลยีทางตรรกะ หรือเครือข่ายโอเวอร์เลย์ (Logical Topology or Overlay Network)

โทโพโลยีทางตรรกะ มีความแตกต่างจากโทโพโลยีทางกายภาย (Physical Topology) หมายถึง การเชื่อมต่อที่มีรูปลักษณะภายนอก เป็นแบบหนึ่ง แต่ภายในกลับมีรูปแบบการทำงานที่ไม่เหมือนกับลักษณะการเชื่อมต่อภายนอก ซึ่งโทโพโลยีทางตรรกะสามารถปรับเปลี่ยนการเชื่อมต่อตามที่ต้องการได้ โดยที่โทโพโลยีทางกายภาพยังคงเดิม เช่น ระบบเครือข่ายอีเทอร์เน็ต (Ethernet) ที่มีรูปลักษณะภายนอกเป็นแบบดาว (Star) แต่ภายในมีการเชื่อมต่อแบบบัส (Bus) และมีการรับส่งข้อมูลในรูปแบบบัส

โทโพโลยีทางตรรกะมีรูปแบบการเชื่อมโยงหลายรูปแบบ รูปแบบที่สำคัญ อาทิ โทโพโลยีแบบดาว, โทโพโลยีแบบบัส, โทโพโลยีแบบวงแหวน, โทโพโลยีแบบต้นไม้, โทโพโลยีแบบตาข่ายซึ่งในที่นี่จะขออธิบายเฉพาะโทโพโลยีทางตรรกะต้นไม้ (Tree Overlay) ดังภาพที่ 2.4 และโทโพโลยีทางตรรกะตาข่าย (Mesh Overlay) ดังภาพที่ 2.5 เท่านั้น



ภาพที่ 2.4 แสดงโครงสร้างโทโพโลยีต้นไม้



ภาพที่ 2.5 แสดงโครงสร้างโทโพโลยีตาข่าย

โทโพโลยีทางตรรกะแบบต้นไม้ หรือโทโพโลยีทางตรรกะแบบลำดับชั้น (Hierarchical Topology) มีโหนดราก (Root Node) ซึ่งอยู่ในลำดับชั้นสูงสุด (Top Level) เป็นศูนย์กลางในการกระจายข้อมูลโดยจะติดต่ออยู่กับโหนดที่อยู่ในลำดับชั้นที่ต่ำลงมา (Lower Level) ซึ่งเรียกโหนดเหล่านี้ว่าโหนดใบ (Leaf Node) ข้อเสียคือ หากเกิดปัญหาการเชื่อมต่อระหว่างโหนดราก และโหนดใบ หรือปัญหาการส่งข้อมูลที่โหนดลำดับชั้นบน โหนดที่ลำดับชั้นถัดมาจะได้รับผลกระทบไปด้วย

โอเวอร์เลย์แบบตาข่าย เป็นโอเวอร์เลย์ที่มีการติดต่อกับโหนดอื่นหลายทาง ดังนั้นการเชื่อมต่อลักษณะนี้ จึงทนทานต่อการเกิดการเชื่อมต่อล้มเหลว (Link Fail) เนื่องจากมีเส้นทางสำรองอื่นๆ

2.1.4 ลักษณะการกระจายข้อมูล (Data Dissemination)

ลักษณะการกระจายข้อมูลสามารถแบ่งได้ 2 แบบ คือ ลักษณะการกระจายข้อมูลแบบดึง (Pull-based Dissemination) และลักษณะการกระจายข้อมูลแบบผลัก (Push-based Dissemination)

1. การกระจายข้อมูลแบบดึง คือ โหนดที่ต้องการข้อมูลต้องร้องขอไปยังโหนดที่มีข้อมูล เมื่อโหนดที่มีข้อมูลได้รับคำร้องขอ จึงจะส่งข้อมูลนั้นกลับมาให้

2. การกระจายข้อมูลแบบผลัก คือ โหนดที่มีข้อมูลจะส่งข้อมูลออกมา โดยไม่ต้องรอคำร้องขอจากผู้รับดังนั้นการกระจายข้อมูลในลักษณะนี้จึงต้องสามารถคาดได้ว่า ผู้รับต้องการข้อมูลใดบ้าง

2.2 เอกสารและงานวิจัยที่เกี่ยวข้อง

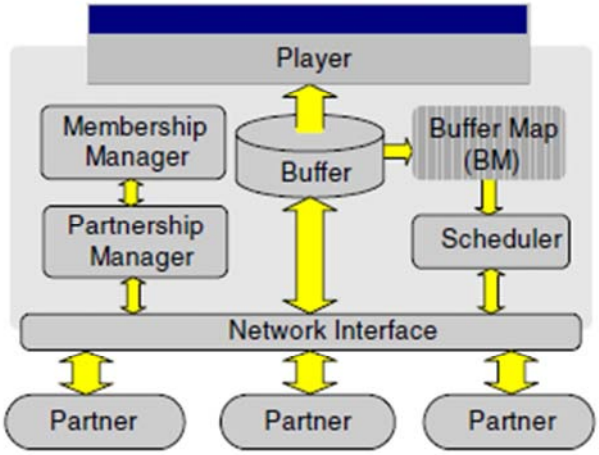
CoolStreaming/Donet [8] ได้นำเสนอรูปแบบการส่งข้อมูลบนเครือข่ายโอเวอร์เลย์ โดยการออกแบบโอเวอร์เลย์คำนึงถึง 3 ปัจจัยหลัก คือ

1. ง่ายต่อการนำไปใช้งาน คือ ไม่ต้องมีโครงสร้างโอเวอร์เลย์ที่ซับซ้อน

2. มีประสิทธิภาพ ข้อมูลที่ส่งไม่จำเป็นต้องมีทิศทางการส่งข้อมูลที่แน่นอน ซึ่งจะแตกต่างจากการส่งชิ้นข้อมูลบนโอเวอร์เลย์ต้นไม้ กล่าวคือ ทิศทางการส่งชิ้นข้อมูลบนโอเวอร์เลย์ต้องสามารถเปลี่ยนแปลงได้ตลอด โดยขึ้นอยู่กับการคงอยู่ของข้อมูลนั้น (Data Availability)

3. ความคงทนของระบบ เมื่อเครือข่ายเกิดการเปลี่ยนแปลง เช่น โหนดผู้ส่งออกไปจากระบบ โหนดผู้รับสามารถเลือกโหนดผู้ส่งใหม่ได้ทันที โดยไม่จำเป็นต้องสร้างโอเวอร์เลย์ใหม่ ทำให้ไม่เกิดความหน่วงของการส่งข้อมูลเนื่องจากการสร้างโอเวอร์เลย์ใหม่

โดยงานวิจัยนี้ ทุกโหนดในระบบ จะมีการแลกเปลี่ยนแผนที่บัฟเฟอร์กับโหนดเพื่อนบ้านอย่างต่อเนื่อง แผนที่บัฟเฟอร์ คือ ข้อความที่ประกาศไปยังเพื่อนบ้าน เพื่อบอกสภาพบัฟเฟอร์ของโหนดตัวเอง ณ ขณะนั้น ให้โหนดเพื่อนบ้านรับรู้ว่ามีชิ้นข้อมูลใดอยู่บ้าง โดยการแลกเปลี่ยนแผนที่บัฟเฟอร์จะกระทำอย่างต่อเนื่อง จากภาพที่ 2.6 แสดงให้เห็นระบบการทำงานของแต่ละโหนดให้ระบบ

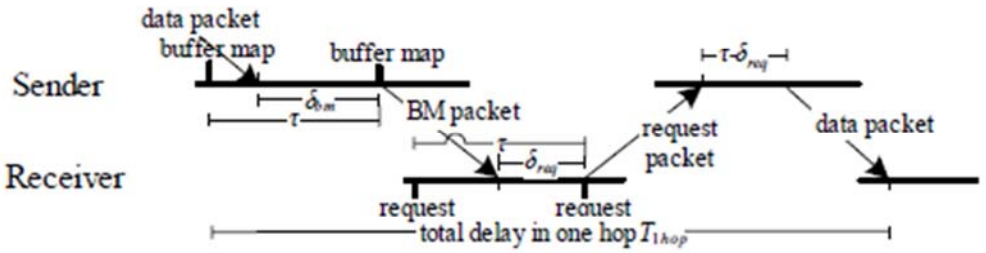


ภาพที่ 2.6 แสดงไดอะแกรมของโหนดในระบบ

งานวิจัยนี้ได้ทำการวัดประสิทธิภาพในเทอมต่างๆ ดังนี้ โอเวอร์เฮดสำหรับควบคุม (Control Overhead) โดยจะเพิ่มขึ้นตามจำนวนของโหนดเพื่อนบ้าน ความต่อเนื่องของการเล่น (Playback Continuity) ซึ่งจะเพิ่มขึ้นตามจำนวนของเพื่อนบ้าน เหตุผลเนื่องจากแต่ละโหนดสามารถเลือกผู้ส่งได้เพิ่มขึ้น และระบบนี้ยังสามารถรองรับการรองรับการขยายเครือข่าย (Scalability) ได้ เนื่องจากการที่แต่ละโหนดจะแลกเปลี่ยนข้อมูลและ แผนที่บัฟเฟอร์กับแค่เฉพาะโหนดเพื่อนบ้านเท่านั้น

แต่เนื่องจากการกระจายข้อมูลแบบดึง ดังนั้นจึงทำให้มีความหน่วงในการกระจายข้อมูลจะสูงมาก ทำให้มีความแตกต่างของตำแหน่งการเล่นระหว่างโหนดแรกที่ได้รับข้อมูล กับโหนดสุดท้ายที่สุดท้ายที่ได้รับข้อมูลมาก ซึ่งทำให้การแสดงข้อมูลไม่เรียลไทม์ (Real Time)

GridMedia [9] งานวิจัย GridMedia ใช้การกระจายข้อมูลโดยการผลัก ร่วมกับการดึง (Push-Pull based) สำหรับข้อมูลไลฟ์สตรีม บนโอเวอร์เลย์ตาข่ายโดยมีการเลือกโหนดเพื่อที่จะมาเป็นโหนดเพื่อนบ้านแบบสุ่ม (Random) โดยมีแนวคิดที่ว่า การส่งข้อมูลโดยการผลักสามารถลดความหน่วงในการส่งในระบบได้ดีกว่าการใช้การกระจายข้อมูลโดยการดึงเพียงอย่างเดียว โดยแสดงขั้นตอนของการกระจายชิ้นข้อมูลแบบดึง ดังภาพที่ 2.7



ภาพที่ 2.7 แสดงขั้นตอนการกระจายข้อมูลแบบดึง

1. หลังจากโหนดผู้รับได้รับชิ้นข้อมูล จะทำการประกาศแผนที่บัฟเฟอร์ให้โหนดผู้รับ โดยจะส่งในจังหวะเวลาถัดไป (Next Interval) ซึ่งมีเวลาการรอเท่ากับ δ_{bm}

2. ถ้าหากโหนดผู้รับต้องการชิ้นข้อมูลที่โหนดผู้ส่งประกาศมา โหนดผู้รับจะส่งคำร้องขอ (Request Massage) ไปยังโหนดผู้ส่งเพื่อขอชิ้นข้อมูลดังกล่าว โดยการส่งคำร้องขอในจังหวะเวลาถัดไป ซึ่งมีเวลาการรอเท่ากับ δ_{req}

3. สุดท้ายหลังจากโหนดผู้ส่งได้รับข้อความร้องขอ โหนดผู้ส่งจะส่งชิ้นข้อมูลกลับไปยังโหนดผู้รับ โดยมีเวลาการรอเท่ากับ τ-δ_{req}

จากขั้นตอนดังกล่าว เป็นการกระจายข้อมูลใน 1 ฮอพ (Hop) โดยสามารถคำนวนผลรวมความหน่วงในการส่งข้อมูลแสดงดังภาพที่ 2.8

$$\delta_{bm} + \delta_{req} + \left(\tau - \delta_{req}\right) + 3\overline{\delta_{EED}} = 3\tau/2 + 3\overline{\delta_{EED}}$$

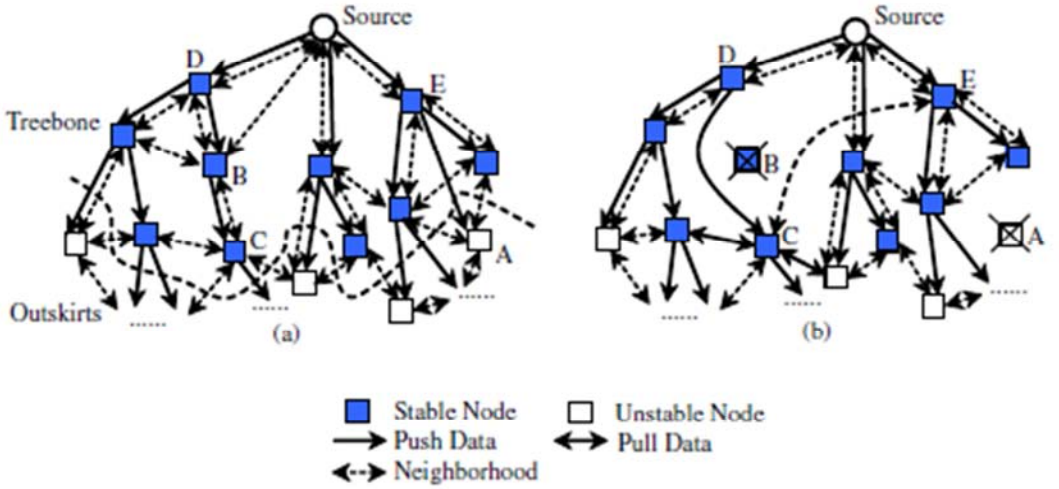
ภาพที่ 2.8 แสดงค่าผลรวมความหน่วงที่เกิดขึ้นจากการกระจายชิ้นข้อมูลแบบดึง

โดยกำหนด τ เป็นจังหวะเวลา (Interval) ของการส่งแผนที่บัฟเฟอร์และการส่งคำร้องขอ , δ_{bm} เป็นเวลาการรอ (Waiting Time) ของการส่งแผนที่บัฟเฟอร์ และ δ_{req} เป็นเวลาการรอของการส่งคำร้องขอ

ผลการทดลองการกระจายชิ้นข้อมูลแบบผลัก และดึง สามารถลดความหน่วงเวลาสำหรับการเล่น (Playback Time) ลงมาอยู่ที่ 3 วินาที เมื่อเปรียบเทียบกับการใช้การกระจายข้อมูลแบบการดึงที่มีเวลาความหน่วงเวลาสำหรับการเล่นเท่ากับ 10 วินาที ในครือข่ายคงที่และในเครือข่ายที่มีการเปลี่ยนแปลงการกระจายชิ้นข้อมูลแบบผลัก และดึงใช้เวลาเพียงแค่ 13 วินาที เมื่อเทียบกับ 22 วินาทีของการใช้การกระจายข้อมูลแบบการดึง

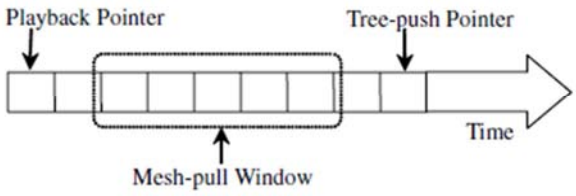
จากงานวิจัยดังกล่าว การกระจายข้อมูลแบบผลักช่วยลดความหน่วงสำหรับการเล่นได้เมื่อเปรียบเทียบกับการกระจายแบบดึงแต่งานวิจัยนี้ได้ใช้การผลักข้อมูลโดยการสุ่มเลือกผู้รับ และสุ่มเลือกชิ้น ทำให้เวลาการกระจายข้อมูลยังไม่ดีเท่าที่ควร

mTreebone[12] งานวิจัย mTreebone นำเสนอโอเวอร์เลย์แบบผสมโดยให้โอเวอร์เลย์ต้นไม้ ทำหน้าที่เสมือนเป็นแกนหลัก (Backbone) จึงเรียกว่า“Treebone” ดังภาพที่ 2.9 โดยเมื่ออยู่ในสภาวะปกติ แต่ละโหนดจะผลักชิ้นข้อมูลอย่างต่อเนื่อง (Sequence) ลงมาตามกิ่งของโอเวอร์เลย์ต้นไม้



ภาพที่ 2.9 (a) โอเวอร์เลย์แบบผสม (b) โอเวอร์เลย์ใหม่เมื่อเครือข่ายเปลี่ยนแปลง

แต่หากเกิดการเปลี่ยนแปลงของโอเวอร์เลย์ต้นไม้ หรือชิ้นข้อมูลส่งมาไม่ทันจะทำให้เกิดช่องว่าง (Gap) หรือชิ้นข้อมูลหายไปในหน้าต่างของการดึง (Mesh-pull Window) ดังภาพที่ 2.10 mTreebone ก็จะส่งคำร้องขอไปยังโหนดเพื่อนบ้าน เพื่อขอชิ้นข้อมูลที่ต้องการ โดยแต่ละโหนดจะสุ่มเลือกบางโหนดให้เป็นโหนดเพื่อนบ้าน และแลกเปลี่ยนแผนที่บัฟเฟอร์ระหว่างกันอยู่ตลอดเวลา



ภาพที่ 2.10 หน้าต่างของการดึง และการผลัก

จากการใช้โอเวอร์เลย์แบบผสมทำให้ความหน่วงเวลาสำหรับการเล่นน้อยกว่างานวิจัย Coolstream/DONET ซึ่งเป็นการกระจายชิ้นข้อมูลแบบดึงและสามารถกระจายชิ้น

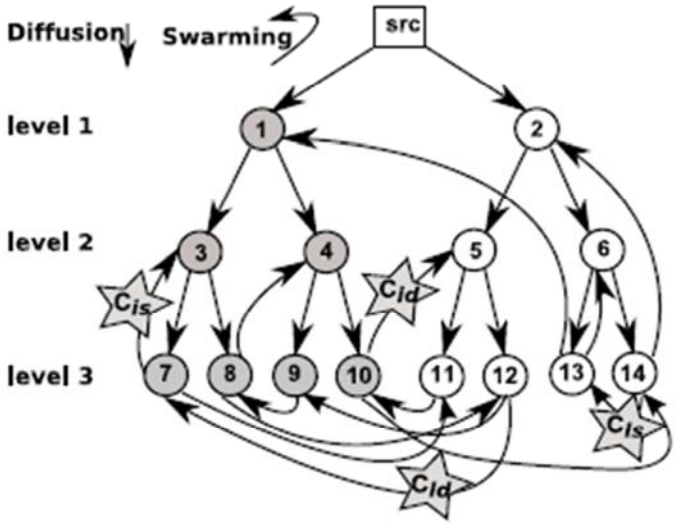
ข้อมูลได้เร็วกว่าเมื่อเทียบกับ การกระจายชิ้นข้อมูลบนโอเวอร์เลย์ต้นไม้หลายต้น (Multiple Tree) และเมื่อเปรียบเทียบโอเวอร์เฮด ระหว่าง mTreebone กับ Coolstream/DONET พบว่า โอเวอร์เฮดของ mTreebone ลดลง

การส่งข้อมูลบนโอเวอร์เลย์ที่ต้องมีโครงสร้าง ดังเช่น โอเวอร์เลย์ต้นไม้ จะมีโอเวอร์เฮดที่เกิดขึ้น และต้องมีการดูแลโครงสร้างอยู่เสมอ ซึ่งงานวิจัยต่อมาจะทดสอบว่า การส่งบนเวอร์เลย์แบบใดที่ให้ประสิทธิภาพที่ดีกว่า

Mesh or Multiple-Tree [18] ทดลองเปรียบเทียบความแตกต่างของการส่งข้อมูลไลฟ์สตรีมมิ่งบนโอเวอร์เลย์ตาข่าย (Mesh Overlay) และโอเวอร์เลย์ต้นไม้หลายต้น (Multiple Tree Overlay)

สำหรับบนโอเวอร์เลย์ตาข่าย จะสร้างโดยการสุ่มเลือกโหนดเพื่อนบ้าน ซึ่งโดยปกติแล้ว การส่งข้อมูลบนโอเวอร์เลย์ตาข่ายจะสามารถส่งข้อมูลได้ 2 ทิศทาง (Bi-Direction) ซึ่งต่างกับการส่งบนโอเวอร์เลย์ต้นไม้หลายต้น แต่ในการศึกษาครั้งนี้ ได้กำหนดให้มีการส่งแบบทิศทางเดียว (Uni-Direction) คล้ายกับความสัมพันธ์แบบพ่อ-ลูก (Parent-Child) เพื่อความสะดวกให้การเปรียบเทียบ

ความคล้ายคลึงกันระหว่างโอเวอร์เลย์สองแบบนี้คือ ข้อมูลมาจากหลายผู้ส่งแล้วสามารถส่งต่อไปยังหลายผู้รับได้ สำหรับโอเวอร์เลย์ต้นไม้หลายต้น ผู้ส่งต้นทาง (Source) จะส่งข้อมูลให้กับโหนดรากของต้นไม้แต่ละต้น ส่วนความแตกต่างกันระหว่าง 2 โอเวอร์เลย์นี้ คือ ในโอเวอร์เลย์ต้นไม้ หากข้อมูลไม่สามารถส่งผ่านไปยังโหนดผู้รับอาจเนื่องจากแบนด์วิดท์ไม่เพียงพอที่ส่วนเชื่อมต่อ (Link) ของต้นไม้ใดต้นไม้หนึ่ง จะทำให้โหนดลำดับชั้นล่างไม่ได้รับข้อมูลบนต้นไม้นั้นเลย แต่ในโอเวอร์เลย์ตาข่ายข้อมูลนั้นยังคงสามารถส่งได้จากโหนดผู้ส่งอื่น ดังภาพที่ 2.11



ภาพที่ 2.11 การจัดการเชื่อมต่อของโอเวอร์เลย์ตาข่าย

จากภาพตัวอย่างที่ 2.11 หากข้อมูลจากโหนดที่ 1 ไปโหนดที่ 4 ไม่สามารถส่งได้ โหนดที่อยู่ในลำดับชั้นถัดไป ซึ่งในที่นี้คือโหนดที่ 9 และโหนดที่ 10 จะได้รับผลกระทบหากส่งบนโอเวอร์เลย์ต้นไม้ แต่สำหรับบนโอเวอร์เลย์ตาข่ายโหนดที่ 4 ยังคงสามารถรับชิ้นข้อมูลจากโหนดที่ 8 ได้ ซึ่งเป็นส่วนสำคัญที่ทำให้การส่งข้อมูลบนโอเวอร์เลย์ตาข่ายดีกว่า

จากการทดลองส่งข้อมูลบนเครือข่ายที่มีการเปลี่ยนแปลง (Dynamic Network) โดยโอเวอร์เลย์ถูกเปลี่ยนรูปตลอดเวลา เนื่องจากโหนดมีการเข้าออก พบว่าโอเวอร์เลย์ตาข่ายยังคงให้อัตราการส่งข้อมูลที่สูงกว่ามาก เมื่อเทียบกับโอเวอร์เลย์ต้นไม้หลายต้น นอกจากนี้ยังแสดงให้เห็นว่าการส่งข้อมูลบนโอเวอร์เลย์ตาข่าย ยังให้ภาพรวมของการใช้แบนด์วิดท์ และคุณภาพของข้อมูลที่ดีกว่า

ซึ่งงานวิจัยนี้ได้สนับสนุนแนวคิดของการนำโอเวอร์เลย์ตาข่ายมาใช้ในการส่งชิ้นข้อมูลประเภทไลฟ์สตรีมมิ่ง

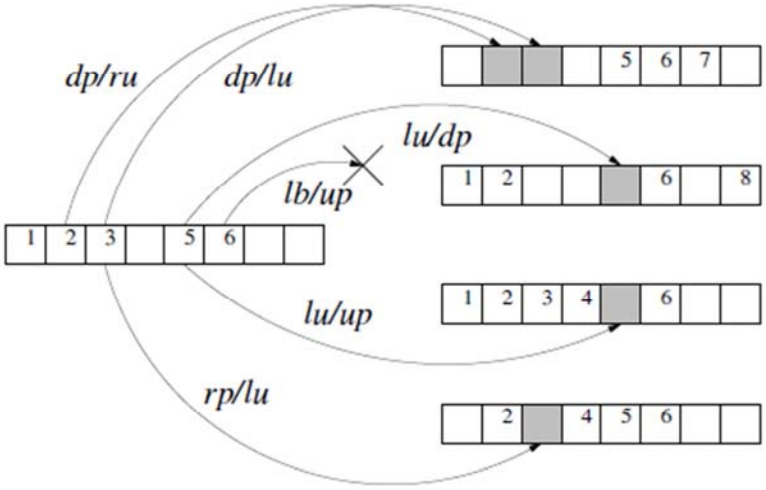
Epidemic Live Streaming [11] นำเสนอรูปแบบของการกระจายข้อมูลแบบผลักบนโอเวอร์เลย์ตาข่าย และดูว่ารูปแบบใดมีอัตราการแพร่ของข้อมูลสูงสุด (Optimal Diffusion Rate) และความหน่วงในการแพร่ข้อมูลน้อยที่สุด (Optimal Diffusion Delay) สำหรับการกระจายข้อมูลแบบผลักแต่ละครั้ง ผู้ส่งจะทำการเลือกผู้รับ และเลือกชิ้นข้อมูลที่จะผลักโดยการทดลองได้แบ่งรูปแบบการเลือกผู้รับและการเลือกชิ้นข้อมูล ดังนี้

การเลือกโหนดผู้รับ

1. Random Peer (Rp) เป็นการเลือกโหนดผู้รับแบบสุ่ม
2. Random Useful Peer (Up) เลือกสุ่มจากเซตของโหนดผู้รับที่ยังไม่มีชิ้นข้อมูลที่ผู้ส่งต้องการส่งให้
3. Most Deprived Peer (Dp) เลือกโหนดผู้รับที่มีความแตกต่างของจำนวนของ เซตชิ้นข้อมูลที่มีในผู้ส่ง แต่ไม่มีในผู้รับ มากที่สุด

การเลือกชิ้นข้อมูลที่จะส่งไปให้โหนดผู้รับ

1. Latest Blind Chunk (Lb) เลือกชิ้นข้อมูลล่าสุดที่มีในผู้ส่ง
2. Latest Useful Chunk (Lu) เลือกชิ้นข้อมูลล่าสุดที่มีในผู้ส่ง และไม่มีในผู้รับ
3. Random Useful Chunk (Ru) เลือกสุ่มชิ้นข้อมูล จากเซตของชิ้นข้อมูลที่มีในผู้ส่ง และไม่มีในผู้รับ



ภาพที่ 2.12 การเลือกโหนด และการเลือกชิ้นข้อมูลของโหนดผู้ส่ง (ซ้าย) ไปยังโหนดผู้รับ (ขวา)

โดยในการทดลอง การส่งแต่ละครั้งจะมีการจับคู่เลือกโหนดผู้รับและเลือกชิ้นข้อมูล และจากผลการทดลอง รูปแบบการส่งที่ให้อัตราการแพร่ข้อมูลมากที่สุด และมีความหน่วงในการแพร่ข้อมูลน้อยที่สุด คือการส่งข้อมูลแบบการผลักในรูปแบบของการจับคู่ระหว่าง Dp/Lu, Lu/Dp, Lu/Up และ Lb/Up เนื่องจากการเลือกแบบ Lu หรือแบบ Lb จะทำให้ผู้รับได้รับชิ้นข้อมูลใหม่ล่าสุด และจะช่วยลดชิ้นข้อมูลซ้ำ แต่สำหรับการเลือกแบบสุ่ม คือแบบ Ru การเลือกสุ่ม ผู้รับจะได้รับข้อมูลซ้ำเพิ่มขึ้น และความหน่วงในการแพร่ข้อมูลก็เพิ่มสูงขึ้น

Is there a future for mesh-based live video Streaming? [10] งานวิจัยนี้กล่าวว่า การออกแบบที่ทำให้มีการใช้แบนด์วิดท์อย่างเต็มที่ (High Bandwidth Utilization) จะนำมาซึ่งอัตราการแพร่ข้อมูลไลฟ์ตรีมที่มากขึ้นด้วย ดังนั้น งานวิจัยนี้จึงได้นำรูปแบบ Dp/Lu มาใช้ โดยผู้ส่งจะส่งโทเค็นของการดึง (Pull-Token) ไปยังผู้รับที่ถูกเลือกโดยใช้การเลือกแบบ Dp หลังจากนั้น ผู้รับจะส่งข้อความร้องขอมายังผู้ส่งเพื่อขอรับข้อมูล งานวิจัยนี้ทดลองเมื่อโหนดมีความจุการอัพโหลด (Upload Capacity) ที่ต่างกัน พบว่าการกำหนดให้แหล่งกระจายข้อมูล (Source) กระจายข้อมูลไปยังโหนดที่มีความจุการอัพโหลดที่มากก่อน จะทำให้ความหน่วงในการแพร่ข้อมูลลดลง และช่วยเพิ่มประสิทธิภาพของระบบ นอกจากนี้ยังได้ทำการศึกษาการจัดการการสร้างโอเวอร์เลย์ (Overlay Management) โดยหากอัตราการรับข้อมูลจากทุกเพื่อนบ้านรวมกันน้อยกว่าอัตราการเล่นข้อมูล จะทำการหาโหนดเพื่อนบ้านใหม่ (Rewiring the Overlay) เพื่อให้สามารถรับข้อมูลในอัตราที่สูงขึ้น

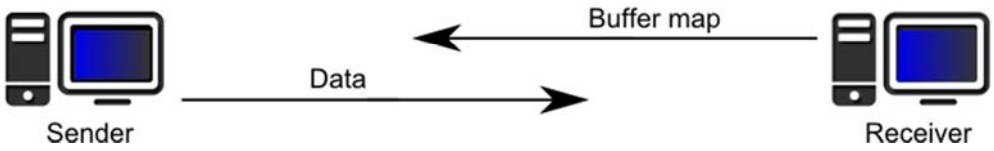
เมื่อพิจารณาถึงงานวิจัยที่กล่าวมา สามารถสรุปหลักทำงานของโปรโตคอลแต่ละแบบได้ตามตารางที่ 2.1

ตารางที่ 2.1 ตารางเปรียบเทียบหลักการทำงานของโปรโคตอลการกระจายข้อมูลไลฟ์สตรีมมิ่ง

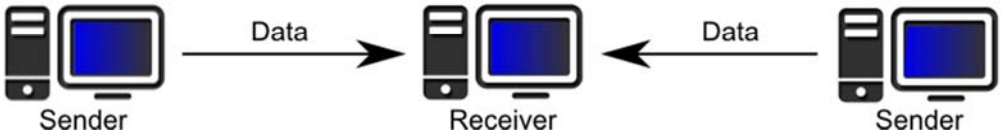
โปรโตคอล	CoolStream/ DONET	GridMedia	mTreebone	Epidemic live streaming	Is there a future for mesh-based?	Proposed
โอเวอร์เลย์ที่ใช้	โอเวอร์เลย์ตาข่าย	โอเวอร์เลย์ตาข่าย	โอเวอร์เลย์ต้นไม้ ร่วมกับ โอเวอร์เลย์ตาข่าย	โอเวอร์เลย์ตาข่าย	โอเวอร์เลย์ตาข่าย	โอเวอร์เลย์ตาข่าย
ลักษณะการกระจายชิ้นข้อมูล	กระจายชิ้นข้อมูลแบบดึง	กระจายชิ้นข้อมูลแบบผสม	กระจายชิ้นข้อมูลแบบผสม	กระจายชิ้นข้อมูลแบบผลัก	กระจายชิ้นข้อมูลแบบดึง	กระจายชิ้นข้อมูลแบบผสม
จำนวนชิ้นข้อมูลซ้ำ	ไม่มีข้อมูลซ้ำเกิดขึ้น เนื่องจากโหนดผู้รับจะส่งข้อความร้องขอ	จำนวนข้อมูลซ้ำเยอะมาก จากการสุ่มเลือกผู้รับ และข้อมูลที่ส่ง	ไม่มีข้อมูลซ้ำเกิดขึ้น เนื่องจากผลักข้อมูลลงมาตามกิ่งของโอเวอร์เลย์ต้นไม้ และดึงข้อมูลเมื่อเกิดการสูญหาย	ยังคงมีข้อมูลซ้ำเกิด เนื่องจากเป็นการผลักข้อมูลบนโอเวอร์เลย์ตาข่าย	ไม่มีข้อมูลซ้ำเกิดขึ้น เนื่องจากโหนดผู้รับจะส่งข้อความร้องขอ	การเกิดข้อมูลซ้ำลดลงมาก
ความเร็วการแพร่ข้อมูล	ไม่ Optimal	ไม่ Optimal	ไม่ Optimal	Optimal	Optimal	Optimal

บทที่ 3
การออกแบบโปรโตคอลกระจายข้อมูลไลฟ์สตรีมมิ่ง บนเครือข่ายเพียร์ทูเพียร์

การกระจายข้อมูลไลฟ์สตรีมมิ่งบนเครือข่ายเพียร์ทูเพียร์ ถูกออกแบบโดยคำนึงถึงปัจจัยสำคัญ 2 ประการดังนี้ 1) ความเร็วสำหรับการกระจายข้อมูล ซึ่งเป็นส่วนสำคัญที่ทำให้ข้อมูลไลฟ์สตรีมมิ่งกระจายไปยังทุกโหนดในระบบภายในระยะเวลาอันสั้น 2) การเกิดข้อมูลซ้ำซึ่งยิ่งข้อมูลซ้ำเกิดขึ้นน้อยเท่าไหร่ ยิ่งทำให้การสูญเสียแบนด์วิทด์ไปกับข้อมูลที่ไม่เป็นประโยชน์น้อยลงเท่านั้น ส่งผลทำให้ระบบมีการแบนด์วิทด์อย่างมีประสิทธิภาพ (Bandwidth Efficiency)



ภาพที่ 3.1 แสดงปัญหาความหน่วงการส่งแผนที่บัฟเฟอร์



ภาพที่ 3.2 แสดงปัญหาการส่งซ้ำระหว่าง 2 โหนดผู้ส่ง

3.1 หลักการทำงานของโปรโตคอลการกระจายข้อมูล

โปรโตคอลการกระจายข้อมูลในวิทยานิพนธ์ฉบับนี้ ได้ถูกเผยแพร่ในผลงานทางวิชาการ โดยใช้ชื่อว่า “A Reinforcement-based Push-Pull Approach for Peer-to-peer Live Streaming”

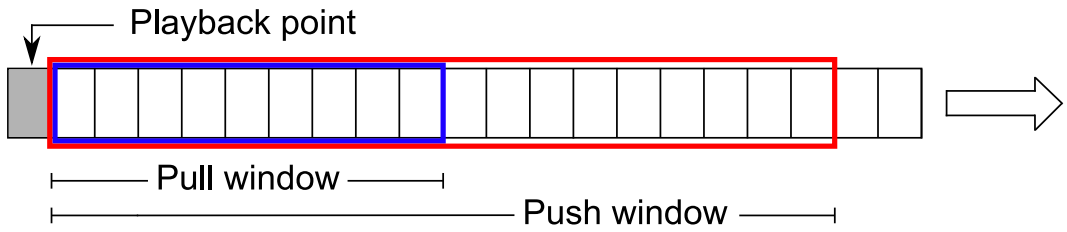
ได้แบ่งการทำงานในส่วนต่างๆ ของโปรโตคอลดังนี้

3.1.1 ส่วนการสร้างโทโพโลยีทางตรรกะ หรือโอเวอร์เลย์ (Logical Topology Construction or Overlay Network Construction Module)

ขั้นตอนการเข้าร่วมระบบของโหนด หากมีโหนดใดที่ต้องการเข้าร่วมในระบบเพียร์ทูเพียร์ไลฟ์สตรีมมิ่ง จะต้องทำการติดต่อกับจุดนับพบ (Rendezvous Point) ซึ่งในที่นี้อาจจะเป็นเครื่องเซิร์ฟเวอร์ หรือเครื่องที่ทำหน้าเก็บข้อมูลของโหนดที่อยู่ในระบบ (Tracker) ซึ่งจะทำหน้าที่เก็บรายชื่อ หรือหมายเลขของโหนดที่กำลังเชื่อมต่ออยู่ในระบบ ณ ขณะนั้นเมื่อแต่ละโหนดได้รับข้อความตอบกลับจากเซิร์ฟเวอร์ ที่ประกอบไปด้วยรายชื่อของโหนดทั้งหมดที่กำลังเชื่อมต่ออยู่จากนั้นโหนดที่ต้องการรับข้อมูลจะทำการสุ่มเลือก (Uniformly Random) โหนดเพื่อนบ้านจากรายชื่อทั้งหมดที่ได้รับมา และส่งข้อความสร้างการเชื่อมต่อ (Connect Message) ระหว่างโหนดตัวเอง กับโหนดเพื่อนบ้าน เพื่อสร้างการโครงสร้างการเชื่อมต่อ ในรูปแบบของโอเวอร์เลย์ตาข่าย

3.1.2 ส่วนหน้าต่างบัฟเฟอร์ (Buffer Window Module)

หน้าต่างบัฟเฟอร์ (Buffer Window) เป็นส่วนที่ใช้ในการสะสมชิ้นข้อมูลก่อนที่ชิ้นข้อมูลนั้นจะถูกเล่น สำหรับหน้าต่างบัฟเฟอร์ในงานวิจัยนี้จะแบ่งออกเป็นสองส่วน คือ ส่วนหน้าต่างการดึงชิ้นข้อมูล (Pull Window) และส่วนหน้าต่างการผลักชิ้นข้อมูล (Push Window) ดังภาพที่ 3.3



ภาพที่ 3.3 หน้าต่างของการผลักชิ้นข้อมูล และหน้าต่างของการดึงชิ้นข้อมูล

ในส่วนของหน้าต่างการดึงชิ้นข้อมูล หากมีชิ้นข้อมูลที่ยังไม่ได้รับเกิดขึ้นในหน้าต่างนี้ จะส่งข้อความร้องขอชิ้นข้อมูล (Request Message) ออกไปยังโหนดเพื่อนบ้านที่มีชิ้นข้อมูลที่ต้องการ (โดยดูจากแผนที่บัฟเฟอร์ที่ได้รับมา ซึ่งจะบอกว่ามีโหนดเพื่อนบ้านใดบ้างที่มีชิ้นข้อมูลที่ต้องการ) และสำหรับชิ้นข้อมูลที่ถูกผลัก จะอยู่ในส่วนของหน้าต่างของการผลักเท่านั้น โดยหน้าต่างบัฟเฟอร์จะเคลื่อนไปเรื่อยๆ ด้วยความเร็วเท่ากับความเร็วของการเล่นไฟล์

3.1.3 ส่วนการแลกเปลี่ยนแผนที่บัฟเฟอร์ (Buffer Map Exchange Module)

จุดประสงค์ของการแลกเปลี่ยนแผนที่บัฟเฟอร์ คือ ให้โหนดเพื่อนบ้านรับรู้ว่าโหนดตัวเองมีชิ้นข้อมูลใดอยู่บ้าง โดยแผนที่บัฟเฟอร์ที่ถูกส่งไปยังโหนดเพื่อนบ้านเป็นไปตามรูปแบบ ดังตารางที่ 3.1

ข้อมูล	รายละเอียด	ขนาดข้อมูล
Node ID	หมายเลขโหนด	4 bytes
Array of Chunk ID	หมายเลขชิ้นข้อมูล	4 bytes *Array size

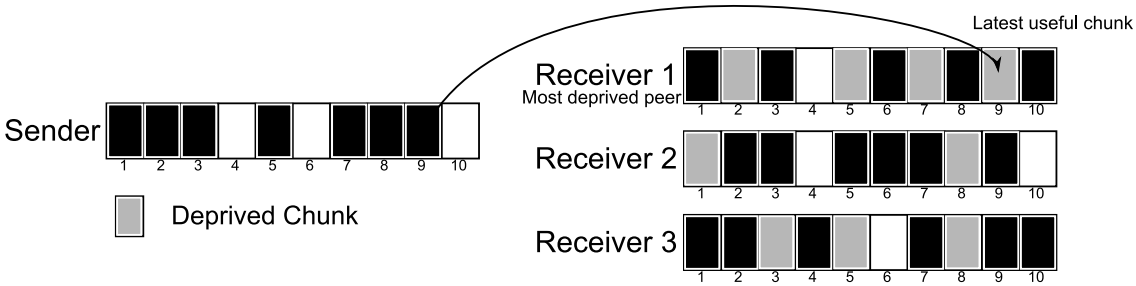
ตารางที่ 3.1 รูปแบบแผนที่บัฟเฟอร์

3.1.4 ส่วนการเลือกโหนด และชิ้นข้อมูลที่ต้องการส่ง (Peer and Chunk Selection Module)

การกระจายชิ้นข้อมูลแบบผลัก ชิ้นข้อมูลจะถูกกระจายโดยมีการเลือกโหนดผู้รับแบบ “Most Deprived Peer” หรือ “Dp” และเลือกชิ้นข้อมูลที่จะส่งแบบ “Latest Useful Chunk” หรือ “Lu” โดยโหนดผู้ส่งจะทำการตัดสินใจในครั้งการส่ง (Send Interval) นั้น

สำหรับขั้นตอนการผลักข้อมูลแต่ละครั้ง จะประกอบไปด้วย 2 ขั้นตอน คือ ขั้นตอนการเลือกโหนดผู้รับ และขั้นตอนการเลือกชิ้นข้อมูลที่ต้องการส่ง

1) การเลือกโหนดผู้รับ โดยโหนดผู้ส่งจะทำการเลือกโหนดผู้รับที่ต้องการส่งข้อมูลให้ โดยเลือกโหนดผู้รับแบบ “Most Deprived Peer” ตัวอย่าง จากภาพที่ 3.4 ช่องสีดำแสดงว่ามีชิ้นข้อมูลอยู่ในบัฟเฟอร์ ช่องสีขาวแสดงให้เห็นว่ายังไม่ได้รับชิ้นข้อมูล ดังนั้นการเลือกโหนดผู้รับรูปแบบ “Most Deprive Peer” จะเลือกโหนดผู้รับที่มีจำนวนชิ้นที่ขาดแคลน (Deprived Chunk) มากที่สุดซึ่งชิ้นข้อมูลที่ขาดแคลน คือ ชิ้นข้อมูลที่มีในโหนดผู้ส่ง แต่ไม่มีในโหนดผู้รับดังช่องที่แรงเงาสีเทา ในตัวอย่างภาพที่ 3.4



ภาพที่ 3.4 แสดงการเลือกโหนดผู้รับ และเลือกชิ้นข้อมูลที่จะส่ง

หมายเลขโหนดผู้รับ	ชิ้นที่มีในผู้ส่ง แต่ไม่มีในผู้รับ	จำนวน Deprived Chunk
1	2, 5, 7, 9	4
2	1, 8	2
3	3, 5, 8	3

ตารางที่ 3.2 อธิบายตัวอย่าง และแสดงจำนวนของ Deprived Chunk ของโหนด 1, 2, 3

ดังนั้นในการส่งครั้งนี้ โหนดผู้ส่งจะเลือกโหนดผู้รับหมายเลข 1 เป็นโหนดหมายเลข 1 เป็นโหนดผู้รับ ซึ่งเป็นไปตามสมการ (1)

$$Dp = \arg\max_{r} |C(s) \backslash C(r)| \ ; \ r \in neighbor(s) \qquad (1)$$

2) เลือกชิ้นข้อมูลที่จะส่งการเลือกชิ้นข้อมูลที่จะส่ง เกิดขึ้นหลังจากเลือกโหนดผู้รับได้แล้ว โดยโหนดผู้ส่งจะเลือกชิ้นข้อมูลที่จะส่งแบบ “Latest Useful Chunk” คือ ข้อมูลชิ้นสุดท้ายที่มีในโหนดผู้ส่ง และโหนดผู้รับที่ถูกเลือกยังไม่มีชิ้นข้อมูลนั้นในบัฟเฟอร์ของโหนดผู้รับนั้นซึ่งงานวิจัยก่อนหน้าได้กล่าวไว้ว่า การผลักข้อมูลในรูปแบบดังกล่าว จะทำให้ได้ความหน่วงในการกระจายข้อมูลน้อยสุด (Delay Optimal) และได้จำนวนชิ้นข้อมูลที่ได้รับต่อข้อมูลที่ส่งสูงสุด (Delivery Rate Optimal)

3.1.5 ส่วนการคำนวณความน่าจะเป็นในการผลักชิ้นข้อมูล (Probabilistic Calculation Module)

ส่วนการคำนวณความน่าในการผลักชิ้นข้อมูล จะเกิดขึ้นทุกครั้งหลังจากที่โหนดได้รับชิ้นข้อมูล โดยการคำนวณในส่วนนี้จะเกิดขึ้นที่ฝั่งของโหนดผู้รับ โหนดผู้รับจะคำนวณ

ความน่าจะเป็น (Probability) ใช้สัญลักษณ์ คือ P'_{ji} โดยจะมีค่าอยู่ระหว่าง 0 ถึง 1 คำนวณโดยดูจากจำนวนชิ้นข้อมูลที่ได้รับที่เป็นประโยชน์ คือ ชิ้นข้อมูลที่ไม่เคยได้รับมาก่อน (None Duplicate Chunk) ต่อจำนวนชิ้นข้อมูลที่เคยได้รับทั้งหมด (Total Received Chunk) จากโหนดผู้ส่งนั้น ตามสมการที่ (2)

$$P'_{ji} = \frac{Number\ of\ none\ duplicate\ chunks\ from\ peer\ j}{Number\ of\ total\ received\ chunk\ from\ peer\ j} \quad (2)$$

หลังจากกระบวนการคำนวณความน่าจะเป็นเสร็จสิ้น โหนดผู้รับจะส่งข้อความความน่าจะเป็น (Probability Message) กลับไปยังโหนดผู้ส่งนั้นโดยรูปแบบของข้อความความน่าจะเป็น ดังตารางที่ 3.3

ข้อมูล	รายละเอียด	ขนาดข้อมูล
Node ID	หมายเลขโหนด	4 bytes
Probability	ความน่าจะเป็นในการผลักชิ้นข้อมูล	8 bytes

ตารางที่ 3.3 รูปแบบข้อความความน่าจะเป็น

3.1.6 ส่วนการปรับความน่าจะเป็น (Adjust Probabilistic Push Module)

การปรับความน่าจะเป็นสำหรับการผลักข้อมูลจะเกิดขึ้นที่ฝั่งของโหนดผู้ส่ง โดยหลังจากที่โหนดผู้ส่งได้รับข้อความความน่าจะเป็นจากโหนดผู้รับ โหนดผู้ส่งจะปรับค่าความน่าจะเป็นสำหรับการผลักข้อมูลใหม่ โดยคำนวณตามสมการที่ (3)

$$P_{ji} = P_{ji}(1-\alpha) + P'_{ji}\alpha \ ; \ i \in neighbor\ j \quad (3)$$

P_{ji} คือ ค่าความน่าจะเป็นสำหรับการผลักชิ้นข้อมูลที่ต้องการปรับ, P'_{ji} คือ ค่าความน่าจะเป็นที่ได้รับ และ α คือ ขั้นในการปรับ (Step Size)

โดยค่า P_{ji} จะมีค่าระหว่าง 0 ถึง 1

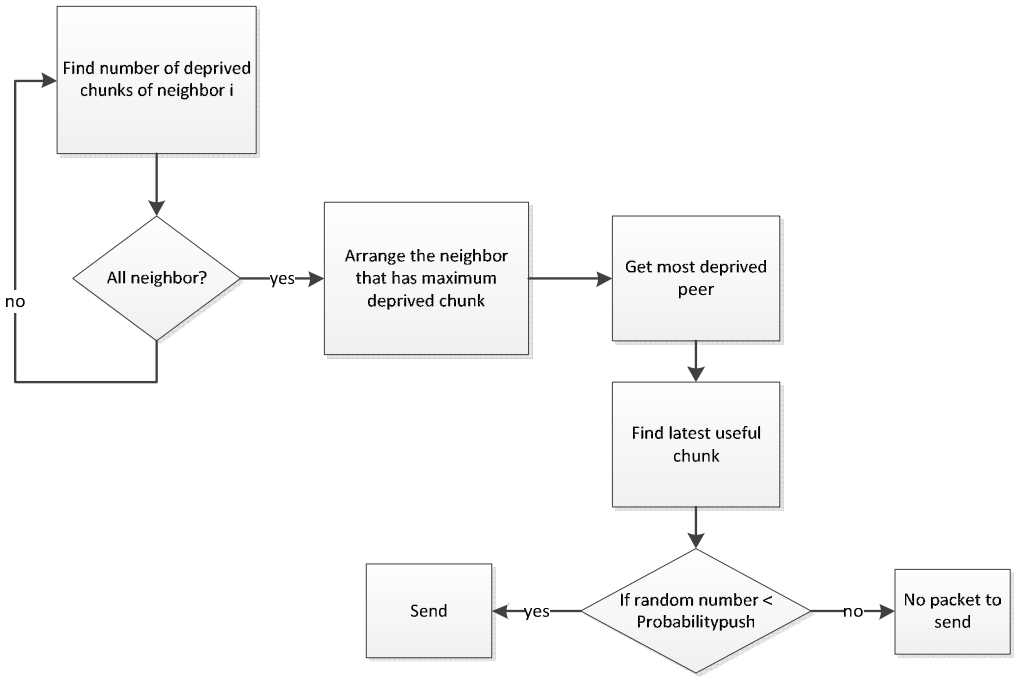
แต่ละโหนดจะเก็บข้อมูลโหนดเพื่อนบ้าน โดยตารางการเก็บเป็นไปในรูปแบบ ดังตารางที่ 3.4

ข้อมูล	รายละเอียด	ขนาดข้อมูล
Node ID	หมายเลขโหนด	4 bytes

Useful_Chunk	จำนวนชิ้นข้อมูลที่ได้รับ และเป็นชิ้นข้อมูลที่ไม่เคยได้รับมาก่อน	4 bytes
Total_Received_Chunk	จำนวนชิ้นข้อมูลที่ได้รับทั้งหมด	4 bytes
Probability_Push	ความน่าจะเป็นที่จะผลักชิ้นข้อมูลไปยังโหนดเพื่อนบ้าน	8 bytes

ตารางที่ 3.4 รูปแบบการเก็บข้อมูลของโหนดเพื่อนบ้าน

ดังนั้น สำหรับการส่งในแต่ละรอบของการส่ง (Send Interval) หลังจากที่เลือกโหนดผู้รับ และชิ้นข้อมูลที่ต้องการส่งแล้ว โหนดผู้ส่งจะสุ่มเลือกค่า (Uniformly Random) ในช่วงระหว่าง 0 ถึง 1 และเปรียบเทียบกับค่าความน่าจะเป็นในการผลักชิ้นข้อมูล โดยหากตัวเลขของการสุ่มอยู่ในช่วงระหว่าง 0 ถึงค่าความจะเป็นในการผลักชิ้นข้อมูล โหนดผู้ส่งจะตัดสินใจส่งชิ้นข้อมูลนั้นออกไป แต่หากตัวเลขของการสุ่มอยู่นอกเหนือช่วงดังกล่าว จะตัดสินใจไม่ส่งชิ้นข้อมูลนั้นผังการทำงานของโปรโตคอลการกระจายข้อมูล โดยเน้นการลดข้อมูลซ้ำที่ได้ออกแบบไว้ แสดงดังภาพที่ 3.5



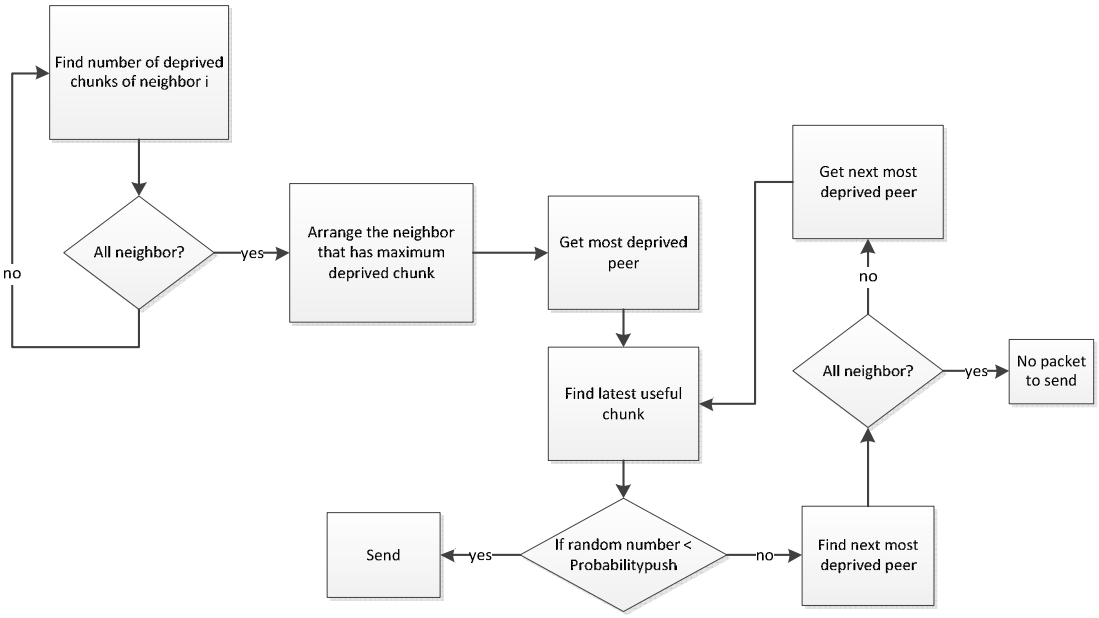
ภาพที่ 3.5 แสดงผังการทำงานของโปรโตคอลการกระจายข้อมูล

3.2 ออกแบบเพิ่มเติมโปรโตคอลการกระจายข้อมูล

จากการทดลองโปรโตคอลที่ออกแบบไว้ในตอนที่แล้ว พบว่าสามารถลดจำนวนชิ้นข้อมูลซ้ำได้ประมาณ 73% รายละเอียดผลการทดลองจะแสดงในบทที่ 4 แต่ผลกระทบที่

เกิดขึ้นคือ ความเร็วการกระจายข้อมูลลดลงด้วย เมื่อเทียบกับการกระจายข้อมูลแบบ Dp/Lu เหตุผลเนื่องจาก การที่โหนดผู้ส่งตัดสินไม่ส่งข้อมูลออกไปในรอบการของส่งครั้งนั้น ทำให้ชิ้นที่ได้รับมาใหม่บางชิ้นไม่ถูกกระจายไปยังโหนดผู้รับอื่นในรอบของการส่งนั้น ซึ่งความหน่วงที่เพิ่มมากขึ้นในการกระจายข้อมูลไปยังโหนดเพื่อนบ้านเกิดจากการรอเพื่อที่จะส่งข้อมูลชิ้นใหม่นั้นในรอบการส่งครั้งถัดไป

จากสิ่งที่เกิดขึ้น จึงได้ทดลองออกแบบส่วนการทำงานของโปรโตคอลเพิ่มเติมจากเดิม โดยมุ่งเน้นเพื่อเพิ่มความเร็วของการแพร่กระจายชิ้นข้อมูลให้กับทุกโหนดในระบบได้เร็วขึ้น จากเดิมหากโหนดผู้ส่งตัดสินใจไม่ส่งข้อมูล จะทำให้ชิ้นใหม่ที่เข้ามาไม่ถูกกระจายไปยังโหนดอื่น จึงทดลองเพิ่มเติมการตัดสินใจครั้งต่อไป คือ หากโหนดผู้ส่งตัดสินใจไม่ส่งข้อมูลให้โหนดผู้รับโหนดแรกซึ่งเป็นโหนดที่ขาดแคลนชิ้นข้อมูลมากที่สุด หรือเป็น Dp โหนดผู้ส่งจะทำการเลือกโหนดผู้รับต่อไป ซึ่งโหนดที่ถูกเลือกจะเป็นโหนดที่ขาดแคลนชิ้นข้อมูลในลำดับถัดไป หรือเป็น Next Dp (Next Most Deprived Peer) และทำการเลือกชิ้น โดยใช้การเลือกชิ้นที่จะส่งแบบ Lu การตัดสินใจจะเกิดขึ้นซ้ำจนกว่าจะครบทุกโหนดซึ่งเป็นโหนดเพื่อนบ้านผังการทำงานของโปรโตคอลการกระจายข้อมูล โดยเน้นความเร็วการกระจายข้อมูลที่ได้ออกแบบไว้แสดงดังภาพที่ 3.6



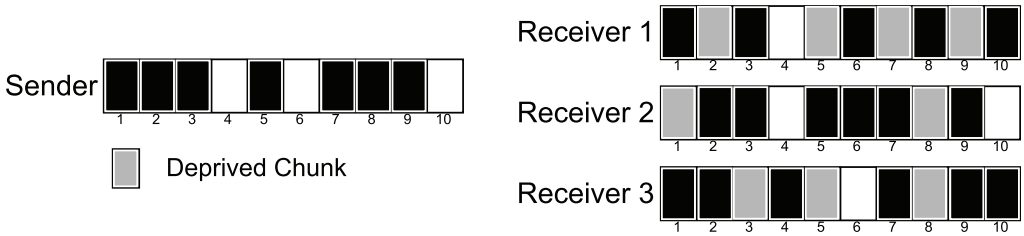
ภาพที่ 3.6 แสดงผังการทำงานของโปรโตคอลการกระจายข้อมูลเมื่อมีการวนเลือกผู้รับต่อไป

ตัวอย่างการเลือกโหนดผู้รับ และเลือกชิ้นข้อมูลที่จะส่งแบบวิธีวนซ้ำดังภาพที่ 3.7 เมื่อถึงรอบของการส่ง โหนดผู้ส่งซึ่งมีโหนดเพื่อนบ้าน 3 โหนด เพื่อนบ้านโหนดที่ 1,2 และ 3 มีจำนวนชิ้นข้อมูลที่ขาดแคลน คือ 4, 2 และ 3 ตามลำดับดังนั้นโหนดผู้ส่งจะเลือกผู้รับที่มีชิ้น

ที่ขาดแคลนมากที่สุดก่อน ซึ่งคือ โหนดผู้รับที่ 1 และเลือกชิ้นข้อมูลที่ 9 หลังจากนั้นจะส่งด้วยความน่าจะเป็นสำหรับการผลักชิ้นข้อมูลของโหนดผู้รับที่ 1

ซึ่งหากโหนดผู้ส่งตัดสินใจไม่ส่งชิ้นข้อมูลออกไป โหนดผู้ส่งจะทำกระบวนการเลือกโหนดผู้รับต่อไป ซึ่งโหนดผู้รับต่อไปจะเป็นโหนดที่มีชิ้นที่ขาดแคลนรองลงมา ซึ่งคือโหนดผู้รับที่ 3 และชิ้นข้อมูลที่จะถูกเลือกเพื่อส่ง คือชิ้นข้อมูลที่ 8 และตัดสินใจจะส่งหรือไม่ส่งด้วยความน่าจะเป็นสำหรับการผลักชิ้นข้อมูลของโหนดผู้รับที่ 3

กระบวนการนี้จะทำซ้ำไปเรื่อยๆ จนกว่าจะครบทุกโหนดเพื่อนบ้าน



ภาพที่ 3.7 แสดงการเลือกโหนดผู้รับ และเลือกชิ้นข้อมูลที่จะส่งแบบวิธีวนซ้ำ

บทที่ 4

ผลการทดลอง และวิเคราะห์ผลการทดลอง

ในบทนี้จะทำการวิเคราะห์สมรรถนะของการกระจายชิ้นข้อมูลจากโปรโตคอล Reinforcement-based โดยเปรียบเทียบกับโปรโตคอลPull-based และโปรโตคอล Dp/Lu เริ่มจากการกำหนดตัววัดสมรรถนะของระบบ (Performance Metrics) ในแต่ละด้าน คือ จำนวนชิ้นข้อมูลซ้ำที่ลดได้ ความเร็วในการแพร่กระจายข้อมูล และค่าใช้จ่าย

4.1 ตัววัดสมรรถนะของโปรโตคอล

ในงานวิจัยนี้ได้ทำการวัดสมรรถนะของโปรโตคอล (Performance Metrics) 3 ด้านดังต่อไปนี้

1. จำนวนชิ้นข้อมูลซ้ำที่ลดได้ (Number of Duplicate Chunks) โดยการ
2. ความเร็วในการแพร่กระจายข้อมูล (Speed of Data Dissemination)
3. ค่าใช้จ่าย (Overhead)

4.2 เครื่องมือในการวัดสมรรถนะของโปรโตคอล

การวัดสมรรถนะของโปรโตคอลใช้โปรแกรมจำลอง ซึ่งประกอบด้วยโปรแกรมดังนี้

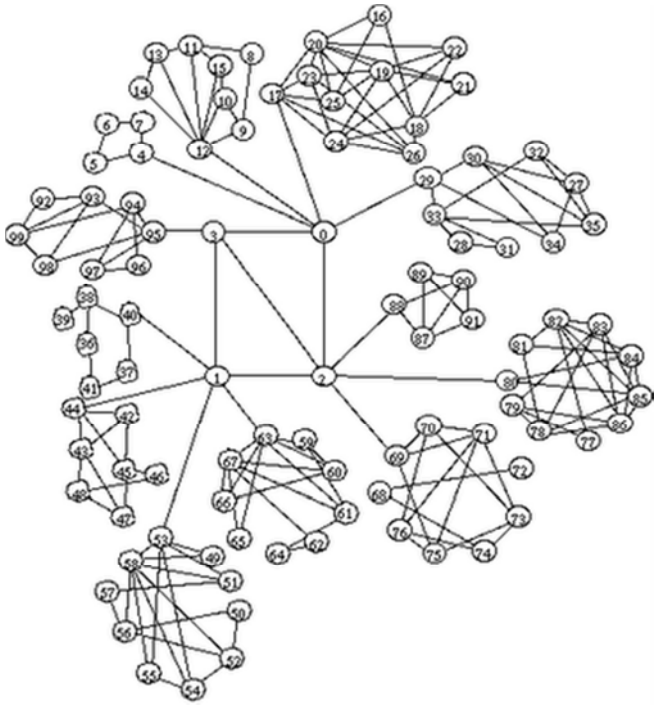
1. เครื่องมือสร้างโทโพโลยี GT-ITM (Georgia Tech Internetwork Topology Models) ซึ่งจะสร้างโทโพโลยีที่มีความเสมือนจริงของเครือข่าย สามารถกำหนดรูปแบบของเครือข่าย ลักษณะของเครือข่าย ขนาดของเครือข่าย ขนาดของช่องสัญญาณ และความหน่วงของการเชื่อมต่อได้

2. โปรแกรมจำลอง NS-2.34 (Network Simulation) เป็นโปรแกรมจำลองลักษณะการทำงานของเครือข่ายเสมือนจริง

3. เฟรมเวิร์ก NS2-appเป็นเฟรมเวิร์กที่ทำงานบนโปรแกรมจำลอง NS-2 ซึ่งช่วยให้การสร้างโปรแกรมสำหรับการจำลองโปรโตคอลในชั้นแอพพลิเคชั่นเลเยอร์ (Application Layer Simulation) สะดวกมากขึ้น

ในการทดลองเริ่มต้นด้วยการสร้างโทโพโลยีเครือข่ายเสมือนโดยใช้ GT-ITM โดยกำหนดค่าต่างๆ ตามที่ต้องการ จากนั้นจึงแปลงกราฟที่ได้ให้เป็นรูปแบบ tcl ซึ่งเป็นรูปแบบที่สามารถให้ได้บนโปรแกรมจำลอง NS-2.34 โดยใช้โปรแกรมแปลง sgb2ns (sgb2ns Conversion Program)

4.3 ผลการทดลอง และการเปรียบเทียบ



ภาพที่ 4.1 แสดงลักษณะโทโพโลยีที่ใช้ในการทดลอง

การทดลองจะใช้โทโพโลยีที่มีลักษณะใกล้เคียงกับเครือข่ายจริงในปัจจุบัน ซึ่งเรียกรูปแบบโทโพโลยีแบบนี้ว่า “Transit-stub Model” เพื่อให้ผลการทดลองมีความสมจริง และได้ผลการทดลองใกล้เคียงกับเมื่อนำไปใช้บนเครือข่ายจริง โดยมีการกำหมดค่าตัวแปรสำหรับการทดลองดังตารางที่ 4.1

ตัวแปรสำหรับการทดลอง	ค่าตัวแปร
จำนวนโหนดทั้งหมดบนเครือข่าย (Number of node in the topology)	650 โหนด
จำนวนโหนดบนโอเวอร์เลย์ (Number of node in the overlay network)	50, 100, 150, 200, 250 โหนด
จำนวนของชิ้นข้อมูล (Number of UDP data packet)	2000 ชิ้น
ขนาดของชิ้นข้อมูล (Chunk size)	1024ไบต์

ขนาดหน้าต่างของการดึง (Pull window size)	15 วินาที
ขนาดหน้าต่างของการผลัก (Push window size)	30 วินาที
อัตราการสตริมมิ่ง (Streaming rate)	300 กิโลบิตต่อวินาที
ความสามารถการอัพโหลดข้อมูล (Upload capacity of peer)	1.8 เมกะบิตต่อวินาที
ขนาดการปรับ (Step size)	0.2

ตารางที่ 4.1 แสดงค่าตัวแปรที่ใช้ในการทดลอง

สำหรับการสร้างโอเวอร์เลย์ตาข่าย ทุกโหนดบนโอเวอร์เลย์จะเลือกโดยการสุ่มแบบเอกรูป (Uniformly Random) โดยแต่ละโหนดจะมีโหนดเพื่อน 10 โหนดโดยเฉลี่ยการทดลองได้สมมุติให้ไม่มีข้อมูลหายในระบบเกิดขึ้นในระบบ

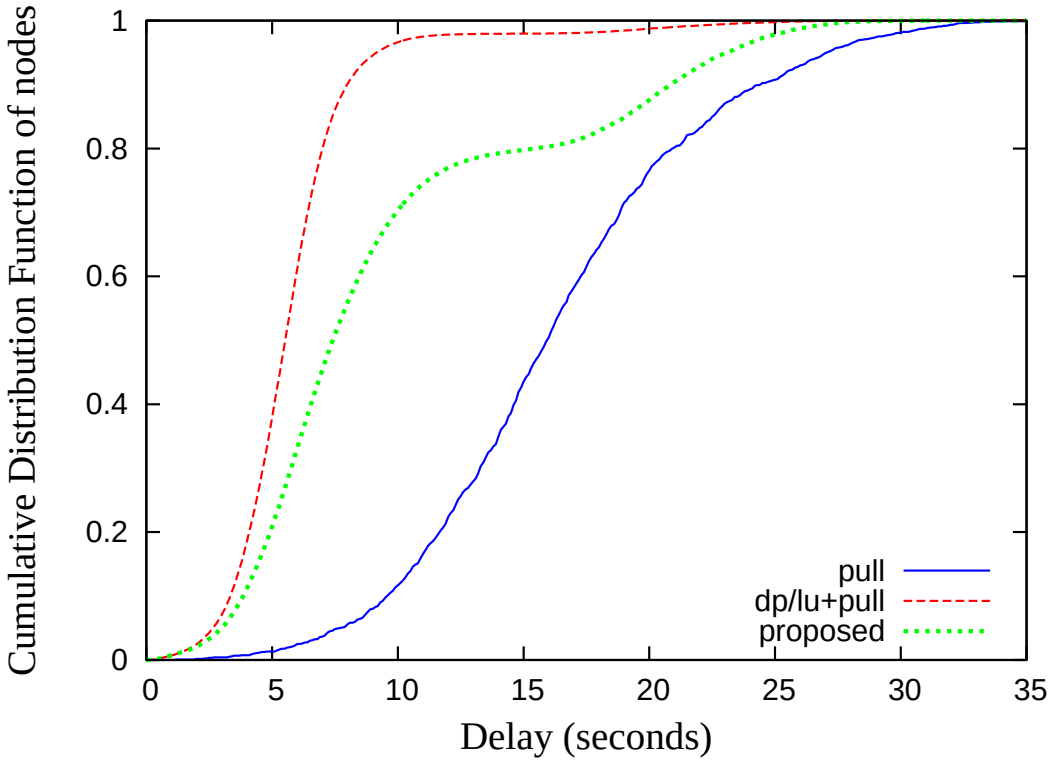
โปรโตคอลในระดับชั้นการส่ง (Transport Layer) ที่ใช้ในการทดลอง ใช้โปรโตคอล UDP (User Datagram Protocol) และ TCP (Transmission Control Protocol) โดยโปรโตคอล TCP จะใช้กับข้อมูลที่ต้องการความน่าเชื่อถือสูง อาทิ ข้อความสร้างโอเวอร์เลย์ และสำหรับโปรโตคอล UDP จะใช้กับข้อมูลทั่วไป เช่น ข้อมูลของไฟล์วิดีโอ ข้อมูลแผนที่บัฟเฟอร์ ข้อมูลการปรับความน่าจะเป็น

ลักษณะการส่งชิ้นข้อมูลจากเครื่องเซิร์ฟเวอร์ พฤติกรรมการส่งข้อมูลที่เครื่องเซิร์ฟเวอร์จะแตกต่างจากโหนดทั่วๆ ไป กล่าวคือ เครื่องเซิร์ฟเวอร์จะสุ่มเลือกโหนดผู้รับจากโหนดที่อยู่ในระบบทั้งหมด โดยไม่สนใจการอัลกอรึทึมการเลือกโหนด Dp หรืออัลกอรึทึมการเลือกชิ้น Lu เครื่องเซิร์ฟเวอร์จะส่งข้อมูลใหม่ที่สุดออกไปยังโหนดผู้รับเสมอ

และเนื่องจากเครื่องเซิร์ฟเวอร์ไม่มีอัลกอรึทึมการเลือกโหนด และการเลือกชิ้น ดังนั้นโหนดในระบบทุกโหนดจะไม่ส่งแผนที่บัฟเฟอร์ให้กับเครื่องเซิร์ฟเวอร์ จึงกล่าวได้ว่าเซิร์ฟเวอร์ในระบบนี้มีความสามารถในการรองรับการขยายตัวของเครือข่าย (Scalable)

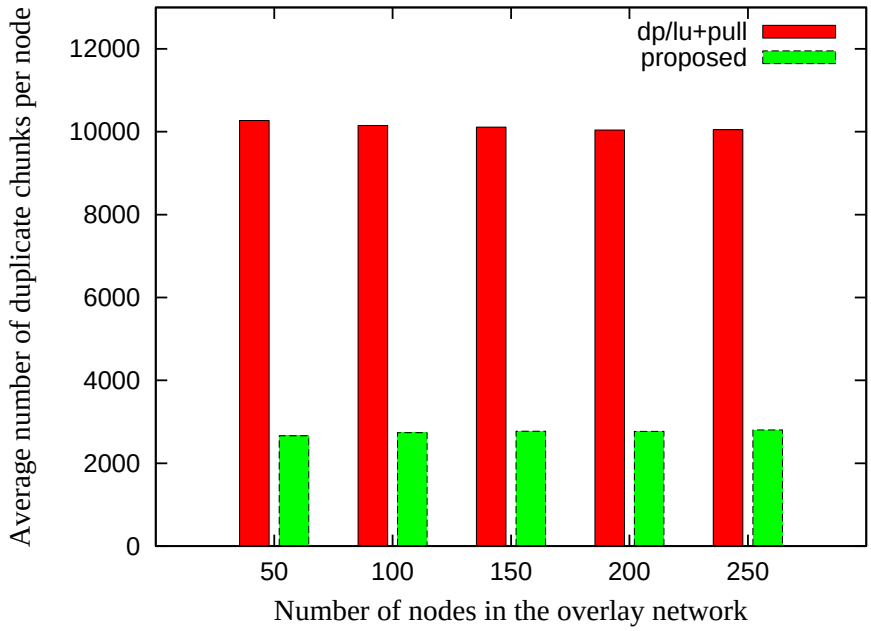
ผู้วิจัยได้ทำการทดลอง และเปรียบเทียบประสิทธิภาพการทำงานของโปรโตคอล 3 แบบ คือ

1. Pull เป็นการกระจายข้อมูลแบบดึงในลักษณะ Request-Response
2. Dp/Lu+Pull เป็นการกระจายข้อมูลแบบผลัก ร่วมกับการดึงข้อมูล
3. Reinforcement-based เป็นวิธีที่นำเสนอ

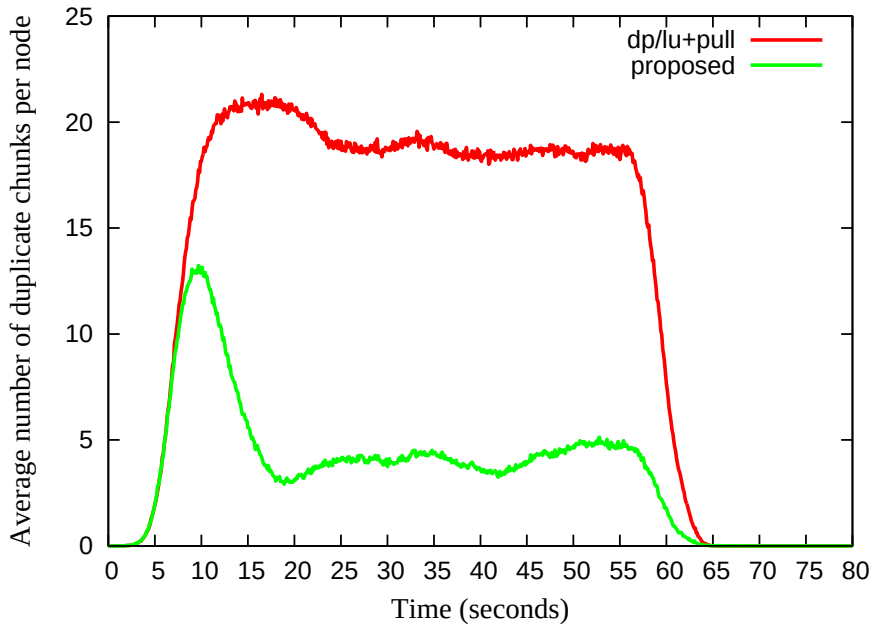


ภาพที่ 4.2 กราฟแสดงจำนวนโหนดที่ได้รับชิ้นข้อมูลต่อความหน่วงในการกระจาย

จากภาพที่ 4.2 แสดงความหน่วงในการกระจายข้อมูล โดยทำการเฉลี่ยจากทุกชิ้นข้อมูล โดยความหน่วงในแกน Y หาจากค่าเวลาความแตกต่างหลังจากที่เซิร์ฟเวอร์สร้างข้อมูลนั้น จะเห็นว่าความหน่วงสำหรับการกระจายข้อมูลแบบ Reinforcement-based ยังคงเร็วกว่าการใช้แบบดึง แต่ช้ากว่าแบบ Dp/Lu+Pull เนื่องจากการกระจายข้อมูลแบบ Reinforcement-based มีการตัดสินใจไม่ส่งบางข้อมูลที่เป็นข้อมูลที่ใหม่ ดังนั้นจึงทำให้การกระจายข้อมูลไปได้ช้ากว่า



ภาพที่ 4.3 กราฟแสดงจำนวนข้อมูลซ้ำเปรียบเทียบระหว่าง Dp/Lu+Pull กับวิธี Reinforcement-based

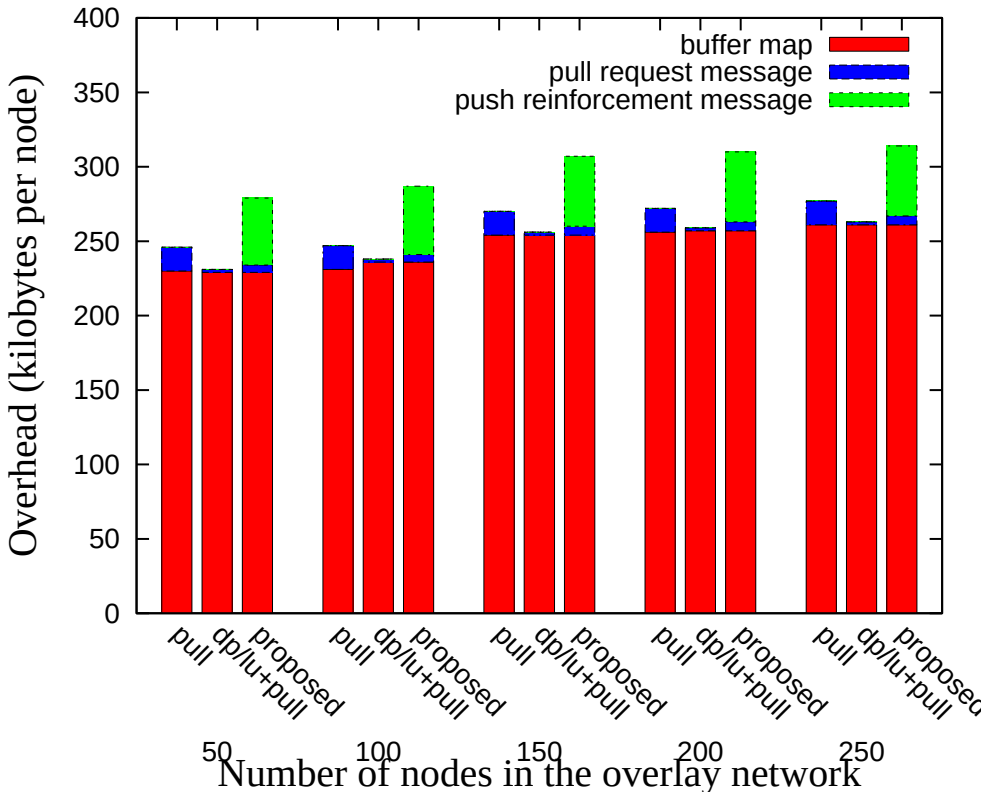


ภาพที่ 4.4 แสดงจำนวนข้อมูลซ้ำที่เกิดขึ้นที่เวลาต่างๆ

จากภาพที่ 4.3 จะเห็นว่าโปรโตคอล Reinforcement-based สามารถลดข้อมูลซ้ำในระบบได้ โดยเมื่อมีจำนวนโหนดบนโอเวอร์เลย์มีจำนวนมากขึ้น แต่โปรโตคอลนี้ก็ยังสามารถลดจำนวนข้อมูลซ้ำที่เกิดขึ้นได้ ซึ่งแสดงให้เห็นว่ายังคงให้ประสิทธิภาพที่ดี แม้ว่าจำนวนโหนดจะเพิ่มขึ้นก็ตาม จะเห็นว่า แม้จำนวนโหนดในโอเวอร์เลย์เพิ่มขึ้น แต่ประสิทธิภาพ

การลดลงของข้อมูลซ้ำยังคงเหมือนเดิม คือ ข้อมูลซ้ำสามารถลดได้ โดยไม่แปรผันตามจำนวนโหนดที่เพิ่มขึ้น ทั้งนี้ เนื่องจาก แต่ละโหนดมีการส่งข้อมูลระหว่างโหนดตัวเอง และโหนดเพื่อนบ้านเท่านั้นไม่ว่าจะเป็น ข้อมูลของไฟล์วิดีโอ และโอเวอร์เฮดต่างๆ และจากเหตุผลนี้ ทำให้เมื่อจำนวนโหนดในโอเวอร์เลย์ที่เพิ่มขึ้น ไม่มีผลทำให้ประสิทธิภาพของการลดข้อมูลซ้ำลดลง กล่าวคือ ตัวแปรที่จะมีผลต่อการลดลงของข้อมูลซ้ำ คือ จำนวนโหนดเพื่อนบ้าน ซึ่งการทดลองนี้ ได้กำหนดให้แต่ละโหนดมีจำนวนสมาชิกของโหนดเพื่อนบ้านเฉลี่ยประมาณ 10 โหนด

จากภาพที่ 4.4 แสดงให้เห็นถึงจำนวนข้อมูลซ้ำที่เกิดขึ้นเมื่อเวลาผ่านไป ที่จำนวนโหนดเท่ากับ 200 โหนด จากกราฟแสดงให้เห็นว่า จำนวนข้อมูลซ้ำลดลงอย่างมาก ซึ่งในช่วงแรกที่เริ่มต้น ข้อมูลซ้ำจะเกิดขึ้นค่อนข้างมาก เนื่องจากมีการตัดสินใจส่งทุกครั้ง แต่เมื่อเวลาผ่านไป และได้รับข้อความความน่าจะเป็นในการส่ง การเกิดข้อมูลซ้ำจึงลดลง โดยขณะที่ทำการทดสอบกราฟอาจมีการแกว่งของจำนวนการเกิดข้อมูลซ้ำ เนื่องจากค่าความน่าจะเป็นจะมีการปรับตัวอยู่ตลอดเวลา ซึ่งที่จำนวนโหนดอื่นๆ อาทิ 50, 100, 150, 200 และ 250 ได้ผลออกมาในลักษณะกราฟที่คล้ายกัน



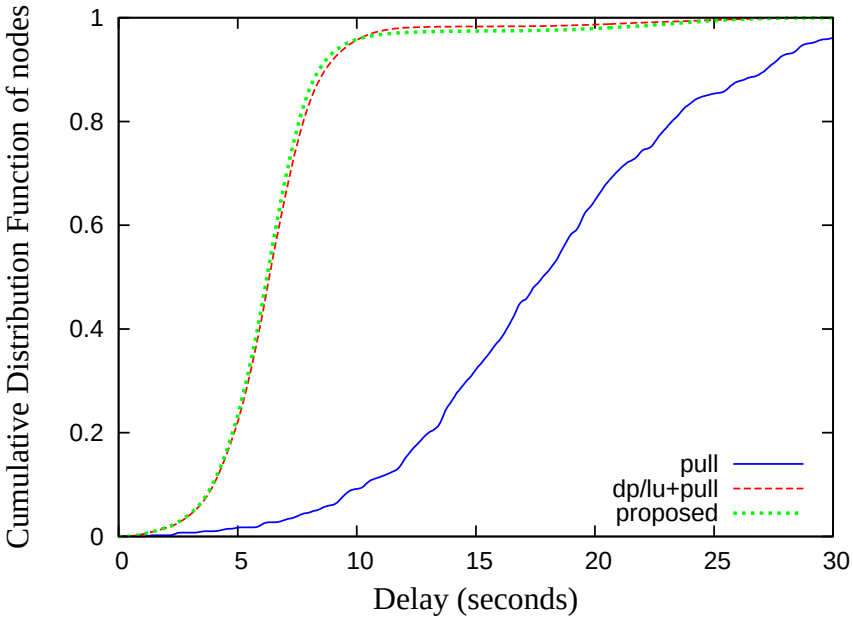
ภาพที่ 4.5 แสดงโอเวอร์เฮดเปรียบเทียบระหว่าง Pull, Dp/Lu+Pull และวิธี Reinforcement-based

จากภาพที่ 4.5 แสดงให้เห็นโอเวอร์เฮดที่เกิดขึ้นในหน่วยของกิโลไบต์ โดยทำการเปรียบเทียบโปรโตคอล Pull, Dp/Lu+Pull และแบบ Reinforcement-based ในจำนวนโหนดบนโอเวอร์เลย์ที่สูงขึ้น จากกราฟจะเห็นว่า จำนวนโอเวอร์เฮดที่เกิดขึ้นแผนที่บัฟเฟอร์มีจำนวนที่เยอะมาก ซึ่งจากทั้งโปรโตคอลทั้งสามแบบจะมีโอเวอร์เฮดในส่วนนี้ในจำนวนที่เท่าๆ กัน ซึ่งโปรโตคอลในแบบการกระจายข้อมูลแบบผลักจะมีโอเวอร์เฮดสำหรับการดึงข้อมูลเมื่อยังไม่ได้รับข้อมูลที่กำลังจะเล่น และโปรโตคอล Reinforcement-based จะมีโอเวอร์เฮดของการอัพเดตความน่าจะเป็นสำหรับการส่งเพิ่มขึ้นมา ซึ่งหากเทียบโอเวอร์เฮดในส่วนนี้ที่เพิ่มขึ้นกับจำนวนข้อมูลซ้ำที่โปรโตคอลนี้สามารถลดลงได้ โปรโตคอลนี้สามารถลดการใช้แบนด์วิทโดยรวม (Bandwidth Consumption) ได้มากกว่า

โหนดได้รับข้อมูลซ้ำ สาเหตุหนึ่งเนื่องมาจาก เหตุผลของความหน่วงในการส่งแผนที่บัฟเฟอร์ ซึ่งทำให้โหนดเพื่อนบ้านไม่รู้ว่า โหนดผู้รับมีชิ้นใดอยู่บ้าง ณ เวลาขณะนั้น จึงทำให้เกิดการตัดสินใจที่ผิดพลาด โดยการส่งชิ้นข้อมูลเดิม ที่โหนดผู้รับได้รับแล้ว (ซึ่งโหนดผู้รับได้รับชิ้นข้อมูลนี้แล้ว แต่แผนที่บัฟเฟอร์ยังส่งไปไม่ถึงโหนดเพื่อบ้าน) จึงทำให้เกิดข้อมูลซ้ำที่โหนดผู้รับขึ้น ดังนั้น ตัวแปรที่สำคัญที่ทำให้การตัดสินใจผิดพลาด คือ ความหน่วงในเครือข่าย ซึ่งหากในเครือข่ายมีความหน่วงมาก การรับรู้ว่าโหนดผู้รับมีชิ้นใดอยู่บ้างก็จะล่าช้าไปด้วย ซึ่งอาจทำให้การตัดสินใจส่งชิ้นข้อมูลทำได้ไม่ดี แต่ถ้าหากในเครือข่ายมีความหน่วงน้อย จะทำให้การตัดสินใจการโหนดผู้ส่งดีขึ้น ซึ่งหมายถึงการส่งข้อมูลซ้ำในจำนวนที่ลดลง

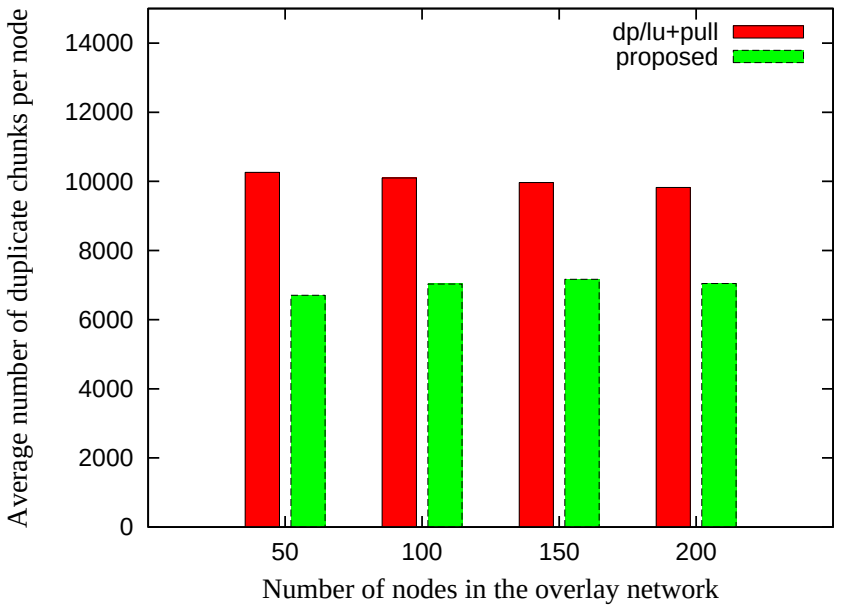
จากการทดลองข้างต้น ให้การเร็วการกระจายข้อมูลที่ไม่ดี เนื่องจากมีการตัดสินใจไม่ส่งข้อมูลเพื่อลดการซ้ำซ้อนข้อมูล ซึ่งกระทบต่อการความเร็วการแพร่กระจายชิ้นข้อมูล เนื่องจากบางชิ้นที่เป็นชิ้นล่าสุดที่ไม่ถูกส่งออกไป

ดังนั้นจึงทำการทดสอบเพิ่มเติมตามวิธีที่ได้ออกแบบมา ดังที่กล่าวในบทที่ 3 ซึ่งทดสอบโดยมีการเลือกโหนดผู้รับ และการเลือกชิ้นข้อมูลแบบวิธีวนซ้ำ ซึ่งวิธีนี้จะมีการหาโหนดที่เป็น “Next Most Deprived Peer” ที่จะเป็นโหนดผู้รับต่อไป ได้ผลทดลองดังภาพที่ 4.6



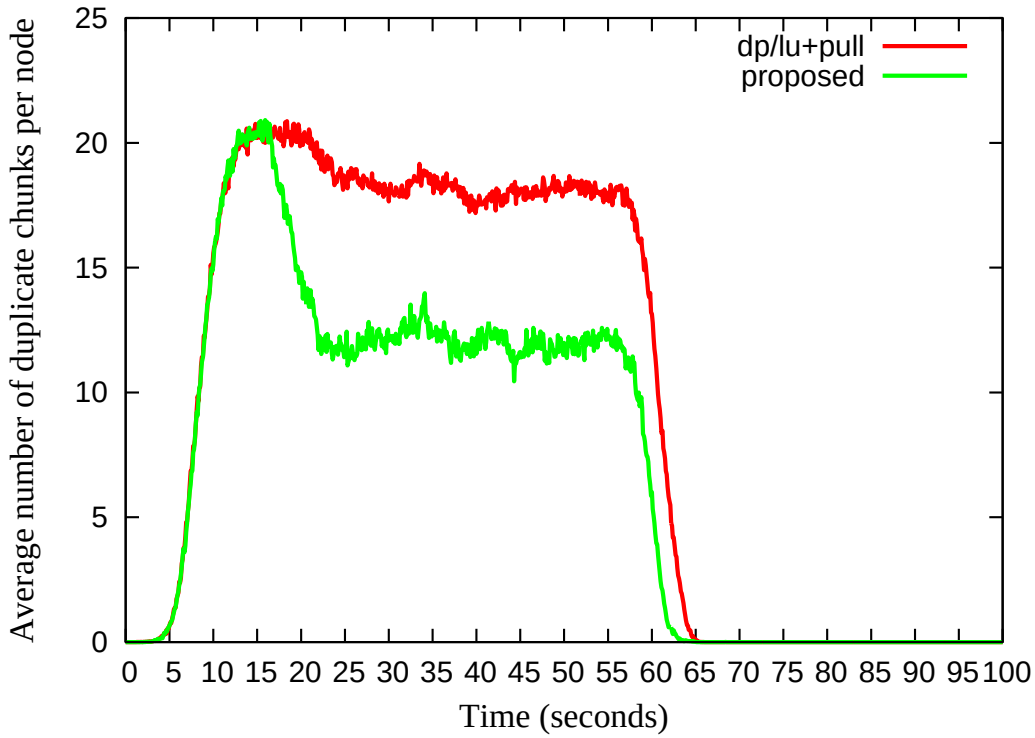
ภาพที่ 4.6 กราฟแสดงจำนวนโหนดที่ได้รับชิ้นข้อมูลต่อความหน่วงในการกระจายข้อมูลโดยการวนโหนดผู้รับซ้ำ

จากภาพที่ 4.6 จะเห็นว่าความเร็วของการแพร่กระจายชิ้นข้อมูลเพิ่มขึ้นมาก ซึ่งเทียบเท่ากับความเร็วการแพร่กระจายข้อมูลของโปรโตคอล Dp/Lu+Pull ซึ่งแสดงให้เห็นว่าการวนโหนดผู้รับซ้ำเพื่อพยายามส่งข้อมูลล่าสุดออกไป แทนที่จะไม่ทำการส่งชิ้นนั้นในการส่งครั้งนั้น ให้ผลการทดลองในเรื่องความเร็วการแพร่ข้อมูลที่ดียิ่งขึ้น



ภาพที่ 4.7 กราฟแสดงจำนวนข้อมูลซ้ำเปรียบเทียบระหว่าง Dp/Lu+Pull กับวิธี Reinforcement-based แบบวนโหนดผู้รับซ้ำ

จากภาพที่ 4.7 จำนวนข้อมูลซ้ำของโปรโคตอลที่นำเสนอยังคงลดลงมากกว่าโปรโตคอล Dp/Lu+Pull แต่ถ้าหากเปรียบเทียบกับวิธี Reinforcement-based แบบไม่วนซ้ำ โปรโตคอลแบบวนซ้ำทำให้เกิดข้อมูลซ้ำเพิ่มขึ้น เนื่องจากการวนโหนดผู้รับซ้ำทำให้เพิ่มความน่าจะเป็นที่ทำให้โหนดผู้รับได้รับข้อมูลซ้ำด้วย



ภาพที่ 4.8 แสดงจำนวนข้อมูลซ้ำที่เกิดขึ้นที่เวลาต่างๆ

จากภาพที่ 4.8 แสดงข้อมูลซ้ำที่ ณ เวลาต่างๆ โดยวัดที่ 200 โหนด จากผลการทดลองโปรโตคอล Reinforcement-based แบบวนซ้ำ เริ่มแรกการเกิดจำนวนข้อมูลซ้ำใกล้เคียงกับโปรโตคอล Dp/Lu+Pull แต่เมื่อผ่านไประยะเวลาหนึ่งโปรโตคอล Reinforcement-based แบบวนซ้ำเริ่มมีการปรับตัวลดการส่งลงตามความน่าจะเป็นที่จะเกิดข้อมูลซ้ำ จึงทำให้การเกิดข้อมูลซ้ำลดลง และช่วงวินาทีที่ 22 จนสิ้นสุดระยะการทดสอบ กราฟจะมีการแกว่ง (Swing) เนื่องจากมีการปรับความน่าจะเป็นสำหรับการส่งตลอด

บทที่ 5

สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ

5.1 สรุปผลการวิจัย

งานวิจัยนี้ได้นำเสนอ การกระจายข้อมูลไลฟ์สตรีมมิ่งบนเครือข่ายเพียร์ทูเพียร์ โดยเริ่มต้นจากการศึกษางานวิจัยทางด้านการกระจายข้อมูลไลฟ์สตรีมมิ่งในรูปแบบต่างๆ ซึ่งมีผลงานทางวิชาการตีพิมพ์ในวารสาร และรายงานการประชุมที่มีชื่อเสียงทั่วโลกอย่างต่อเนื่อง และผู้วิจัยได้ออกแบบโปรโตคอล โดยคำนึงถึงการนำไปใช้งานบนเครือข่ายจริง และได้นำเอาการออกแบบดังกล่าวไปทำให้เกิดผล รวมทั้งทดลองบนเครือข่ายที่มีความเสมือนจริง บนระบบปฏิบัติการ Linux Mint [16]

ผู้วิจัยได้นำเสนอการกระจายข้อมูลโดยลดการเกิดข้อมูลซ้ำ ซึ่งสามารถลดการเกิดข้อมูลซ้ำในระบบได้อย่างมีประสิทธิภาพ ไม่ทำให้คุณภาพของวิดีโอลดลง โดยทุกโหนดสามารถได้รับชิ้นข้อมูลทันเวลาก่อนการเล่นชิ้นนั้นโดยผลที่ได้รับจากการลดชิ้นข้อมูลซ้ำ คือ ช่วยลดการสูญเสียช่องสัญญาณที่ไม่เกิดประโยชน์

ซึ่งการลดการเกิดข้อมูลซ้ำมีผลทำให้ความเร็วการแพร่ข้อมูลลดลง ดังนั้นการเพิ่มการตัดสินใจส่งครั้งต่อไปช่วยทำให้ความเร็วสำหรับการแพร่เพิ่มขึ้น และจำนวนซ้ำที่เกิดขึ้นยังคงมีน้อยกว่าการใช้โปรโตคอลการกระจายข้อมูลแบบ Dp/Lu+Pull

โปรโตคอลดังกล่าว ผู้วิจัยได้แสดงให้เห็น และเปรียบเทียบผลการทดลองกับโปรโตคอลในงานวิจัยก่อน โดยสามารถลดการเกิดข้อมูลซ้ำได้โดยไม่ทำให้คุณภาพของวิดีโอลดลง ซึ่งเหมาะสมกับการกระจายข้อมูลบนเครือข่าย การลดลงของข้อมูลซ้ำที่เกิดขึ้น ทำให้ช่องสัญญาณ หรือแบนวิดท์ที่ไม่ถูกใช้ไปกับการส่งข้อมูลซ้ำ สามารถนำไปใช้ประโยชน์อย่างอื่นได้ อาทิ สามารถใช้ส่งข้อมูลไลฟ์สตรีมไฟล์อื่นๆ ได้ หรือสามารถนำมาใช้ส่งข้อมูลเพื่อเพิ่มคุณภาพของไฟล์

5.2 ข้อจำกัด

การตัดสินใจเพื่อกระจายข้อมูลนั้นขึ้นอยู่กับข้อความแผนที่บัฟเฟอร์ที่โหนดเพื่อนบ้านได้อัพเดตมาให้ ดังนั้นการรับรู้อัพเดตช้าหรือเร็ว ซึ่งขึ้นอยู่กับความหน่วงเกิดขึ้นในเครือข่าย ซึ่งปัจจัยนี้ทำให้ความเร็วการกระจายข้อมูลมีประสิทธิภาพที่ลดลงด้วย

5.3 ข้อเสนอแนะ

โปรโตคอลการกระจายข้อมูลไลฟ์ตรีมมิ่งบนเครือข่ายเพียร์ทูเพียร์ทำให้เกิดโอเวอร์เฮดที่ค่อนข้างมาก เพราะมีข้อความแผนที่บัฟเฟอร์บอกโหนดเพื่อนบ้านทุกครั้งที่ได้รับข้อมูลใหม่ ซึ่งอาจลดโอเวอร์เฮดด้วยการลดจำนวนครั้งในการบอกสถานะบัฟเฟอร์ของตัวเองให้กับโหนดเพื่อนบ้านคือไม่จำเป็นต้องบอกทุกครั้งที่ได้รับชิ้นข้อมูลใหม่

เนื่องจากการกระจายข้อมูลให้กับโหนดอื่นๆ จำเป็นต้องให้โหนดทุกโหนดยินยอมอัพโหลดข้อมูลที่ตัวเองมีให้กับโหนดอื่นๆ ที่อยู่ในเครือข่ายด้วย เพื่อให้การกระจายข้อมูลเป็นไปอย่างมีประสิทธิภาพ ดังนั้น ประเด็นของการโจมตีจากโหนดไม่ประสงค์ดี หรือโหนดที่ไม่ยอมอัพโหลดข้อมูลให้กับโหนดอื่นยังคงเป็นประเด็นที่น่าสนใจ ซึ่งหากมีโหนดประเภทนี้ในเครือข่ายมาก ก็จะส่งผลถึงคุณภาพของข้อมูล และประสิทธิภาพการแพร่กระจายข้อมูลด้วย

นอกจากนั้น หากมีการทดสอบโปรโตคอลที่นำเสนอบนเครื่องคอมพิวเตอร์จริงและบนเครือข่ายจริง ก็จะทำให้ผลการทดลองมีความสมจริงมากยิ่งขึ้น

รายการอ้างอิง

[1] Yang-hua, C., Sanjay G., R, and Hui, Z. A case for end system multicast. In Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems (June 2000)

[2] JumpTV. Available from : http://www.jumptv.com

[3] PPLive. Available from : http://www.pplive.com

[4] Sopcast. Available from : http://www.sopcast.com

[5] Suman, B., Bobby, B., and Christopher, K. Scalable application layer multicast, In Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications (August 2002)

[6] Miguel, C., Peter, D., Anne-Marie, K., Animesh, N., Antony, R., and Atul, S., SplitStream: High-bandwidth multicast in cooperative environments. In Proceedings of the nineteenth ACM symposium on Operating systems principles (October 2003)

[7] Vidhyashankar, V., Paul, F., and John, C. Chunkyspread: Multi-tree unstructured peer-to-peer multicast. In Proceedings of the international conference on Peer-to-peer systems (2006)

[8] Xinyan, Z., JC, L., Bo, L., and Tak-Shing Peter, Y. CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming. In Proceedings of IEEE INFOCOM (March 2005)

[9] Meng, Z., Jian-Guang, L., Li, Z., and Shi-Qiang, Y. A peer-to-peer network for live media streaming using a push-pull approach. In Proceedings of the ACM international conference on Multimedia (November 2005)

[10] Fabio, P., and Laurent, M. Is there a future for mesh-based live video streaming?. In Proceedings of the 2008 Eighth International Conference on Peer-to-Peer Computing (September 2008)

[11] Thomas, B., Laurent, M., Fabien, M., Diego, P., and Andrew, T. Epidemic live streaming: Optimal performance trade-offs. In Proceedings of the 2008 ACM SIGMETRICS international conference on Measurement and modeling of computer systems (June 2008)

[12] Feng, W., Yongqiang, X., and Jiangchuan, L. mTreebone: A Collaborative Tree-Mesh Overlay Network for Multicast Video Streaming. Journal IEEE transactions on parallel and distributed systems, IEEE Journal 21,3 (March 2010)

[13] Meng, Z., Qian, Z., Lifeng, S., and Shiqiang, Y. Understanding the Power of Pull-Based Streaming Protocol: Can We Do Better?. Journal selected areas in communications, IEEE Journal25,9 (December 2007)

[14] NS-2. Available from : http://www.isi.edu/nsnam/ns

[15] GT_ITM. Available from : http://www.cc.gatech.edu/projects/gtitm

[16] Linux Mint, http://www.linuxmint.com

[17] Yong, L., Yang, G., and Chao, L.A survey on peer-to-peer video streaming systems. In Proceeding of Peer-to-peer Networking and Applications (January 2008)

[18] Nazanin, M., Reza, R., and Yang, G. Mesh or Multiple-Tree: A Comparative Study of Live P2P Streaming Approaches. In Proceedings of the INFOCOM (May 2007)

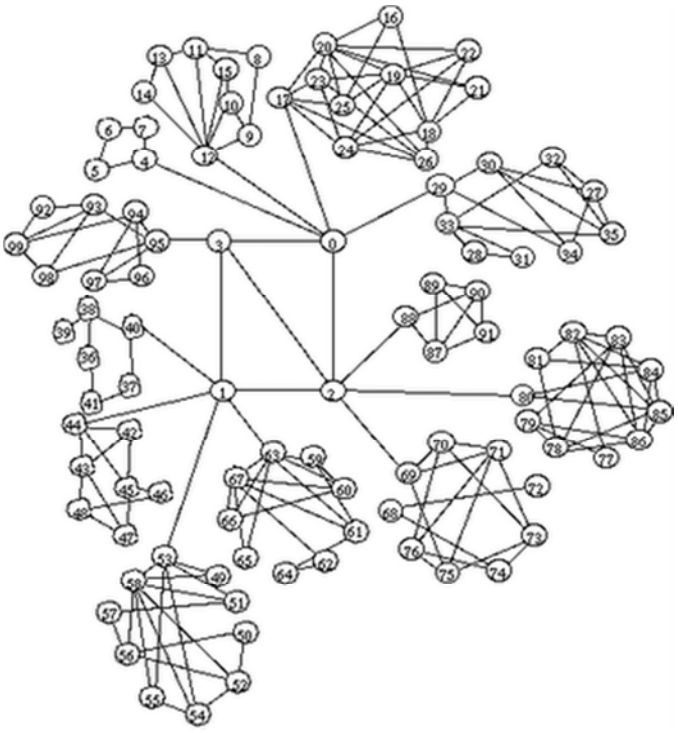
ภาคผนวก

ภาคผนวก ก

รายละเอียดโปรแกรม และเครื่องมือที่ใช้ในการทดลอง

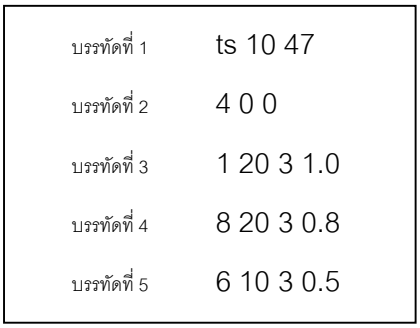
เครื่องมือสร้างโทโพโลยี (Topology Generator)

เครื่องมือสร้างโทโพโลยีชนิดนี้ สามารถสร้างกราฟได้ 3 รูปแบบ คือ “Flat random Graph”, “N-level hierarchical Graph” และ “Transit-stub hierarchical Graph” ซึ่งลักษณะของกราฟในแต่ละรูปแบบจะแตกต่างกันไป สามารถดาวน์โหลดโปรแกรมเพื่อใช้งานได้ที่เว็บไซต์ http://www.cc.gatech.edu/projects/gtitm/ จากภาพที่ ก.1 เป็นตัวอย่างการสร้างกราฟในรูปแบบ “Transit-stub hierarchical” ขนาด 200 โหนด



ภาพที่ ก.1ตัวอย่างโทโพโลยีรูปแบบ Transit-stub

ตัวอย่างรูปแบบข้อความ (Syntax) การสร้าง โทโพโลยีแบบ Transit-stub ดังนี้



ภาพที่ ก.2 รูปแบบข้อความการสร้างโทโพโลยีแบบ Transit-stub

อธิบายรูปแบบข้อความข้างต้น

บรรทัดที่ 1 <คีย์เวิร์ดของรูปแบบกราฟ><จำนวนของกราฟ><กำหนดค่าเริ่มต้น (Initial Seed)>

สร้างกราฟในรูปแบบ Transit-stub จำนวน 10 กราฟ โดยมีค่าเริ่มต้นเท่ากับ 47

บรรทัดที่ 2 <โดเมน Stub ต่อ Transit><จุดเชื่อมต่อ Transit-stub><จุดเชื่อมต่อ Stub-stub>

มี 4 stub domains ต่อ 1 transit ไม่มีการเชื่อมต่อที่ transit-stub และ Stub-stub

บรรทัดที่ 3 <จำนวนโหนด><ขนาด><ลักษณะการเชื่อมต่อ><ความน่าจะเป็นของการเชื่อมต่อ>

มี 1 Transit domain และเชื่อมต่อแบบสมบูรณ์

บรรทัดที่ 4 <จำนวนโหนด><ขนาด><ลักษณะการเชื่อมต่อ><ความน่าจะเป็นของการเชื่อมต่อ>

ที่ Transit domain มีจำนวนโหนดเฉลี่ย 8 โหนด เชื่อมต่อด้วยความน่าจะเป็น 0.8

บรรทัดที่ 5 <จำนวนโหนด><ขนาด><ลักษณะการเชื่อมต่อ><ความน่าจะเป็นของการเชื่อมต่อ>

ที่ Stub domain มีจำนวนโหนดเฉลี่ย 6 โหนด เชื่อมต่อด้วยความน่าจะเป็น 0.5

หมายเหตุ คีย์เวิร์ดของกราฟรูปแบบ “Flat randomGraph” คือ “**geo**” สำหรับ “N-level hierarchical Graph” คือ “**tier**” และ “Transit-stub hierarchical Graph” คือ “**ts**”

จำนวนโหนดทั้งหมดที่เกิดขึ้น โดยใช้รูปแบบการสร้างดังรูปแบบข้อความข้างต้น คือ 1 * 8 (1+ (4 * 6))= 200 โหนด

การสร้างกราฟจะใช้คำสั่ง itm ตามด้วยชื่อไฟล์ข้อความรูปแบบข้างต้น (เช่น itm file.txt) กราฟทั้งหมดจะถูกสร้างออกมาในไฟล์ชื่อ file-[0-9].gb จำนวน 10 กราฟซึ่งกราฟ

ดังกล่าว เป็นกราฟในรูปแบบของ Stanford Graph Base Format ขั้นตอนต่อไป คือ การเปลี่ยนกราฟให้เป็นรูปแบบtcl (tcl format) เพื่อให้โปรแกรมจำลอง NS-2 เข้าใจ

การเปลี่ยนรูปแบบกราฟ Stanford Graph Base Format ให้เป็นรูปแบบ tcl สามารถทำได้โดย ใช้โปรแกรมแปลง sgb2ns(sgb2ns Conversion Program) โดยการใช้คำสั่ง sgb2ns ตามด้วยไฟล์ .gb ซึ่งเป็นอินพุท และตามด้วย ชื่อไฟล์เอาท์พุทซึ่งเป็นไฟล์ .tcl (เช่น sgb2ns file.gb file.tcl)

สำหรับการสร้างกราฟในรูปแบบอื่นๆ จะมีรูปแบบข้อความที่แตกต่างกันไปดังภาพที่ก.3 เป็นข้อความการสร้างโทโพโลยีแบบ N-Level Hierarchicalและภาพที่ ก.4 สำหรับสร้างโทโพโลยีแบบ Flat Random

บรรทัดที่ 1	hier 10 47
บรรทัดที่ 2	3 0 0
บรรทัดที่ 3	10 10 1 0.7 0.2
บรรทัดที่ 4	10 10 1 0.5 0.2
บรรทัดที่ 5	10 10 1 0.4 0.2

ภาพที่ ก.3 รูปแบบข้อความการสร้างโทโพโลยีแบบ N-Level Hierarchical

บรรทัดที่ 1	geo 10 47
บรรทัดที่ 2	10 10 1 0.2

ภาพที่ ก.4 รูปแบบข้อความการสร้างโทโพโลยีแบบ Flat Random

จากภาพที่ ก.3 กำหนดให้กราฟแบบHierarchical มี 3 ลำดับชั้น (Levels) โดยแต่ละลำดับชั้นมี 10 โหนด และภาพที่ ก.4 กำหนดให้สร้างแบบ Flat Random มีโหนดทั้งหมด 10 โหนด และความน่าเป็นสำหรับการเชื่อมต่อระหว่างโหนดคือ 0.2

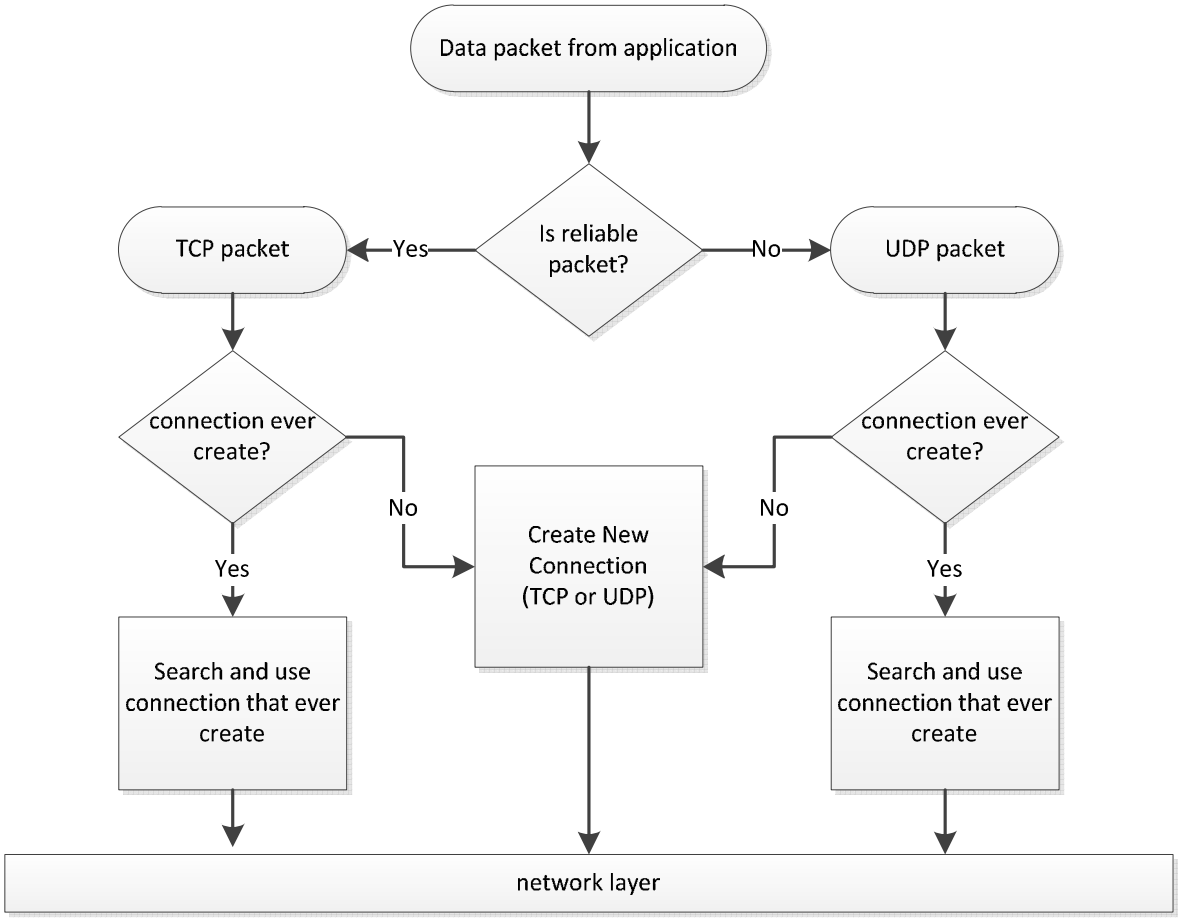
โปรแกรมเฟรมเวิร์ก NS2-app

งานวิจัยฉบับนี้ใช้ NS2-app ซึ่งเป็นเฟรมเวิร์กบนโปรแกรมจำลอง NS-2 ช่วยทำให้การสร้างโปรแกรมบนโปรแกรมจำลอง NS-2 สำหรับแอพพลิเคชั่นเลเยอร์ (Application Layer Simulation) ง่ายขึ้น โดยโปรแกรมจะประกอบไปด้วย 3 ส่วนใหญ่ๆ คือ ADU, RealApplication และ Wrapper

ในส่วนแรก ADU จะคืนค่าขนาดของทุกข้อมูล ซึ่งมาจากชั้นแอพพลิเคชั่น ที่จะถูกส่งบนเครือข่าย โดยมีหน่วยเป็นไบต์ ส่วนที่สอง RealApplication เป็นอินเตอร์เฟซสำหรับติดต่อกับโปรแกรมบนแอพพลิเคชั่นเลเยอร์ ผ่านฟังก์ชั่น send() เพื่อส่งข้อมูล และ recv() เพื่อรับข้อมูลซึ่งในส่วนนี้จะสร้างคอนเนคชั่นขึ้นโดยอัตโนมัติ และส่วนสุดท้าย Wrapper จะคอยจัดการการรับ-ส่งของแพกเกจข้อมูล

เฟรมเวิร์ค NS2-app สามารถดาวน์โหลดได้ที่ http://code.google.com/p/ns2-app/

เดิมโปรแกรมเฟรมเวิร์คนี้ ได้สร้างการเชื่อมต่อโดยใช้โปรโตคอล TCP (TCP Connection) เพียงอย่างเดียวซึ่งผู้เขียนได้เขียนการเชื่อมต่อโดยใช้โปรโตคอล UDP (UDP Connection) เพิ่ม เพื่อรองรับการส่งข้อมูลไลฟ์สตรีมมิ่ง ซึ่งมีลักษณะการส่งแบบ UDP โดยก่อนการสร้างการเชื่อมต่อ ข้อมูลจะถูกแยกประเภทระหว่าง ข้อมูลที่จะถูกส่งด้วย TCP และข้อมูลที่จะถูกส่งด้วย UDP จากนั้นหากไม่เคยมีการสร้างการเชื่อมต่อระหว่างโหนดนั้นมาก่อน โปรแกรมจะสร้างการเชื่อมต่อใหม่ขึ้นมาแต่ถ้าหากเคยมีการสร้างการเชื่อมต่อระหว่างโหนดนั้นมาแล้ว โปรแกรมจะนำการเชื่อมต่อเดิมมาใช้ ภาพที่ ก.2 แสดงผังการทำงานของเฟรมเวิร์ค NS2-app

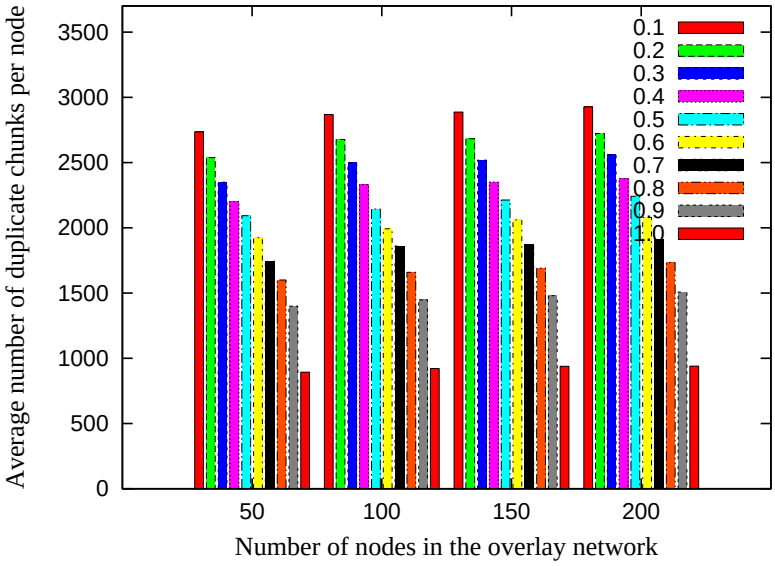


ภาพที่ ก.5แสดงผังงานการทำงานของเฟรมเวิร์ค NS2-app

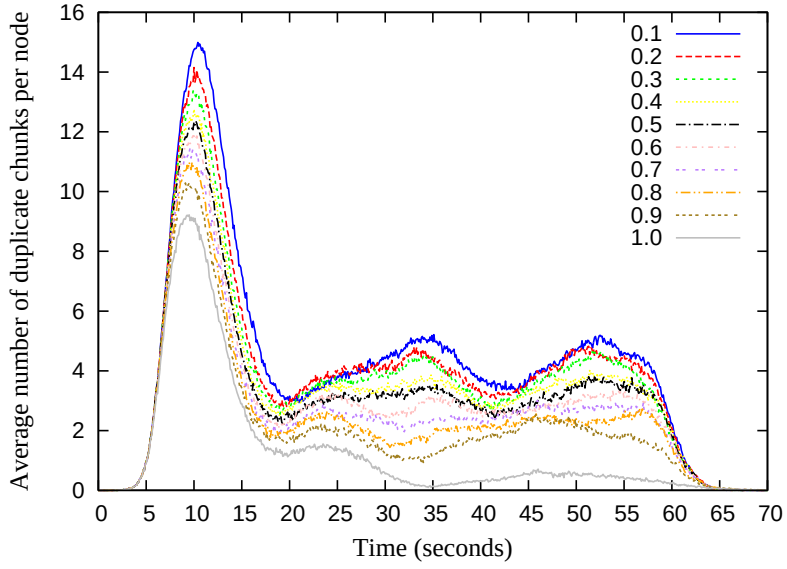
ภาคผนวก ข

การทดสอบค่าตัวแปรที่ใช้ในการทดลอง

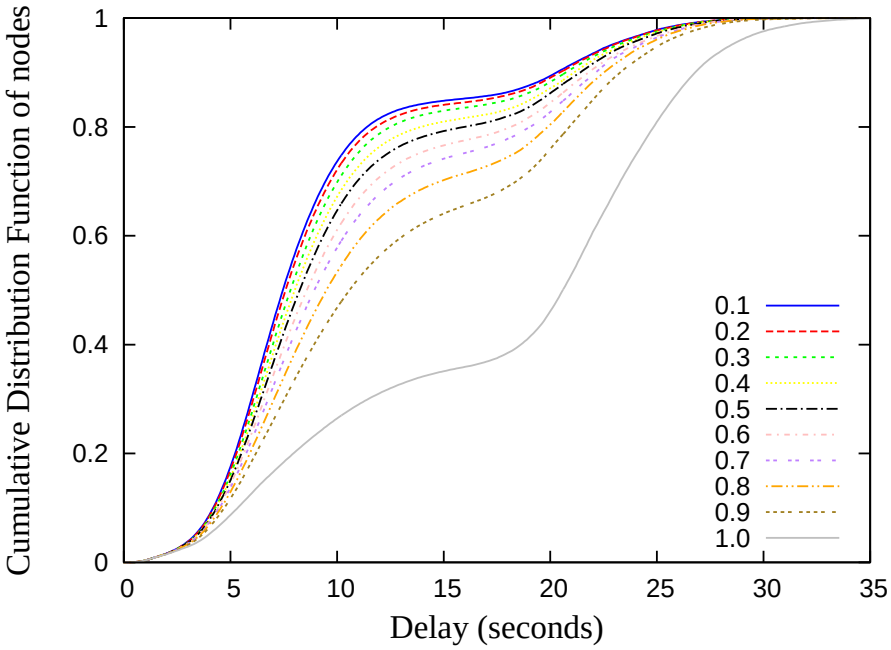
พิจารณาตัวแปรขั้นการปรับ (Step Size) ของโปรโตคอล Reinforcement-based แบบไม่มีการวนซ้ำ โดยทำการทดสอบที่ค่า 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 และ 1.0 จากกราฟที่ได้ แสดงให้เห็นว่า ผลการทดลองตามภาพข้างล่างนี้ ทดลองเมื่อทุกโหนดมีจำนวนโหนดเพื่อนบ้านเฉลี่ย 10 โหนด



ภาพที่ ข.1 แสดงจำนวนข้อมูลซ้ำที่ค่าการปรับระดับต่างๆ



ภาพที่ ข.2 แสดงจำนวนข้อมูลซ้ำเมื่อระยะเวลาผ่านไปที่ค่าการปรับต่างๆ



ภาพที่ ข.3 แสดงความเร็วการกระจายชิ้นข้อมูลที่ค่าการปรับต่างๆ

จากผลการทดลองการเปลี่ยนตัวแปรขั้นการปรับ (Step Size) จะเป็นว่าผลการทดลองจะมีผลต่อกัน ดังเช่น ที่ค่าการปรับ 0.1 จะให้ความเร็วสำหรับการกระจายข้อมูลดีที่สุด แต่จะทำให้เกิดข้อมูลซ้ำมากที่สุดด้วย ดังนั้นจึงสรุปผลว่า ค่าขั้นการปรับทำให้ความเร็วการกระจายข้อมูลกับการเกิดชิ้นข้อมูลซ้ำมีผลต่อกัน (Trade-off)

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวขวัญจิรา นาคเดช เกิดเมื่อวันที่ 12 กรกฎาคม พ.ศ. 2529 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาระดับมัธยมศึกษาจากโรงเรียนเซนต์โยเซฟ ทิพวัล จังหวัดสมุทรปราการ สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมอิเล็กทรอนิกส์และโทรคมนาคม ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ และโทรคมนาคม คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี ในปีการศึกษา 2551 และเข้าศึกษาในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2552

ผู้เขียนได้ตีพิมพ์ผลงานทางวิชาการในรายงานการประชุม “2011 Wireless Communication, Networking and Mobile Computing” ในชื่อว่า “A Reinforcement-Based Push-Pull Approach for Peer-to-peer Live Streaming Data” ณ เมืองอู่ฮั่น ประเทศจีน เมื่อเดือนกันยายน พ.ศ. 2554