การปรับปรุงประสิทธิภาพของโปรโตคอลทีซีพีแรพพิดในเครือข่ายความเร็วสูง

นายนพพร คงวัดใหม่

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมศาสตร์คอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

IMPROVING TCP RAPID PERFORMANCE IN HIGH-SPEED NETWORKS

Mr Nopporn Kongwatmai

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Computer Science

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2011

นพพร คงวัดใหม่ : การปรับปรุงประสิทธิภาพของโปรโตคอลทีซีพีแรพพิดในเครือข่ายความเร็วสูง. (IMPROVING TCP RAPID PERFORMANCE IN HIGH-SPEED NETWORKS) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร. กุลธิดา โรจน์วิบูลย์ชัย, 43 หน้า.

ในหนึ่งทศวรรษที่ผ่านมาได้มีผู้นำเสนอโปรโตคอลสำหรับควบคุมการส่งข้อมูลบนเครือข่ายคอมพิวเตอร์ความเร็วสูงหลายโปรโตคอลด้วยกัน โดยโปรโตคอลดังกล่าวได้พัฒนาขึ้นมาเพื่อแก้ปัญหาในเรื่องประสิทธิภาพหากนำไปใช้บนเครือข่ายความเร็วสูง กล่าวคือ โปรโตคอลต้องสามารถใช้ประโยชน์จากขนาดความจุหรือแบนด์วิดธ์ที่เพิ่มมากขึ้นได้อย่างมีประสิทธิภาพ และต้องไม่ก้าวร้าวต่อโปรโตคอลอื่นที่ทำงานร่วมกัน ซึ่งทีซีพีแรพพิดก็เป็นหนึ่งในโปรโตคอลดังกล่าวที่ถูกนำเสนอออกมาล่าสุด โดยมีคุณสมบัติที่ค่อนข้างโดดเด่นหลายประการด้วยกันคือ สามารถวัดค่าแบนด์วิดธ์ที่เหลืออยู่บนเครือข่ายได้อย่างรวดเร็วภายในเวลาอันสั้น ไม่ส่งข้อมูลออกไปบนเครือข่ายมากเกินกว่าที่ควรจะเป็น และไม่ก้าวร้าวกับโปรโตคอลอื่น เป็นต้น โดยเทคนิคที่ใช้ในโปรโตคอลแรพพิดคือการวัดค่าระยะเวลาที่แต่ละแพ็กเก็ตไปค้างอยู่ในบัฟเฟอร์ของเร้าเตอร์ ซึ่งทำให้สามารถค้นหาแบนด์วิดธ์ได้รวดเร็วและไม่ก้าวร้าวต่อโปรโตคอลตัวอื่น แต่อย่างไรก็ตามยังพบว่าแรพพิดไม่สามารถทำงานร่วมกับโปรโตคอลตัวอื่นที่มีความก้าวร้าว เช่นโปรโตคอลที่มีการปรับเปลี่ยนอัตราการส่งข้อมูลโดยดูจากการสูญหายของแพ็กเก็ตได้ จากการที่นำโปรโตคอลแรพพิดไปทำงานคู่กับโปรโตคอลนิวเรโนซึ่งเป็นโปรโตคอลที่มีการใช้งานกันอย่างแพร่หลายในปัจจุบันนั้น พบว่าแรพพิดไม่สามารถทำงานต่อไปได้อย่างมีประสิทธิภาพ และจากการวิเคราะห์พบว่าแรพพิดยังไม่มีความเสถียรภาพมากพอในการปรับเปลี่ยนอัตราการส่งข้อมูล ดังนั้นในวิทยานิพนธ์นี้จึงได้นำเสนอวิธีการแก้ไขโดยการนำอัลกอริทึมที่ชื่อว่าไอจีไอมาทำงานร่วมกันอัลกอริทึมของแรพพิด เพื่อช่วยให้แรพพิดสามารถทำงานร่วมกับโปรโตคอลอื่นที่มีความก้าวร้าวมากกว่าได้

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

5271428221: MAJOR COMPUTER SCIENCE

KEYWORDS : High-speed Networks / TCP RAPID / Available Bandwidth Estimate

NOPPORN KONGWATMAI : IMPROVING TCP RAPID PERFORMANCE IN HIGH-SPEED NETWORKS. ADVISOR : ASSISTANT PROFESSOR KULTIDA ROJVIBOONCHAI,PH.D, 43 pp.

Several Transmission Control Protocols have been proposed to address the significant problem on congestion control avoidance in order to be efficient in achieving high throughput on high-speed network in non-intrusive manner. TCP RAPID has recently been proposed as a new TCP that can search and adapt to the available bandwidth within a few RTT. TCP RAPID is an outstanding protocol in many aspects especially in queue friendliness, fairness, and rapidly searching for dynamic available bandwidth. The bandwidth estimation technique used in TCP RAPID is based on queuing delay. The delay based protocol maintains high speed in bandwidth estimation and in non-intrusive manner. However, we found that TCP RAPID cannot co-exist with other loss based protocols such as TCP NewReno, which is commonly-used TCP version on the Internet. In this paper, we propose a solution to address this problem for TCP RAPID that is to use the Initial Gap Increasing (IGI) algorithm to increase the stability of TCP RAPID. This can cause TCP RAPID to survive in the aggressive environment as it is in the today Internet.

Department : Computer Engineering Student's Signature .....

Field of Study : Computer Science Advisor's Signature .....

Academic Year : 2011

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ ด้วยความช่วยเหลือของอาจารย์ ดร. กุลธิดา โรจน์วิบูลย์ชัย อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งอาจารย์ได้ให้คำปรึกษา ความรู้ และคำแนะนำที่เป็นประโยชน์ในการทำวิจัยทั้งทางตรงและทางอ้อม รวมไปถึงคำวิจารณ์ที่เป็นประโยชน์ต่อการทำวิทยานิพนธ์และการนำเสนอผลงานทางวิชาการที่เกี่ยวข้อง อาจารย์มีความเมตตา และให้ความใส่ใจที่ดีมาโดยตลอด ผู้วิจัยจึงขอกราบขอบพระคุณมา ณ ที่นี้

ขอขอบพระคุณผู้ช่วยศาสตราจารย์ ดร. เฉลิมเอก อินทนากรวิวัฒน์ ประธานกรรมการสอบวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร. เกริก ภิรมย์โสภา กรรมการสอบวิทยานิพนธ์ และผู้ช่วยศาสตราจารย์ ดร. สุขุมาล กิติสิน กรรมการสอบวิทยานิพนธ์จากภายนอกมหาวิทยาลัย ที่ได้สละเวลาและให้คำแนะนำเพื่อให้วิทยานิพนธ์ฉบับนี้สมบูรณ์ยิ่งขึ้น

นอกจากนี้ขอขอบพระคุณผู้ช่วยศาสตราจารย์ Jasleen Kaur และคุณ Vishnu Konda จาก University of North Carolina at Chapel Hill ที่ให้ความกรุณาแบ่งปันซอร์สโค้ดของโปรโตคอลแรพพิด ทำให้ผู้วิจัยสามารถทำงานได้ง่ายขึ้น

ขอกราบขอบพระคุณบิดา มารดา ครอบครัว อาจารย์และเจ้าหน้าที่ภาควิชาวิศวกรรมศาสตร์คอมพิวเตอร์ รวมทั้งเพื่อน ๆ ที่ให้กำลังใจ ให้ความรู้ และความช่วยเหลือในด้านต่าง ๆ ในการทำวิจัยด้วยดีเสมอมา

สารบัญ

หน้า

บทที่ **หน้า**

สารบัญตาราง

สารบัญภาพ

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากปัจจุบันเทคโนโลยีทางด้านเครือข่ายคอมพิวเตอร์มีความก้าวหน้าไปอย่างมาก ส่งผลให้สื่อกลางสำหรับการส่งข้อมูลบนเครือข่ายคอมพิวเตอร์มีวิวัฒนาการก้าวหน้าไปอย่างรวดเร็ว ดังจะเห็นได้ว่าสื่อกลางสำหรับส่งข้อมูลนั้นมีขนาดความจุหรือแบนด์วิดธ์ (Bandwidth) ที่สูงขึ้นเป็นอย่างมาก ส่งผลให้การส่งข้อมูลทำได้รวดเร็วมากยิ่งขึ้น (High-speed Networks) แต่อย่างไรก็ตามโปรโตคอลสำหรับควบคุมการส่งข้อมูล (Transmission Control Protocol หรือ TCP) บางโปรโตคอลยังไม่สามารถใช้ประโยชน์จากขนาดของแบนด์วิดธ์ที่มีขนาดกว้างมากดังกล่าวได้อย่างมีประสิทธิภาพ เนื่องจากไม่สามารถวัดค่าแบนด์วิดธ์ที่มีอยู่ได้อย่างรวดเร็วและถูกต้องเท่าที่ควร ยกตัวอย่างเช่น โปรโตคอล NewReno [1] ถือได้ว่าเป็นโปรโตคอลที่มีการใช้งานอย่างแพร่หลายในปัจจุบัน ยังต้องใช้เวลานานกว่าที่จะส่งข้อมูลได้เต็มขนาดของแบนด์วิดธ์ที่มีอยู่ เนื่องด้วยโปรโตคอล NewReno ใช้อัลกอริทึม Additive Increase/Multiplicative Decrease หรือ AIMD ซึ่งจะค่อย ๆ เพิ่มอัตราความเร็วในการส่งข้อมูลขึ้นในปริมาณที่เท่า ๆ กันในทุกช่วงเวลาถึงแม้ว่าขณะที่ส่งข้อมูลนั้นมีแบนด์วิดธ์เหลืออยู่จานวนมากก็ตาม โดยการทำงานในส่วนนี้จะเกี่ยวข้องกับอัลกอริทึมในการควบคุมความคับคั่ง (Congestion Control) ซึ่งถือว่าเป็นส่วนสำคัญของโปรโตคอลสาหรับควบคุมการส่งข้อมูล โดยทำหน้าที่ในการกำหนดหรือควบคุมปริมาณการส่งข้อมูลให้สอดคล้องกับแบนด์วิดธ์ที่มีอยู่เพื่อให้การส่งข้อมูลมีประสิทธิภาพมากที่สุด และถือได้ว่าปัญหาการควบคุมความคับคั่งนั้นเป็นปัญหาที่สำคัญติดหนึ่งในสิบของปัญหาทางด้านเครือข่ายในช่วงทศวรรษที่ผ่านมา เนื่องจากเป็นงานที่ยาก มีหลากหลายประเด็นที่ต้องคำนึงถึง เช่น ต้องทำหน้าที่ค้นหาค่าของแบนด์วิดธ์ที่มีอยู่ให้ได้อย่างรวดเร็ว แม่นยำ และไม่รบกวนการทำงานของโปรโตคอลตัวอื่น

จากที่ผ่านมานั้นมีหลาย ๆ โปรโตคอลที่ถูกนาเสนอขึ้นมาเพื่อใช้แก้ปัญหาดังกล่าว ได้แก่ HighSpeed TCP [2], CUBIC [3], BIC[4], FAST [5], PCP [6], SRF [7] และ Scalable [8] เป็นต้น แต่อย่างไรก็ตามโปรโตคอลต่าง ๆ เหล่านี้ก็ยังใช้เวลานานสำหรับการค้นหาแบนด์วิดธ์ที่มีอยู่ โดยเฉพาะอย่างยิ่งบนเครือข่ายที่มีขนาดแบนด์วิดธ์เป็นระดับกิกะบิต (Gigabits) ดังนั้นจึงได้มีการคิดค้นอัลกอริทึมหรือเครื่องมือสาหรับวัดค่าแบนด์วิดธ์ขึ้นมาโดยใช้เทคนิคที่แตกต่างกันไป รวมถึงมีการประเมินประสิทธิภาพของเครื่องมือต่าง ๆ เหล่านั้น [9] ซึ่งผลจากการ

ประเมินพบว่า pathChirp [10] เป็นเครื่องมือหนึ่งที่มีประสิทธิภาพดีที่สุด โดยสามารถวัดแบนด์วิดธ์ได้แม่นยำ รวดเร็ว และมีผลค่าใช้จ่ายน้อยกว่าเครื่องมือตัวอื่น ด้วยสาเหตุนี้จึงได้มีการนำเทคนิคของ pathChirp มาพัฒนาเป็นโปรโตคอลควบคุมการส่งข้อมูลที่มีชื่อว่าแรพพิด (RAPID) [11] โดยผลการทางานของ RAPID นั้นถือได้ว่าเป็นโปรโตคอลที่ให้ประสิทธิภาพดีที่สุดเมื่อเทียบกับ TCP สาหรับเครือข่ายความเร็วสูงตัวอื่น ๆ ที่ได้นำเสนอมาก่อนหน้านี้ ซึ่งคุณสมบัติเด่นของ RAPID ก็คือสามารถค้นหาแบนด์วิดธ์ได้อย่างรวดเร็ว และสามารถปรับตัวเองให้สอดคล้องกับการเปลี่ยนแปลงของแบนด์วิดธ์ได้อย่างรวดเร็วเช่นเดียวกัน ดังนั้น RAPID เปรียบเสมือนโปรโตคอลในการควบคุมการส่งข้อมูลที่ดีและทันสมัยที่สุดตัวหนึ่ง ซึ่งเหมาะสมที่จะนำมาศึกษา ปรับปรุง หรือพัฒนาต่อ

ดังที่กล่าวข้างต้น RAPID เป็นโปรโตคอลที่สามารถปรับตัวเองให้เข้ากับการเปลี่ยนแปลงของแบนด์วิดธ์ได้อย่างรวดเร็ว ทำให้ RAPID เป็นโปรโตคอลที่ไม่ก้าวร้าว (Aggressive) และสามารถนาไปใช้งานร่วมกับโปรโตคอลอื่น ๆ ได้โดยไม่รบกวนการทำงานของโปรโตคอลอื่น แต่อย่างไรก็ตามข้อดีดังกล่าวก็ก่อให้เกิดผลเสียตามมาเช่นกัน กล่าวคือจากการทดลองบน NS-2 [13] ซึ่งนำ RAPID ไปทำงานร่วมกับโปรโตคอลอื่น ๆ ที่มีคุณลักษณะค่อนข้างก้าวร้าวหรือไม่สามารถปรับตัวเองได้เร็วพอกับความคับคั่งบนเส้นทางการสื่อสาร ทำให้ RAPID เป็นฝ่ายที่ต้องลดความเร็วในการส่งข้อมูลลงหรือถอยตัวเองออกไปก่อนโปรโตคอลตัวอื่น ๆ เมื่อระดับความคับคั่งบนเครือข่ายเพิ่มมากขึ้น ซึ่งหมายถึงว่าประสิทธิภาพของ RAPID จะลดลงเมื่อปฏิบัติงานบนเครือข่ายที่มีความคับคั่งหรือมีการเปลี่ยนแปลงสูง หรือทำงานร่วมกับโปรโตคอลที่ไม่สามารถปรับตัวเองได้เร็วพอต่อสภาวการณ์คับคั่งดังกล่าว อย่างเช่น NewReno เป็นต้น อย่างไรก็ตามใน [11] ได้ประเมินผลการทางานของ RAPID เมื่อปฏิบัติงานร่วมกับ TMIX [14] ซึ่งเป็นตัวจำลองการจราจรบนอินเตอร์เน็ต โดยกำหนดให้ TMIX จำลองการใช้งานแบนด์วิดธ์ที่ 70 Mbps บนเครือข่ายที่มีขนาดแบนด์วิดธ์ 100 Mbps พบว่า RAPID สามารถใช้งานแบนด์วิดธ์ที่เหลืออยู่ 30 Mbps ได้อย่างเต็มประสิทธิภาพ แต่อย่างไรก็ตามผลการทดลองดังกล่าวไม่ได้สะท้อนความจริงของการจราจรบนอินเตอร์เน็ตที่ว่าแต่ละโปรโตคอลบนเครือข่ายมีสิทธิเท่าเทียมกันในการเข้าใช้งานแบนด์วิดธ์ โดยไม่สามารถกำหนดขนาดแบนด์วิดธ์ที่เท่าเทียมกันล่วงหน้าให้กับแต่ละแอปพลิเคชันได้ ดังนั้นโปรโตคอลแต่ละตัวจะแข่งขันกันเพื่อให้ตนเองสามารถส่งข้อมูลได้อย่างเต็มประสิทธิภาพมากที่สุด ด้วยสาเหตุนี้จึงต้องการที่จะปรับปรุง RAPID ซึ่งมีคุณสมบัติที่ดีอยู่แล้วในด้านอื่น ๆ ให้สามารถทำงานร่วมกับโปรโตคอลที่ค่อนข้างมีลักษณะก้าวร้าวโดยเฉพาะอย่างยิ่ง

NewReno ได้อย่างมีประสิทธิภาพมากยิ่งขึ้น ซึ่งหมายความว่า RAPID จะต้องสามารถทนทานต่อสถานการณ์ที่แบนด์วิดธ์ถูกแย่งใช้งานจากโปรโตคอลอื่นค่อนข้างเยอะได้ดียิ่งขึ้น

สาเหตุที่ต้องมีการเปรียบเทียบการทำงานกับโปรโตคอล NewReno เนื่องจากในปัจจุบันนี้ระบบปฏิบัติการที่ได้รับความนิยมมากที่สุด และถือว่าเป็นระบบปฏิบัติการหลักของผู้ใช้ในโลกนี้คือระบบปฏิบัติการวินโดวส์ (Windows) ซึ่งโปรโตคอลที่ใช้ในการควบคุมการส่งข้อมูลที่ใช้ในวินโดวส์จะสืบทอดมาจากโปรโตคอล NewReno ถึงแม้ว่าจะมีการติดตั้งโปรโตคอลตัวใหม่อย่างเช่น Compound TCP (CTCP) ลงไปในวินโดวส์เวอร์ชั่นใหม่ ๆ เพื่อช่วยให้สามารถปรับอัตราการส่งข้อมูลให้สอดคล้องกับแบนด์วิดธ์ที่มีอยู่ได้รวดเร็วขึ้น แต่โปรโตคอลดังกล่าวก็ยังคงอิงอยู่กับโปรโตคอล NewReno อยู่เช่นเดียวกัน ดังนั้นจึงมีความจำเป็นอย่างยิ่งในการประเมินผลในส่วนนี้ โปรโตคอล NewReno จึงถูกเลือกมาใช้เป็นตัวแทนของโปรโตคอลที่ถูกติดตั้งใช้งานในปัจจุบัน

1.2 วัตถุประสงค์ของการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อนำเสนอวิธีการในการปรับปรุงโปรโตคอล RAPID เพื่อให้โปรโตคอลสามารถทำงานได้ดีขึ้น ซึ่งจากปัญหาที่พบกับ RAPID ดังที่กล่าวข้างต้นว่าโปรโตคอลจะถอยตัวเองออกมาเร็วกว่าโปรโตคอลตัวอื่นเท่าให้ RAPID สูญเสียความเสมอภาคไป ดังนั้นงานวิจัยนี้จึงต้องการปรับปรุง RAPID โดยเน้นในการเพิ่มความเสถียรภาพและความเสมอภาคให้กับ RAPID เพื่อแก้ปัญหาดังกล่าว

1.3 ขอบเขตของการวิจัย

1. ทำการปรับปรุง RAPID เพิ่มเติมบน NS-2.34 โดยจะเน้นในเรื่องของการเพิ่มความเสถียรภาพ และความเสมอภาคของโปรโตคอล

2. สร้างแบบจำลองบน NS-2.34 เพื่อดูผลการทำงานของ RAPID ที่มีการแก้ไขแล้ว โดยแบบจำลองดังกล่าวจะต้องเป็นแบบจำลองแบบเดียวกันกับที่ใช้ใน [11] เพื่อให้แน่ใจว่า RAPID ที่ปรับปรุงดังกล่าวยังคงคุณสมบัติที่ดีของ RAPID เดิมไว้ได้ นอกจากนี้ต้องสร้างแบบจำลองเพื่อดูการทำงานของ RAPID เมื่อทำงานร่วมกับโปรโตคอล NewReno โดยเฉพาะเวลาที่โปรโตคอล NewReno เริ่มอิ่มตัวแล้ว

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ทำให้โปรโตคอล RAPID มีความเสถียรภาพมากขึ้น โดยเฉพาะอย่างยิ่งเวลาที่นำไปใช้ร่วมกับโปรโตคอลที่มีคุณลักษณะก้าวร้าวอย่างเช่น NewReno ซึ่งเป็นโปรโตคอลที่มีการใช้งานอย่างแพร่หลายในปัจจุบัน ซึ่งหาก RAPID ได้รับการแก้ไขปัญหาดังที่กล่าวข้างต้นได้ก็จะสามารถเพิ่มความมั่นใจได้ว่า RAPID จะสามารถนำไปติดตั้งใช้งานจริงได้อย่างไม่มีปัญหา ซึ่งในสถานการณ์จริงนั้นเราไม่สามารถจะควบคุมได้ว่า RAPID จะต้องไปทำงานร่วมกับโปรโตคอลใดโดยเฉพาะอย่างยิ่งโปรโตคอล NewReno ซึ่งเป็นโปรโตคอลที่มีการใช้งานอยู่อย่างแพร่หลายในปัจจุบัน

2. สามารถชี้ให้เห็นถึงข้อดีข้อเสียของแต่ละเทคนิคที่ใช้ในการวัดแบนด์วิดธ์ โดยสามารถนำข้อดีของแต่ละเทคนิคมาประยุกต์ใช้ในการแก้ปัญหากับการจัดการความคับคั่งสาหรับโปรโตคอลควบคุมการส่งข้อมูล ซึ่งยังไม่มีงานวิจัยมากนักที่นำเครื่องมือต่าง ๆ เหล่านี้มาประยุกต์ใช้ในการแก้ปัญหาดังกล่าวนี้

1.5 วิธีดำเนินการวิจัย

1. นำซอร์สโค้ด (Source Code) ของ RAPID ซึ่งได้มาจากผู้พัฒนา RAPID มาทำการศึกษาอัลกอริทึม

2. ออกแบบวิธีการแก้ไขอัลกอริทึมของ RAPID โดยประยุกต์ใช้อัลกอริทึมของ IGI เข้ามาร่วม

3. นำวิธีการแก้ไขดังกล่าวมาพัฒนาหรือปรับปรุง RAPID บน NS-2.34

4. สร้างแบบจำลองสาหรับทดสอบการทำงานบน NS-2.34

5. วิเคราะห์ผลการทดลอง

6. ปรับปรุงส่วนที่ทำงานผิดพลาดของ RAPID

7. สรุปผลและเรียบเรียงวิทยานิพนธ์

1.6 ลำดับขั้นตอนในการเสนอผลการวิจัย

งานวิจัยฉบับนี้แบ่งเนื้อหาออกเป็น 5 บท ดังต่อไปนี้

บทที่ 1 บทนำ ซึ่งประกอบด้วย ความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ของการวิจัย ขอบเขตของการวิจัย ประโยชน์ที่คาดว่าจะได้รับ วิธีดำเนินการวิจัย และสุดท้ายคือลำดับขั้นตอนในการเสนอผลการวิจัย

บทที่ 2 เอกสารและงานวิจัยที่เกี่ยวข้อง เป็นบทที่นำเสนอทฤษฎีและงานวิจัยที่เกี่ยวข้องกับการทำงานของโปรโตคอลสำหรับควบคุมการส่งข้อมูลบนเครือข่ายความเร็วสูง

บทที่ 3 โปรโตคอลที่นำเสนอ เป็นส่วนที่อธิบายอัลกอริทึมที่ผู้วิจัยได้นำเสนอเพื่อปรับปรุงโปรโตคอล RAPID

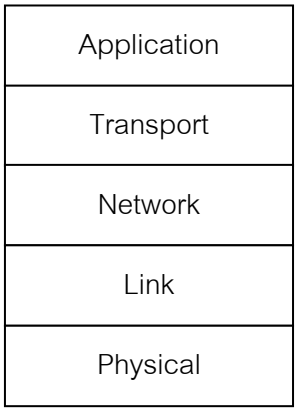
บทที่ 4 การจำลองและวิเคราะห์ผลการจำลอง จะอธิบายแบบจำลองบน NS-2 ที่สร้างขึ้นเพื่อทำการทดสอบการทำงานของโปรโตคอลที่นำเสนอ รวมถึงการวิเคราะห์ผลที่ได้รับ

บทที่ 5 สรุปและข้อเสนอแนะ จะเป็นการสรุปผลรวมถึงข้อเสนอแนะที่สามารถนำไปใช้ในการศึกษาต่อไป

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 สถาปัตยกรรมเครือข่ายอินเตอร์เน็ต

สถาปัตยกรรมเครือข่ายอินเทอร์เน็ตนั้นในบางครั้งจะเรียกว่าสถาปัตยกรรม TCP/IP เนื่องจากมี Transmission Control Protocol (TCP) และ Internet Protocol (IP) เป็นโปรโตคอลหลักของสถาปัตยกรรม สถาปัตยกรรมเครือข่ายอินเตอร์เน็ตเป็นแบบจำลองที่ประกอบด้วย 4 ลำดับชั้นด้วยกัน ซึ่งเป็นสถาปัตยกรรมที่นำเสนอมาเพื่อใช้แทน Open Systems Interconnection model (OSI model) โดยในแต่ละชั้นจะทำงานหรือมีหน้าที่หลักของตัวเองแตกต่างกันออกไป ดังนั้นในแต่ละชั้นก็จะมีการใช้งานโปรโตคอลที่แตกต่างกันออกไปขึ้นอยู่กับหน้าที่การทำงานของตนเองดังกล่าว แต่อย่างไรก็ตามในแต่ละลำดับชั้นจะทำหน้าที่ในการให้บริการแก่ชั้นที่อยู่ด้านบนของตัวเอง โดยสำหรับสถาปัตยกรรมเครือข่ายอินเตอร์เน็ตนั้นสามารถแสดงด้วยรูปภาพที่ 2.1 ดังต่อไปนี้



รูปภาพที่ 2.1 สถาปัตยกรรมเครือข่ายอินเตอร์เน็ต

1. Application Layer – เป็นระดับชั้นของแอปพลิเคชันโดยจะประกอบด้วยหลาย ๆ โปรโตคอลด้วยกันตามแต่ละแอปพลิเคชันที่ต้องการใช้งานเครือข่ายอินเตอร์เน็ต เช่น Hypertext Transfer Protocol (HTTP), Simple Mail Transfer Protocol (SMTP) เป็นต้น

2. Transport Layer – เป็นระดับชั้นที่ทำหน้าที่ควบคุมหรือจัดการกับการส่งข้อมูลให้กับแอปพลิเคชันที่อยู่ในระดับชั้นของแอปพลิเคชัน ซึ่งประกอบด้วย 2 โปรโตคอลที่สำคัญ คือ Transmission Control Protocol (TCP) และ User Datagram Protocol (UDP) โดยความแตกต่างระหว่างสองโปรโตคอลคือ TCP จะทำหน้าที่ส่งข้อมูลโดยรับประกันว่าข้อมูลจะส่งออกไป

ยังผู้รับได้อย่างถูกต้อง แต่สำหรับ UDP นั้นจะส่งข้อมูลออกไปอย่างเดียวโดยไม่รับประกันว่าข้อมูลที่ส่งออกไปจะถึงผู้รับหรือไม่

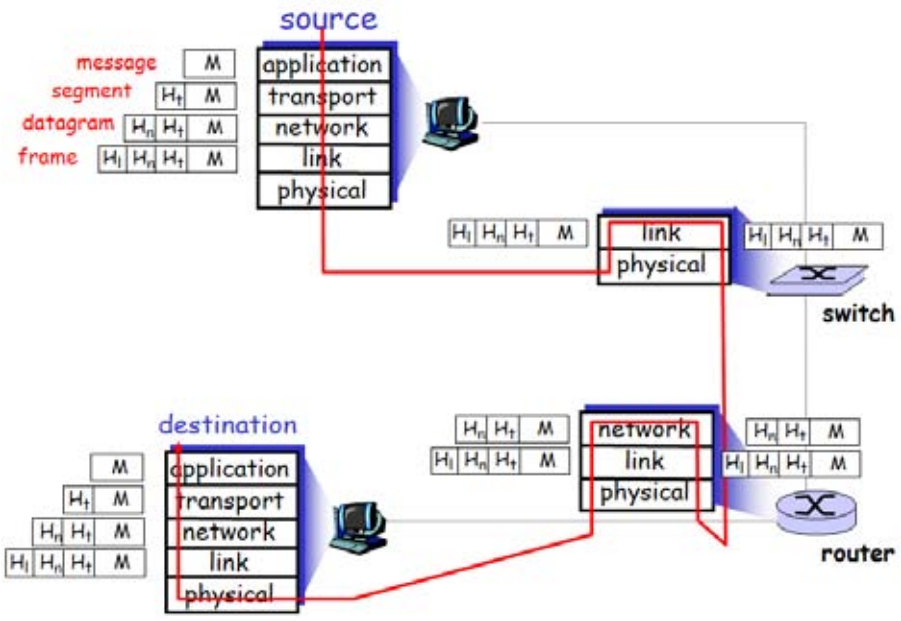
3. Network Layer – เป็นระดับชั้นที่ทำหน้าที่ในการจัดการเรื่องเส้นทางในการส่งข้อมูล (Routing) โดยโปรโตคอลที่ใช้คือ Internet Protocol (IP)

4. Link Layer – เป็นระดับชั้นที่ควบคุมการส่งข้อมูลระหว่างโหนดที่อยู่ติดกันซึ่งเรียกว่าลิงค์

5. Physical Layer – เป็นระดับชั้นที่เกี่ยวข้องกับอุปกรณ์เครือข่ายที่ทำหน้าที่ส่งข้อมูลออกไปยังช่องทางการสื่อสารหรือเครือข่ายคอมพิวเตอร์

2.2 Messages, Segments, Datagram และ Frames

การส่งข้อมูลตามสถาปัตยกรรมเครือข่ายอินเตอร์เน็ตจะใช้แนวคิดของการห่อหุ้ม (Encapusulation) กล่าวคือในการส่งข้อมูลนั้น แต่ละชั้นจะรับข้อมูลหรือแพ็กเก็ต (Packet) จากระดับชั้นที่อยู่ด้านบนแล้วทำการแปะข้อมูลเพิ่มเติมของระดับชั้นตัวเองเข้าไปในแพ็กเก็ตที่รับเข้ามาดังกล่าว โดยไม่สนใจว่าแพ็กเก็ตที่รับเข้ามานั้นมีโครงสร้างเป็นอย่างไร หลังจากนั้นจึงค่อยส่งลงไปยังระดับชั้นด้านล่างต่อไป โดยการทำงานสำหรับการส่งข้อมูลสามารถแสดงได้ดังรูปภาพที่ 2.2 ต่อไปนี้



ภาพที่ 2.2 ตัวอย่างการไหลของข้อมูลบนสถาปัตยกรรมอินเตอร์เน็ต

ฝั่งส่งจะเริ่มต้นส่งข้อมูล โดยเริ่มจากการที่ Application Layer ส่งข้อมูลหรือเรียกว่าเมสเสจ (Message) ที่ต้องการส่งลงไปยัง Transport Layer หลังจากนั้น Transport Layer จะทำการแปะข้อมูลเพิ่มเติมของตนเอง (Transport-layer header information) ไปยังส่วนหัวของเมสเสจที่รับเข้ามา โดยจะเรียกข้อมูลที่ถูกแปะข้อมูลเพิ่มเติมจาก Transport Layer เข้าไปแล้วว่าเซกเมนต์ (Segment) หรือกล่าวได้ว่าเซกเมนต์ดังกล่าวได้ห่อหุ้มเมสเสจที่ได้รับมาจาก Application Layer เอาไว้ โดยที่ข้อมูลที่เพิ่มเข้าไปนั้นจะถูกแกะและนำไปใช้ประโยชน์โดย Transport Layer ในฝั่งรับ ยกตัวอย่างเช่นผู้รับจะใช้ข้อมูลดังกล่าวเพื่อสามารถที่จะส่งเมสเสจที่รับเข้ามาไปให้แอปพลิเคชันที่ถูกต้องได้ หลังจากนั้น Transport Layer จะส่งเซกเมนต์ที่ได้ลงไปยัง Network Layer และด้วยวิธีการเดียวกันนั้น Network Layer ก็จะเพิ่มข้อมูลเพิ่มเติมของตัวเองเข้าไปยังเซกเมนต์ที่รับเข้ามา โดยจะเรียกเซกเมนต์ที่รับมาจาก Transport Layer ที่ถูกเพิ่มข้อมูลเพิ่มเติมของ Network Layer เข้าไปว่าดาต้าแกรม (Datagram) หลังจากนั้นก็ส่งลงไปให้ Link Layer หลังจากนั้น Link Layer ก็จะเพิ่มข้อมูลของตนเองเข้าไปยังดาต้าแกรมที่รับเข้ามา โดยจะเรียกข้อมูลดังกล่าวว่าเฟรม (Frame) แล้วจึงค่อยส่งไปยัง Physical Layer เพื่อส่งข้อมูลไปบนสื่อที่ใช้ส่งข้อมูลตามลำดับ

เมื่อข้อมูลมาถึงยังฝั่งรับโดยจะผ่านมาทาง Physical Layer ของผู้รับ หลังจากนั้นข้อมูลที่รับเข้ามาก็จะถูกส่งขึ้นไปยัง Link Layer โดย Link Layer จะถอดข้อมูลที่ Link Layer ของฝั่งส่งแปะมาในส่วนหัวของข้อมูลเพื่อใช้งานต่อไป โดยข้อมูลที่โดนถอดข้อมูลของ Link Layer ออกแล้วนั้นก็จะกลายเป็นดาต้าแกรมเพื่อส่งไปยัง Network Layer หลังจากนั้น Network Layer ก็จะถอดข้อมูลของ Network Layer ออกจากดาต้าแกรมเช่นเดียวกัน ดังนั้นข้อมูลดังกล่าวก็จะกลายเป็นเซกเมนต์เพื่อส่งขึ้นไปยัง Transport Layer หลังจากนั้นในทำนองเดียวกันเซกเมนต์ก็จะถูกถอดเอาข้อมูลของ Transport Layer ที่ผู้ส่งเพิ่มเข้ามาออกเพื่อนำมาใช้งาน และส่งข้อมูลที่ถูกถอดข้อมูลของชั้นตัวเองออกเรียบร้อยแล้วหรือที่เรียกว่าเมสเสจนั้นไปให้แอปพลิเคชันที่ถูกต้องใน Application Layer ต่อไป

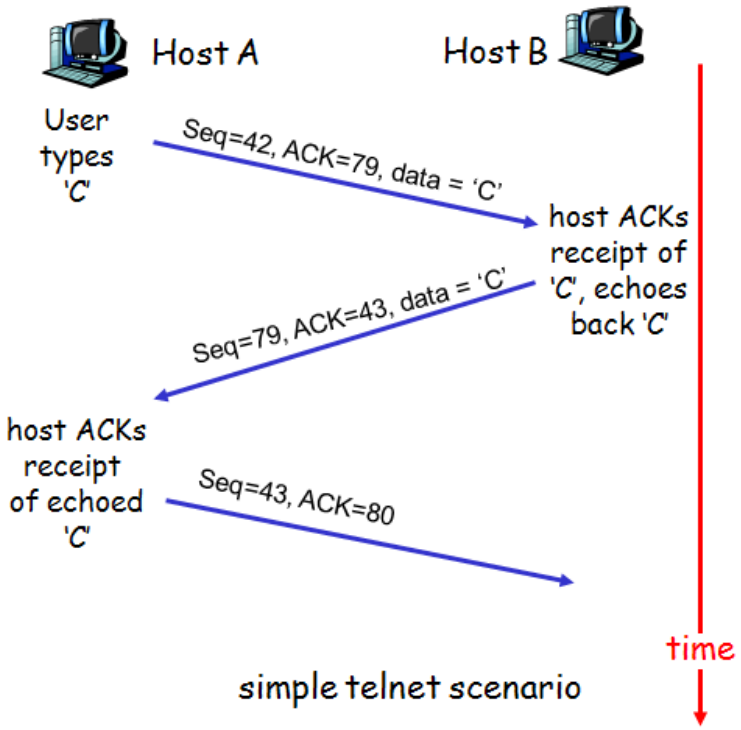
2.3 Transport Layer

เป็นระดับชั้นหนึ่งที่อยู่ในสถาปัตยกรรมสำหรับเครือข่ายอินเตอร์เน็ต ทำหน้าที่ให้เป็นผู้สร้างช่องทางการสื่อสารเชิงตรรกะ (Logical Communication) ระหว่างโปรเซสของแอปพลิเคชัน (Application Processes) ที่อยู่ต่างโฮสกันบนเครือข่ายอินเตอร์เน็ต หรือที่เรียกว่าการสื่อสารระหว่างระหว่างโปรเซส (Process-to-Process Communication) กล่าวคือ Network Layer จะรับผิดชอบในการส่งดาต้าแกรมไปยังโฮสปลายทางเท่านั้น (Host-to-Host Communication) โดยไม่สนใจว่าแต่ละดาต้าแกรมที่ส่งออกไปนั้นมีความเกี่ยวข้องหรือสัมพันธ์กัน

อย่างไร แต่การส่งข้อมูลออกไปนั้นยังไม่ถือว่าเป็นการส่งข้อมูลที่สมบูรณ์ ดังนั้นจึงเป็นหน้าที่ของ Transport Layer ที่จะรับดาต้าแกรมดังกล่าวเข้ามา เพื่อนำส่งไปยังโปรเซสที่ถูกต้องต่อไป (Process-to-Process Communication)

2.4 Transmission Control Protocol (TCP)

เป็นหนึ่งโปรโตคอลหลักในชุดโปรโตคอลสำหรับเครือข่ายอินเตอร์เน็ต ซึ่งทำหน้าที่ควบคุมการรับส่งข้อมูลระหว่างโฮส (Host) ต้นทางถึงโฮสปลายทางบนเครือข่าย ดังนั้นจึงมั่นใจได้ว่าข้อมูลที่ส่งด้วย TCP จะไปถึงยังผู้รับได้อย่างถูกต้องและเป็นลำดับ (Reliable Transmission) จึงถือได้ว่า TCP เป็นโปรโตคอลที่รับประกันการส่งข้อมูล ซึ่งการที่ TCP จะสามารถรับประกันการส่งข้อมูลได้นั้น TCP จะต้องกำหนดหมายเลข (Sequence Number) ให้กับแต่ละเซกเมนต์ซึ่งเป็นตัวเลขที่ไม่ซ้ำกันและต่อเนื่องกันไปตามลำดับของเซกเมนต์ที่จะส่ง เพื่อให้ฝั่งรับสามารถรับรู้ลำดับของเซกเมนต์ และสามารถเรียงลำดับของเซกเมนต์ที่รับเข้ามาได้อย่างถูกต้อง หลังจากนั้นเมื่อผู้รับได้รับเซกเมนต์ก็จะทำการตอบกลับ (Acknowledgment) ไปยังผู้ส่ง เพื่อให้ผู้ส่งรับรู้ว่าผู้รับได้รับเซกเมนต์ที่ส่งมาเรียบร้อยแล้ว โดยในการตอบกลับดังกล่าวผู้รับจะระบุว่าตนเองต้องการหรือรอรับเซกเมนต์ลำดับที่เท่าไหร่ (Acknowledgement Number) เป็นลำดับต่อไป ซึ่งการรับส่งข้อมูลสามารถแสดงเป็นรูปได้ดังรูปภาพที่ 2.3



รูปภาพที่ 2.3 ตัวอย่างการส่งเซกเมนต์ของ TCP

แต่อย่างไรก็ตามสำหรับการส่งเซกเมนต์นั้น ถ้าหากดูจาก ฟอร์แมต (Format) ของเซกเมนต์จะมีฟิลด์ (Field) ชื่อ Sequence Number และ Acknowledgment Number ไว้ให้เพื่อใช้สำหรับระบุหมายเลขไบต์ (Byte) ซึ่งไม่ใช่หมายเลขเซกเมนต์ กล่าวคือแต่ละไบต์ของข้อมูลจะถูกกำหนดหมายเลยไว้ ซึ่งในแต่ละเซกเมนต์สามารถประกอบไปด้วยหลาย ๆ ไบต์ ดังนั้นหมายเลขที่จะระบุให้กับฟิลด์ Sequence Number นั้นจะเป็นหมายเลขของไบต์แรกที่อยู่ในเซกเมนต์นั้น และในทางเดียวกันผู้รับก็จะสามารถตอบกลับผู้ส่ง โดยระบุ Acknowledgement Number เป็นหมายเลขของไบต์ที่ต่อจากไบต์สุดท้ายที่ได้รับมาในเซกเมนต์นั้น ซึ่งก็คือหมายเลขไบต์สุดท้ายที่ได้รับบวกเพิ่มอีกหนึ่ง (Cumulative Acknowledgement) เพื่อบอกให้ฝั่งส่งรู้ว่าต้องส่งไบต์ลำดับที่เท่าไหร่เป็นลำดับต่อไป นอกจากนี้การทำงานของ TCP ยังจะแบ่งออกได้เป็นสองส่วนสำคัญด้วยกันดังต่อไปนี้

1. ส่วนควบคุมการไหลของข้อมูล (Flow Control) จะทำหน้าที่เกี่ยวข้องกับการประสานงานระหว่างผู้ส่งกับผู้รับ เพื่อป้องกันไม่ให้ผู้ส่งทำการส่งข้อมูลออกไปจนเกินความสามารถที่ผู้รับจะรับได้

2. ส่วนควบคุมความคับคั่ง (Congestion Control) จะทำหน้าที่เกี่ยวข้องกับการป้องกันไม่ให้ผู้ส่งทำการส่งข้อมูลออกไปสู่เครือข่ายมากเกินไป หรือเกินกว่าขนาดแบนด์วิดธ์ที่มีอยู่

2.5 Round-Trip Time Estimation and Timeout

ในการส่งข้อมูลของ TCP จะมีการตั้งเวลา (Timeout) เพื่อตรวจสอบว่าเซกเมนต์ที่ส่งออกไปนั้นได้รับการตอบกลับจากผู้รับภายในเวลาที่ควรจะเป็นหรือไม่ ถ้าหากไม่ได้รับการตอบกลับมาภายในระยะเวลาดังกล่าวก็จะถือว่าเซกเมนต์ที่ส่งออกไปเกิดสูญหาย และจำเป็นที่จะต้องส่งเซกเมนต์ซ้ำอีกครั้ง สำหรับการตั้งเวลาที่เหมาะสมนั้นจำเป็นที่ผู้ส่งจะต้องวัดระยะเวลาที่ใช้ในการส่งข้อมูล โดยเริ่มนับตั้งแต่เวลาที่ผู้ส่งส่งเซกเมนต์ออกไปจะถึงเวลาที่ผู้ส่งได้รับการตอบกลับสำหรับเซกเมนต์นั้นจากผู้รับ (Round-trip Time หรือ RTT) เพื่อนำมาใช้ในการคำนวณหาระยะเวลาที่เหมาะสมเพื่อนำไปใช้สำหรับตั้งเวลาในการส่งเซกเมนต์ต่อไป

2.6 Flow Control

จากที่กล่าวมาข้างต้น ผู้ส่งจำเป็นต้องส่งข้อมูลไปยังผู้รับในปริมาณที่ผู้รับสามารถรองรับได้หรือนำไปประมวลผลได้ทัน ในการส่งข้อมูลผ่าน TCP นั้นตัวข้อมูลที่รับเข้ามาเพื่อจะส่งออกไปจะถูกแบ่งออกเป็นชิ้นย่อยหรือเซกเมนต์ (segment) ตามขนาดที่เหมาะสมที่

สามารถส่งผ่านไปบนเครือข่ายได้อย่างมีประสิทธิภาพ สำหรับการควบคุมการไหลของข้อมูลนั้นผู้ส่งจะมีตัวแปรสำหรับเก็บค่าว่าผู้รับมีพื้นที่ว่างของหน่วยความจำหรือขนาดของบัฟเฟอร์ (Buffer) สำหรับเก็บข้อมูลมากแค่ไหน เพื่อที่ผู้ส่งจะสามารถส่งข้อมูลออกไปให้พอดีกับปริมาณที่ผู้รับสามารถรองรับได้ โดยอัลกอริทึมที่ใช้ในการควบคุมการไหลของข้อมูลที่ใช้กันในปัจจุบันเรียกว่าอัลกอริทึมหน้าต่างบานเลื่อน (Sliding Window) กล่าวคือผู้ส่งและผู้รับจะมีหน้าต่าง (Window) ของตัวเองเพื่อเก็บสถานะของการส่งและการอ่านข้อมูลตามลำดับ เพื่อให้ทั้งสองฝั่งประสานการส่งและรับข้อมูลกันได้อย่างถูกต้อง และไม่เกินความสามารถที่ผู้รับจะรับได้

จากการที่แต่ละด้านของการส่งข้อมูลจะมีหน้าต่างของตัวเอง ด้วยสาเหตุนี้ฝั่งผู้รับก็จะมีตัวแปรสำคัญต่าง ๆ ซึ่งใช้เก็บสถานะของการรับข้อมูลดังนี้ต่อไปนี้

RcvBuffer – สำหรับเก็บขนาดของบัฟเฟอร์ที่ผู้รับสามารถรองรับได้

LastByteRead – สำหรับเก็บค่าหมายเลขไบต์ล่าสุดที่ถูกอ่านออกไปจากบัฟเฟอร์เพื่อส่งให้แอปพลิเคชัน

LastByteRcvd – สำหรับเก็บค่าหมายเลยไบต์ล่าสุดที่รับเข้ามาและถูกนำไปเก็บไว้ในบัฟเฟอร์แล้ว

ดังนั้นขนาดของหน้าต่างในฝั่งรับแทนด้วยตัวแปร rwnd จะเท่ากับ RcvBuffer – [LastByteRcvd - LastByteRead] ซึ่งจะเป็นขนาดของบัฟเฟอร์ที่ยังว่างอยู่ ฝั่งรับจะส่งค่าดังกล่าวนี้ไปยังผู้ส่งทุกครั้งที่มีการตอบกลับโดยระบุค่าไปกับฟิลด์ Window Size

สำหรับฝั่งส่งก็จะมีการเก็บค่าตัวแปรต่าง ๆ ในลักษณะเดียวกันเพื่อเก็บสถานะของการส่งข้อมูลดังนี้คือ LastByteSent สำหรับเก็บค่าหมายเลขไบต์ล่าสุดที่ส่งออกไป และ LastByteAcked สำหรับเก็บค่าหมายเลยไบต์ล่าสุดที่ได้รับการตอบกลับจากผู้ส่ง ดังนั้นจำนวนของไบต์ที่ส่งออกไปแล้ว และยังไม่ได้รับการตอบกลับจากผู้รับก็จะเท่ากับค่าของ LastByteSent หักลบด้วยค่าของ LastByteAcked ซึ่งถ้าหากผู้ส่งไม่ต้องการส่งข้อมูลให้เกินความสามารถของผู้รับ ก็จำเป็นจะต้องรักษาสถานะไม่ให้จำนวนของไบต์ที่ยังไม่ได้รับการตอบกลับดังกล่าวเกินกว่าค่า Window Size ที่ผู้รับแจ้งมา

2.7 Congestion Control

การควบคุมความคับคั่งเป็นการควบคุมการส่งข้อมูลไม่ให้ผู้ส่งปล่อยหรือส่งข้อมูลออกไปมากกว่าที่เครือข่ายหรือเส้นทางในการส่งข้อมูลจะรับได้ กล่าวคืออัตราความเร็วของการส่งข้อมูลจะต้องไม่เกินปริมาณแบนด์วิดธ์ที่มีอยู่ สำหรับการควบคุมความคับคั่งนั้นจะกระทำในฝั่งผู้ส่ง โดยที่ฝั่งส่งจะมีตัวแปรเพิ่มขึ้นมาเพื่อเก็บจำนวนของไบต์ (Congestion Window หรือ cwnd) ที่ผู้ส่งสามารถส่งออกไปได้ภายในแต่ละ RTT ซึ่งค่าของ cwnd สามารถทำให้รู้ได้ว่าขนาดของแบนด์วิดธ์โดยประมาณมีค่าเท่าไหร่ (cwnd/RTT ไบต์ต่อวินาที) ดังนั้นในการส่งข้อมูลของผู้ส่งจะต้องรักษาสถานะไม่ให้จำนวนไบต์ที่ยังไม่ได้รับการตอบกลับจากผู้รับมีค่าเกินกว่าค่าที่น้อยที่สุดระหว่าง cwnd และ rwnd (min{cwnd, rwnd}) นอกจากนี้ผู้ส่งจำเป็นต้องมีการปรับค่า cwnd ให้เหมาะสมกับขนาดของแบนด์วิดธ์ที่มีเหลืออยู่ในแต่ละช่วงเวลาที่ส่งข้อมูล จะเห็นว่าเป็นสิ่งที่ค่อนข้างยากที่จะปรับค่า cwnd ให้เหมาะสมในการส่งข้อมูล ถ้าหากผู้ส่งส่งข้อมูลออกไปเร็วเกินไปก็จะทำให้เกิดความคับคั่งขึ้นบนเส้นทางการสื่อสาร และส่งผลให้ข้อมูลเกิดการสูญหายตามมา และในทางกลับกันถ้าหากผู้ส่งส่งข้องมูลช้าเกินไปก็จะทำให้ไม่สามารถใช้ประโยชน์จากแบนด์วิดธ์ได้อย่างเต็มที่ โดยปกติแล้ว TCP จะตรวจสอบความคับคั่งบนเส้นทางการสื่อสาร โดยดูว่าเซกเมนต์ที่ส่งไปนั้นเกิดการสูญหายหรือไม่ ถ้าหากมีการสูญหายเกิดขึ้นผู้ส่งจำเป็นที่จะต้องลดค่า cwnd ลง แต่เมื่อใดก็ตามที่ผู้ส่งได้รับการตอบกลับจากผู้รับสำหรับการส่งเซกเมนต์ใด ๆ ก็จะหมายถึงว่าเซกเมนต์นั้นได้ไปถึงผู้รับเรียบร้อยแล้วและผู้ส่งสามารถที่จะเพิ่มค่า cwnd ได้

2.8 Packet Switching

สำหรับวิธีการส่งข้อมูลภายในเครือข่ายคอมพิวเตอร์ ซึ่งประกอบไปด้วยโหนด (Node) ต่าง ๆ ที่ถูกเชื่อมต่อกันอยู่มากมายจะมี 2 วิธีด้วยกันคือ

1. Circuit Switching – เป็นกระบวนการที่ใช้วิธีการจัดสรรหรือจองทรัพยากรที่จำเป็นต้องใช้ในการส่งข้อมูลไว้ก่อนที่จะส่งข้อมูลจริง ไม่ว่าจะเป็นบัฟเฟอร์ เส้นทางการส่งข้อมูล และอัตราความเร็วของการส่งข้อมูล เป็นต้น โดยทรัพยากรที่ถูกจองไว้จะใช้ได้แค่การเชื่อมต่อหนึ่งการเชื่อมต่อเท่านั้น หลังจากนั้นถึงจะทำการส่งข้อมูล ซึ่งข้อมูลทั้งหมดจะไม่โดนหั่นแต่จะโดนส่งออกไปทั้งหมดบนเส้นทางที่ได้จองไว้ดังกล่าว

2. Packet Switching – เป็นกระบวนการที่การส่งข้อมูลจะเกิดขึ้นโดยไม่มีการจองทรัพยากรไว้ก่อนล่วงหน้า โดยข้อมูลหรือเมสเสจที่รับเข้ามาจากชั้นการทำงานด้านบนที่จะส่งออกจะต้องถูกนำมาแบ่งเป็นชิ้นย่อย ๆ หรือแพ็กเก็ต (Packet) ก่อนถึงจะส่งออกไปทีละแพ็กเก็ต และฝั่งรับก็จะรับเข้ามาทีละแพ็กเก็ต แล้วค่อยนำมาประกอบต่อเข้าด้วยกันเป็นเมสเสจอย่างเดิมก่อนที่จะส่งให้กับชั้นการทำงานด้านบน เนื่องจากไม่มีการจองทรัพยากรไว้ล่วงหน้า ดังนั้นแต่ละอุปกรณ์เชื่อมต่อบนเครือข่ายจำเป็นจะต้องมีกลไกในการค้นหาเส้นทางที่จะใช้ส่งข้อมูลอีกด้วย

ถึงแม้ว่าวิธีการส่งข้อมูลทั้ง 2 วิธีการดังกล่าวข้างต้นมีการใช้งานอย่างแพร่หลายในเครือข่ายสำหรับการส่งข้อมูลทางไกล แต่แนวโน้มความนิยมจะเป็นวิธี Packet Switching เนื่องจากเป็นวิธีการส่งข้อมูลที่มีประสิทธิภาพมากกว่า โดยการส่งข้อมูลด้วยวิธี Packet switching นั้นจะเป็นการส่งข้อมูลในลักษณะของ Store-and-forward กล่าวคือก่อนที่ผู้ส่งจะส่งข้อมูลออกไปได้นั้นจำเป็นจะต้องรอให้ทุกบิต (Bit) ของแพ็กเก็ตเข้ามาครบเรียบร้อยก่อน ถึงจะสามารถส่งบิตแรกของแพ็กเก็ตออกไปได้ ซึ่งจะก่อให้เกิดความล่าช้า (store-and-foreward delay) ได้ นอกจากนี้แพ็กเก็ตที่วิ่งเข้ามาอาจจะต้องถูกนำไปเก็บไว้ในบัฟเฟอร์หรือคิว (queue) เพื่อรอการส่งออกหากไม่สามารถส่งแพ็กเก็ตออกไปได้ทันทีเนื่องจากมีความคับคั่งบนช่องทางการสื่อสารเกิดขึ้น ดังนั้นจึงก่อให้เกิดความล่าช้า (queuing delay) ได้เช่นกัน นอกจากนี้ความล่าช้าอาจเกิดจากที่ขนาดของคิวที่ใช้เก็บแพ็กเก็ตที่รอการส่งออกนั้นมีขนาดที่จำกัด ดังนั้นเมื่อมีจำนวนแพ็กเก็ตวิ่งเข้ามามากเกินกว่าที่จะเก็บเข้าคิวได้หมดก็จะทำให้แพ็กเก็ตที่เกินมานั้นถูกโยนทิ้ง หรือเกิดการสูญหายของแพ็กเก็ต (Packet Loss) อีกด้วย

2.9 Delays in Packet-Switched Networks

1. Processing Delay คือความล่าช้าในการประมวลผลของผู้ส่งและผู้รับเช่น เวลาที่ใช้ในการตรวจสอบส่วนหัวของแพ็กเก็ตเพื่อนำมาหาว่าจะส่งแพ็กเก็ตออกไปทิศทางใด เป็นต้น

2. Queuing Delay คือความล่าช้าหรือระยะเวลาที่แพ็กเก็ตถูกนำไปพักรอไว้ในบัฟเฟอร์ก่อนที่จะถูกส่งออก ความล่าช้าดังกล่าวจะมีมากขึ้นหากมีแพ็กเก็ตก่อนหน้ากำลังรออยู่ในบัฟเฟอร์เพื่อรอการส่งออกเช่นเดียวกันเป็นจำนวนมาก ดังนั้น Queuing Delay จึงเป็นตัวแปรหนึ่งที่สามารถนำมาใช้ในการตรวจสอบความคับคั่งของเส้นทางการสื่อสารได้

3. Transmission Delay คือความล่าช้าในการส่งบิตทั้งหมดในแพ็กเก็ตลงไปยังช่องทางการสื่อสารหรือลิงค์ (Link) โดยความล่าช้าจะขึ้นอยู่กับขนาดของแบนด์วิดธ์ที่ใช้ส่งข้อมูล

4. Propagation Delay คือระยะเวลาที่ใช้ในการเดินทางของแพ็กเก็ตจากต้นทางไปยังปลายทาง โดยความเร็วจะขึ้นอยู่กับความยาวของสื่อที่ใช้ในการส่งข้อมูล

2.10 Bandwidth Estimation Techniques

1. Active Measurement เป็นการหาค่าแบนด์วิดธ์โดยการส่งกลุ่มของแพ็กเก็ตออกไปบนเส้นทางการสื่อสารเพื่อตรวจสอบแบนด์วิดธ์ โดยวิธีนี้อาจจำเป็นที่ผู้ส่งและผู้รับต้องประสานการทำงานร่วมกัน หรือใช้โปรโตคอลเดียวกัน นอกจากนี้กลุ่มของแพ็กเก็ตที่ใช้อาจจะมีจำนวนมากและอาศัยการส่งติดต่อกันเป็นเวลานาน ซึ่งอาจไปกระทบกับการส่งข้อมูลของแอปพลิเคชันอื่นได้อีกด้วย โดยสามารถแบ่งออกได้เป็นเทคนิคย่อยดังนี้

1.1 Probe Gap Model (PGM) เป็นการส่งคู่แพ็กเก็ต (Packet Pair) หรือเป็นกลุ่มของคู่แพ็กเก็ตออกไปบนเส้นทางการสื่อสาร และเมื่อแพ็กเก็ตดังกล่าวไปถึงผู้รับก็จะทำการวัดค่าแบนด์วิดธ์ โดยแนวคิดหลักคือดูการกระจายของแพ็กเก็ต โดยผู้รับจะดูระยะเวลาว่าแพ็กเก็ตปัจจุบันกับแพ็กเก็ตก่อนหน้ามาถึงปลายทางห่างกันมากน้อยแค่ไหน โดยระยะห่างที่มากขึ้นจะบ่งบอกว่าแพ็กเก็ตดังกล่าวถูกแทรกด้วยแพ็กเก็ตของแอปพลิเคชันอื่น ซึ่งสะท้อนให้เห็นถึงความคับคั่งของเส้นทางการสื่อสาร เครื่องมือในการวัดแบนด์วิดธ์ที่ใช้เทคนิคดังกล่าวนี้ได้แก่ Spruce [19] และ IGI [15] เป็นต้น

1.2 Probe Rate Mode (PRM) เป็นการส่งกลุ่มของแพ็กเก็ตออกไปด้วยอัตราความเร็วที่ต่ำกว่าแบนด์วิดธ์ที่มีอยู่ ดังนั้นแพ็กเก็ตดังกล่าวจะต้องไปถึงผู้รับด้วยอัตราความเร็วที่สอดคล้องกับอัตราการส่งที่ฝั่งส่ง ซึ่งหมายความว่าไม่เกิดความล่าช้าขึ้น แต่ถ้าแพ็กเก็ตถูกส่งออกไปด้วยความเร็วที่สูงกว่าแบนด์วิดธ์ที่มีอยู่ก็จะทำให้แพ็กเก็ตดังกล่าวไปค้างอยู่ในบัฟเฟอร์ทำให้เกิด Queuing Delay ดังนั้นผู้รับก็สามารถวัดค่าแบนด์วิดธ์ที่เหมาะสมในการส่งข้อมูลได้จาก Queuing Delay ดังกล่าว ซึ่งเครื่องมือที่ใช้เทคนิคนี้ได้แก่ TOPP, PathLoad [M. Jain and C. Dovrolis, 2002] และ pathChirp [10] เป็นต้น

2. Passive Measurement เป็นการหาค่าแบนด์วิดธ์โดยอาศัยข้อมูลที่ได้จากการส่งข้อมูลที่ผ่านมาของการเชื่อมต่อนั้น ๆ เช่น RTT หรือระยะเวลาในการรับ Acknowledgement เป็นต้น ดังนั้นวิธีนี้จึงค่อนข้างรวดเร็วและประหยัด แต่อย่างไรก็ตามการวัดแบนด์วิดธ์จะทำได้ก็ต่อเมื่อมีการส่งข้อมูลมาแล้วก่อนหน้านี้ แต่อย่างไรก็ตามก็ยังมีปัจจัยอื่นที่ส่งผลต่อความแม่นยำ เช่นอัลกอริทึมในการควบคุมความคับคั่ง ขนาดของบัฟเฟอร์ และการแข่งขันกันส่งข้อมูลของแต่ละการเชื่อมต่อ เป็นต้น

2.11 งานวิจัยที่เกี่ยวข้อง

RAPID [11] เป็นโปรโตคอลที่พัฒนาขึ้นมาโดยมีจุดประสงค์เพื่อนำไปใช้งานบนเครือข่ายอินเตอร์เน็ตความเร็วสูง ซึ่งจุดเด่นของ RAPID คือ ลดระยะเวลาในการค้นหาแบนด์วิดธ์โดยใช้เวลาไม่กี่ RTT ก็สามารถรู้ถึงขนาดของแบนด์วิดธ์ที่มีอยู่ได้อย่างถูกต้อง ซึ่งในการค้นหาแบนด์วิดธ์นั้น RAPID ได้นำอัลกอริทึมของ pathChirp [10] มาใช้ซึ่งถือได้ว่าเป็นเครื่องมือในการวัดแบนด์วิดธ์ที่ได้รับการยอมรับในวงการ โดยอัลกอริทึมของ RAPID ที่อธิบายไว้ใน [12] นั้นจะต้องอาศัยการส่งข้อมูลเป็นลักษณะกลุ่มของแพ็กเก็ต (ที่มีขนาด N แพ็กเก็ต) ดังที่แสดงอยู่ในรูปภาพที่ 2.4 ซึ่งเรียกว่า multi-rate probe stream (p-stream) ซึ่งในแต่ละแพ็กเก็ตจะถูกส่งออกไปที่อัตราความเร็วในการส่งข้อมูลของแพ็กเก็ตก่อนหน้า โดยสมมุติฐานที่ว่าแพ็กเก็ตก่อนหน้าถูกส่งเสร็จ ดังนั้นแพ็กเก็ตปัจจุบันกับแพ็กเก็ตก่อนหน้าจะถูกส่งเป็นเวลาห่างกันเท่ากับขนาดของแพ็กเก็ตหารด้วยความเร็วในการส่งข้อมูลของแพ็กเก็ตก่อนหน้า จากการส่งข้อมูลดังกล่าวจะเห็นได้ว่าแต่ละแพ็กเก็ตจะต้องถูกกำหนดอัตราความเร็วในการส่งข้อมูลไว้เพื่อนำไปใช้ในการตั้งเวลาสำหรับการส่งข้อมูลแต่ละแพ็กเก็ตใน p-stream

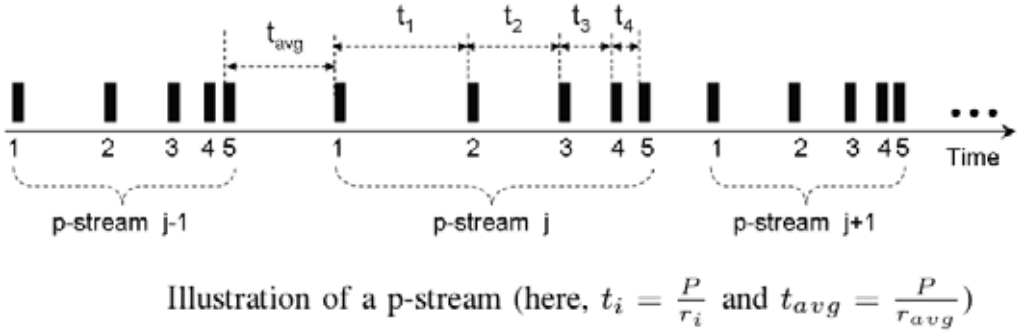


Illustration of a p-stream (here, $t_i = \frac{P}{r_i}$ and $t_{avg} = \frac{P}{r_{avg}}$)

รูปภาพที่ 2.4 ตัวอย่าง p-stream ของโปรโตคอล RAPID

ในการกำหนดอัตราการส่งข้อมูลของแต่ละแพ็กเก็ตใน p-stream นั้นจะถูกควบคุมด้วยอัตราการส่งข้อมูลหรือแบนด์วิดธ์เฉลี่ยที่กำหนดไว้สำหรับแต่ละ p-stream ซึ่งแต่ละ p-stream จะส่งออกไปเพื่อสามารถค้นหาช่วงของแบนด์วิดธ์ที่ต่ำกว่าอัตราแบนด์วิดธ์เฉลี่ย และช่วงของแบนด์วิดธ์ที่สูงกว่าแบนด์วิดธ์เฉลี่ยในคราวเดียวกัน ซึ่งอัตราการส่งเฉลี่ยของทั้ง p-stream ก็จะเท่ากับอัตราการส่งเฉลี่ยดังกล่าว โดยสามารถแสดงการคำนวณหาอัตราการส่งข้อมูลเฉลี่ย (แทนด้วย Avg. Rate) ได้ดังสมการที่ (2.1) ต่อไปนี้

$$Avg.Rate = \frac{N-1}{\frac{1}{Rate_1}+\frac{1}{Rate_2}+\frac{1}{Rate_{N-1}}}, i>1, Rate_i > Rate_{i-1} \tag{2.1}$$

ดังนั้นในการสร้าง p-stream สามารถทำได้โดยนำขนาดแบนด์วิดธ์เฉลี่ยมากระจายเป็นช่วงของแบนด์วิดธ์ตามจำนวนของแพ็กเก็ตใน p-stream ซึ่งจำเป็นจะต้องใช้พารามิเตอร์ (Parameter) สำหรับให้น้ำหนักในการกระจายค่าแบนด์วิดธ์ให้กับแต่ละแพ็กเก็ต โดยเรียกพารามิเตอร์ดังกล่าวว่า spread factor ในการคำนวณหาอัตราการส่งข้อมูลสำหรับแต่ละแพ็กเก็ตสามารถทำได้โดยนำค่าแบนด์วิดธ์เฉลี่ย (แทนด้วย Avg.Rate) จำนวนแพ็กเก็ตของ p-stream (แทนด้วย N) และค่า spread factor (แทนด้วย m) มาเข้าสูตรการคำนวณดังสมการที่ (2.2) และ (2.3) ดังต่อไปนี้

$$Rate_1 = \frac{m^{N-1}-1}{(N-1)(m-1)m^{N-2}} Avg.Rate \tag{2.2}$$

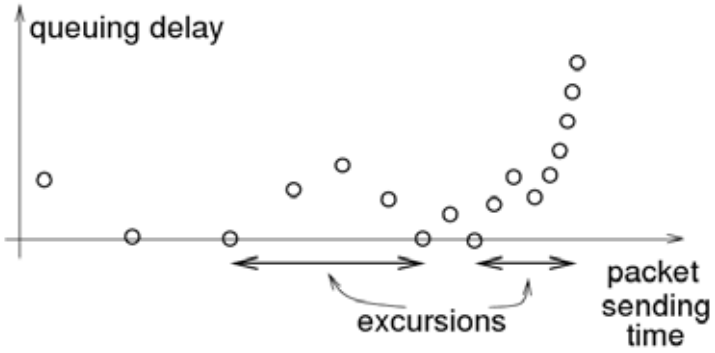
$$Rate_i = m^{i-1} \times Rate_1 \tag{2.3}$$

ยกตัวอย่างเช่น กำหนดให้แบนด์วิดธ์เฉลี่ยมีค่าเท่ากับ 100Mbps ขนาดของ p-stream เท่ากับ 10 แพ็กเก็ต และ spread factor มีค่าเท่ากับ 1.2 ก็จะได้ p-stream ที่มีอัตราการส่งข้อมูลของแต่

ละแพ็กเก็ตดังนี้คือ {100 Mbps, 53.75 Mbps, 64.50 Mbps, 77.39 Mbps, 92.87 Mbps, 111.49 Mbps, 133.74 Mbps, 160.49 Mbps, 192.58 Mbps, และ 231.10 Mbps} จะสังเกตได้ว่าอัตราการส่งของแพ็กเก็ตแรกจะเท่ากับอัตราการส่งข้อมูลเฉลี่ย เนื่องจากเป็นแพ็กเก็ตที่ต่อจากแพ็กเก็ตสุดท้ายของ p-stream ก่อนหน้า

แต่อย่างไรก็ตามในการคำนวณหาอัตราการส่งข้อมูลสำหรับแพ็กเก็ตแรกจะแตกต่างออกไปหากโปรโตคอลทำงานอยู่ในช่วงเริ่มต้นของการทำงาน (Slow Start) ดังนี้คืออัตราการส่งข้อมูลของแพ็กเก็ตแรกจะเท่ากับอัตราการส่งข้อมูลเฉลี่ยหรือค่าแบนด์วิดธ์ที่คำนวณได้จาก p-stream ก่อนหน้า หากค่าแบนด์วิดธ์ที่คำนวณได้มีค่ามากกว่าอัตราการส่งข้อมูลของแพ็กเก็ตสุดท้ายของ p-stream ก่อนหน้าพร้อมทั้งเพิ่มค่าขนาดของ p-stream เป็น 2 เท่า แต่ถ้าค่าแบนด์วิดธ์ที่คำนวณได้น้อยกว่าก็จะหลุดออกจากช่วง Slow Start และเข้าสู่ช่วงป้องกันความคับคั่ง หรือ Congestion Avoidance ซึ่งขนาดของ p-stream จะคงที่ แต่ยังใช้การส่งแบบเดิมเพื่อหาแบนด์วิดธ์ที่มีอยู่บนช่องทางการสื่อสารต่อไป หลังจากที่ได้ p-stream เรียบร้อยแล้วผู้ส่งก็จะตั้งเวลาสำหรับส่งข้อมูลแต่ละแพ็กเก็ตใน p-stream ออกไปสู่ช่องทางการสื่อสารตามลำดับ

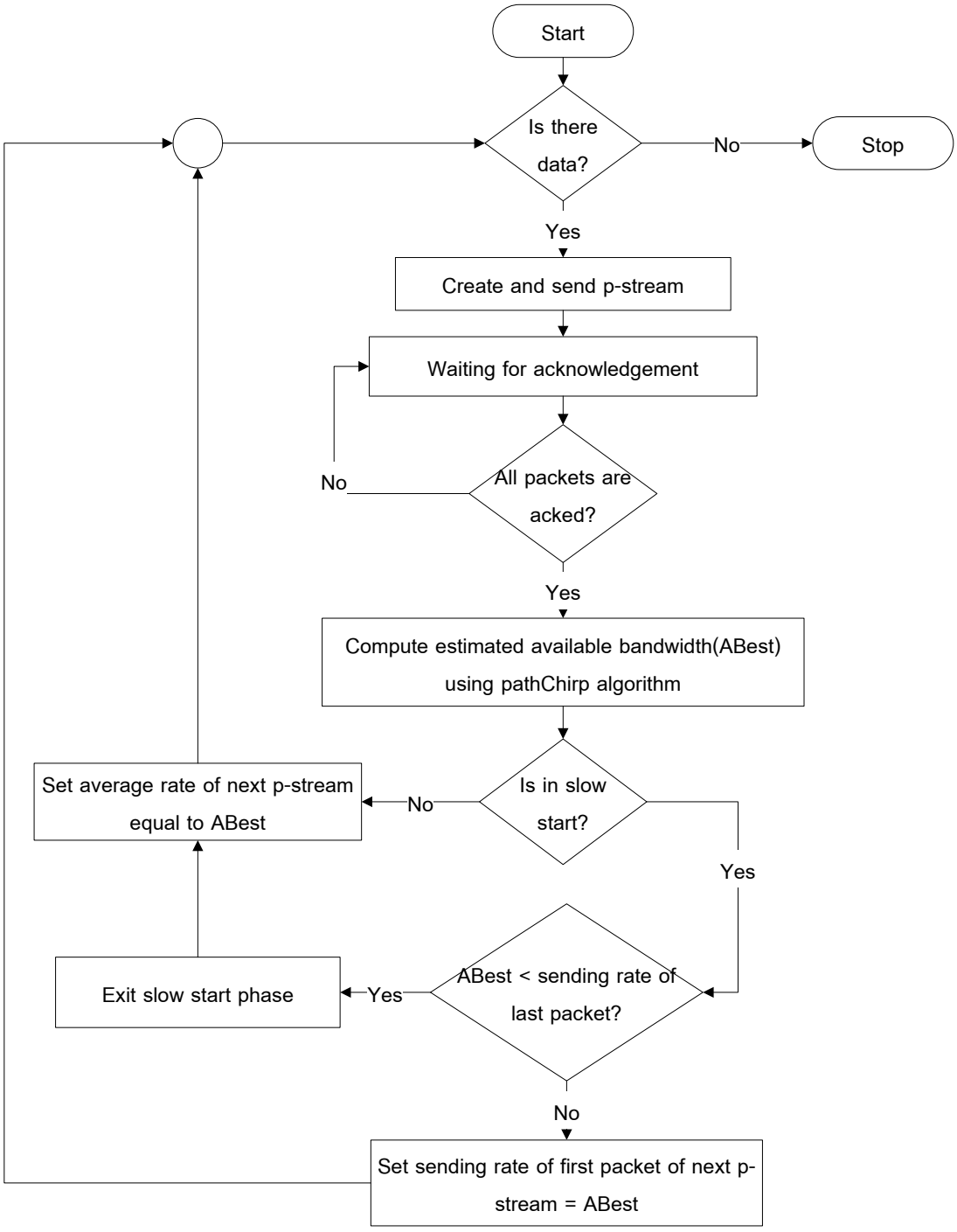
หลังจากนั้นเมื่อผู้รับได้รับแต่ละแพ็กเก็ตก็จะบันทึกระยะเวลาที่แต่ละแพ็กเก็ตนั้นใช้ในการเดินทางมายังผู้รับ เมื่อทุกแพ็กเก็ตใน p-stream มาครบแล้วก็จะคำนวณหาค่าแบนด์วิดธ์โดยดูว่าเกิด Queuing Delay ขณะส่งแพ็กเก็ตของ p-stream หรือไม่ ซึ่งความล่าช้าดังกล่าวสามารถบ่งบอกได้ว่ามีความคับคั่งเกิดขึ้นบนช่องทางการสื่อสาร



รูปภาพที่ 2.5 ตัวอย่างการเกิด Queuing Delay ใน p-stream

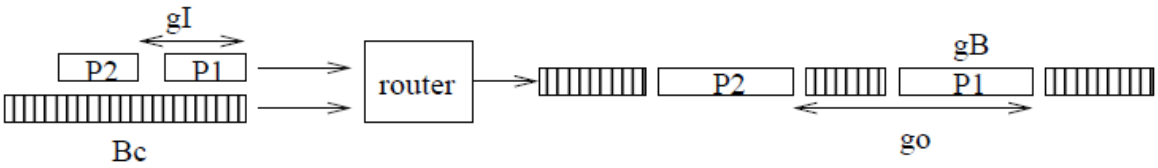
อีกหนึ่งสาเหตุที่ RAPID มีการส่งข้อมูลเป็นลักษณะกลุ่มของแพ็กเก็ต เนื่องจากไม่สามารถนำ Queuing Delay ของแต่ละแพ็กเก็ตเดี่ยว ๆ มาใช้ในการคำนวณได้ เนื่องจากความล่าช้าอาจจะเกิดขึ้นด้วยสาเหตุอื่นได้ในขณะที่ส่งข้อมูล ดังนั้นจึงต้องดูแนวโน้มของการเกิด Queuing Delay ว่า

เกิดความล่าช้าติดต่อ ๆ กันหลายแพ็กเก็ตหรือไม่ ซึ่งทำให้มั่นใจได้ว่าเป็นความล้าช้าที่เกิดจากความคับคั่งบนช่องทางการสื่อสารที่เพิ่มมากขึ้นจริง หลังจากนั้นผู้รับจึงรู้ได้ว่าความเร็วใดที่จะสามารถส่งข้อมูลได้โดยไม่ก่อให้เกิดความล่าช้า ซึ่งสะท้อนให้เห็นถึงขนาดของแบนด์วิดธ์ที่เหลืออยู่นั่นเอง สำหรับแผนผังการทำงานของโปรโตคอล RAPID สามารถดูได้ที่รูปภาพที่ 2.6



รูปภาพที่ 2.6 แผนผังการทำงานของ RAPID

Initial Gap Increasing (IGI) [15] เป็นเทคนิคที่ใช้ในการวัดแบนด์วิดธ์โดยการส่งกลุ่มของแพ็กเก็ตติดต่อกันออกไปเพื่อดูการกระจายตัวของแพ็กเก็ต (gap distribution) ดังแสดงอยู่ในรูปภาพที่ 2.7 ซึ่งช่องว่างระหว่างแพ็กเก็ตสามารถวัดได้เมื่อแพ็กเก็ตไปถึงผู้รับ ซึ่งจะเป็นตัวสะท้อนขนาดแบนด์วิดธ์ที่ถูกใช้งานอยู่จากแอปพลิเคชันอื่น (Competing Bandwidth) นอกจากนี้ IGI จะมีอัลกอริทึมในการหาค่าแบนด์วิดธ์ของช่องทางการสื่อสารในส่วนที่เป็นคอขวด (Bottleneck Bandwidth) ดังนั้นเมื่อได้ค่าแบนด์วิดธ์ดังกล่าวแล้วก็สามารถหาค่าแบนด์วิดธ์ที่เหลืออยู่ได้โดยการหักลบกัน



รูปภาพที่ 2.7 จำลองการจราจรบนเครือข่ายและแพ็กเก็ตที่ใช้ในการหาแบนด์วิดธ์ของ IGI

บทที่ 3
โปรโตคอลที่นำเสนอ

เนื่องจากเทคนิคในการหาค่าแบนด์วิดธ์ที่ใช้ใน RAPID คือการดูแนวโน้มของการเกิด Queuing Delay ซึ่งสามารถเกิดขึ้นได้ง่ายเมื่อมีการแย่งกันใช้งานแบนด์วิดธ์บนช่องทางการสื่อสาร ดังนั้นจึงเป็นสาเหตุหลักทำให้ RAPID ค่อนข้างอ่อนไหวต่อการเกิด Queuing Delay มากเกินไป จนทำให้ไม่สามารถทนทานต่อการแข่งขันกันเข้าใช้งานแบนด์วิดธ์บนเครือข่าย ถึงแม้ว่ามีการปรับค่าพารามิเตอร์ (Parameter) ที่เกี่ยวกับการวัดแบนด์วิดธ์แล้วก็ตาม ทำให้ RAPID ลดอัตราการส่งข้อมูลลงเมื่อเกิด Queuing Delay ค่อนข้างเร็ว ถึงแม้ว่าบนช่องทางการสื่อสารยังเกิดความคับคั่งไม่ถึงระดับที่ควรจะลดอัตราการส่งข้อมูลก็ตาม ทั้งนี้เนื่องจากโปรโตคอลบางตัวไม่สามารถตรวจจับความคับคั่งได้เร็วอย่างที่ควรหรือรอจนกว่าจะเกิดการสูญหายของแพ็กเก็ตถึงจะลดปริมาณการส่งข้อมูลลง ทำให้โปรโตคอลดังกล่าวส่งข้อมูลออกไปและไปค้างอยู่ในบัฟเฟอร์เป็นปริมาณค่อนข้างมาก ซึ่งเป็นสาเหตุให้เกิด Queuing Delay ตามมา ดังนั้นจึงจำเป็นจะต้องหาเทคนิคหรือตัวกรองบางอย่างเพื่อป้องกันไม่ให้ RAPID อ่อนไหวจนเกินไป จากการค้นคว้าพบว่าการวัดแบนด์วิดธ์ด้วยเทคนิค Probe Gap Model จะให้ค่าแบนด์วิดธ์ที่ค่อนข้างคงที่กว่า ดังนั้นจึงได้เลือกนำอัลกอริทึมที่มีชื่อว่า Initial Gap Increasing (IGI) ของ IGI/PTR มาเป็นตัวแทนของ Probe Gap Model เพื่อใช้ในการหาค่าแบนด์วิดธ์ร่วมกันกับอัลกอริทึมของ RAPID โดยทำหน้าที่เป็นอัลกอริทึมเสริม ซึ่งสาเหตุที่เลือกนำ IGI มาใช้เนื่องมาจากการส่งข้อมูลของอัลกอริทึม IGI จะเป็นลักษณะเช่นเดียวกันกับ RAPID คือมีการส่งข้อมูลแบบกลุ่มของแพ็กเก็ต ดังนั้นจึงสามารถนำมาประยุกต์ใช้ได้สะดวกมากยิ่งขึ้น นอกจากนี้ได้มีงานวิจัย [16] ที่ทำการทดลองนำ IGI มาทำงานเสริมกับ pathChirp และพบว่าให้ค่าแบนด์วิดธ์ที่ค่อนข้างคงที่กว่าการวัดด้วย pathChirp เพียงลำพัง โดยจะเรียกโปรโตคอลใหม่ที่นำเสนอว่า NewRAPID

3.1 กระบวนการทำงานของโปรโตคอล

การทำงานของโปรโตคอลในส่วนของการส่งและรับแพ็กเก็ตจะเหมือนกับการทำงานของ RAPID แต่จะแตกต่างกันที่การวัดค่าแบนด์วิดธ์ กล่าวคือโปรโตคอลจะใช้อัลกอริทึมเดิมของ RAPID ในการตรวจจับ Queuing Delay ในขณะเดียวกันก็ใช้อัลกอริทึม IGI ในการตรวจจับการกระจายตัวของแพ็กเก็ต ไปพร้อม ๆ กัน โดยสามารถแสดงขั้นตอนการทำงานของโปรโตคอลได้ดังต่อไปนี้ นอกจากนี้สำหรับผังการทำงานของโปรโตคอลที่นำเสนอสามารถดูได้จากรูปภาพที่ 3.1

1. เริ่มต้นจากการที่ผู้ส่งเริ่มส่งกลุ่มของแพ็กเก็ต (p-stream) ไปยังผู้รับ โดยการส่งข้อมูลนั้นจะส่งตามอัลกอริทึมเดิมของ RAPID กล่าวคือแต่ละแพ็กเก็ตจะส่งเป็นลำดับกันไปโดยระยะห่างในการส่งของแต่ละแพ็กเก็ตจะขึ้นอยู่กับแบนด์วิดธ์หรืออัตราความเร็วที่ใช้ในการส่งแพ็กเก็ตก่อนหน้าที่อยู่ในกลุ่มเดียวกัน โดยทุกครั้งที่ส่งแพ็กเก็ตจะต้องบันทึกเวลาส่งไว้เพื่อนำมาใช้ในการคำนวณหาแบนด์วิดธ์เมื่อได้รับ Acknowledgement ของทุกแพ็กเก็ตในกลุ่มเดียวกันครบแล้ว

2. ผู้รับตอบ Acknowledgement กลับมายังผู้ส่ง ผู้ส่งจะบันทึกเวลาที่ผู้รับได้รับแพ็กเก็ตดังกล่าว โดยดูได้จากค่า timestamp ของ Acknowledgement ที่รับเข้ามา ซึ่งจะแปะมาโดยผู้รับขณะส่ง Acknowledgement

3. ผู้รับเมื่อได้รับ Acknowledgement ครบทุกแพ็กเก็ตแล้ว จึงนำข้อมูลที่ได้มาคำนวณหาค่าแบนด์วิดธ์ด้วยอัลกอริทึมของ RAPID เดิม ซึ่งก็คืออัลกอริทึมของ pathChirp มาใช้ในการคำนวณ โดยดูจากแนวโน้มของการเกิด Queuing Delay จากกลุ่มของแพ็กเก็ต

4. หลังจากนั้นเตรียมข้อมูลเพื่อนำไปใช้ในการคำนวณหาค่าแบนด์วิดธ์ด้วยอัลกอริทึมของ IGI กล่าวคือนำข้อมูลเวลาส่งและเวลารับของแต่ละแพ็กเก็ตใน p-stream มาหักลบด้วยระยะห่างในการส่งข้อมูลของแพ็กเก็ตนั้น ๆ กับแพ็กเก็ตก่อนหน้า เพื่อให้ตรงกับสมมุติฐานของ IGI ที่ว่าแต่ละแพ็กเก็ตจะถูกส่งต่อท้ายกันกันไปโดยไม่มีการหยุดรอ ด้วยสาเหตุนี้ทำให้ผลการวัดแบนด์วิดธ์ด้วย IGI อาจจะผิดพลาดไปได้ แต่อย่างไรก็ตามบนเครือข่ายความเร็วสูงนั้นขนาดของระยะห่างในการส่งข้อมูลสำหรับแต่ละแพ็กเก็ตจะค่อนข้างน้อย

5. หลังจากนั้นนำข้อมูลดังกล่าวที่ได้ไปคำนวณหาค่าแบนด์วิดธ์อีกครั้งด้วยอัลกอริทึม IGI โดยดูจากการกระจายตัวของแพ็กเก็ต แต่อย่างไรก็ตามอัลกอริทึม IGI จะถูกนำมาใช้ก็ต่อเมื่อการทำงานของโปรโตคอลหลุดออกจากช่วง Slow Start ไปแล้วเท่านั้น นอกจากนี้อัลกอริทึมของ IGI จะคำนวณค่าแบนด์วิดธ์ของส่วนที่เป็นคอขวด (Bottleneck Link) ในทุกรอบของการคำนวณ ซึ่งพบว่าค่าจะแกว่งไปมาเรื่อย ๆ ซึ่งค่าอาจจะไม่ตรงกับแบนด์วิดธ์ในส่วนที่เป็นคอขวดจริง ๆ ดังนั้นในการพัฒนา NewRAPID จะมีการเก็บค่าสูงสุดของแบนด์วิดธ์ในส่วนคอขวดที่คำนวณได้จากทุก ๆ รอบ แล้วจึงใช้ค่าสูงสุดดังกล่าวในการหาค่าแบนด์วิดธ์ต่อไป

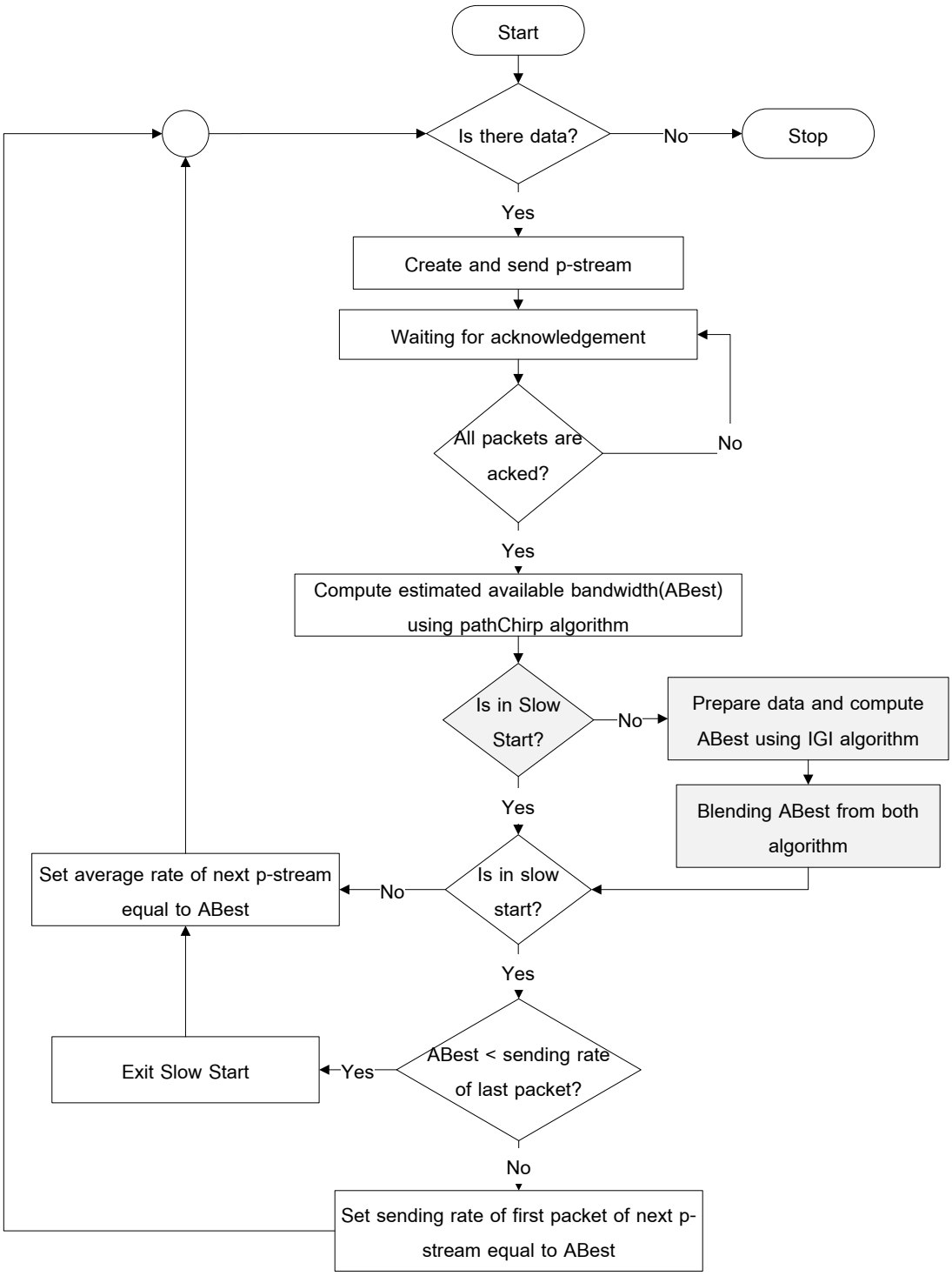
6. เมื่อได้ค่าแบนด์วิดธ์จากทั้งสองอัลกอริทึมมาแล้ว จึงค่อยนำค่าแบนด์วิดธ์ทั้งสองมาคำนวณรวมกันเพื่อให้ได้ค่าแบนด์วิดธ์สุดท้าย โดยการนำค่าแบนด์วิดธ์ที่ได้จากอัลกอริทึมของ RAPID มาคูณด้วยค่าน้ำหนัก (Alpha) ที่ให้กับอัลกอริทึมของ RAPID หลังจากนั้นนำค่า

แบนด์วิดธ์ที่ได้จากอัลกอริทึม IGI มาคูณด้วยค่าน้ำหนักที่คำนวณได้จาก (1 - Alpha) แล้วจึงนำสองค่ามารวมกันเป็นค่าแบนด์วิดธ์สุดท้าย โดยสาเหตุที่ต้องมีการให้น้ำหนักเนื่องจากต้องการให้ผลการคำนวณของ RAPID มีความสำคัญมากกว่า IGI ซึ่งใน [18] นั้นพบว่าเทคนิคของ Probe Gap Model จะให้แม่นความแม่นยำน้อยกว่า Probe Rate Model ซึ่งสมการสำหรับการรวมค่าแบนด์วิดธ์จะเป็นดังที่แสดงในสมการที่ 3.1 คือ

$$AB=\alpha X + (1-\alpha)Y \tag{3.1}$$

จากสมการข้างต้นนั้นตัวแปร α จะแทนค่าน้ำหนักที่ใช้ในการรวมค่าแบนด์วิดธ์ ตัวแปร X จะหมายถึงค่าแบนด์วิดธ์ที่ได้จากอัลกอริทึมของ RAPID และตัวแปร Y จะหมายถึงค่าแบนด์วิดธ์ที่ได้จากอัลกอริทึม IGI เมื่อผ่านสมการนี้ก็จะได้ค่าแบนด์วิดธ์สุดท้ายพร้อมนำไปใช้

7. นำค่าแบนด์วิดธ์ที่ได้นำไปปรับใช้ในการส่งข้อมูลสำหรับ p-stream ถัดไป โดยจะนำไปใช้เป็นค่าแบนด์วิดธ์เฉลี่ยของ p-stream ถัดไปนั่นเอง



รูปภาพที่ 3.1 ผังงานแสดงการทำงานของโปรโตคอลที่นำเสนอ

จากขั้นตอนการทำงานข้างต้นสามารถนำมาเขียนเป็นคำสั่งลำลอง (Pseudocode) ได้ดังรูปภาพที่ 3.2 ดังนี้

```
compute_av_bw (snd_time [N], rcv_time [N], gap [N], alpha)
{
  // compute available bandwidth using RAPID algo
  rapid_bw = rapid_av_bw (snd_time [N], rcv_time [N])
  // normalize data for IGI
  FOR i = 1 to N
    snd_time [i] = snd_time [i] – gap [i]
    rcv_time [i] = rcv_time [i] – gap [i]
  END FOR
  // compute bottleneck banwidth using IGI algo
  b_bw = igi_b_bw (snd_time [N], rcv_time [N])

  // compute competing bandwidth using IGI algo
  c_bw = igi_compete_bw (snd_time [N], rcv_time [N])

  IF b_bw > max_b_bw
    // track maximum bottleneck bandwidth
    max_b_bw = b_bw
  END IF

  // compute available bandwidth using IGI
  igi_bw = max_b_bw – c_bw

  // blending available banwidth
  av_bw = (alpha*rapid_bw)+((1-alpha)*igi_bw)
}
```

รูปภาพที่ 3.2 คำสั่งลำลองสำหรับโปรโตคอลที่นำเสนอ

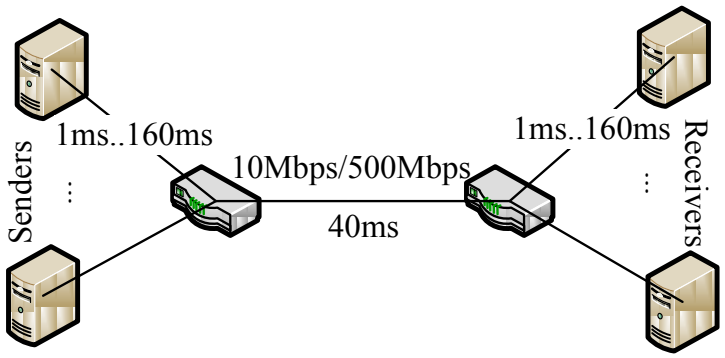
บทที่ 4

การจำลองและวิเคราะห์ผลการจำลอง

จากโปรโตคอลที่ได้นำเสนอข้างต้นนั้นได้ถูกนำไปพัฒนาบน NS-2 เวอรชั่น 2.34 โดยทำการแก้ไขซอร์สโค้ดของโปรโตคอล RAPID เพื่อให้ได้โปรโตคอล NewRAPID ตามที่ได้นำเสนอ และออกแบบจำลองเพื่อทดสอบผลการทำงานของโปรโตคอลดังกล่าว

4.1 โทโพโลยีที่ใช้ในการจำลอง

ในการประเมินผลการทำงานของโปรโตคอลนั้น ได้นำโทโพโลยี (Topology) แบบดัมบ์เบลโทโพโลยี (Dumbbell Topology) ดังที่แสดงในรูปภาพที่ 4.1 มาใช้ในการสร้างแบบจำลอง กล่าวคือโทโพโลยีจะประกอบไปด้วยกลุ่มของโหนด 2 ฝั่งด้วยกันคือฝั่งผู้ส่งและผู้รับ ซึ่งในแต่ละฝั่งจะประกอบไปด้วยโหนดหลาย ๆ โหนดต่อเข้ากับเร้าเตอร์ (Router) ตัวเดียวกัน โดยทั้งสองฝั่งจะถูกเชื่อมโยงเข้าหากันโดยนำทั้งสองเร้าเตอร์มาต่อกัน ซึ่งจะมีลักษณะเหมือนดัมบ์เบลดังรูป



รูปภาพที่ 4.1 ดัมบ์เบลโทโพโลยี

โดยเส้นทางที่เชื่อมเร้าเตอร์เข้าด้วยกันนั้นจะกลายเป็นส่วนหนึ่งของเส้นทางการสื่อสารที่จะต้องแบ่งกันเข้าใช้งานจากทุก ๆ โหนดของทั้งสองฝั่ง ดังนั้นเราจะกำหนดให้ส่วนดังกล่าวเป็นจุดคอขวดของเส้นทางการสื่อสารด้วยการกำหนดให้ขนาดแบนด์วิดธ์บนจุดดังกล่าวต่ำกว่าขนาดของแบนด์วิดธ์ของเส้นทางระหว่างโหนดไปยังเร้าเตอร์ แต่อย่างไรก็ตามเวลาที่ใช้ในการแพร่กระจาย (Propagation Delay) และขนาดของแบนด์วิดธ์ของแต่ละเส้นทางอาจจะมีค่าแตกต่างกันออกไปแล้วแต่รูปแบบของการประเมินผล นอกจากนี้สำหรับแต่ละการทดลองจำเป็นที่จะต้องกำหนดค่าพารามิเตอร์สำหรับแต่ละโปรโตคอล โดยรายละเอียดระบุไว้ในตารางที่ 4.1

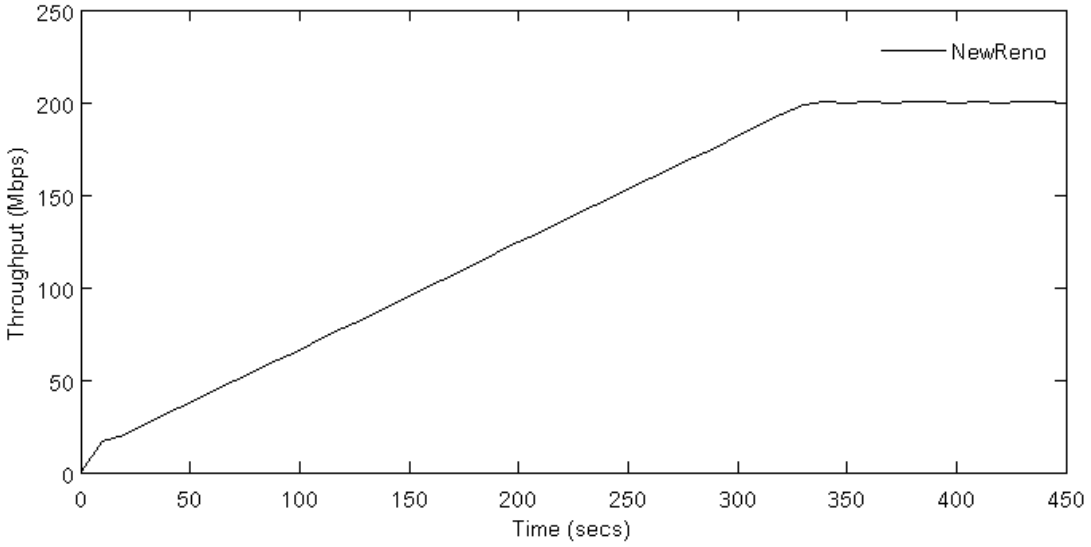
NewReno	packet_size = 1000 Bytes, max_window = 30000
CUBIC	packet_size = 1000 Bytes, max_window = 30000
RAPID	packet_size = 1000 Bytes, initial_p_stream = 5, ss_spread_factor = 2, init_min_rate = 100Kbps, spread_factor = 1.04 (high-speed) and 1.02 (low-speed), p_stream_size = 30 (high-speed) and 12 (low-speed), excursion_threshold = 4
NewRAPID	Same as RAPID for RAPID algorithm, packet_size = 1000 Bytes, hybrid_alpha = 0.8

ตารางที่ 4.1 พารามิเตอร์ของแต่ละโปรโตคอลที่ใช้ในการทดลอง

4.2 Inter-protocol Fairness Evaluation

ในส่วนนี้เป็นการวัดความสามารถของโปรโตคอล NewRAPID หากนำไปทำงานร่วมกันกับโปรโตคอลตัวเก่าที่ถูกติดตั้งใช้งานในเครือข่ายจริง ๆ ในปัจจุบัน ซึ่งหากโปรโตคอลที่นำเสนอมาไม่สามารถทำงานร่วมกันกับโปรโตคอลดังกล่าวได้ก็ยากที่จะนำโปรโตคอลดังกล่าวไปใช้ในสถานการณ์จริงได้ โดยในปัจจุบันนี้ระบบปฏิบัติการที่ได้รับความนิยมมากที่สุด และถือว่าเป็นระบบปฏิบัติการหลักของผู้ใช้ในโลกนี้คือระบบปฏิบัติการวินโดวส์ (Windows) ซึ่งโปรโตคอลที่ใช้ในการควบคุมการส่งข้อมูลที่ใช้ในวินโดวส์จะสืบทอดมาจากโปรโตคอล NewReno ดังที่กล่าวข้างต้น ดังนั้นจึงมีความจำเป็นอย่างยิ่งในการประเมินผลในส่วนนี้ โปรโตคอล NewReno จึงถูกเลือกมาใช้เป็นตัวแทนของโปรโตคอลที่ถูกติดตั้งใช้งานในปัจจุบัน

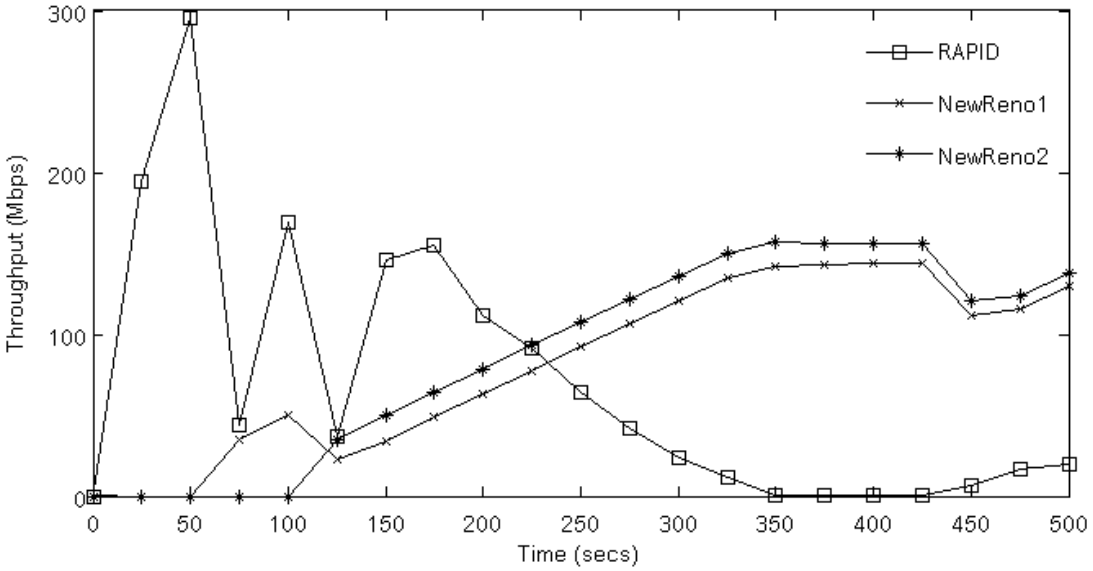
จากการทดลองพบว่าจุดบกพร่องที่เห็นได้ชัดของโปรโตคอล NewReno คือความล่าช้าในการค้นหาค่าแบนด์วิดธ์ที่มีอยู่บนช่องทางการสื่อสาร ดังจะเห็นได้ว่า NewReno จะใช้เวลาหลาย RTT กว่าจะปรับตัวเองให้ใช้ประโยชน์จากแบนด์วิดธ์ได้อย่างเต็มที่



รูปภาพที่ 4.2 ประสิทธิภาพของโปรโตคอล NewReno

จากรูปภาพที่ 4.2 ข้างต้นเป็นการนำโปรโตคอล NewReno มาทำงานบนเครือข่ายขนาดแบนด์วิดธ์ 200 Mbps จะเห็นได้ว่าโปรโตคอลใช้เวลามากกว่า 300 วินาที กว่าจะใช้ประโยชน์จากแบนด์วิดธ์ได้อย่างเต็มที่ นอกจากนี้โปรโตคอล NewReno จะตรวจจับความคับคั่งของช่องทางการสื่อสารโดยดูจากการสูญหายของแพ็กเก็ต กล่าวคือจะลดจำนวนของแพ็กเก็ตที่ส่งออกลงก็ต่อเมื่อพบว่ามีแพ็กเก็ตสูญหายขึ้น ทำให้ค่อนข้างล่าช้าในการปรับเปลี่ยนอัตราการส่งข้อมูล ซึ่งยังส่งผลให้โปรโตคอลมีคุณลักษณะค่อนข้างก้าวร้าว เนื่องจากไม่สามารถปรับลดอัตราการส่งข้อมูลได้อย่างทันท่วงทีเมื่อเกิดความคับคั่งบนเครือข่าย

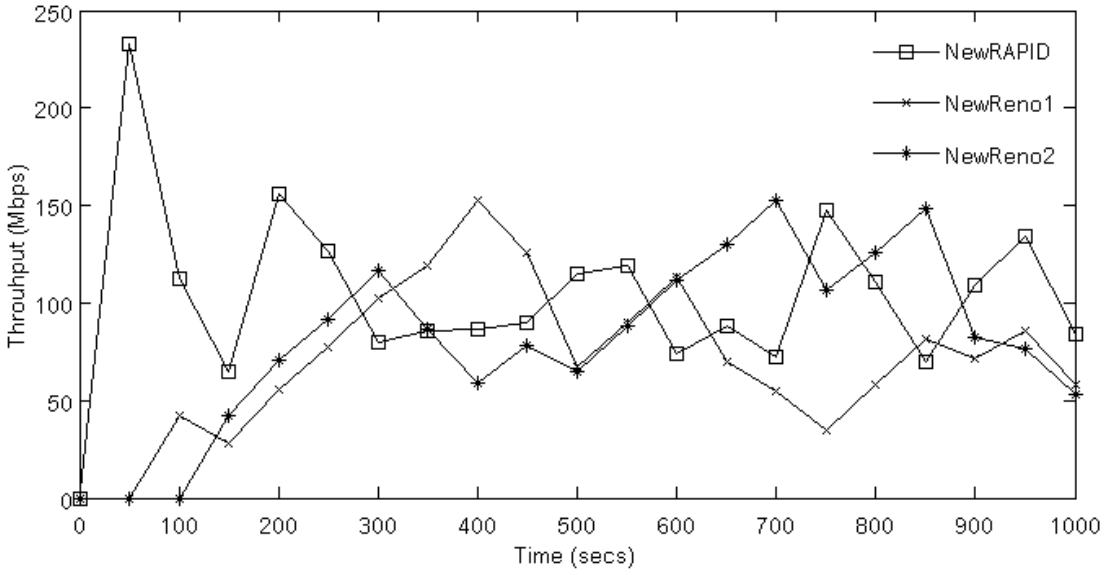
ดังที่กล่าวข้างต้นว่าโปรโตคอล RAPID ไม่สามารถทำงานร่วมกันกับโปรโตคอล NewReno ได้ ซึ่งจากการทดลองพบว่า RAPID ลดอัตราการส่งข้อมูลของตัวเองเร็วมากดังรูปภาพที่ 4.3 ในขณะที่ NewReno ก็ค่อย ๆ ส่งข้อมูลไปเรื่อย ๆ โดยไม่สามารถรับรู้ได้ว่ามีการใช้งานจากผู้อื่นจนกว่าจะเกิดแพ็กเก็ตสูญหาย สำหรับการทดลองนั้นจะสร้างเส้นทางการส่งข้อมูลระหว่างผู้ส่งและผู้รับเป็น 3 เส้นทางด้วยกัน โดยเส้นทางแรกจะกำหนดให้ใช้โปรโตคอล RAPID และอีกสองเส้นทางที่เหลือจะใช้โปรโตคอล NewReno ซึ่งการเชื่อมต่อทั้ง 3 เส้นทางนั้นจะต้องเข้าใช้ช่องทางการสื่อสารที่เป็นส่วนของคอขวดเดียวกัน โดยกำหนดขนาดแบนด์วิดธ์ไว้ที่ 300 Mbps นอกจากนี้จะสั่งให้ผู้ส่งเริ่มต้นการทำงานไม่พร้อมกัน เพื่อต้องการดูการปรับตัวของโปรโตคอล



รูปภาพที่ 4.3 ปริมาณการส่งข้อมูลที่ RAPID และ NewReno ทำได้เมื่อทำงานร่วมกัน

สาเหตุที่ทำให้ RAPID ลดอัตราการส่งข้อมูลค่อนข้างเร็ว เนื่องจากเป็นโปรโตคอลที่อิงอยู่กับการเกิด Queuing Delay ของแพ็กเก็ต ดังนั้นเมื่อมีจำนวนแพ็กเก็ตไปรออยู่ในบัฟเฟอร์ที่เร้าเตอร์ค่อนข้างนานและเยอะ ซึ่งเป็นสาเหตุมาจากการที่มีแพ็กเก็ตปล่อยออกมาจากโปรโตคอล NewReno อยู่ในเครือข่ายค่อนข้างมากกว่าที่ควรจะเป็นนั่นเอง ทำให้ RAPID ลดอัตราการส่งข้อมูลของตัวเองลงทันที และจากเหตุการณ์ดังกล่าวจึงสามารถจะสรุปได้ว่าโปรโตคอลที่ใช้หลักการตรวจจับความล้าช้าของแพ็กเก็ตมาเป็นตัวชี้วัดความคับคั่งของเครือข่าย เพื่อปรับเปลี่ยนอัตราการส่งข้อมูลนั้น อาจจะไม่สามารถนำไปติดตั้งใช้งานในเครือข่ายอินเตอร์เน็ตในปัจจุบันได้

เมื่อนำโปรโตคอล NewRAPID ที่ได้นำเสนอในงานวิจัยชิ้นนี้ ซึ่งเป็นโปรโตคอลที่พัฒนามาจากโปรโตคอล RAPID โดยได้นำอัลกอริทึม IGI มาเป็นตัวเสริมการทำงานของโปรโตคอล RAPID มาทดลองโดยใช้แบบจำลองเดียวกันกับข้างต้น ซึ่งจากผลการทดลองพบว่าโปรโตคอล NewRAPID จะมีความเสถียรภาพมากกว่าโปรโตคอล RAPID โดยสามารถทำงานควบคู่กับโปรโตคอล NewReno ได้เป็นอย่างดี จากการที่กล่าวว่าโปรโตคอล NewRAPID มีความเสถียรภาพกว่านั้น จะวัดจากการแกว่ง (Volatility) ของอัตราการส่งข้อมูลที่โปรโตคอลใช้ในการส่งข้อมูลในแต่ละช่วงเวลา กล่าวคืออัตราการส่งข้อมูลของโปรโตคอล NewRAPID จะไม่แกว่งไปมามากจนเกินความจำเป็นเหมือนอย่างที่เกิดขึ้นกับโปรโตคอล RAPID

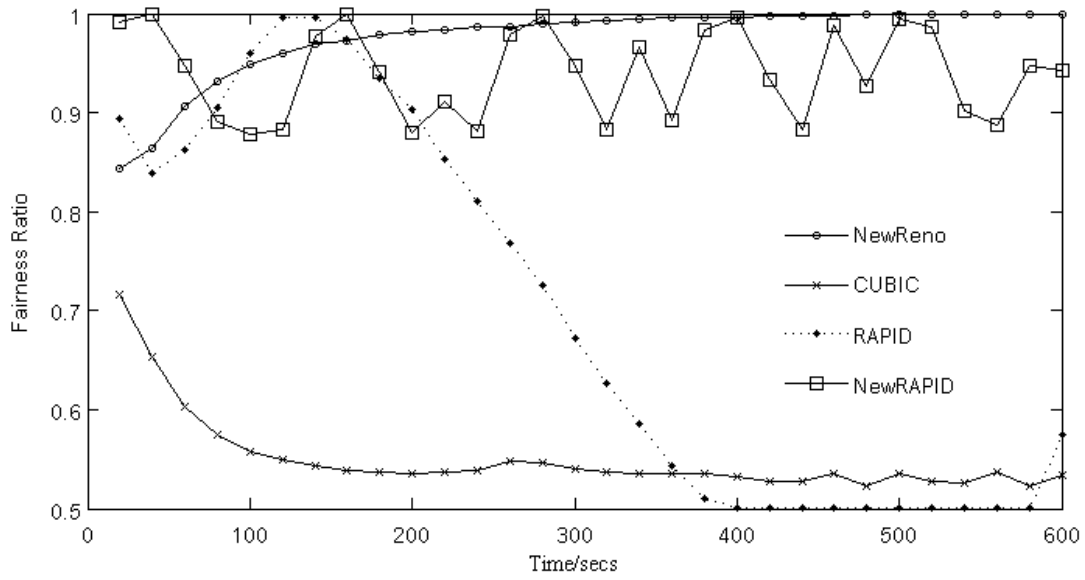


รูปภาพที่ 4.4 ปริมาณการส่งข้อมูลที่ NewRAPID และ NewReno ทำได้เมื่อทำงานร่วมกัน

จากรูปภาพที่ 4.4 จะเห็นได้ว่า NewRAPID สามารถทำงานต่อไปได้เมื่อทำงานร่วมกันโปรโตคอล NewReno ถึงแม้ว่าปริมาณการส่งข้อมูลที่ NewRAPID ทำได้จะตกลงบ้าง เนื่องจากความก้าวร้าวของ NewReno ที่ส่งแพ็กเก็ตออกไปค้างที่บัฟเฟอร์ของเร้าเตอร์เป็นจำนวนมาก ซึ่งจะเห็นได้ว่าค่อนข้างนานกว่าที่ NewReno จะลดปริมาณการส่งแพ็กเก็ตลง จนกระทั่งเมื่อ NewReno พบว่ามีแพ็กเก็ตสูญหายก็จะลดอัตราการส่งข้อมูลลง ทำให้ NewRAPID ตรวจจับได้ว่ามีแบนด์วิดธ์ที่ว่างอยู่เพิ่มมากขึ้นก็จะค่อย ๆ เพิ่มอัตราการส่งข้อมูลสลับกันไป จากการที่ค่าแบนด์วิดธ์ที่วัดได้จาก IGI จะไม่ค่อยแกว่งไปมามากนัก จึงทำให้โปรโตคอลไม่ปรับเปลี่ยนการส่งข้อมูลเร็วมากจนเกินไป แต่ก็ควรที่จะไม่ช้าจนเกินไปเช่นเดียวกัน ดั้งนั้นเราจึงยังต้องอาศัยอัลกอริทึมเดิมของ RAPID อยู่ โดยจากผลที่ได้รับสามารถประเมินได้ว่าอัลกอริทึม IGI เป็นตัวเสริมที่ดีสำหรับช่วยให้ RAPID มีความเสถียรภาพในการปรับอัตราการส่งข้อมูลมากยิ่งขึ้น สำหรับค่าน้ำหนักที่ใช้ในการทดลองจะอยู่ที่ 0.8 ซึ่งถูกกำหนดไว้เป็นค่าเริ่มต้นของโปรโตคอล

นอกจากนี้ได้มีการวัดค่าความเสมอภาคโดยใช้สมการของ Jain's Fairness Index [20] เพื่อเปรียบเทียบอัตราส่วนของปริมาณการส่งข้อมูล (Throughput) ที่แต่ละคู่ของผู้ส่งและผู้รับสามารถทำได้ โดยการทดลองจะนำโปรโตคอล CUBIC RAPID และ NewRAPID แต่ละตัวมาทำงานร่วมกันกับโปรโตคอล NewReno แล้วจึงวัดปริมาณการส่งข้อมูลที่แต่ละโปรโตคอลและ NewReno ทำได้ หลังจากนั้นจึงนำมาหาค่าอัตราส่วนความเสมอภาคกัน (Fairness Ratio) สาเหตุที่เลือกใช้โปรโตคอล CUBIC มาเป็นตัวเปรียบเทียบในการทดลอง เนื่องจากโปรโตคอล CUBIC ได้ถูกติดตั้งใช้งานจริงบนระบบปฏิบัติการลีนุกซ์ (Linux) มาหลายเวอร์ชั่นแล้ว [21] ดังนั้น

ถือได้ว่าเป็นโปรโตคอลที่ประสบความสำเร็จตัวหนึ่ง อย่างไรก็ตามผู้ใช้ส่วนใหญ่ก็ยังคงใช้ระบบปฏิบัติการวินโดวส์ ซึ่งใช้โปรโตคอลสำหรับควบคุมการส่งข้อมูลโดยอิงมาจากโปรโตคอล NewReno ซึ่งได้อยู่ในการทดลองแล้ว อย่างไรก็ตามระบบปฏิบัติการวินโดวส์ในเวอร์ชั่นหลัง ๆ เช่น วินโดวส์วิสต้า (Windows Vista) นั้นได้ถูกติดตั้งโปรโตคอลตัวใหม่ที่ชื่อว่า Compound TCP (CTCP) ไปแล้ว แต่ในขณะที่ทำการทดลองนั้น ผู้วิจัยไม่สามารถตรวจสอบได้ว่าโปรโตคอล CTCP ที่อยู่บน NS-2 จะเป็นเวอร์ชั่นล่าสุดที่ใช้อยู่บนระบบปฏิบัติการวินโดวส์ จึงไม่ได้นำมาใช้ในการทดลองครั้งนี้ แต่อย่างไรก็ตาม CTCP ก็ยังคงอิงอยู่กับการทำงานแบบ NewReno อยู่เช่นเดียวกัน สำหรับการทดลองครั้งนี้จะกำหนดให้แบนด์วิดธ์ที่คอขวด (Bottleneck Link) มีค่าเท่ากับ 100 Mbps เพื่อต้องการให้ NewReno ขึ้นถึงจึงสูงสุดของแบนด์วิดธ์ได้อย่างรวดเร็ว และใช้เวลาในการทดลอง 600 วินาที



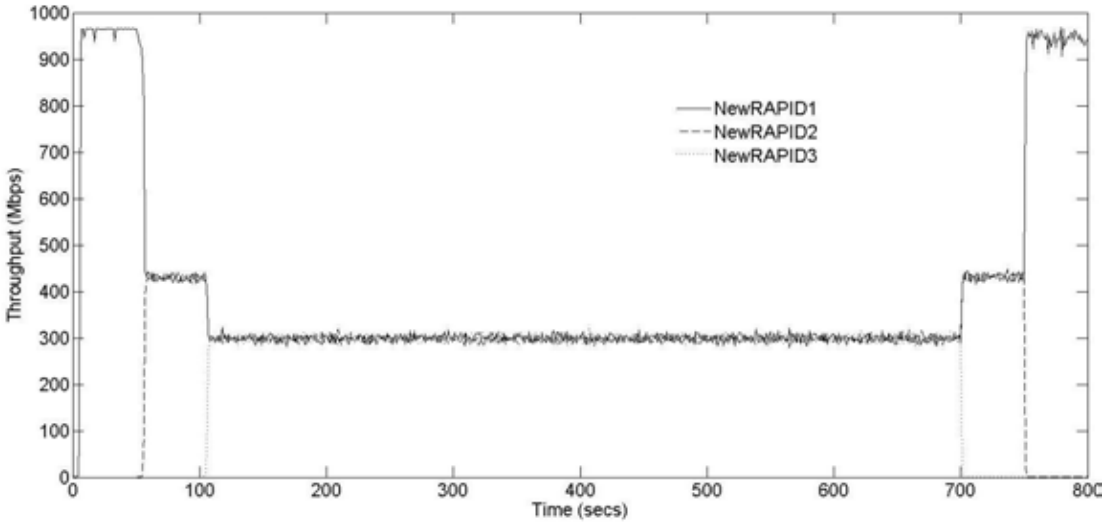
รูปภาพที่ 4.5 อัตราส่วนปริมาณการส่งข้อมูลของแต่ละโปรโตคอลเมื่อทำงานร่วมกับ NewReno

จากรูปภาพที่ 4.5 จะเห็นได้ว่า NewRAPID สามารถทำงานควบคู่กับ NewReno ได้ดีกว่าโปรโตคอลตัวอื่น ๆ สำหรับ CUBIC นั้นพบว่าเป็นโปรโตคอลที่ค่อนข้างก้าวร้าวกว่า NewReno ทำให้ปริมาณการส่งข้อมูลค่อนข้างมากกว่า NewReno เป็นอย่างมากทำให้ค่าความเสมอภาคต่ำลงมา แต่อย่างไรก็ตามเป็นสาเหตุมาจากการที่จำนวนผู้ใช้เครือข่ายค่อนข้างน้อย ถ้ามีจำนวนผู้ใช้งานมากกว่านี้ก็จะทำให้แพ็กเก็ตของ CUBIC สูญหายบ่อยและมากครั้งขึ้น ซึ่งจะส่งให้ CUBIC ลดการอัตราการส่งข้อมูลของตัวเองลงได้เร็วมากยิ่งขึ้น และส่งผลให้สัดส่วนของปริมาณการส่งข้อมูลของแต่ละผู้ใช้เท่าเทียมกันมากขึ้น นอกจากนี้ผู้วิจัยได้ลองทดลองเพิ่มเติมโดยเพิ่ม

จำนวนของผู้ใช้ทั้ง NewReno และ NewRAPID เป็นอย่างละ 6 และเพิ่มขนาดแบนด์วิดธ์เป็น 400Mbps ก็พบว่า NewRAPID ก็ยังคงให้ผลที่ดีกว่าโปรโตคอลอื่นเช่นเดียวกัน กล่าวคือ NewReno และ NewRAPID สามารถสลับกันเข้าใช้แบนด์วิดธ์ขนานกันไป

4.2 Intra-protocol Fairness Evaluation

การประเมินในส่วนนี้เป็นการประเมินประมาณการส่งข้อมูลที่โปรโตคอล NewRAPID แต่ละตัวทำได้เมื่อนำมาทำงานร่วมกัน ดังที่ทราบว่าโปรโตคอล RAPID นั้นสามารถค้นหาค่าแบนด์วิดธ์ได้อย่างรวดเร็วและไม่ก้าวร้าว ทำให้ RAPID เป็นโปรโตคอลที่มีความยุติธรรมกับเพื่อนโปรโตคอลเดียวกัน กล่าวคือตัว RAPID จะไม่ส่งแพ็กเก็ตออกไปมากเกินกว่าส่วนแบ่งแบนด์วิดธ์ที่ตัวเองควรจะได้รับ ในการทดลองนี้เป็นการนำโปรโตคอล NewRAPID มาเป็นโปรโตคอลสำหรับควบคุมการส่งข้อมูลระหว่างผู้รับและผู้ส่ง 3 คู่ด้วยกัน โดยกำหนดขนาดของแบนด์วิดธ์ในส่วนที่เป็นคอขวดไว้ที่ 1 Gbps โดยกำหนดให้ผู้ส่งของแต่ละเส้นทางเริ่มต้นการทำงานไม่พร้อมกัน โดยจะค่อย ๆ เริ่มต้นการทำงานขึ้นมาทีละตัวเป็นลำดับต่อเนื่องกันไป เพื่อจะดูว่าโปรโตคอลจะปรับตัวเองอย่างไรเมื่อมีผู้อื่นเข้าใช้เส้นทางการสื่อสารเดียวกัน

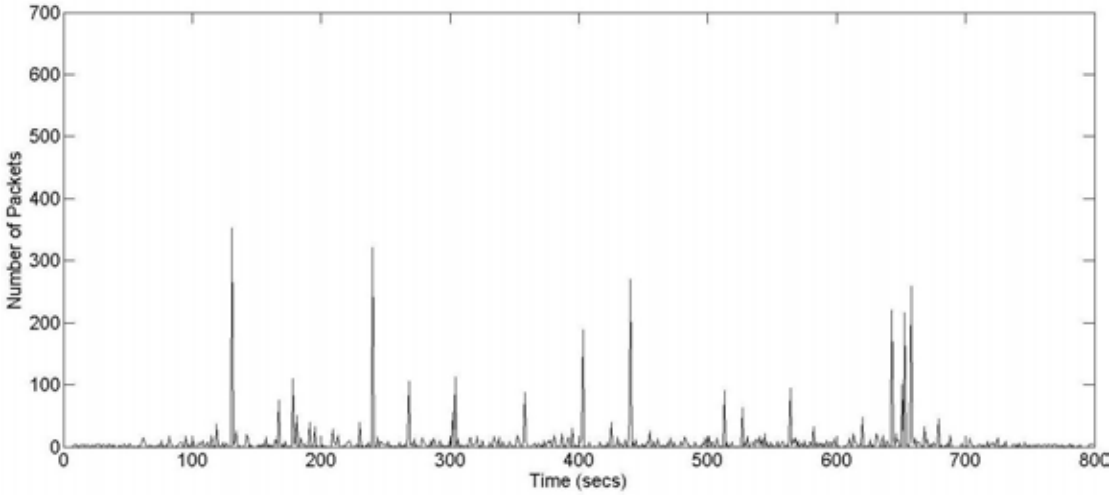


รูปภาพที่ 4.6 ปริมาณการส่งข้อมูลที่ NewRAPID ทำได้เมื่อทำงานร่วมกัน

จากรูปภาพที่ 4.6 จะพบว่าโปรโตคอล NewRAPID สามารถค้นหาค่าแบนด์วิดธ์ และตรวจจับการเปลี่ยนแปลงของค่าแบนด์วิดธ์ได้อย่างรวดเร็วเช่นเดียวกับโปรโตคอล RAPID ดังนั้นทำให้โปรโตคอลปรับเปลี่ยนอัตราการส่งข้อมูลได้อย่างเหมาะสม จะเห็นได้ว่าเมื่อมีผู้ใช้คนใหม่เข้ามาใช้เส้นทางการสื่อสารเดียวกัน โปรโตคอล NewRAPID ก็สามารถลดอัตราการส่งข้อมูลลงมาได้อย่าง

รวดเร็ว เพื่อให้คนอื่นสามารถส่งข้อมูลได้เท่าเทียมกันกับตนเอง ซึ่งสามารถสร้างความเสมอภาคระหว่างผู้ใช้ได้รวดเร็วเป็นอย่างมาก

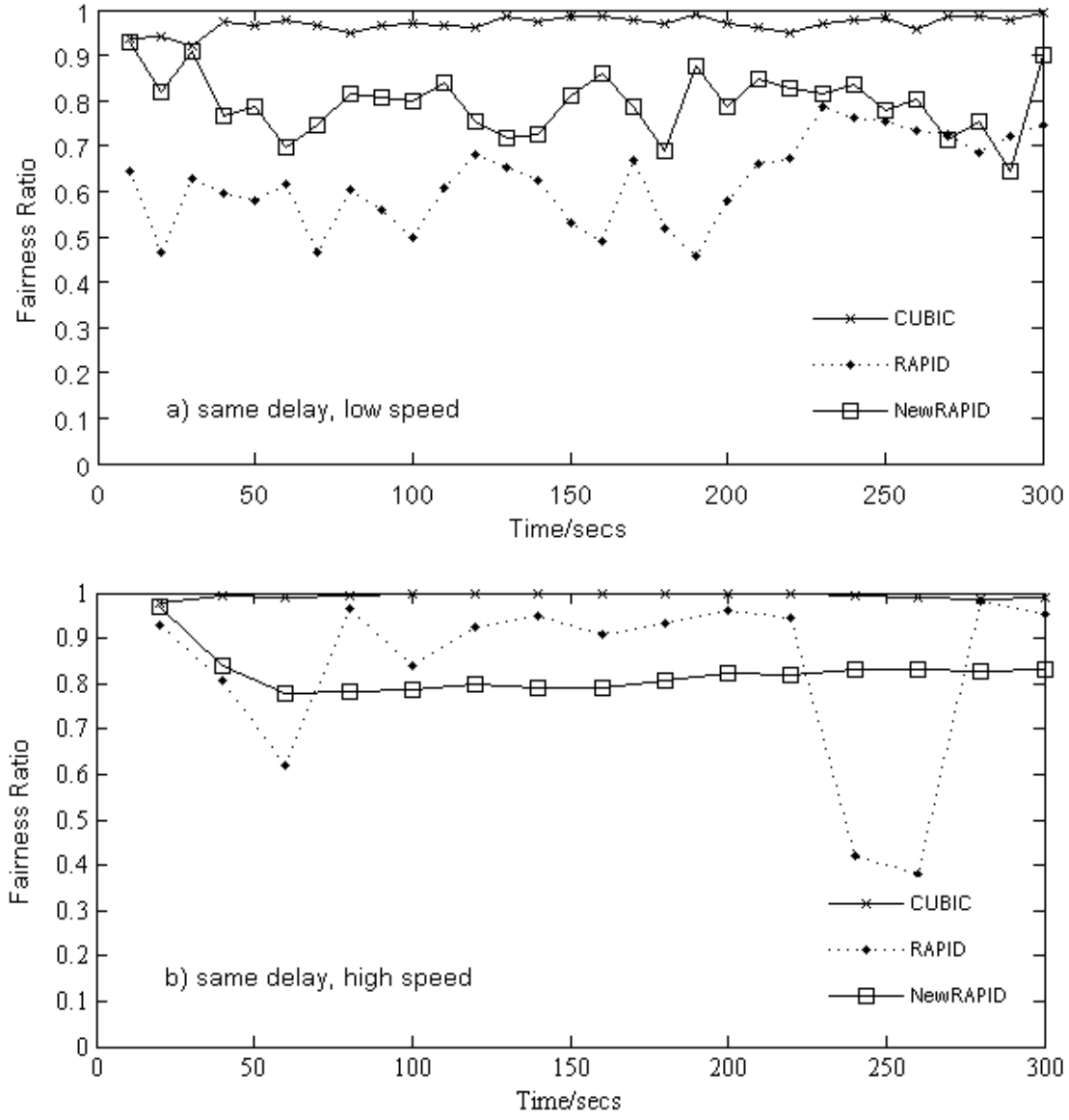
จากการทดลองข้างต้น ได้วัดค่าขนาดของบัฟเฟอร์ตามช่วงเวลาไปพร้อม ๆ กันด้วย ซึ่งพบว่าขนาดของบัฟเฟอร์ที่เร้าเตอร์จะเพิ่มขึ้นและลดลงอย่างรวดเร็ว โดยไม่ทำให้บัฟเฟอร์ที่เร้าเตอร์เกิดเต็มเป็นเวลานาน ซึ่งเป็นคุณสมบัติเด่นของ RAPID และสืบทอดมายัง NewRAPID นั่นเอง โดยมีสาเหตุมาจากการที่ทั้งสองโปรโตคอลใช้หลักการของการตรวจจับค่า Queuing Delay จึงทำให้โปรโตคอลไม่ก้าวร้าว และปรับเปลี่ยนตัวเองได้ให้สอดคล้องกับการเปลี่ยนแปลงของขนาดแบนด์วิดธ์ได้อย่างรวดเร็ว ดังนั้นโปรโตคอลจะไม่พยายามส่งแพ็กเก็ตออกไปในเครือข่ายมากจนเกินความจำเป็น ส่งผลให้จำนวนแพ็กเก็ตที่ไปคั่งค้างที่บัฟเฟอร์ของเร้าเตอร์น้อยลง ดังนั้นจะส่งผลให้โอกาสเกิดแพ็กเก็ตสูญหายก็ลดน้อยลงอีกด้วย โดยจำนวนของแพ็กเก็ตที่อยู่ในบัฟเฟอร์ของเร้าเตอร์ที่วัดได้จะแสดงอยู่ในรูปภาพที่ 4.6

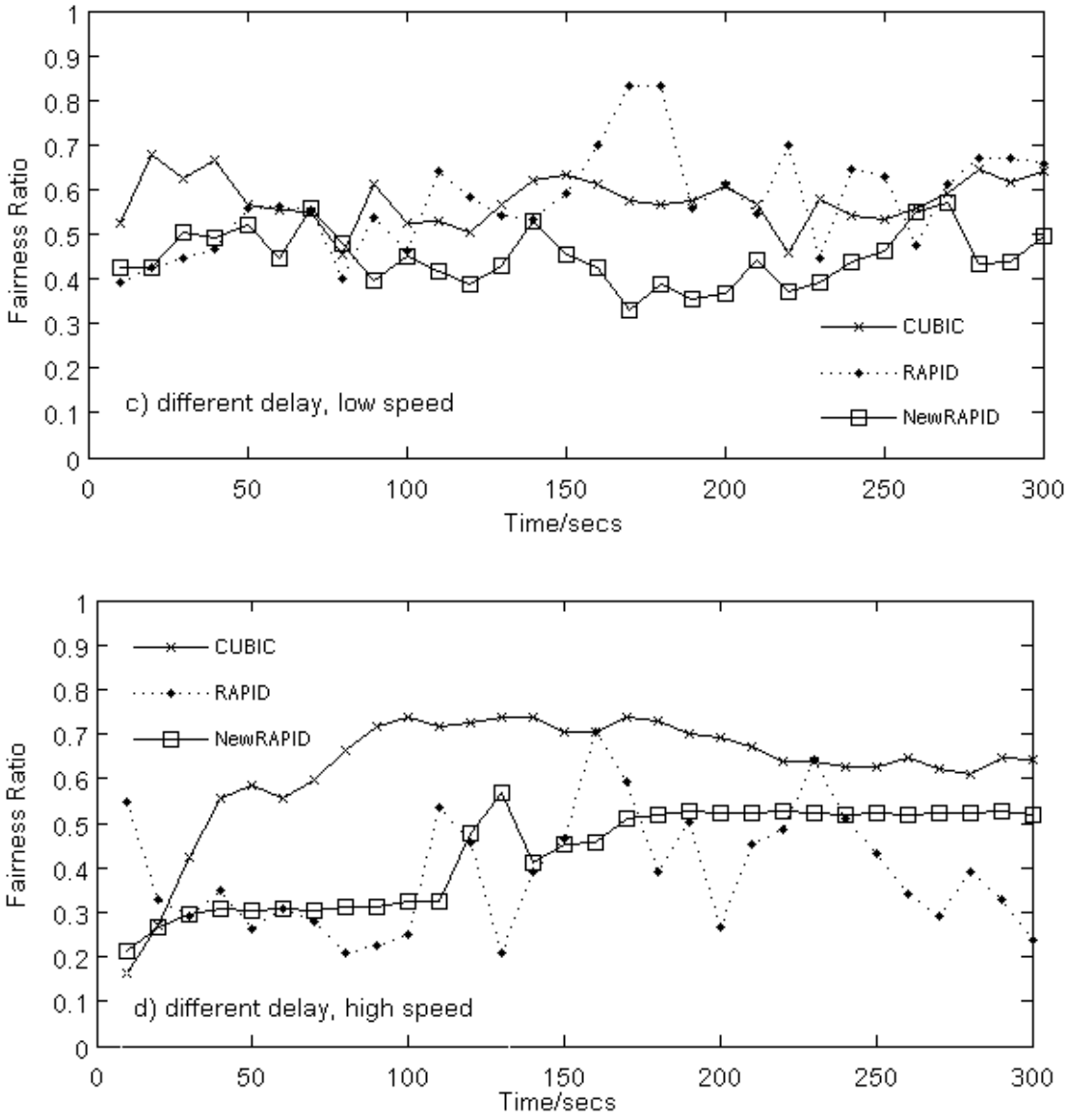


รูปภาพที่ 4.7 จำนวนแพ็กเก็ตที่ค้างอยู่ในบัฟเฟอร์ของเร้าเตอร์

หลังจากการทดลองข้างต้น ผู้วิจัยได้ทำการทดลองเพิ่มเติมโดยการกำหนดให้มีจำนวนผู้ใช้จำนวนมากขึ้น และทำการทดสอบทั้งบนเครือข่ายความเร็วต่ำและสูง หลังจากนั้นใช้สมการของ Jain's Fairness Index มาวัดค่าความเสมอภาคที่แต่ละโปรโตคอลทำได้ ในการทดลองได้กำหนดขนาดแบนด์วิดธ์ในส่วนที่เป็นคอขวดสำหรับเครือข่ายความเร็วต่ำไว้ที่ 10 Mbps และจำนวนผู้ใช้เท่ากับ 10 การเชื่อมต่อ สำหรับเครือข่ายความเร็วสูงได้กำหนดขนาดของแบนด์วิดธ์ในส่วนที่เป็นคอขวดไว้ที่ 500Mbps และจำนวนผู้ใช้เท่ากับ 20 การเชื่อมต่อ โดยผลการทดลองที่ได้จะเป็นดังรูปภาพที่ 4.8 โดย 2 รูปย่อยแรก (a และ b) จะแสดงผลการทดลองในกรณีที่

แต่ละการเชื่อมต่อระหว่างผู้ส่งและผู้รับมีค่า Propagation Delay ที่เท่ากัน และอยู่บนเครือข่ายความเร็วต่ำและสูงตามลำดับ หลังจากนั้น 2 รูปย่อยหลังสุด (c และ d) จะแสดงผลการทดลองในกรณีที่แต่ละการเชื่อมต่อระหว่างผู้ส่งและผู้รับมีค่า Propagation Delay ที่แตกต่างกันออกไปโดยจะสุ่มค่ามาให้ขณะทำการทดลอง ซึ่งจะมีค่าอยู่ที่ระหว่าง 20 – 340ms และอยู่บนเครือข่ายความเร็วต่ำและสูงตามลำดับเช่นกัน



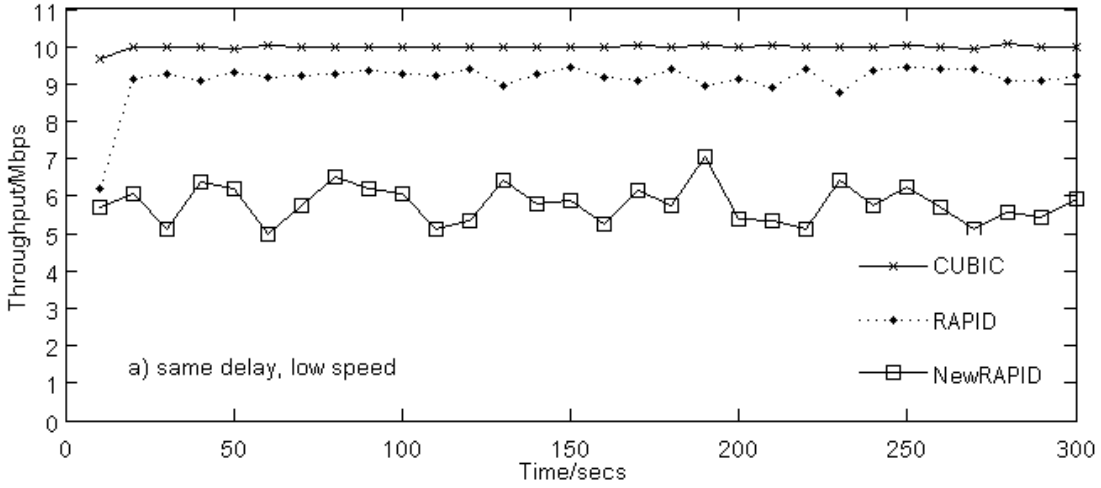


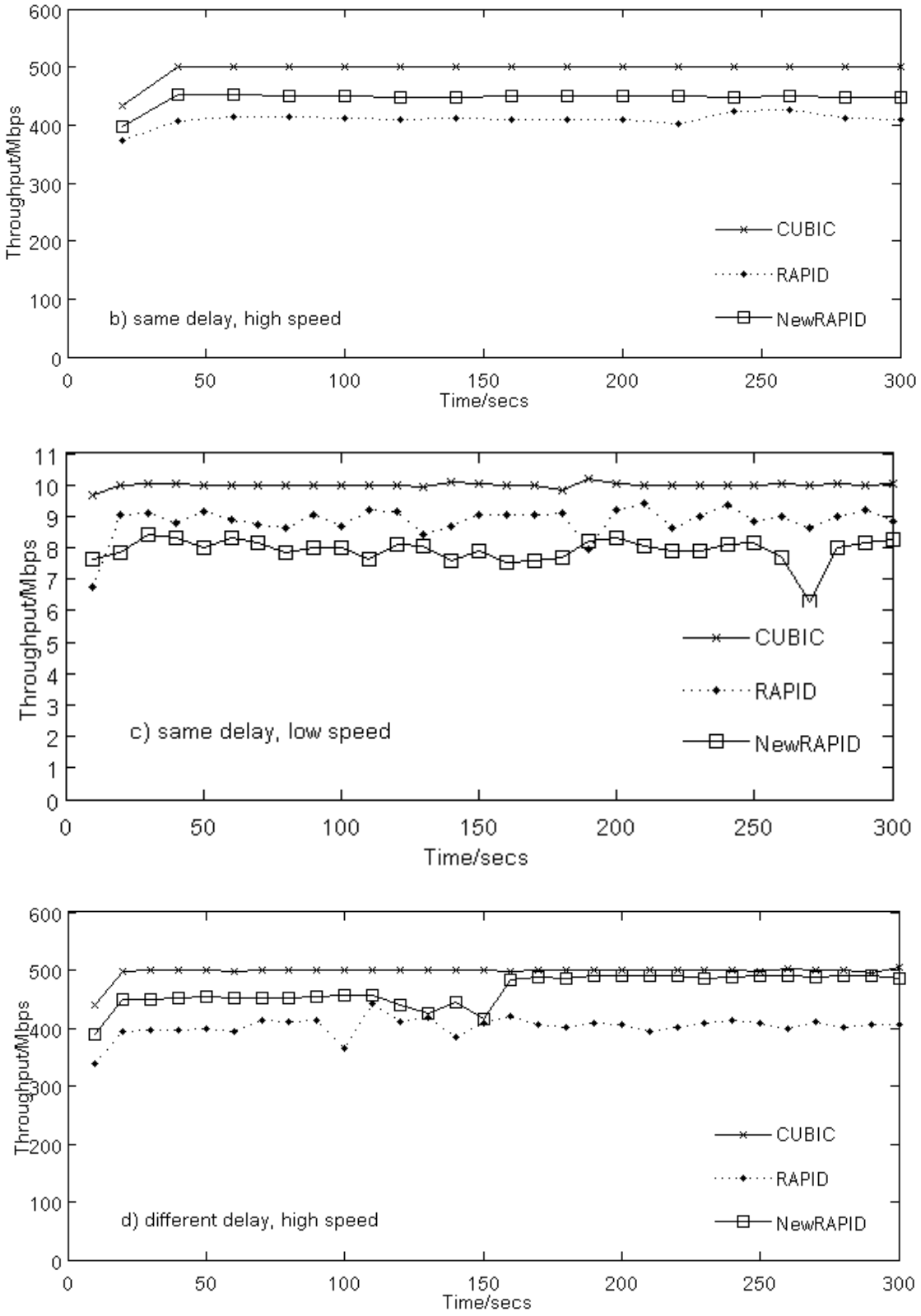
รูปภาพที่ 4.8 เปรียบเทียบค่าความเสมอภาคที่แต่ละโปรโตคอลทำได้

จากผลการทดลองข้างต้นพบว่าในกรณีที่ทุก ๆ การเชื่อมต่อมีค่า Propagation Delay เท่ากันนั้น CUBIC จะให้ค่าความเสมอภาคที่ดีกว่า RAPID และ NewRAPID โดยทั้งสองโปรโตคอลจะมีค่าที่ใกล้เคียงกัน นอกจากนี้การที่ RAPID และ NewRAPID ใช้ Queuing Delay ทำให้ค่าแบนด์วิดธ์ที่วัดได้ค่อนข้างแกว่งไปมามากกว่า CUBIC โดยเฉพาะอย่างยิ่งในเครือข่ายความเร็วต่ำ แต่ค่าความเสมอภาพจะเพิ่มสูงขึ้นสำหรับการทำงานบนเครือข่ายความเร็วสูง ซึ่งหมายความว่าในเครือข่ายความเร็วสูงโปรโตคอล RAPID และ NewRAPID จะทำงานได้ดีขึ้นนั่นเอง

สำหรับกรณีที่แต่ละการเชื่อมต่อมีค่า Propagation Delay ต่างกันนั้น ทุก ๆ โปรโตคอลจะให้ค่าความเสมอภาคที่ต่ำลง ดังนั้นจะเห็นได้ว่าเกิด RTT un-fairness โดยการเชื่อมต่อที่มีค่า RTT ที่สั้นกว่าก็จะได้รับ acknowledgement ที่เร็วกว่า ดังนั้นโอกาสในการปรับอัตราการส่งข้อมูลจึงทำได้เร็วกว่าและบ่อยกว่า ทำให้สามารถเข้าใช้แบนด์วิดธ์ที่ว่างอยู่ได้รวดเร็วกว่า แต่ถึงอย่างไรโปรโตคอล CUBIC ก็ยังให้ค่าความเสมอภาคที่ดีกว่าโปรโตคอลตัวอื่น

นอกจากนี้ค่าปริมาณข้อมูลที่แต่ละผู้ใช้สามารถรับส่งกันได้ในการทดลองข้างต้นสามารถนำมาหาผลรวมเพื่อดูปริมาณข้อมูลทั้งหมดที่แต่ละโปรโตคอลสามารถรับส่งได้ เพื่อดูประสิทธิภาพในการเข้าใช้แบนด์วิดธ์ของแต่ละโปรโตคอลนั่นเอง โดยผลที่ได้จะแสดงอยู่ในรูปภาพที่ 4.9 โดย 2 รูปย่อยแรก (a และ b) จะแสดงผลการทดลองในกรณีที่แต่ละการเชื่อมต่อระหว่างผู้รับและผู้ส่งมีค่า Propagation Delay ที่เท่ากัน และอยู่บนเครือข่ายความเร็วต่ำและสูงตามลำดับ หลังจากนั้น 2 รูปย่อยหลังสุด (c และ d) จะแสดงผลการทดลองในกรณีที่แต่ละการเชื่อมต่อมีค่า Propagation Delay ที่แตกต่างกัน และทำงานอยู่บนเครือข่ายความเร็วต่ำและสูงตามลำดับเช่นกัน



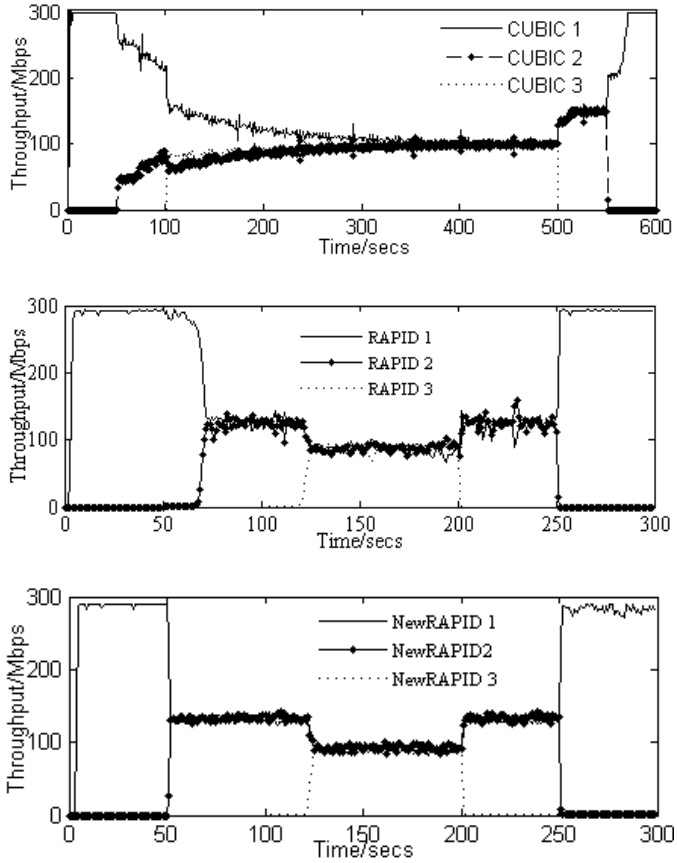


รูปภาพที่ 4.9 เปรียบเทียบประสิทธิภาพในการใช้แบนด์วิดธ์ของแต่ละโปรโตคอล

จากรูปภาพข้างต้นพบว่าทั้ง 4 สถานการณ์ดังกล่าวนั้น CUBIC สามารถใช้แบนด์วิดธ์ได้อย่างมีประสิทธิภาพมากที่สุด เนื่องจากปริมาณการส่งข้อมูลที่ทำได้นั้นใกล้เคียงกับขนาดแบนด์วิดธ์ที่กำหนดไว้ในส่วนที่เป็นคอขวดมากที่สุด สำหรับ RAPID และ NewRAPID นั้นสามารถเข้าใช้แบนด์วิดธ์ได้ประสิทธิภาพต่ำลงมา กล่าวคือ RAPID สามารถใช้แบนด์วิดธ์ได้อย่างมีประสิทธิภาพมากกว่า NewRAPID ในกรณีที่ทำงานบนเครือข่ายความเร็วต่ำ แต่ NewRAPID จะสามารถเข้าใช้งานแบนด์วิดธ์ได้อย่างมีประสิทธิภาพมากกว่า RAPID หากทำงานบนเครือข่ายความเร็วสูง นอกจากนี้ยังสามารถประเมินได้ว่าการที่โปรโตคอลที่ใช้ Queuing Delay เป็นตัวตรวจจับความคับคั่งนั้นจะค่อนข้างอ่อนไหวต่อการจราจรบนเครือข่ายโดยเฉพาะอย่างยิ่งหากเกิดความแออัดขึ้นบนเครือข่ายค่อนข้างมาก ทำให้โปรโตคอลปรับเปลี่ยนอัตราการส่งข้อมูลค่อนข้างเยอะและเร็วจนเกินไป หากมีการลดลงค่อนข้างมากก็จะส่งผลให้ตัวเองให้สูญเสียปริมาณการส่งข้อมูลมากตามไปด้วย

4.3 Convergence Time Evaluation

สำหรับการประเมินผลในส่วนนี้ จะเป็นการวัดความเร็วของโปรโตคอลในการปรับตัวเองเมื่อเกิดการเปลี่ยนแปลงของจำนวนผู้ใช้งานบนเครือข่าย โดยได้ทดลองนำโปรโตคอลบนเครือข่ายความเร็วสูงแต่ละตัวมาทำงานร่วมกัน โดยกำหนดขนาดแบนด์วิดธ์ไว้ที่ 300 Mbps และกำหนดจำนวนการเชื่อมต่อระหว่างผู้ส่งและผู้รับไว้ 3 คู่ด้วยกัน โดยผู้ส่งแต่ละตัวจะเริ่มต้นและหยุดทำงานที่ช่วงเวลาต่างกัน เพื่อต้องการดูว่าเมื่อมีผู้ใช้คนอื่นเข้ามาร่วมใช้แบนด์วิดธ์ แต่ละโปรโตคอลจะปรับตัวเองอย่างไร และเมื่อมีผู้ใช้บางคนเลิกใช้งานเครือข่าย แต่ละโปรโตคอลจะปรับตัวเองได้รวดเร็วมากน้อยแค่ไหนเช่นเดียวกัน ดังรูปภาพที่ 4.7 โปรโตคอลความเร็วสูงตัวอื่นที่เลือกมาใช้ในการเปรียบเทียบคือ CUBIC ซึ่งจากการทดลองพบว่า CUBIC จะค่อนข้างช้าในการปรับเปลี่ยนอัตราการส่งข้อมูลของตนเอง โดยจะเห็นได้ว่าต้องใช้เวลาทดลองนานถึง 600 วินาทีกว่าจะเห็นภาพการทำงานของ CUBIC ชัดเจน ทั้งนี้เป็นสาเหตุมาจากที่ CUBIC มีคุณลักษณะค่อนข้างก้าวร้าว โดยจะลดอัตราการส่งข้อมูลลงค่อนข้างช้า ทำให้เกิดความไม่เสมอภาคกันระหว่างผู้ใช้เป็นระยะเวลาค่อนข้างนาน สำหรับโปรโตคอล RAPID และ NewRAPID นั้นสามารถปรับตัวเองได้รวดเร็ว โดยแต่ละเส้นมาบรรจบกันภายในเวลาอันสั้น ซึ่งทั้งนี้เป็นข้อดีของการนำ Queuing Delay มาใช้ในการตรวจจับความคับคั่ง นอกจากนั้นจะเห็นได้ว่าแต่ละโปรโตคอลยังสามารถใช้ประโยชน์จากแบนด์วิดธ์ได้อย่างเต็มที่ โดยดูผลรวมของปริมาณการส่งข้อมูลจากของแต่ละผู้ใช้พบว่าค่อนข้างใกล้เคียงกับขนาดของแบนด์วิดธ์ที่กำหนดให้กับส่วนที่เป็นคอขวด



รูปภาพที่ 4.10 ปริมาณข้อมูลที่แต่ละโปรโตคอลสามารถทำได้เมื่อทำงานร่วมกัน

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

งานวิจัยนี้ได้เสนอวิธีการแก้ปัญหาเรื่องการขาดเสถียรภาพในการปรับอัตราการส่งข้อมูลของโปรโตคอล RAPID เมื่อนำไปใช้งานบนเครือข่ายที่มีความก้าวร้าว ที่แต่ละผู้ส่งก็พยายามส่งข้อมูลออกไปเยอะเกินความจำเป็น โดยวิธีการแก้ปัญหาคือการนำอัลกอริทึม IGI เข้ามาเสริมการทำงานของ RAPID โดยจากการทดลองพบว่าทำให้ RAPID ทำงานได้ดีขึ้นในสภาพแวดล้อมที่มีความก้าวร้าว และยังคงคุณสมบัติเด่นข้ออื่น ๆ ของ RAPID ไว้ได้ในระดับเดียวกัน ได้แก่ สามารถปรับอัตราการส่งข้อมูลของตัวเองเมื่อมีผู้ใช้งานเพิ่มมากขึ้นได้อย่างรวดเร็ว มีความเสมอภาคกับเพื่อนร่วมโปรโตคอลเดียวกัน เป็นต้น แต่อย่างไรก็ตามทุก ๆ โปรโตคอลที่นำมาทดสอบนั้นให้ค่าความเสมอภาคที่ต่ำลงเมื่อทำงานบนเครือข่ายที่ระยะทางระหว่างผู้ส่งและผู้รับแต่ละคู่แตกต่างกันออกไป (RTT un-fairness) ซึ่งจำเป็นที่จะต้องมีการปรับปรุงในอนาคต นอกจากนี้พบว่าโปรโตคอลที่ใช้ Queuing Delay เป็นตัวตรวจจับความคับคั่งนั้นจะค่อนข้างอ่อนไหวต่อการจราจรบนเครือข่าย โดยเฉพาะอย่างยิ่งกรณีที่มีความคับคั่งบนเครือข่ายค่อนข้างสูง หรือบนเครือข่ายความเร็วต่ำ ทำให้อัตราการส่งข้อมูลของโปรโตคอลแกว่งไปมามากจนเกินไป ซึ่งส่งผลให้ปริมาณการส่งข้อมูลเพิ่มและลดบ่อยมากจนเกินไปไม่คงที่ ดังจะเห็นได้ว่าประสิทธิภาพในการใช้งานแบนด์วิดธ์ต่ำลงมาดังที่รายงานไว้ในผลการทดลอง แต่สำหรับบนเครือข่ายความเร็วสูงโปรโตคอลที่ใช้ Queuing Delay ดังกล่าวสามารถทำงานได้ดีขึ้นค่อนข้างมาก

5.2 ข้อเสนอแนะ

จากการศึกษาพบว่าข้อควรแก้ไขของโปรโตคอล RAPID คือส่วนของการค้นหาค่าแบนด์วิดธ์ ซึ่งจำเป็นต้องส่งข้อมูลเป็นกลุ่มของแพ็กเก็ต และใช้ค่า spread factor เป็นตัวกำหนดขนาดของแบนด์วิดธ์ที่ต้องการค้นหา ซึ่งผู้วิจัยพบว่าตัวแปร 2 ตัวนี้จะมีผลต่อการประสิทธิภาพของโปรโตคอลเป็นอย่างมาก กล่าวคือถ้าขนาดของกลุ่มแพ็กเก็ตต่ำจนเกินไปก็จะทำให้ช่วงของแบนด์วิดธ์ที่ต้องการค้นหาในครั้งนั้นแคบจนเกินไปทำให้การค้นหาแบนด์วิดธ์ทำได้ช้าตามมา แต่ถ้ากำหนดขนาดที่สูงมากขึ้นก็จะส่งให้ผลจำนวนแพ็กเก็ตที่ส่งออกไปบนเครือข่ายเยอะมากจนเกินความจำเป็น และส่งผมให้เกิดแพ็กเก็ตเกิดสูญหายมากขึ้น นอกจากนี้สำหรับค่า spread factor นั้นถ้ากำหนดไว้ต่ำ ๆ ก็จะทำให้การค้นหาค่าแบนด์วิดธ์แม่นยำใกล้เคียงกับขนาดแบนด์วิดธ์จริงมากขึ้น แต่ก็จะทำให้การค้นหาแบนด์วิดธ์ทำได้ช้าจนเกินไปเช่นกัน แต่ถ้ากำหนดให้มีค่าที่สูงมากขึ้นก็จะทำให้การค้นหาทำได้เร็วขึ้น แต่จะลดความแม่นยำในการค้นหาค่าแบนด์วิดธ์ นอกจากนี้

โปรโตคอล NewRAPID จำเป็นต้องใช้ค่าน้ำหนักในการรวมค่าแบนด์วิดธ์ ซึ่งถ้าเรากำหนดค่าน้ำหนักต่ำ ๆ จะหมายถึงว่าเราเชื่อค่าแบนด์วิดธ์ที่ได้จากอัลกอริทึม IGI มากกว่า ซึ่งส่งให้โปรโตคอลค่อนข้างก้าวร้าวเพราะจะเปลี่ยนอัตราการส่งข้อมูลช้ากว่า แต่ถ้ากำหนดค่าน้ำหนักให้สูงขึ้น ก็จะหมายความว่าจะเชื่อค่าแบนด์วิดธ์ที่ได้จาก RAPID เดิมมากกว่า ก็จะส่งให้ผลให้การปรับตัวของโปรโตคอลทำได้เร็วขึ้น แต่ก็อาจจะเร็วจนเกินไปดังที่ได้กล่าวมาข้างต้น

ดังนั้นผู้วิจัยแนะนำว่าควรจะมีการศึกษาผลกระทบของตัวแปรเหล่านี้ที่มีต่อโปรโตคอลให้มากยิ่งขึ้น พร้อมกับควรจะทำให้การหาค่าตัวแปรเหล่านี้เป็นอัตโนมัติมากขึ้น เพราะในสถานการณ์จริงเราไม่สามารถกำหนดสภาพแวดล้อมที่ใช้ในการปฏิบัติงานได้

รายการอ้างอิง

[1] Allman, M., Paxson, V., and Stevens, W. TCP Congestion Control RFC 2581 (Proposed Standard) [Online]. 1999. Available from: http://www.ietf.org/rfc/rfc2581.txt [2011, October 1]

[2] Floyd, S. HighSpeed TCP for Large Congestion Windows. RFC 3649 (Experimental) [Online]. 2003. Available from: http://www.ietf.org/rfc/rfc3649.txt [2011, October 1]

[3] Rhee, I., Xu, L., and Ha, S. CUBIC for fast long-distance networks [Online]. Available from: http://tools.ietf.org/html/draft-rhee-tcp-cubic-00 [2011, October, 1]

[4] Xu, L., HarFoush, X., and Rhee, I. Binary increase congestion control for fast, long distance networks. Proceedings of IEEE INFOCOM (March 2004).

[5] Wei, D., Jin, C., and Low, S. FAST TCP: Motivation, architecture, algorithms, performance. IEEE/ACM Transactions on Networking (2006).

[6] Anderson, T., Collins, A., Krishnamurthy, A., and Zoharjan, J. PCP: Efficient endpoint congestion control. Proceedings of the 3rd Symposium on Networked Systems Design and Implementation (NSDI) (May 2006).

[7] Fukuhara, M., Hirose, F., Hatano, T., Shigeno, H., and Okada, K. SRF TCP: A TCP-friendly and fair congestion control method for high-speed networks. Proceedings of the International Conference on Principles of Distributed Systems (OPODIS) (2005).

[8] Kelly, T. Scalable TCP: Improving performance in highspeed wide area networks. First International Workshop on Protocols for Fast Longdistance Networks (February 2003).

[9] Shriram, A., and Kaur J. Empirical evaluations of techniques for measuring available bandwidth. Proceedings of IEEE INFOCOM 2007 (May 2007).

[10] Ribeiro, V., Riedi, R., Baraniuk, R., Navratil, J., and Cottrell, L. pathChirp: Efficient available bandwidth estimation for network paths. Passive and Active Measurement Workshop (April 2003).

[11] Konda, V., and Kaur, J. RAPID: Shrinking the Congestion-control Timescale. Proceedings of IEEE INFOCOM, Rio de Janeiro, Brazil (April 2009).

[12] Konda, V., and Kaur, J. Shrinking the timescales at which congestion control

operates. Technical Report Department of Computer Science, University of North Carolina at Chapel Hill (August 2008).

[13] NS2 Network simulator [Online]. Available from: http://www.isi.edu/nsnam/ns/ [2011, October 1]

[14] Weigle, M., Adurthi, P., Hernandez-Compos, F., Jeffay, K., and Smith, F. Tmix: A tool for generating realistic application workloads in ns-2. ACM SIGCOMM Computer Communication Review 26, 3 (2006): 67-76.

[15] Hu, N., and Steenkiste, P. Evaluation and Characterization of Available Bandwidth Probing Techniques. IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling 21 (August 2003): 6.

[16] Tan, W., Zhanikeev, M., and Tanaka, Y. Rate-based and Gap-Based Available Bandwidth Estimation Techniques in Cross-Traffic Context. Proceedings of 9th Asia-Pacific Network Operations and Management Symposium (September 2006).

[17] Heng, C. Empirical Study of Active Available Bandwidth Measurement Techniques [Online]. Available from: http://www.tlc-networks.polito.it/croce/cui_thesis.pdf [2011, Octorber 1]

[18] Lao, L., Dovrolis, C., and Sanadidi, M. The Probe Gap Model can Underestimate the Available Bandwidth of Multihop Paths. ACM SIGCOMM Computer Communication Review 36 (Octorber 2006): 5.

[19] Strauss, J., Katabi, D., and Kaashoek, F. A Measurement Study of Available Bandwidth Estimation Tools. Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement (2003).

[20] Jain, R., Chiu, D. W., and Hawe, W. R. A quantitative measure of fairness and discrimination for resource allocation in shared computer system. dEC Research Report TR-301 (September 1984).

[21] Xu, L., Ha, S., and Rhee, I. CUBIC: A new TCP-Friendly high-speed TCP variant. Proceeding of Workshop on Protocols for Fast Long Distance Networks (2005).

ประวัติผู้เขียนวิทยานิพนธ์

นายนพพร คงวัดใหม่ เกิดเมื่อวันที่ 5 ตุลาคม พ.ศ. 2524 จังหวัดนครศรีธรรมราช เป็นบุตรของนายสงคราม และนางกิมหวน คงวัดใหม่ สำเร็จการศึกษาระดับปริญญาตรี หลักสูตรสารสนเทศศาสตรบัณฑิต สาขาระบบสารสนเทศเพื่อการจัดการ จากมหาวิทยาลัยวลัยลักษณ์ เมื่อปี พ.ศ. 2547 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ในปีการศึกษา 2552 ณ ภาควิชาวิศวกรรมศาสตร์คอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย สำหรับชีวิตการทำงานนั้น ปัจจุบันทำงานที่บริษัท Reuters Software (Thailand) โดยดำรงตำแหน่ง Senior Software Engineer และมีประสบการณ์ทำงานรวมทั้งสิ้น 7 ปี

บทความทางวิชาการที่เกี่ยวข้องกับงานวิจัย และได้รับการเผยแพร่ในงานวิชาการ ประกอบด้วย 2 บทความ ตามลำดับดังนี้คือ

Enhancing TCP RAPID using IGI. The 11th International Symposium on Communications & Information Technologies (October 2011): 349.

Qualitative Study of TCP NewRAPID. 2012 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (May 2012).