การสร้างเหตุการณ์จำลองเพื่อแสดงประสิทธิภาพการทำงานของกลไกการจัดการคิวในบัฟเฟอร์ของเกตเวย์บนเครือข่าย TCP/IP โดยใช้โปรแกรมจำลองการทำงานของระบบเครือข่าย

PERFORMANCE SIMULATION OF BUFFER QUEUE MANAGEMENT MECHANISM IN TCP/IP NETWORK GATEWAY USING NETWORK SIMULATION TOOLS

โดย

ภควัต เชี่ยวเจริญวงศ์

ภัทรจิตร์ เปี่ยมด้วยธรรม

อารยะ จิระภิญโญ

อาจารย์ที่ปรึกษา

อาจารย์สุเมธ ประภาวัต

อาจารย์ที่ปรึกษาร่วม

รศ.ดร.โชติพัชร์ ภรณวลัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 2 ปีการศึกษา 2549

PERFORMANCE SIMULATION OF BUFFER QUEUE MANAGEMENT MECHANISM IN TCP/IP NETWORK GATEWAY USING NETWORK SIMULATION TOOLS

Pakavat Cheawjaleanvong

Phatrachit Piamduaytham

Araya Jirapinyo

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2/2006

ใบรับรองปริญญานิพนธ์ ประจำปีการศึกษา 2549

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การสร้างเหตุการณ์จำลองเพื่อแสดงประสิทธิภาพการทำงานของกลไกการจัดการคิวในบัฟเฟอร์ของเกตเวย์บนเครือข่าย TCP/IP โดยใช้โปรแกรมจำลองการทำงานของระบบเครือข่าย

Performance Simulation of Buffer Queue Management mechanism in TCP/IP Network Gateway using Network Simulation Tools

ผู้จัดทำ

1. นายภควัต	เชี่ยวเจริญวงศ์	รหัสประจำตัว 46060078
2. นางสาวภัทรจิตร์	เปี่ยมด้วยธรรม	รหัสประจำตัว 46060079
3. นายอารยะ	จิระภิญโญ	รหัสประจำตัว 46060102

.....อาจารย์ที่ปรึกษา

(อาจารย์สุเมธ ประภาวัต)

.....อาจารย์ที่ปรึกษาร่วม

(รศ.ดร. โชติพัชร์ ภรณวลัย)

หัวข้อ	การสร้างเหตุการณ์จำลองเพื่อแสดงประสิทธิภาพการทำงานของกลไกการจัดการคิวในบัฟเฟอร์ของเกตเวย์บนเครือข่าย TCP/IP โดยใช้โปรแกรมจำลองการทำงานของระบบเครือข่าย	
นักศึกษา	นายภควัต เชี่ยวเจริญวงศ์	**รหัสนักศึกษา** 46060078
	นางสาวภัทรจิตร์ เปี่ยมด้วยธรรม	**รหัสนักศึกษา** 46060079
	นายอารยะ จิระภิญโญ	**รหัสนักศึกษา** 46060102
ปริญญา	วิทยาศาสตรบัณฑิต	
สาขาวิชา	เทคโนโลยีสารสนเทศ	
ปีการศึกษา	2549	
อาจารย์ที่ปรึกษา	อาจารย์สุเมธ ประภาวัต	
อาจารย์ที่ปรึกษาร่วม	รศ.ดร. โชติพันธ์ ภรณวลัย	

บทคัดย่อ

โครงงานนี้ศึกษาการสร้างแบบจำลองด้วยโปรแกรมเอ็นเอสทู เพื่อศึกษาผลการทำงานของกลไกจัดการคิวในบัฟเฟอร์ของเกตเวย์โดยการจัดลำดับแพคเก็ตในคิว (Packet Scheduler) อย่างเช่น กลไกเอชทีบี ที่เป็นกลไกหนึ่งซึ่งใช้กันมาก และมีประสิทธิภาพในการควบคุมการจราจรบนเครือข่าย อีกทั้งถูกพัฒนาบนระบบปฏิบัติการลีนุกซ์ และสามารถเรียกใช้งานได้บนระบบปฏิบัติการลีนุกซ์รุ่นปัจจุบัน

กลไกเอชทีบีจึงถูกเลือกมาศึกษาโดยการสร้างแบบจำลองเลียนแบบการทำงานในโครงงานนี้ โดยใช้ภาษาซีพลัสพลัส ในการสร้างส่วนการจำลองการทำงานย่อยต่างๆ ของกลไก และใช้ภาษาโอทีซีแอล เพื่อรวบรวมการทำงานเข้าด้วยกัน แล้วสร้างการติดต่อกับผู้ใช้ เพื่อให้ได้แบบจำลองที่นำไปจำลองการทำงานได้ ในส่วนท้ายของโครงงานจะเป็นการแสดงผลการจำลองเลียนแบบเพื่อแสดงประสิทธิภาพของเครือข่าย ในกรณีที่ใช้กลไกการควบคุมการจราจรบนเครือข่ายกับกรณีที่ไม่มีการใช้กลไกดังกล่าว แล้วเปรียบเทียบผลที่ได้จากการจำลองเลียนแบบดังกล่าว กับผลที่คาดการณ์โดยการวิเคราะห์ด้วยทฤษฎีและหลักการทางคณิตศาสตร์ เพื่อแสดงความน่าเชื่อถือของแบบจำลองที่สร้างขึ้น

Title	Performance Simulation of Buffer Queue Management mechanism in TCP/IP Network Gateway using Network Simulation Tools
Student	Mr. Pakavat Cheawjaleanvong Mrs. Phatrachit Piamduaytham Mr. Araya Jirapinyo
Degree	Bachelor of Science
Programme	Information Science
Academic Year	2006
Advisor	Sumet Prabhavat
Co-advisor	Assoc. Prof. Dr. Chotipat Pornavalai

ABSTRACT

This project proposes simulation modeling using Network Simulator Tool version 2 (NS-2) to study the network performance enhancement when implements Hierachical Token Bucket (HTB) mechanism as a packet scheduler at network gateway. We choose the HTB because this mechanism has been developed and currently existed on several Linux platforms. In addition, the HTB has been widely implemented and performed as a traffic shaper.

Our simulation model is developed by C++ and OTCL (Object Tool Command Language) programming. We model the HTB subsystem functions using C++ coding; integrate all functions and provide user interface using OTCL coding. Our project provides two simple models for the samples and the guideline which may help in modifying models or adding more models. Finally, we represent the effectiveness of the HTB mechanism by simulation and analytical results in order to prove our proposed model.

กิตติกรรมประกาศ

โครงงานนี้จะไม่สามารถประสบความสำเร็จได้ถ้าขาดความรัก ความเอาใจใส่ ความช่วยเหลือและแรงสนับสนุนจากบุคคลหลายท่าน ผู้จัดทำใคร่ขอแสดงความระลึกถึงบุคคลสำคัญผู้อยู่เบื้องหลังต่อไปนี้

ขอขอบคุณ คุณแม่ สำหรับแรงใจ ที่มีให้ไม่เคยขาดแม้ยามท้อแท้และสิ้นหวัง แรงกาย ที่แม้ท่านเหนื่อยล้าเพียงใด ท่านสามารถเป็นทุ่นที่คอยทุ่มแรงเสมอ และทุนทรัพย์ สำหรับความเหนื่อยยากลำบากที่ส่งเสียตลอดมา

ขอขอบคุณ พี่สาว สำหรับอาหารว่างที่คอยสนับสนุนไม่ให้ขาด น้องสาว สำหรับการดูแลเรื่องจิปาถะที่ไม่ให้รำคาญใจ จนผู้เขียนสามารถมีวันนี้ได้

ขอขอบคุณ อาจารย์สุเมธ ประภาวัต เป็นอาจารย์ที่ปรึกษาโครงการ ที่กรุณาให้คำปรึกษา ให้กำลังใจและให้คำแนะนำทั้งในเรื่องต่างๆที่เป็นประโยชน์ยิ่งจนทำให้โครงการนี้สำเร็จลุล่วงไปได้ด้วยดี

ขอขอบคุณ เพื่อนๆ ทั้งในคณะและนอกคณะทุกท่านที่เสมือนพลังขับเคลื่อนเมื่อยามหมดแรงคอยเติมความช่วยเหลือ และเติมกำลังใจมาโดยตลอด

ขอขอบคุณ คอมพิวเตอร์พกพายี่ห้อ BenQ รุ่น T31 และ Apple รุ่น MacBook ที่ช่วยประมวลผลและเก็บข้อมูลรายงานทั้งหมดตลอดที่ทำมา

ขอขอบคุณ ห้องแลป 510 ที่ให้ได้ใช้เป็นที่จัดทำโครงงานและค้างแรม

ขอขอบคุณ คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้โอกาสพวกเราได้ศึกษาและจบการศึกษาเป็นนักศึกษาของสถาบันแห่งนี้ ซึ่งเป็นที่ภูมิใจของพวกเราอย่างยิ่ง

และสุดท้าย ขอขอบคุณ คณะผู้จัดทำโครงงานที่เสียสละ ตรากตรำ บากบั่น รักและสามัคคีจนเป็นพลังที่สามารถพายเรือลำนี้ให้ถึงฝั่ง

จึงใคร่ขอขอบคุณบุคคลดังกล่าวข้างต้นมา ณ โอกาสนี้

นายภควัต เชี่ยวเจริญวงศ์
นางสาวภัทรจิตร์ เปี่ยมด้วยธรรม
นายอารยะ จิระภิญโญ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

สารบัญรูป (ต่อ)

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันการวิเคราะห์ประสิทธิภาพของเครือข่ายสามารถกระทำได้ 3 วิธี การวิเคราะห์ด้วยหลักการทางคณิตศาสตร์ (Analysis Model) การจำลอง (Simulation) และการวัดจากระบบจริง (Measurement) ซึ่งปัญหาที่เกิดขึ้นของการวิเคราะห์ประสิทธิภาพของเครือข่าย คือการกระทำด้วยวิธีใดสามารถทำให้ได้ผลลัพธ์ที่ถูกต้อง แม่นยำอีกทั้งสามารถกระทำได้ง่ายและประหยัดค่าใช้จ่าย

การจำลองคือรูปแบบของการจำลองระบบโดยรูปแบบการจำลองเป็นไปตามระบบจริงมากที่สุด สามารถแก้ปัญหาที่กล่าวมาข้างต้นได้เนื่องจากเป็นความเป็นจริงที่การวิเคราะห์ประสิทธิภาพของเครือข่ายจากระบบจริงได้ผลลัพธ์ที่ถูกต้อง แม่นยำแต่มีการลงทุนสูงและกระทำได้ลำบาก หรือการวิเคราะห์ประสิทธิภาพของเครือข่ายด้วยหลักการทางคณิตศาสตร์ผลลัพธ์ที่ได้เป็นไปแนวทางทฤษฎี ถึงแม้ว่ามีการลงทุนที่ต่ำกว่าและกระทำได้ง่าย การจำลองเป็นการวิเคราะห์ประสิทธิภาพเครือข่ายที่มีประโยชน์เนื่องจากเป็นวิธีการที่ง่ายและรวดเร็ว อีกทั้งการจำลองมีความยืดหยุ่นในการสร้างแบบจำลองให้ได้แบบจำลองที่เสมือนจริงที่สุด อย่างไรก็ตามการจำลองอาจทำให้ผลที่ผิดพลาดได้เช่น ไม่มีความรู้ในการจำลองแบบจำลอง หรือมีความรู้ทางด้านสถิติไม่เพียงพอ

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

รายงานฉบับนี้มุ่งหวังเพื่อสร้างแบบจำลองเพิ่มเติมลงบนโปรแกรม NS-2 การสร้างแบบจำลองเพิ่มเติมลงบนโปรแกรม NS-2 สามารถจำลองระบบเครือข่ายให้เสมือนจริง ไม่ว่าเป็นการจำลองระบบเครือข่ายแบบง่าย จนถึงระบบเครือข่ายที่มีความซับซ้อน อีกทั้งยังสามารถใช้แบบจำลองนี้เพื่อสร้างสถานการณ์จำลองในการวิเคราะห์ประสิทธิภาพของเครือข่ายและนำผลลัพธ์ที่ได้มากระทำการศึกษาเพื่อปรับปรุงหรือแก้ไขให้ได้ประสิทธิภาพดีที่สุดเพื่อนำไปอ้างอิงการสร้างระบบจริง

ดังนั้นในรายงานฉบับนี้นำเสนอการเพิ่มเติมโมดูลการสร้างแบบจำลองการควบคุมการจราจรบนเครือข่ายโดยลอกเลียนการทำงานบางส่วนของ HTB ในโปรแกรม NS-2 เพื่อใช้สำหรับจำลองการควบคุมการจราจรบนเครือข่ายและนำแบบจำลองที่จำลองได้มาเปรียบเทียบกับทฤษฎีและหลักการทางคณิตศาสตร์ของการควบคุมจราจรบนเครือข่ายเพื่อให้ได้ผลลัพธ์ของการจำลองเครือข่ายที่เสมือนจริงและสามารถนำไปใช้ในการสร้างแบบจำลองเพื่อศึกษาเกี่ยวกับเรื่องการควบคุมการจราจรบนเครือข่าย

1.3 สมมติฐานของการศึกษา

ข้อจำกัดของการสร้างแบบจำลองด้วยโปรแกรม NS-2 คือ ขาดโมดูลที่ต้องสร้างแบบจำลอง เนื่องจากบางครั้งในการสร้างแบบจำลองต้องการบางโมดูลที่ไม่มีในโปรแกรม NS-2 อีกทั้งต้องมีความเข้าใจในทฤษฎีและหลักการของโมดูลที่ต้องการเพิ่มเติมในโปรแกรม NS-2 เพื่อสามารถนำมาสร้างแบบจำลองที่เสมือนจริงให้กับการจำลองระบบเครือข่ายได้อย่างถูกต้อง

การแก้ปัญหาข้างต้นนี้ ต้องศึกษาวิธีเพิ่มโมดูลที่ต้องการในการสร้างแบบจำลอง และศึกษาทฤษฎีและหลักการของโมดูลที่ต้องการเพิ่มเติมในโปรแกรม NS-2 เพื่อสามารถเพิ่มเติมโมดูลในโปรแกรม NS-2 ให้ได้การทำงานที่ถูกต้อง ซึ่งนำไปสู่การสร้างแบบจำลองที่ได้ผลลัพธ์ที่เสมือนจริงมากที่สุด

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

การเพิ่มเติมโมดูลการสร้างแบบจำลองการควบคุมการจราจรบนเครือข่ายในโปรแกรม NS-2 ก่อนอื่นต้องรู้วิธีในการเพิ่มเติมโมดูลการทำงานใน โปรแกรม NS-2 จากนั้นเข้าใจเรื่องทฤษฎีและหลักการทำงานของกลไลการจัดการจราจรบนเครือข่ายโดยลอกเลียนการทำงานบางส่วนของ HTB และขั้นตอนสุดท้าย คือการเพิ่มอัลกอริทึมของโมดูลในการสร้างแบบจำลองการควบคุมการจราจรบนเครือข่าย ลงในโปรแกรม NS-2 และเพื่อให้การสร้างแบบจำลองของโมดูลการสร้างแบบจำลองการควบคุมการจราจรบนเครือข่ายในโปรแกรม NS-2 มีความน่าเชื่อถือและสามารถเชื่อมั่นได้ว่าการจำลองบนโปรแกรม NS-2 นั้นสามารถนำไปสร้างเหตุการณ์จำลองในลักษณะอื่นเพื่อวัดประสิทธิภาพของระบบเครือข่าย เราจึงต้องนำผลลัพธ์ที่ได้จากการวิเคราะห์ด้วยการจำลองเหตุการณ์การจราจรบนเครือข่าย โดยใช้แบบจำลองโปรแกรม NS-2 เปรียบเทียบกับทฤษฎีและหลักการทางคณิตศาสตร์ของการควบคุมจราจรบนเครือข่าย ซึ่งในรายงานฉบับนี้ได้แสดงผลลัพธ์ของการวิเคราะห์ประสิทธิภาพของระบบเครือข่ายด้วยกราฟเพื่อเปรียบเทียบวิธีการที่นำเสนอกับหลักการทางคณิตศาสตร์ ในการวิเคราะห์ประสิทธิภาพของระบบเครือข่ายต้องมีความรู้และความเข้าใจเรื่องทฤษฎีและหลักการทำงานของกลไลการจัดการจราจรบนเครือข่ายและการวัดประสิทธิภาพของระบบเครือข่าย เช่น หลักการและทฤษฎีทรูพุต(Throughput) หลักการและทฤษฎีความหน่วง(Delay) เพื่อที่สามารถนำผลลัพธ์ที่ได้มาวิเคราะห์หาประสิทธิภาพของระบบเครือข่าย

1.5 ขอบเขตการวิจัย

ในรายงานฉบับนี้นำเสนอการเพิ่มเติมโมดูลการสร้างแบบจำลองการควบคุมการจราจรบนเครือข่ายโดยลอกเลียนการทำงานบางส่วนของ HTB ในโปรแกรม NS-2 ซึ่งใช้ผลลัพธ์ที่ได้จากการจำลองเสมือนจริง (Simulation) เปรียบเทียบกับทฤษฎีและหลักการทางคณิตศาสตร์ของการควบคุมจราจรบนเครือข่าย เพื่อสามารถนำโมดูลการสร้างแบบจำลองการควบคุมการจราจรบนเครือข่ายโดยลอกเลียนการทำงานของ HTB ไปใช้แทนระบบจริง

1.6 ขั้นตอนของการศึกษา

รายงานฉบับนี้ได้แบ่งเนื้อหาออกเป็น 5 บทด้วยกันคือ

บทที่ 1 กล่าวถึงความเป็นมาของงานวิจัยและความสำคัญของปัญหา ความมุ่งหมายและวัตถุประสงค์ สมมติฐาน ทฤษฎีที่ใช้ ขอบเขตของการวิจัย และขั้นตอนการศึกษา

บทที่ 2 กล่าวถึงทฤษฎีและหลักการทำงานของการควบคุมการจราจรบนเครือข่ายด้วยกลไกชนิด HTB

บทที่ 3 กล่าวถึงการเพิ่มเติมโมดูลการสร้างแบบจำลองการควบคุมการจราจรบนเครือข่ายโดยลอกเลียนการทำงานบางส่วนของ HTB

บทที่ 4 กล่าวถึงการทดลองเพื่อแสดงให้เห็นถึงประสิทธิภาพของโมดูลการสร้างแบบจำลองการควบคุมการจราจรบนเครือข่ายโดยลอกเลียนการทำงานบางส่วนของ HTB

บทที่ 5 เป็นบทสรุปผลการดำเนินโครงงานและข้อเสนอแนะ

บทที่ 2
หลักการพื้นฐานการควบคุมปริมาณทราฟฟิคในระบบเครือข่าย

ในหัวข้อนี้อธิบายถึงทฤษฎีและหลักการพื้นฐานการจัดการจราจรบนเครือข่ายและกลไกการจัดการจราจรบนเครือข่ายแบบ HTB

2.1 แนวคิดเกี่ยวกับการจัดการจราจรบนเครือข่าย

ในปัจจุบันมีการใช้ระบบเครือข่ายเพิ่มมากขึ้น และเพื่อให้เกิดความยุติธรรมในการใช้ทรัพยากรที่มีอยู่จำกัด จำเป็นต้องทำการแบ่งแบนด์วิดท์สำหรับทุกการสื่อสารให้เท่าเทียมกันทำให้อาจจะมีผลต่อทราฟฟิคบางประเภทได้ นอกจากนี้ในการควบคุมปริมาณทราฟฟิคของผู้ให้บริการอินเตอร์เน็ต (Internet Service Provider) ได้ควบคุมปริมาณทราฟฟิคการแชร์ไฟล์ในรูปแบบ Peer-to-Peer เนื่องจากทราฟฟิคในลักษณะนี้มีการใช้แบนด์วิดท์มากเกินกว่าที่คาดการณ์ไว้ ทำให้ต้องมีการควบคุมการใช้แบนด์วิดท์เพื่อลดค่าความแปรปรวนของค่าความหน่วง (Delay Jitter) และลดจำนวนการสูญหายของแพคเก็ต (Packet loss) ในระบบเครือข่าย

2.2 หลักการการจัดการจราจรบนเครือข่าย

การจัดการจราจรบนเครือข่ายเป็นกลไกที่ใช้ในการควบคุมปริมาณข้อมูลที่มีการรับส่งภายในระบบเครือข่าย โดยมีการจัดลำดับสำหรับทราฟฟิคที่จะถูกส่งออกเพื่อให้เครือข่ายทำงานได้อย่างมีประสิทธิภาพดีที่สุด ดังนั้นจำเป็นต้องทำการคัดแยกประเภททราฟฟิคในเครือข่าย , จัดระบบแถวคอยสำหรับการเข้าใช้บริการทรัพยากร , กำหนดนโยบายการให้บริการ และความยุติธรรมในการจัดการทราฟฟิค โดยมีผู้ดูแลระบบจัดการหรือปรับปรุงการทำงานระบบแถวคอยของแพคเก็ตตามความเหมาะสม และผลที่ได้รับจากการควบคุมปริมาณทราฟฟิคคือ ค่าความแปรปรวนของค่าความหน่วงลดลง , การสูญหายของแพคเก็ตลดลง และค่าความหน่วงในการส่งข้อมูลลดลง

ในปัจจุบันมีรูปแบบในการจัดการแบนด์วิดท์หลักๆด้วยกัน 3 วิธี ดังนี้

2.2.1 การจัดการแบนด์วิดท์ตามแอพพลิเคชั่น (Per-application rules)

สามารถกำหนดได้ว่าจะควบคุมทราฟฟิคชนิดใด ประเภทใด ลักษณะใด โดยการกำหนดแบนด์วิดท์ให้กับแต่ละทราฟฟิคจะต้องกำหนดค่าไม่เกินค่าที่ต้องการจัดการ เช่น ต้องการจำกัดปริมาณทราฟฟิคชนิด FTP ไม่เกิน 6 เมกะบิตต่อวินาที หมายความว่า ในหนึ่งวินาทีจะไม่สามารถส่งข้อมูลชนิด FTP ได้มากกว่า 6 เมกะบิต นอกจากใช้ประเภท และลักษณะของแอพพลิเคชั่นใน

การควบคุมปริมาณทราฟฟิคแล้ว ยังสามารถจะกำหนดตามลักษณะที่มองเห็นได้ (macroscopic characteristic) เช่น ทราฟฟิคโปรโตคอล (Traffic's protocol) IP , IPX , AppleTalk , DECNet , etc. หรือควบคุมโดยใช้พอร์ตของแอพพลิเคชั่น เช่น Kazaa ใช้พอร์ต 1214

2.2.2 การจัดการแบนด์วิดท์ตามการใช้งานของผู้ใช้ (Per-user rules)

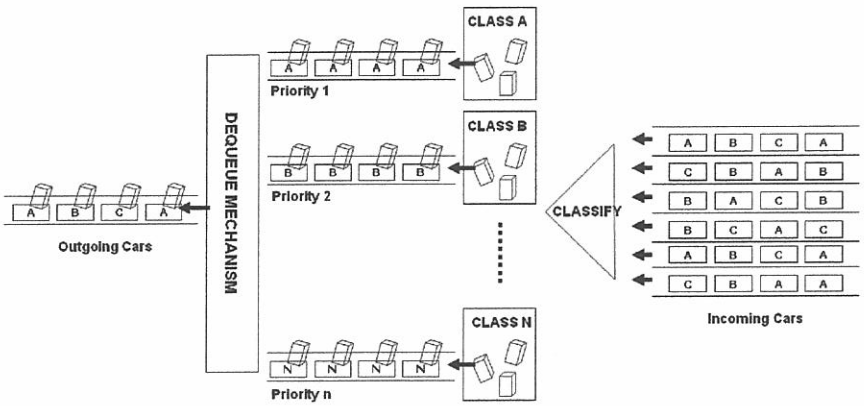
สามารถทำการกำหนดปริมาณทราฟฟิคของผู้ใช้แต่ละคนได้ เพื่อให้มั่นใจได้ว่าผู้ใช้แต่ละคนได้รับแบนด์วิดท์เพื่อใช้งานเท่ากันทุกคน เช่น กำหนดให้ผู้ใช้แต่ละคนได้รับแบนด์วิดท์ในการใช้งาน 256 กิโลบิตต่อวินาที เมื่อมีการจำกัดปริมาณแบนด์วิดท์ให้ผู้ใช้แต่ละคนแล้ว ทำให้สามารถแก้ปัญหาการยึดครองช่องสัญญาณเป็นเวลานานๆ หรือว่าใช้ปริมาณแบนด์วิดท์มากๆ ได้

2.2.3 การจัดการแบนด์วิดท์ตามความสำคัญในการใช้งาน (Priority management)

สามารถทำการกำหนดปริมาณทราฟฟิคตามระดับความสำคัญของการใช้งานซึ่งแตกต่างกันไปในแต่ละประเภทของทราฟฟิคที่ใช้ เช่น กลุ่มงานวิจัยเครือข่าย ใช้ในการเรียนการสอน และเพื่อการวิจัยจะได้รับระดับความสำคัญสูง แต่สำหรับทราฟฟิคอื่นๆ เช่น เกมออนไลน์ หรือ การแชร์ไฟล์ลักษณะ Peer-to-Peer จะได้รับการจัดสรรแบนด์วิดท์สำหรับใช้งานเครือข่ายก็ต่อเมื่อแอพพลิเคชั่นที่สำคัญไม่ต้องการใช้งาน

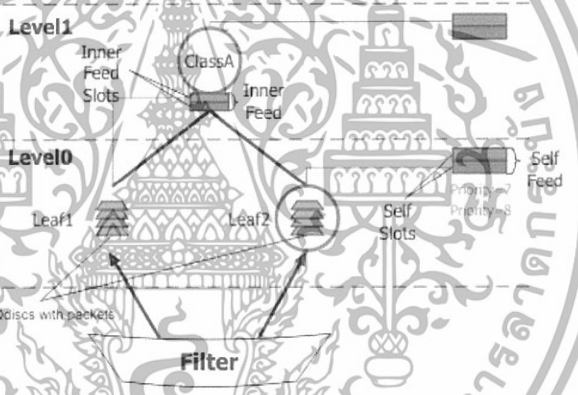
2.3 ทฤษฎีและหลักการทำงานของโทเคน/บัคเก็ตแบบลำดับชั้น (Hierarchical Token Bucket: HTB)

การควบคุมปริมาณเครือข่ายด้วยเทคนิค HTB ถูกพัฒนาจากแนวคิดของกลไก Token Bucket และการทำงานของ CBQ ซึ่งถูกนำไปประยุกต์ใช้งานในเรื่องการควบคุมปริมาณเครือข่ายได้หลายด้าน เช่น การจำกัดแบนด์วิดท์ (Bandwidth Shaping) และการรับประกันแบนด์วิดท์ (Bandwidth Guarantee) เป็นต้น ในโครงสร้างการทำงานของ HTB แพคเก็ตจะถูกจับคู่กับโทเคนก่อต่อเมื่อแพคเก็ตนั้นถูกจัดอยู่ในคลาสสุดท้าย ซึ่งโครงสร้างแบบลำดับชั้นสามารถสร้างคลาสได้สูงสุด 10,000 คลาส โดยมีความลึกได้ 8 ระดับ



รูปที่ 2.1 แสดงตัวอย่างการควบคุมปริมาณทราฟฟิคด้วย HTB

2.3.1 ส่วนประกอบของ HTB



รูปที่ 2.2 แสดงส่วนประกอบของกลไก HTB

- **queuing discipline (qdisc)** - ข้อกำหนดของคิว เป็นอัลกอริธึมที่ใช้สำหรับจัดการแพคเก็ตในคิว ทำให้สามารถกำหนดลำดับการส่งออกของแพ็ตเก็ต อีกทั้งยังสามารถกำหนดว่าแพ็ตเก็ตถูกดร๊อปหรือถูกหน่วง
- **filter** - กระบวนการที่ใช้สำหรับการแยกและจัดกลุ่มแพ็คเก็ต
- **level** - ระดับชั้นของคลาสในโครงสร้างแบบลำดับชั้น(hierarchy)
- **parent class** - คลาสราก ซึ่งถือได้ว่าเป็นคลาสแม่ตั้งต้น
- **inner class** - คลาสแม่ที่ประกอบด้วยคลาสลูกมากกว่าหนึ่ง ซึ่ง Inner classes มีหน้าที่ในการควบคุมการจราจรบนเครือข่าย ดังนั้นในคลาสนี้จึงไม่มีการจัดเก็บแพคเก็ต

- **leaf class** - คลาสที่มีคลาสแม่แต่ไม่มีคลาสลูกต่อท้ายหรือถือได้ว่าเป็นคลาสระดับชั้นล่างสุด(ระดับชั้น เท่ากับ ศูนย์)ของโครงสร้างแบบลำดับชั้น

คลาสทุกคลาสที่เป็นคลาสแม่จะประกอบไปด้วยคลาสลูกมากกว่า 1 คลาส ทำหน้าที่ควบคุมการจราจรบนเครือข่าย ให้กับคลาสระดับชั้นถัดมา และ สำหรับคลาสทุกคลาสที่ไม่มีคลาสลูกประกอบถือว่าเป็นคลาสในระดับศูนย์ หรือเราเรียกว่า Leaves class ซึ่งคลาสในระดับชั้นนี้จะถูกจัดการด้วยคิว

- **Real traffic**(R') - ทราฟฟิคที่วิ่งเข้าสู่ระบบ
- **Assured rate**(AR) - อัตราการส่งข้อมูลขั้นต่ำที่ทราฟฟิคแต่ละคลาสได้รับและถือว่าเป็นการการันตีอัตราการส่งข้อมูลให้กับทราฟฟิคคลาสนั้น
- **Ceil rate**(CR) - อัตราการส่งข้อมูลสูงสุดที่ทราฟฟิคคลาสนั้นสามารถที่จะส่งข้อมูลได้
- **Actual rate**(R) - อัตราการส่งข้อมูลที่เชปเปอร์อนุญาตให้ทราฟฟิคในคลาสนั้นใช้ ณ ช่วงเวลาขณะหนึ่ง
- **Real rate** (R'') - อัตราการส่งข้อมูลออกจริงของคลาส ณ ช่วงเวลาขณะหนึ่ง
- **Remain bandwidth**($\hat{R}$) - แบนด์วิดท์ที่เหลือให้คลาสที่ต้องการยืมแบนด์วิดท์ยืมได้ ณ ช่วงเวลาขณะหนึ่ง
- **priority** - ลำดับของคลาสที่ถูกรับบริการในระดับชั้นเดียวกัน(โดยความสำคัญเรียงจาก 1 ถึง 8 โดยที่ 1 มีความสำคัญสูงสุด)

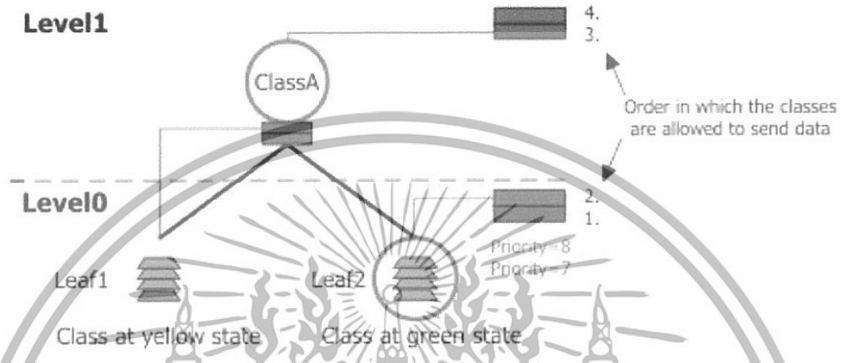
2.3.2 สถานะของการทำงานของ HTB คลาส

คลาสใน HTB สามารถเปลี่ยนสถานะได้ 3 สถานะ โดยขึ้นอยู่กับอัตราในการส่งข้อมูล

- **green** - สถานะของคลาสที่มีทราฟฟิคที่วิ่งเข้าสู่ระบบจริงที่น้อยกว่า Assured rate ดังนั้นคลาสในสถานะนี้จะได้อัตราในการส่งข้อมูลเท่ากับทราฟฟิคจริง
- **yellow** - สถานะของคลาสที่มีทราฟฟิคที่วิ่งเข้าสู่ระบบจริงมากกว่าหรือเท่ากับ Assured rate แต่น้อยกว่าหรือเท่ากับ Ceil rate ดังนั้นคลาสในสถานะนี้จะได้อัตราในการส่งข้อมูลอย่างน้อยเท่ากับอัตราการส่งข้อมูลที่การันตี หรือ มากกว่านั้นในกรณีที่คลาสแม่มีแบนด์วิดท์ให้คลาสนั้นยืม(คลาสแม่ต้องมีสถานะ green)
- **red** - สถานะของคลาสที่มีทราฟฟิคที่วิ่งเข้าสู่ระบบจริงมากกว่า Ceil rate ดังนั้นคลาสในสถานะนี้จะได้อัตราในการส่งข้อมูลไม่เกิน Ceil rate

2.3.3 ระดับความสำคัญ (Priority)

เมื่อคลาสระดับชั้นสุดท้ายต้องการส่งแพคเก็ตออกจากคลาส กระบวนการ HTB มีการตรวจสอบ ระดับความสำคัญ ก่อนซึ่งคลาสในระดับชั้นสุดท้ายที่มีความสำคัญสูงสุด (Priority=1) จะได้รับการดำเนินการก่อนและไล่ตามลำดับความสำคัญไปจนถึงคลาสในระดับชั้นสุดท้ายที่มีความสำคัญน้อยที่สุด (Priority =8) ซึ่งในการเปรียบเทียบระดับความสำคัญใช้ได้กับคลาสที่อยู่ในระดับชั้นเดียวกันเท่านั้น



รูปที่ 2.3 แสดงระดับความสำคัญของคลาส HTB

จากรูป 2.3 เป็นการเปรียบเทียบลำดับความสำคัญของคลาสที่อยู่ระดับชั้นเดียวกันระหว่างคลาสที่มีการยืมแบนด์วิดท์จากคลาสแม่ กับคลาสที่ไม่มีการยืมแบนด์วิดท์ ถึงแม้ว่าตามหลักการแล้วคลาสใดมีระดับความสำคัญสูงกว่าจะได้รับบริการก่อนซึ่งในที่นี้ Leaf1 ก็มีความสำคัญสูงกว่า Leaf2 (Priority Leaf1=7, Priority Leaf2=8) ดังนั้นควรได้รับบริการก่อน แต่ HTB ให้ลำดับความสำคัญของระดับชั้นมากกว่าระดับความสำคัญ ดังนั้นในกรณีนี้ Leaf1 มีการยืมจากคลาส A จะถือว่า Leaf1 มีระดับชั้นเดียวกับคลาส A คือ ระดับชั้น เท่ากับ หนึ่ง ในขณะเดียวกัน Leaf2 (level=0) ซึ่งหมายความว่า Leaf2 มีระดับชั้นที่ต่ำกว่า Leaf1 (level=1) ดังนั้น ใน HTB จะถือว่า Leaf2 มีความสำคัญสูงกว่า Leaf1 และ Leaf2 ได้รับบริการก่อน Leaf1

แต่สำหรับกรณีที่คลาสลูกที่มีแม่เดียวกันมีระดับความสำคัญเท่ากันและมีสถานะเดียวกัน ดังนั้น ในการให้บริการของคลาสแต่ละคลาสจะถูกจัดการด้วยวิธีวนรอบ(Round robin) คือ ทุกคลาสจะได้รับบริการไปทีละคลาสและวนเป็นรอบเท่าๆกัน

2.3.4 ข้อแนะนำในการกำหนดค่า Assure rate และ Ceil rate ให้กับแต่ละคลาส

$$AR_B + AR_C + AR_D \leq AR_A \qquad (2.1)$$

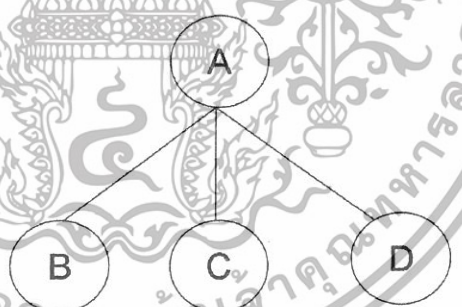
$$CR_i \leq AR_A \qquad (2.2)$$

เมื่อ i = {B, C, D}

จากสมการ 2.1 และ 2.2 เป็นทฤษฎีการกำหนดขอบเขตของ AR และ CR ซึ่งสามารถนำมาประยุกต์สำหรับการกำหนด AR และ CR ในคลาสทรีรูปแบบอื่น ซึ่งจะยกตัวอย่าง คลาสทรี 2 แบบคือ คลาสทรี 2 ระดับชั้น 3 Leaves class และ คลาสทรี 3 ระดับชั้น และ 3 Leaves class สำหรับการกำหนด Assure rate จำเป็นต้องรู้สัดส่วนระหว่าง Leaves class ในระดับชั้นเพื่อใช้ในการแบนด์วิดท์ที่ HTB อนุญาตให้ทราฟฟิคคลาสนั้นส่งออก

ตัวอย่าง 1 การกำหนด Assure rate และ Ceil rate คลาสทรี 2 ระดับชั้น และ 3 Leaves class

กำหนดให้ แบนด์วิดท์ที่ได้รับเท่ากับ X Mbps และมีอัตราส่วนของ Assure rate ระหว่างคลาส B, C และ D เป็น b:c:d ตามลำดับ



รูปที่ 2.4 แสดงคลาสทรี 2 ระดับชั้น และ 3 Leave class

จากสมการหลักข้างต้นทั้ง 2 สมการสามารถประยุกต์เข้ากับการใช้คลาสทรีนี้ ได้สมการสรุปสำหรับการกำหนดค่า Assure rate และ Ceil rate ดังนี้

โดยปกติแล้ว AR_A จะกำหนดให้เท่ากับ CR_A และเท่ากับ แบนด์วิดท์ที่ได้รับ

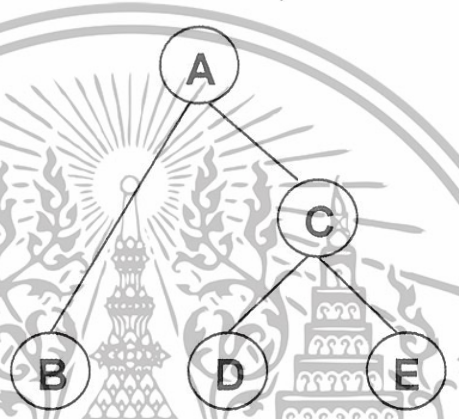
$$AR_B = \frac{b}{b+c+d} AR_A \quad , CR_B \leq AR_A \qquad (2.3)$$

$$AR_C = \frac{c}{b+c+d} AR_A \ , \ CR_C \le AR_A \qquad (2.4)$$

$$AR_D = \frac{d}{b+c+d} AR_A \ , CR_D \le AR_A \qquad (2.5)$$

ตัวอย่าง 2 การกำหนด Assure rate และ Ceil rate คลาสทรี 3 ระดับชั้น และ 3 Leaves class

กำหนดให้ แบนด์วิดท์ที่ได้รับเท่ากับ X Mbps และมีอัตราส่วนของ Assure rate ระหว่างคลาสเป็น B,C เป็น b:c และคลาส D,E เป็น d:e ตามลำดับ



รูปที่ 2.5 แสดงคลาสทรี 3 ระดับชั้น และ 3 Leave class

จากสมการหลักข้างต้นสามารถประยุกต์เข้ากับคลาสทรีนี้ได้และได้สมการสรุปสำหรับการกำหนดค่า Assure rate และ Ceil rate ดังนี้

โดยปกติแล้ว AR_A จะกำหนดให้เท่ากับ CR_A และเท่ากับ แบนด์วิดท์ที่ได้รับ

$$AR_E = \frac{1}{(b+1)(1+\frac{d}{e})} AR_A \ , \ CR_E \le AR_C \qquad (2.6)$$

$$AR_D = \frac{d}{e}(\frac{1}{(b+1)(1+\frac{d}{e})}) AR_A \ , \ CR_D \le AR_C \qquad (2.7)$$

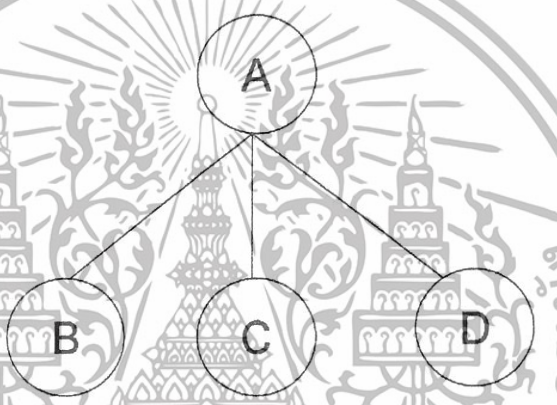
$$AR_C = (1+\frac{d}{e}) AR_E \ , \ CR_C \le AR_A \qquad (2.8)$$

$$AR_B = AR_A - (1+\frac{d}{e}) AR_E \ , \ CR_B \le AR_A \qquad (2.9)$$

2.3.5 การยืมคืนแบนด์วิดท์ (Bandwidth Borrowing)

การยืมแบนด์วิดท์เป็นกลไกลพื้นฐานส่วนหนึ่งของ HTB ซึ่งคลาสลูกจะสามารถยืมโทเค็นจากคลาสแม่ได้ถ้ามีการใช้งานแบนด์วิดท์เกินกว่าโทเค็นที่มีอยู่ โดยคลาสลูกจะพยายามยืมแบนด์วิดท์จากคลาสแม่จนกระทั่งค่าแบนด์วิดท์รวมทั้งหมดไม่เกินค่า Ceil rate ที่มีกำหนดไว้

ใน HTB คลาสลูกที่ต้องการยืมแบนด์วิดท์จากแม่ ได้มีการคำนวณขนาดแบนด์วิดท์ที่คลาสแม่สามารถแบ่งให้คลาสลูกแต่ละคลาสยืมได้ ซึ่งแบนด์วิดท์ที่แต่ละคลาสลูกสามารถยืมได้จะมีขนาดไม่เท่ากันโดยพื้นฐานขึ้นกับค่าควันตั้ม กับขนาดแบนด์วิดท์ของคลาสแม่ที่สามารถให้ยืมได้ โดยสามารถคำนวนได้จากสมการ ดังนี้



รูปที่ 2.6 แสดงรูปแบบพื้นฐานของกลไก HTB

AR_i เมื่อ i = {A, B, C, D} คือ Assure rate ของคลาส i

CR_i เมื่อ i = {A, B, C, D} คือ Ceil rate ของคลาส i

R_i เมื่อ i = {B, C, D} คือ Actual rate ของคลาส i

R_p คือ ค่า Actual rate ของคลาสราก (Root class) ในที่นี้ คือ คลาส A

R_i' เมื่อ i = {B, C, D} คือ ทราฟฟิคที่วิ่งเข้าสู่ระบบจริง

R_i'' เมื่อ i = {B, C, D} คือ Real rate ของคลาส i

$\hat{R}$ คือ แบนด์วิดท์ที่เหลือให้คลาสลูกยืม

B_i เมื่อ i = {B, C, D} คือ Borrow rate ของคลาส i

Q_i เมื่อ i = {B, C, D} คือ ค่าควันตั้มของคลาส i

$$B_i = \frac{Q_i \hat{R}}{\sum_{i=B}^{D} Q_i} \qquad (2.10)$$

เมื่อ $\sum_{i=B}^{D} Q_i$ เป็นผลรวมค่า Q เฉพาะคลาสที่ต้องการยืมแบนด์วิดท์

i = {B, C, D}

จากสมการที่ 2.10 เราสามารถหา แบนด์วิดท์ที่คลาส i สามารถยืมจากคลาสแม่ ได้จากการนำค่าควันตั้มของคลาสนั้น หารด้วยผลรวมค่าควันตั้มของคลาสลูกที่มีการยืมแบนด์วิดท์ จากนั้นคูณด้วย อัตราการส่งข้อมูลที่เหลือให้คลาสลูกยืม จากสมการนี้สรุปได้ว่า แบนด์วิดท์ที่ยืมได้ของคลาส i หาได้จากการ การแบ่งส่วนของแบนด์วิดท์ที่เหลือให้ยืมด้วยค่าน้ำหนักของคลาส i

$$B_i = 0 \quad \text{Otherwise} \qquad (2.11)$$

จากสมการที่ 2.11 บ่งบอกว่าเมื่อคลาสนั้นไม่มีการยืมแบนด์วิดท์จากคลาสแม่ ดังนั้นแบนด์วิดท์ที่คลาส i สามารถยืมจากคลาสแม่จึงมีค่าเท่ากับศูนย์ กรณีนี้ถือว่าคลาสนั้นเป็นคลาสแม่จึงไม่มีการยืมแบนด์วิดท์ หรือ คลาสนั้นไม่มีการยืมแบนด์วิดท์

$$\hat{R} = R_p - \sum_{i=B}^{D} (R_i' + AR_i) \qquad (2.12)$$

เมื่อ $R_i' = 0$ เมื่อ คลาส i มีการยืม และ

$AR_i = 0$ เมื่อ คลาส i ไม่มีการยืม

i คือ คลาสลูก(leaves class), i = {B, C, D}

จากสมการที่ 2.12 เราสามารถหา แบนด์วิดท์ที่เหลือให้คลาสยืมได้จากการนำ อัตราการส่งข้อมูลขั้นต่ำที่การันตีของคลาสราก (Root class) หักออกด้วยทราฟฟิคจริงที่เข้าสู่คลาสในทุกๆคลาสลูก (ในกรณีที่ทราฟฟิคจริงที่เข้าสู่คลาส i น้อยกว่า อัตราการส่งข้อมูลขั้นต่ำที่การันตีของคลาส i) หรือ หักออกด้วยอัตราการส่งข้อมูลขั้นต่ำที่การันตีของคลาสในทุกๆคลาสลูก (ในกรณีที่ทราฟฟิคจริงที่เข้าสู่คลาส i มากกว่า อัตราการส่งข้อมูลขั้นต่ำที่การันตีของคลาส i) เพื่อให้ได้อัตราการส่งข้อมูลที่เหลือเพื่อสามารถนำไปใช้ในการคำนวณอัตราการส่งข้อมูลที่แต่ละคลาสสามารถยืมได้

$$R_i = \min(CR_i, AR_i + B_i) \quad (2.13)$$

เมื่อ $i = \{B, C, D\}$

จากสมการที่ 2.13 อัตราการส่งข้อมูลที่เชปเปอร์อนุญาตให้ทราฟฟิคในคลาส i ใช้ได้ หาได้จากการเลือกค่าที่ต่ำที่สุดระหว่างอัตราการส่งข้อมูลสูงสุดที่คลาสนั้นได้รับ กับ อัตราการส่งข้อมูลขั้นต่ำที่คลาสนั้นได้รับรวมกับแบนด์วิดท์ที่คลาสนั้นสามารถยืมได้จากคลาสแม่ จากสมการนี้เราจะสรุปได้ว่า อัตราการส่งข้อมูลที่เชปเปอร์อนุญาตให้ทราฟฟิคในคลาส i ใช้ได้อยู่ในช่วงระหว่างอัตราการส่งข้อมูลขั้นต่ำที่คลาสนั้นได้รับบวกกับแบนด์วิดท์ที่คลาสนั้นสามารถยืมได้จากคลาสแม่ (ในกรณี ทราฟฟิคที่วิ่งเข้าสู่ระบบจริงมากกว่า อัตราการส่งข้อมูลขั้นต่ำ ซึ่งจะเกิดการยืมแบนด์วิดท์เกิดขึ้น) หรือ อัตราการส่งข้อมูลสูงสุดที่คลาสนั้นได้รับ (ในกรณีที่ อัตราการส่งข้อมูลขั้นต่ำบวกกับแบนด์วิดท์ที่คลาสนั้นสามารถยืมได้จากคลาสแม่ เกิน อัตราการส่งข้อมูลสูงสุดที่คลาสนั้นได้รับ)

$$R_i'' = \min\left(R_i, R_i'\right) \quad (2.14)$$

เมื่อ $i = \{B, C, D\}$

จากสมการที่ 2.14 เราสามารถคำนวณหาอัตราการส่งข้อมูลที่ทราฟฟิคคลาสนั้นส่งออกจริง จากการเลือกค่าที่ต่ำที่สุดระหว่างอัตราการส่งข้อมูลที่เชปเปอร์อนุญาตให้ทราฟฟิคในคลาส i ใช้ได้ กับ ทราฟฟิคที่วิ่งเข้าสู่ระบบจริง ซึ่งหมายถึงเราสามารถรู้ได้ว่า ณ ขณะนั้นคลาสนั้นมีอัตราการส่งออกของข้อมูลเท่าไร

2.3.6 หลักการควอนตัม และ อาร์ทูคิว (Quantum and r2q)

แนวคิดพื้นฐานของการทำงาน ควอนตัม คือ ในความเป็นจริงเมื่อหลายคลาสต้องการที่จะยืม แบนด์วิดท์ จำเป็นต้องใช้ตัวเลขไบต์ค่าหนึ่งในการขอใช้ แบนด์วิดท์ ก่อนที่จะได้รับ แบนด์วิดท์จากคลาสแม่ โดยตัวเลขดังกล่าวเรียกว่า **"quantum"** โดยอาจจะสังเกตได้ว่าหลายคลาสจะได้รับจำนวน แบนด์วิดท์ ที่ยืมมาจากคลาสแม่ในอัตราส่วนเดียวกันกับ ควอนตัม ซึ่งมีความสำคัญมากที่ควรรู้ คือในการทำงาน ค่า ควอนตัม ควรที่จะมีค่าน้อยที่สุดเท่าที่จะทำได้ แต่ควรที่มีขนาดใหญ่กว่าค่า MTU เพื่อสามารถส่งแพ็กเกตได้มากที่สุดในหนึ่งครั้ง แต่ควรจะมีค่าน้อยกว่า 60000 เพื่อป้องกันปัญหา starvation และ ความหน่วงที่นานเกินไป

การคำนวณค่า ควอนตัม ในแต่ละคลาส หาได้จากการนำ Assured rate ของแต่ละคลาสหารด้วยค่าพารามิเตอร์ **"r2q"** ตามสูตร

Quantum = Assured rate (in byte) / r2q (2.15)

แต่ถึงอย่างไรความเร็วในการส่งข้อมูลเมื่อคลาสนั้นได้รับอนุญาติให้ทำการส่งก็ยังคงขึ้นอยู่กับค่าพารามิเตอร์ rate,ceil,burst,cburst ดังนั้นถึงแม้ว่าในคลาสนั้นๆ จะมีค่า ควอนตัม ขนาดใหญ่ก็ตามจะไม่มีผลให้เกิดการส่งข้อมูลแบบ burst เพราะไม่ได้รับอนุญาตจากพารามิเตอร์ burst/cburst

2.3.7 ตัวอย่างการคำนวณหาแบนด์วิดท์ที่สามารถยืมได้ในแต่ละคลาสที่ต้องการยืม

สมมติให้คลาสแม่หรือคลาส A มี แบนด์วิดท์ 30 และมีคลาสลูกอยู่ 3 คลาส คือ C_1,C2,C3 โดยที่ทั้งสามคลาสมี Assured rate เท่ากันคือ 10 และค่า ควอนตัม ของ C2 เท่ากับ 1, C3 เท่ากับ 2 และ C1 เท่ากับ 4 ถ้าในการทำงาน C2, C3 ทราฟฟิคมากกว่า AR ของตน ขณะที่ทราฟฟิคคลาส C1 เท่ากับ 4 ทำให้ แบนด์วิดท์รวมเหลือเท่ากับ 6 (จาก 30-10-10-4 = 6) ซึ่งแบนด์วิดท์ เหลือจะถูกนำไปแบ่งสัดส่วนด้วยค่า ควอนตัม ของ C2 และ C3 ตามลำดับ ซึ่ง ค่า ควอนตัม ของ C2 เท่ากับ 1 และ ควอนตัม ของ C3 เท่ากับ 2 ดังนั้น แบนด์วิดท์ ที่ C2 และ C3 ยืมหาได้จาก

$$B_{C2} = \frac{Q_{C2}(R - R'_{C1} - R'_{C2} - R'_{C3})}{Q_{C2} + Q_{C3}} \quad \text{และ} \quad B_{C3} = \frac{Q_{C3}(R - R'_{C1} - R'_{C2} - R'_{C3})}{Q_{C2} + Q_{C3}}$$

$$= \frac{1(30-10-10-4)}{(1+2)} \qquad = \frac{2(30-10-10-4)}{(1+2)}$$

$$= 2 \qquad = 4$$

จากการคำนวณตามทฤษฎีข้างต้นผลลัพธ์ที่ได้ คือ $B_{C2} = 2$, $B_{C3} = 4$ ดังนั้น class C2 จึงมี แบนด์วิดท์ รวมทั้งหมดที่สามารถใช้ได้ เท่ากับ 12 class C3 จึงมี แบนด์วิดท์ รวมทั้งหมดที่สามารถใช้ได้ เท่ากับ 14

ตารางที่ 2.1 แสดงสรุปผลลัพธ์

Class	Assured rate	Traffic	Quantum	Bandwidth
C1	10	4	1*	4
C2	10	>10	1	10+2=12
C3	10	>10	2	10+4=14

***หมายเหตุ** ค่า ควอนตัม Class C1 จะไม่ถูกนำมาคิดในการคิดค่าน้ำหนักแบนด์วิดท์ที่เหลือ

2.3.8 การจัดการทราฟฟิคเข้า และออกจากระบบของกลไก HTB Scheduler

เพื่อให้เข้าใจง่ายสำหรับการยืมแบนด์วิดท์โดยข้อกำหนดคิวชนิด HTB ได้ใช้ตัวอย่างในการอธิบาย

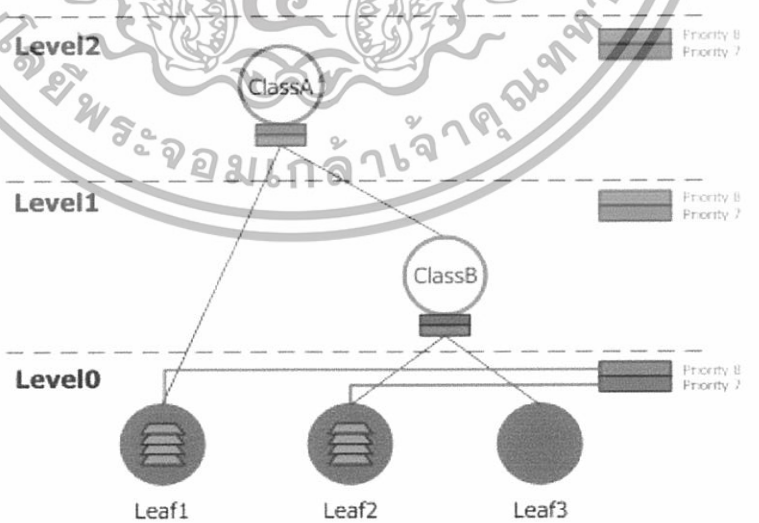
2.3.8.1 ตัวอย่างการยืม-คืนแบนด์วิดท์โดยใช้ข้อกำหนดคิวชนิด HTB

ตารางที่ 2.2 แสดงค่า Assured rate และ Ceil rate ได้จากการคำนวณในหัวข้อ 2.3.4

กำหนดให้ Leaves class ซึ่งประกอบด้วยคลาส Leaf1, Leaf2 และ Leaf3 มีอัตราส่วนในการส่งข้อมูลเท่ากับ 1:1:1 และ Bandwidth ที่ได้รับเท่ากับ 3 Mbps

Class	Assured rate	Ceil rate
A	2 Mbps	3 Mbps
B	1 Mbps	2 Mbps
Leaf1	512 Kbps	1 Mbps
Leaf2	512 Kbps	1 Mbps
Leaf3	512 Kbps	1 Mbps

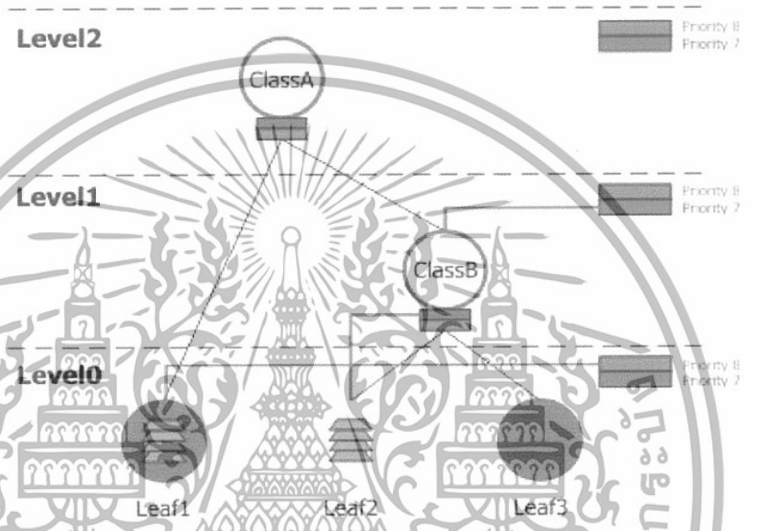
2.3.8.1.1 สถานการณ์แรก เมื่อ คลาส Leaf2 มี ระดับความสำคัญสูงกว่า Leaf1 และทุกคลาสมีสถานะ green



รูปที่ 2.7 แสดงสถานการณ์แรก

คลาส Leaf1 เส้นสีฟ้าใน ระดับชั้น= 0 ด้วย ระดับความสำคัญ = 8 และ คลาส Leaf2 เส้นสีแดงใน ระดับชั้น= 0 ด้วย ระดับความสำคัญ = 7ในสถานการณ์แรกนี้ทั้ง Leaf1 และ Leaf2 มีสถานะ green และ Leaf2 มี ระดับความสำคัญ สูงกว่า Leaf1 ดังนั้น เมื่อมีทราฟฟิคผ่านคลาส Leaf1 และ Leaf2 พร้อมกัน ทำให้ทราฟฟิคผ่านออกจากคลาส Leaf2 ได้รับบริการก่อน Leaf1

2.3.8.1.2 <u>สถานการณ์ที่สอง</u> เมื่อคลาส Leaf2 มี ระดับความสำคัญ สูงกว่า Leaf1 และ คลาส Leaf2 มีสถานะ yellow ส่วน คลาส Leaf1 มีสถานะ green

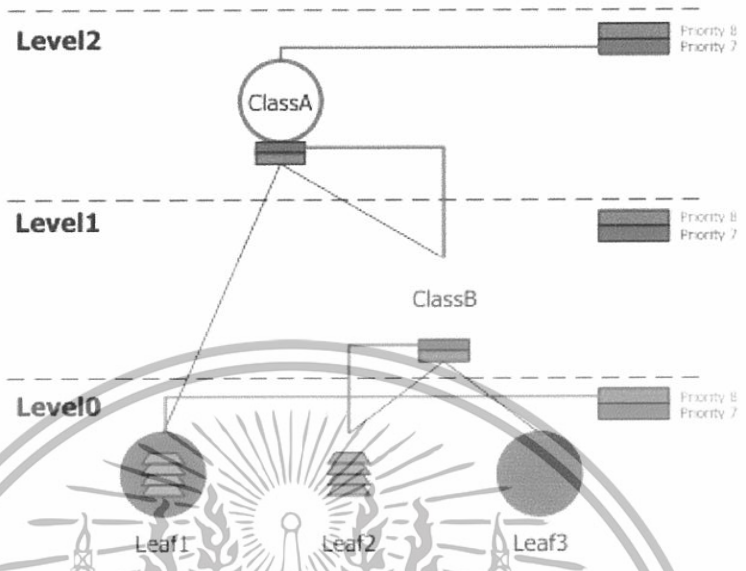


รูปที่ 2.8 แสดงสถานการณ์ที่สอง

คลาส Leaf2 มีการอัตราการส่งข้อมูลมากกว่า 512kbps (ซึ่งมากกว่าอัตราการส่งข้อมูลขั้นต่ำที่กำหนดไว้) ด้วยเหตุนี้ Leaf2 จึงมีสถานะ Yellow และมีการยืมแบนด์วิดท์จากคลาส B ซึ่งจากเดิม Leaf2 จะเป็นเส้นสีแดงในระดับชั้น = 0 แต่เนื่องจากมีการยืมจากคลาส B ดังนั้น ทำให้ขณะนี้ Leaf2 ถือได้ว่าอยู่ในระดับชั้น 1(level 1) และมีระดับความสำคัญ =7 ส่วน Leaf1 ยังคงมีสถานะ green เนื่องจากมีอัตราการส่งข้อมูลไม่เกินขั้นต่ำที่กำหนดไว้ (1Mbps)

ซึ่งในสถานการณ์สองนี้ ถึงแม้ว่า Leaf1 มี ระดับความสำคัญ = 8 ซึ่งต่ำกว่า Leaf2 มี ระดับความสำคัญ = 7 แต่เป็นเพราะ Leaf2 มีการยืมแบนด์วิดท์จากคลาสแม่ในที่นี้คือ คลาส B ซึ่งอยู่บนระดับชั้น = 1 ดังนั้น ระดับชั้นของ Leaf2 จึงถือได้ว่าเท่ากับ 1 ซึ่งเป็นผลทำให้ Leaf1 มีลำดับความสำคัญสูงกว่า Leaf2 ดังนั้น เมื่อมีทราฟฟิคผ่านคลาส Leaf1 และ Leaf2 พร้อมกัน ทำให้ทราฟฟิคผ่านออกจากคลาส Leaf1 ได้รับบริการก่อน Leaf2 (เนื่องจาก HTB จะถือว่าระดับชั้นมีลำดับความสำคัญมากกว่าระดับความสำคัญ)

2.3.8.1.3 สถานการณ์ที่สาม เมื่อคลาส Leaf2 มี ระดับความสำคัญ สูงกว่า Leaf1 และ คลาส Leaf2 มีสถานะ yellow ส่วน คลาส Leaf1 มีสถานะ red แต่ คลาส A มีสถานะ green

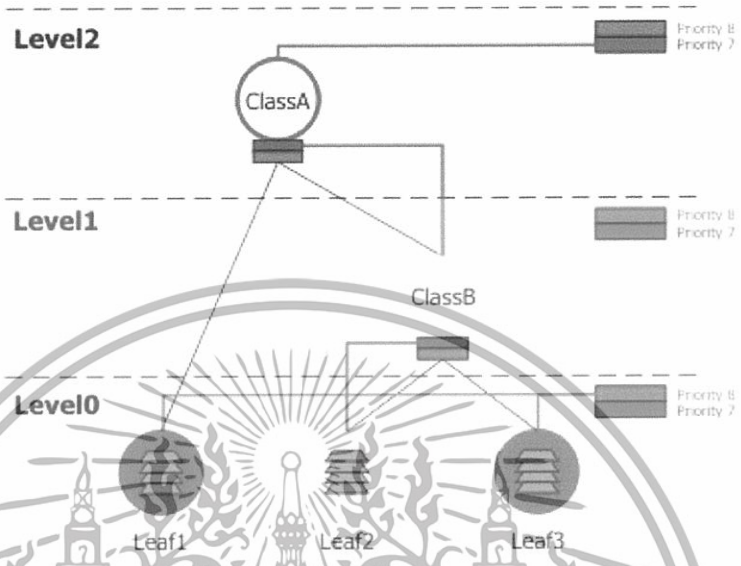


รูปที่ 2.9 แสดงสถานการณ์ที่สาม

สถานะของ Leaf1 เป็น red เนื่องจากมีอัตราการส่งข้อมูลที่สูงกว่าค่า Ceil rate และในขณะเดียวกัน Leaf2 มีอัตราการส่งข้อมูลที่มากกว่า 512 Kbps จึงมีการยืมจากคลาสแม่หรือคลาส B ซึ่งคลาส B มีแบนด์วิดท์ให้ยืม ไม่เพียงพอ ดังนั้น คลาส B จึงทำการยืมจากคลาสแม่ของคลาส B ให้ ในที่นี้คือ คลาส A ดังนั้นคลาส B จึงเปลี่ยนสถานะจาก green เป็น Yellow

ซึ่งในสถานการณ์สามนี้ ด้วยคลาส Leaf1 ไม่สามารถยืมแบนด์วิดท์จากคลาสแม่ได้อีก ดังนั้น Leaf1 จึงมีระดับความสำคัญ เท่ากับ 8 ซึ่งมากกว่า Leaf2 มี ระดับความสำคัญ = 7 ดังนั้น Leaf2 ควรมีลำดับความสำคัญมากกว่า Leaf1 แต่เป็นเพราะ Leaf2 มีการยืมแบนด์วิดท์จากคลาสแม่ ในที่นี้คือ คลาส B และคลาส B มีการยืมแบนด์วิดท์จากคลาสแม่ในที่นี้คือ คลาส A ซึ่งอยู่บนระดับชั้น = 2 ดังนั้น ระดับชั้นของ Leaf2 จึงถือได้ว่าเท่ากับ 2 ส่วนระดับชั้นของ Leaf1 จึงถือได้ว่าเท่ากับ 0 ซึ่งเป็นผลทำให้ Leaf1 มีลำดับความสำคัญสูงกว่า Leaf2 ดังนั้น เมื่อมีทราฟฟิคผ่านคลาส Leaf1 และ Leaf2 พร้อมกัน ทำให้ทราฟฟิคผ่านออกจากคลาส Leaf1 ได้รับบริการก่อน Leaf2

2.3.8.1.4 <u>สถานการณ์ที่สี่</u> เมื่อคลาส Leaf2 มี ระดับความสำคัญ สูงกว่า Leaf 1 และ คลาส Leaf3 มีสถานะ green คลาส Leaf2 มีสถานะ yellow ส่วน คลาส Leaf1 มีสถานะ red แต่ คลาส A มีสถานะ red

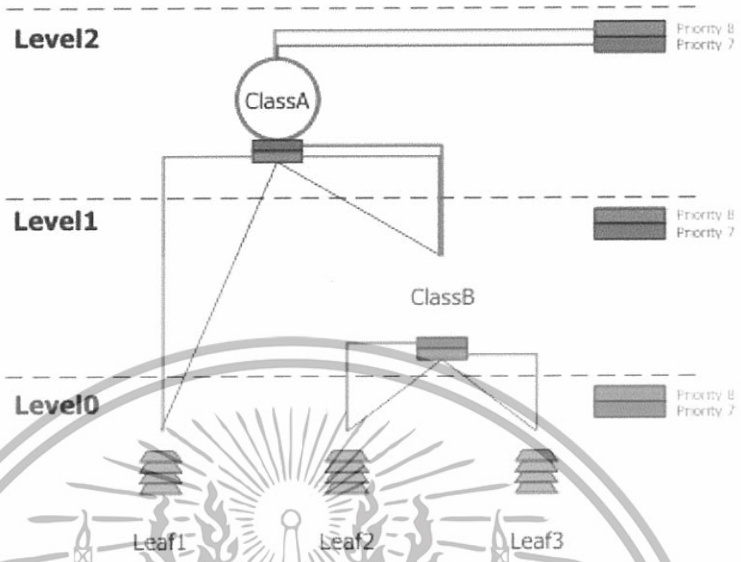


รูปที่ 2.10 แสดงสถานการณ์ที่สี่

Leaf2 มีสถานะ Yellow และต้องการยืมจากคลาส B และ คลาส B ยืมจากคลาส A (เช่นเดียวกับ สถานการณ์สาม) แต่ในขณะนี้สถานะของคลาส A เป็น red ซึ่งหมายถึง คลาส A มีอัตราการส่งข้อมูลสูงกว่าค่า Ceil rate (ในที่นี้คือ 3Mbps)

ซึ่งในสถานการณ์สี่นี้ ด้วยคลาส Leaf1 ไม่สามารถยืมแบนด์วิดท์จากคลาสแม่ได้อีก ดังนั้น Leaf1 จึงมีระดับความสำคัญ เท่ากับ 8 ซึ่งมากกว่า Leaf2 มี ระดับความสำคัญ = 7 ดังนั้น Leaf2 ควรมีลำดับความสำคัญมากกว่า Leaf1 แต่เป็นเพราะ Leaf2 มีการยืมแบนด์วิดท์จากคลาสแม่ในที่นี้คือ คลาส B และคลาส B ต้องการยืมแบนด์วิดท์จากคลาสแม่ในที่นี้คือ คลาส A แต่ไม่สามารถยืมได้ เพราะ คลาส A มีสถานะ red ซึ่งทำให้อยู่บนระดับชั้น = 1 ดังนั้น ระดับชั้นของ Leaf2 จึงถือได้ว่าเท่ากับ 1 ส่วนระดับชั้นของ Leaf1 จึงถือได้ว่าเท่ากับ 0 ซึ่งเป็นผลทำให้ Leaf1 มีลำดับความสำคัญสูงกว่า Leaf2 ดังนั้น เมื่อมีทราฟฟิคผ่านคลาส Leaf1 และ Leaf2 พร้อมกัน ทำให้ทราฟฟิคผ่านออกจากคลาส Leaf1 ได้รับบริการก่อน Leaf2

2.3.8.1.5 <u>สถานการณ์ที่ห้า</u> เมื่อคลาส Leaf2 มี ระดับความสำคัญ สูงกว่า Leaf 1 และ Leaf3 และ คลาส Leaf1 Leaf2 และ Leaf3 มีสถานะเป็น yellow แต่ คลาส A มีสถานะ green



รูปที่ 2.11 แสดงสถานการณ์ที่ห้า

Leaf1 Leaf2 Leaf3 และคลาส B มีสถานะเป็น Yellow ส่วนคลาส A มีสถานะ green

ในสถานการณ์ห้านี้คลาส Leaf1 ยืมจากคลาส A ส่วน Leaf2 และ Leaf3 ยืมจากคลาส B และคลาส B ยืมจากคลาส A ตามลำดับ ซึ่ง ณ ตอนนี้ Leaf1 Leaf2 Leaf3 อยู่ในระดับชั้น 2 เหมือนกันดังนั้น Leaf ใดที่มีระดับความสำคัญสูงสุดจะได้รับการบริการก่อน ซึ่งในที่นี้ Leaf 2 มีระดับความสำคัญที่สูงสุดเนื่องจาก ระดับความสำคัญ = 7 ส่วน Leaf 2 และ Leaf3 มี ระดับความสำคัญ = 8 เมื่อมีทราฟฟิคผ่านคลาส Leaf1 ,Leaf2 และ Leaf3 พร้อมกัน ทำให้ทราฟฟิคผ่านออกจากคลาส Leaf2 ได้รับบริการก่อน Leaf1 และ Leaf3 ส่วน Leaf1 และ Leaf3 ได้รับบริการลำดับถัดมาด้วยวิธีวนรอบ(Round robin)

บทที่ 3
การเพิ่มโมดูลกลไก HTB ให้กับโปรแกรม NS-2

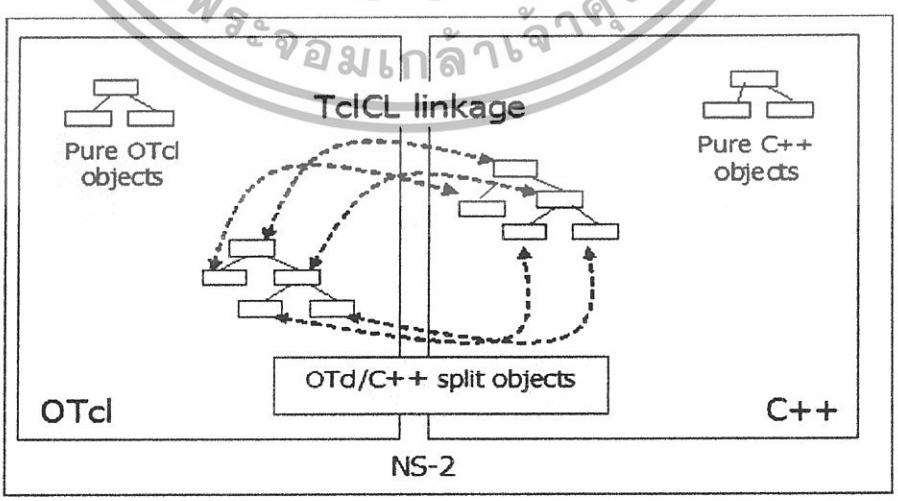
ในหัวข้อนี้จะกล่าวถึงวิธีการและแนวทางในการเพิ่มโมดูลเพื่อเติมความสามารถให้กับโปรแกรม NS-2 พร้อมทั้งแสดงการแก้ไขไฟล์สำหรับการเพิ่มโมดูลกลไก HTB ให้กับโปรแกรม NS-2

3.1 หลักการเพิ่มโมดูลในโปรแกรม NS-2 เบื้องต้น (Extending NS)

การเพิ่มเติมความสามารถให้กับโปรแกรม NS-2 โดยการเพิ่มเน็ตเวิร์คออบเจ็คใหม่ให้กับโปรแกรม NS-2 มีความเกี่ยวข้องกับการทำงานร่วมกันระหว่าง OTCL และ C++ เนื่องจากโครงสร้างของโปรแกรม NS-2 ถูกเขียนขึ้นด้วยภาษา C++ โดย OTCL ทำหน้าที่เป็นฟรอนต์เอนด์ (Front end) ซึ่งการเชื่อมต่อระหว่าง 2 ภาษานี้ถูกจัดการโดย TCLCL (TCL with Classes)

3.1.1 หลักการทำงานเบื้องต้นของ TCLCL

โปรแกรม NS-2 สนับสนุนโครงสร้างแบบลำดับชั้นบน C++ ทั้งโครงสร้างลำดับชั้นแบบคอมไพล์ (Hierarchy Compile) และ โครงสร้างลำดับชั้นแบบอินเตอร์พลิต(Hierarchy interpreted) โดยมีความสัมพันธ์ระหว่างออบเจ็คแบบ 1 ต่อ 1 ซึ่งออบเจ็คที่สร้างขึ้นจะอยู่บนพื้นฐานโครงสร้างลำดับชั้นแบบคอมไพล์ โดยเรียกใช้งานผ่านพื้นฐานโครงสร้างลำดับชั้นแบบอินเตอร์พริต



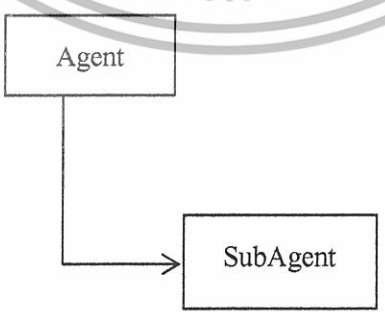
รูปที่ 3.1 แสดงการเชื่อมต่อระหว่าง OTcl และ C++ โดยมี TCLCL เป็นตัวเชื่อม

องค์ประกอบหรือคลาสที่มีความเกี่ยวข้องกับการเชื่อมต่อระหว่าง OTCL กับ C++ ประกอบด้วย 6 คลาสหลักที่เกี่ยวข้อง

ตารางที่ 3.1 แสดงองค์ประกอบที่เกี่ยวข้องกับการเชื่อมต่อระหว่าง OTCL และ C++

TclObject	เป็นรากของโครงสร้างลำดับชั้นของNs-2 bind(): เชื่อมระหว่างตัวแปร OTCL กับตัวแปร C++ command(): เชื่อมระหว่างฟังชั่นของ OTcl กับเมทอด C++
TclClass	ประกาศและสร้าง TclObject
Tcl	ประกอบด้วย ฟังชั่นของ C ++ สำหรับเชื่อมต่อกับ OTcl interpreter เช่น Tcl.eval(char* s)
TclCommand	ประกอบด้วยฟังชั่น command ของ C ++ สำหรับเชื่อมระหว่างฟังชั่นของ OTcl กับเมธอด C++
EmbeddedTcl	ประกอบด้วยฟังชั่นที่สามารถเรียกคำสั่งที่ถูกฝังมาตั้งแต่ต้นได้เพื่อสร้างการจำลองให้เป็นไปได้ง่าย
InstVar	ประกอบด้วยฟังชั่นที่จัดการการเข้าถึง ตัวแปร C++ จาก ตัวแปร OTCL

3.1.2 ขั้นตอนการเชื่อมต่อ OTCL และ C++



รูปที่ 3.2 แสดงภาพโมดูลใหม่ที่ทำการเพิ่มเข้าไปภายใต้โมดูลเดิม

จากรูปที่ 3.3 แสดงให้เห็นถึงโมดูลใหม่ในรูปคือ SubAgent ที่ต้องการเพิ่มเข้าไปใหม่ภายใต้โมดูลที่มีอยู่เดิมคือ Agent โดยมีขั้นตอนดังต่อไปนี้

3.1.2.1 Export C++ class to OTCL

ทำการสร้างการเชื่อมต่อระหว่าง C++ class กับ OTCL กล่าวคือออบเจ็คที่สร้างขึ้นมาจะถูกเรียกใช้จาก C++ class ผ่าน OTCL เช่น set subagent [new Agent/SubAgentOtcl]

โดยสามารถแบ่งการทำงานออกเป็น 2 ส่วนได้ดังนี้

ตัวอย่างที่ 1

```
static class SubAgentClass : public TclClass {
public:
        SubAgentClass() : TclClass("Agent/SubAgentOtcl") {}
        TclObject* create(int, const char*const*) {
                return(new SubAgent());
        }
} class_sub_agent;
```

จากโค้ดตัวอย่างแบ่งการสร้างออกเป็น 2 ส่วนคือ

1) การสร้างชื่อ "Agent/SubAgentOtcl" เพื่อให้ OTCL ใช้อ้างอิงในการสร้างออบเจ็ค

2) การสร้างลิงค์ระหว่าง OTCL object และ C++

สองส่วนนี้จะถูกสร้างเมื่อมีการรันโปรแกรม NS-2 จากไฟล์ .tcl

ตัวอย่างที่ 2

```
class SubAgent : public Agent {
public:
        SubAgent();
protected:
        int command(int argc, const char*const* argv);
private:
        int   var1;
        double var2;
        void   SubPrivFunc(void);
};
```

ตัวอย่างที่ 2 เป็นส่วนการทำงานของคลาส SubAgent โดยมีการเรียกใช้จากคลาส SubAgentClass โดยไฟล์ตัวอย่างเป็นไฟล์เฮดเดอร์ (.h) ที่ประกาศการทำงานของ SubAgent Class ซึ่งการเข้าถึงฟังก์ชันหรือตัวแปรของ C++ object ไม่สามารถเรียกผ่าน C++ object จากการสร้าง Instance จาก OTCL

3.1.2.2 Export C++ class variables to OTCL

ทำการสร้างความสัมพันธ์ของตัวแปรภายใน C++ class เพื่อให้ OTCL สามารถเรียกใช้ตัว

แปรได้ โดยการ export ตัวแปรที่ต้องการเรียกจาก C++ class ด้วยการใช้ bind function ของโปรแกรม NS-2

ตัวอย่างที่ 3

ส่วนการเรียกใช้จาก OTCL

```
$subagent set var1_otcl 2
$subagent set var2_otcl 3.14
```

ตัวอย่างที่ 3 แสดงการใช้ bind function และตัวแปรที่ต้องการ bind ซึ่งเรียกใช้จาก OTCL โดยมีการกำหนดค่าตัวแปร var1_otcl และ var2_otcl ผ่านทาง OTCL ด้วย

ตัวอย่างที่ 4

ส่วนการทำงาน C++ object

```
SubAgent::SubAgent() : Agent(PT_UDP) {
      bind("var1_otcl", &var1);
      bind("var2_otcl", &var2);
}
```

ตัวอย่างที่ 4 การทำ bind function ฝั่งการทำงาน C++ object ประกอบด้วยตัวแปร 2 ส่วน คือ ชื่อตัวแปรที่เรียกใช้ใน OTCL และ ชื่อตัวแปรที่ทำงานภายใน C++ object

"var1_otcl" -> ชื่อตัวแปรที่เรียกใช้ใน OTCL

var1 -> ชื่อตัวแปรที่ทำงานภายใน C++ object

ในการทำ Binding โปรแกรม NS-2 สนับสนุน binding function ของตัวแปร 4 ชนิด

1. bind():ชนิดตัวแปรแบบจำนวนจริง
2. bind_bool():ชนิดตัวแปรแบบ Boolean
3. bind_time():ชนิดตัวแปรเวลา
4. bind_bw():ชนิดตัวแปรbandwidth

โดยเราสามารถกำหนดค่า default ให้กับตัวแปรที่ถูก bind ได้ใน "ns-2/tcl/lib/ns-lib.tcl" มิฉะนั้นจะมีการสร้างข้อความเตือนเมื่อมีการสร้าง instance จาก object ที่สร้างขึ้นใหม่

3.1.2.3 Export C++ object command to OTCL

สร้างการควบคุมจาก OTCL เพื่อเรียกใช้ฟังก์ชันการทำงานภายใน C++ object โดยใช้ฟังก์ชันควบคุม (Command function) ซึ่งเป็นหนึ่งในฟังก์ชันของ C++ class ในความเป็นจริงแล้วฟังก์ชันที่เรียกใช้จาก OTCL เป็นฟังก์ชันเดียวกับฟังก์ชันที่ถูกกำหนดอยู่ภายฟังก์ชันควบคุมของ C++ class

ตัวอย่างที่ 5

```
# ส่วนที่ทำการเรียกฟังชั่น call-sub-function จาก subagent object
$subagent call-sub-function
```

ตัวอย่างที่ 5 การเรียกใช้ฟังชั่นภายใน OTCL object และ การจัดการฟังก์ชันภายใน command ฟังก์ชันซึ่งจะเห็นได้ว่ามีการเรียกใช้ฟังชั่น call-sub-function จาก subagent object

ตัวอย่างที่ 6

```
int SubAgent::command(int argc, const char*const* argv) {
    if(argc == 2) {
        if(strcmp(argv[1], "call-sub-function") == 0) {
            SubPrivFunc();
            return(TCL_OK);
        }
    }
    return(Agent::command(argc, argv));
}
```

ตัวอย่างที่ 6 อธิบาย command ฟังชั่นภายใน C++ class โดยทำการตรวจสอบชื่อฟังชั่น call-sub-function เป็นสมาชิกหนึ่งที่ถูกกำหนดในคอมมานด์ฟังชั่นหรือไม่ถ้าตรวจสอบแล้วพบว่ามีฟังชั่นนั้นอยู่จริงและทำงานตามฟังชั่นนั้นๆซึ่งในที่นี้คือ SubPrivFunc( )

3.1.2.4 Execute an OTCL command from C++

เมื่อทำการสร้าง Network object ใน C++ แล้วต้องมีการกระทำการ(execute) OTCL command จาก C++ object

ตัวอย่างที่ 7

```
void SubAgent::SubPrivFunc(void) {
    Tcl& tcl = Tcl::instance();
    tcl.eval("puts \"Message From SubPrivFunc\"");
    tcl.evalf("puts \"    var1_otcl = %d\"", var1);
    tcl.evalf("puts \"    var2_otcl = %f\"", var2);
}
```

ตัวอย่างที่ 7 แสดงให้เห็นถึงการสร้างฟังก์ชั่น SubPrivFunc( ) จากการเรียกใช้ฟังชั่น command ดังตัวอย่างที่ 6 โดยการทำงานภายในฟังชั่นทำให้ OTCL Interpreter สามารถแสดงค่าตัวแปร var_1 และ var_2 ซึ่งเป็นตัวแปรประเภท private

ในการ Execute OTCL command จาก C++ object หากต้องการส่งค่าผลลัพธ์จาก C++ object ไปยัง OTCL จำเป็นต้องอ้างอิงถึง "Tcl::instance()" เพื่อให้สามารถเชื่อมการส่งผลลัพธ์ที่ได้จากการเรียกฟังก์ชั่น SubPrivFunc( ) ไปยัง interpreter ของ OTCL ได้ สามารถทำการส่งผลลัพธ์ไปยัง interpreter ได้ 2 วิธี ดังตัวอย่างที่ 8

ตัวอย่างที่ 8

```
tcl.eval("puts \"Message From SubPrivFunc\"");
tcl.evalf("puts \"     var1_otcl = %d\"", var1);
```

3.1.2.5 Run

จากตัวอย่างที่ 1-8 เมื่อทำการสร้างไฟล์ .tcl เพื่อเรียกใช้ออบเจ็คของ SubAgent

ตัวอย่างที่ 9

```
#เรียกใช้ SubAgent oject ผ่าน OTCL
set subagent [new Agent/SubAgentOtcl]

# กำหนดค่าให้กัตัวแปร
$ subagent set var1_otcl 2
$ subagent set var2_otcl 3.14

# เรียกใช้ฟังชั่นใน SubAgent oject
$ subagent call-sub-function
```

ตัวอย่างที่ 9 เป็นการนำ SubAgent มาใช้งานจริง จากตัวอย่างข้างต้นได้ทำการกำหนดค่าตัวแปรให้กับออบเจ็ค subagent พร้อมทั้งทำการเรียกใช้ฟังก์ชันการทำงานภายใน SubAgent

ผลลัพธ์ที่ได้คือ

ตัวอย่างที่ 10

```
Message From SubPrivFunc
var1_otcl = 2
var2_otcl = 3.14000
```

ตัวอย่างที่ 10 เนื่องจากเราได้กำหนดค่าให้กับตัวแปร var1_otcl และ var 2_otcl ดังนั้นเมื่อทำการเรียกฟังชั่น call-sub-function ใน OTCL ค่าที่ส่งกลับมาจะเป็นไปตามการทำงานของฟังชั่น SubPrivFunc() ในส่วนของ C++ ซึ่งมีการคืนค่าผ่านไปยัง interpreter เพื่อแสดงผลลัพธ์

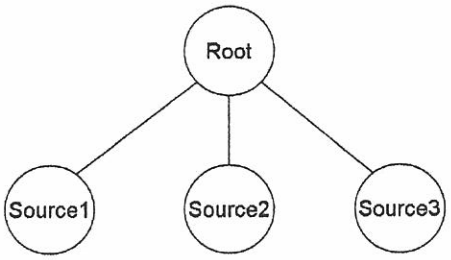
3.2 การเพิ่มกลไก HTB ในโปรแกรม NS-2

จากทฤษฎีและหลักการทำงานของ HTB ในบทที่ 2 จะนำมาสร้างเป็นโมดูลเพิ่มเติมในโปรแกรม NS-2 ได้ดังนี้

3.2.1 การแบ่งคลาสและลำดับชั้น

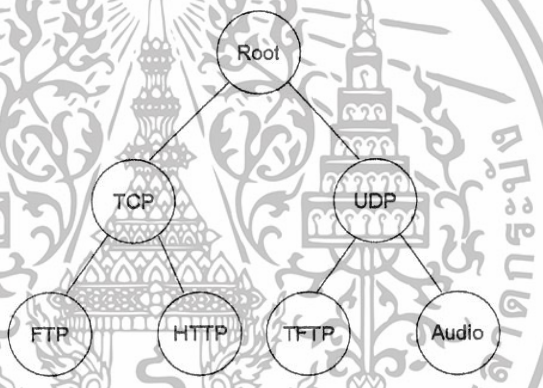
ในรายงานฉบับนี้ได้ทำการเพิ่มโมดูลของ HTB โดยได้ทำการกำหนดคลาสลงในโปรแกรมใน 2 ลักษณะดังนี้

1. แบ่งตามลักษณะของผู้ใช้



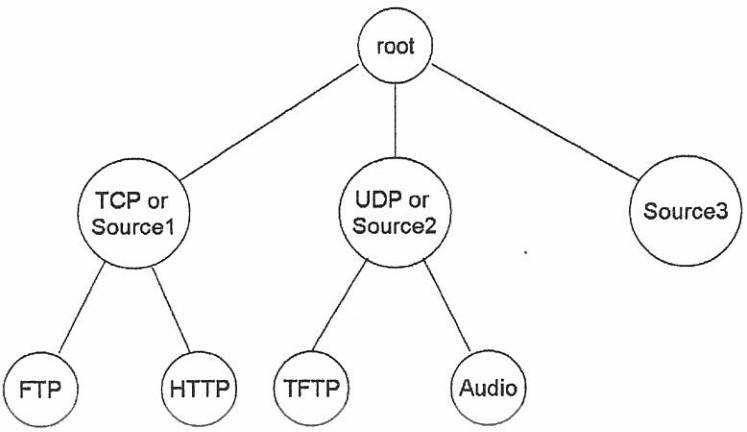
รูปที่ 3.3 แสดงการแบ่งคลาสตามลักษณะผู้ใช้

2. แบ่งตามลักษณะแอพพลิเคชั่น



รูปที่ 3.4 แสดงการแบ่งคลาสตามลักษณะแอพพลิเคชั่น

จากรูปการแบ่งคลาสทั้ง 2 ลักษณะ ได้ทำการกำหนดให้คลาสของ Source1 ใช้ร่วมกับคลาสของ TCP และคลาสของ Source2 ใช้ร่วมกับ UDP โดยขึ้นอยู่กับการเลือกใช้ว่าจะแบ่งการกำจัดทราฟฟิคตามลักษณะผู้ใช้หรือลักษณะแอพพลิเคชั่น โดยจะทำการกำหนดโครงสร้างลำดับชั้นและการแบ่งคลาสลงไปในโมดูลของ HTB ในโปรแกรม NS-2 ดังรูป



รูปที่ 3.5 แสดงการแบ่งคลาสที่กำหนดเข้าไปในโปรแกรม NS-2

3.2.1.1 ข้อจำกัดในการเพิ่มคลาสทรี

ในโครงงานนี้มีข้อจำกัดเกี่ยวกับการกำหนดคลาสทรีในโมดูล HTB ที่สร้างเพิ่มเข้าไปในโปรแกรม NS-2 ให้เป็นไปดังรูปที่ 3.6 เท่านั้น ดังนั้นหากมีความต้องการเพิ่มคลาสทรีรูปแบบอื่นจำเป็นที่จะต้องแก้ไขโค้ดของโมดูล HTB ในส่วนการสร้างคลาสทรีใหม่

3.2.2 ผังงานการทำงาน

การทำงานของโมดูล HTB ที่ได้เพิ่มเข้าไปในโปรแกรม NS-2 มีการแบ่งขั้นตอนการทำงานออกเป็น 5 ส่วนดังต่อไปนี้

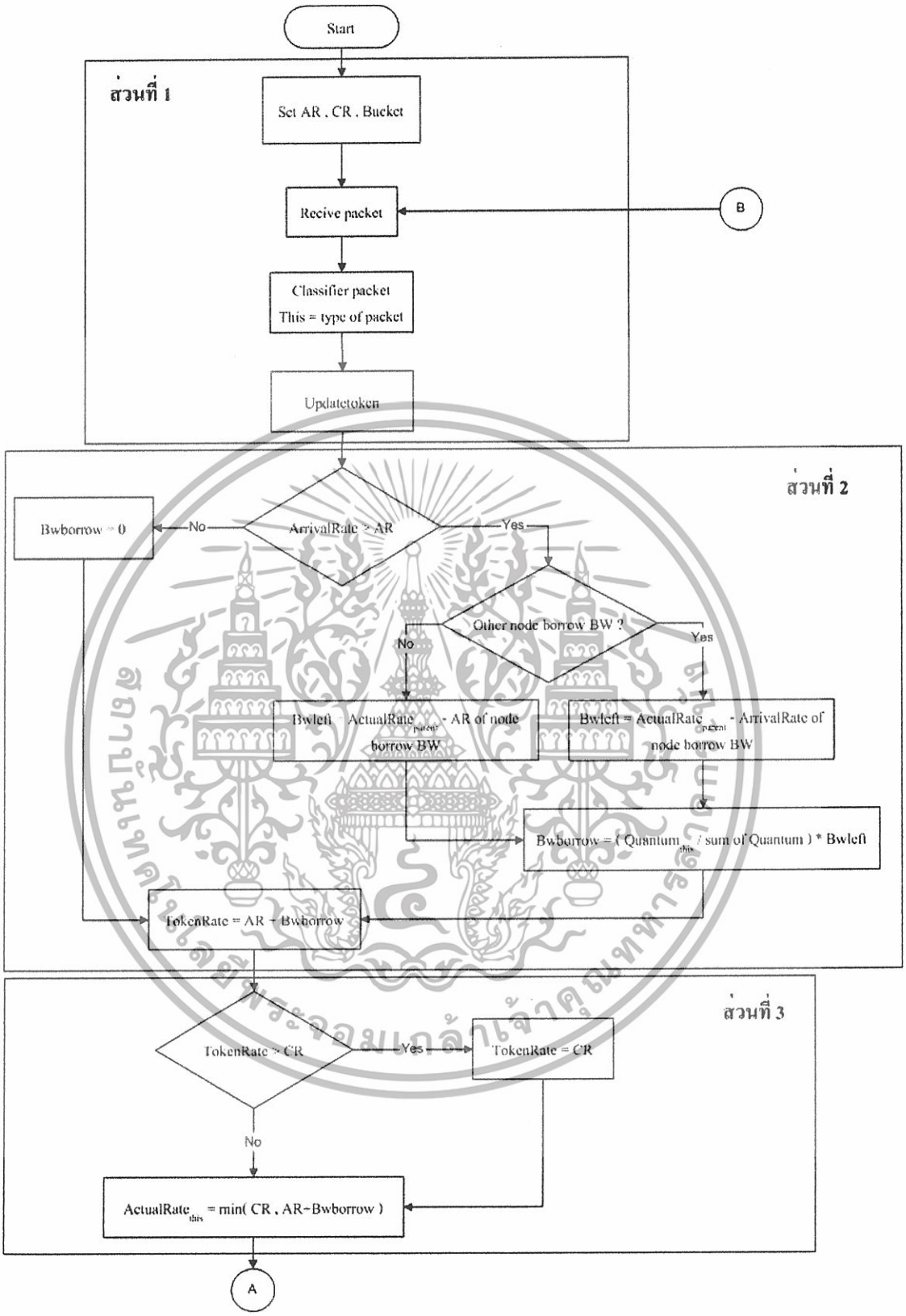
ส่วนที่ 1 ทำการรับแพคเก็ตเข้าสู่การทำงานของกลไก HTB และแบ่งชนิดแพคเก็ตตามคลาส

ส่วนที่ 2 ทำการตรวจสอบอัตราการเข้ามาของแพคเก็ตชนิดนั้นและเช็คกับค่า AR ของคลาสว่าถ้าอัตราการเข้ามาของแพคเก็ตมีค่ามากกว่า AR จะมีการคำนวณค่าการยืมแบนด์วิทด์

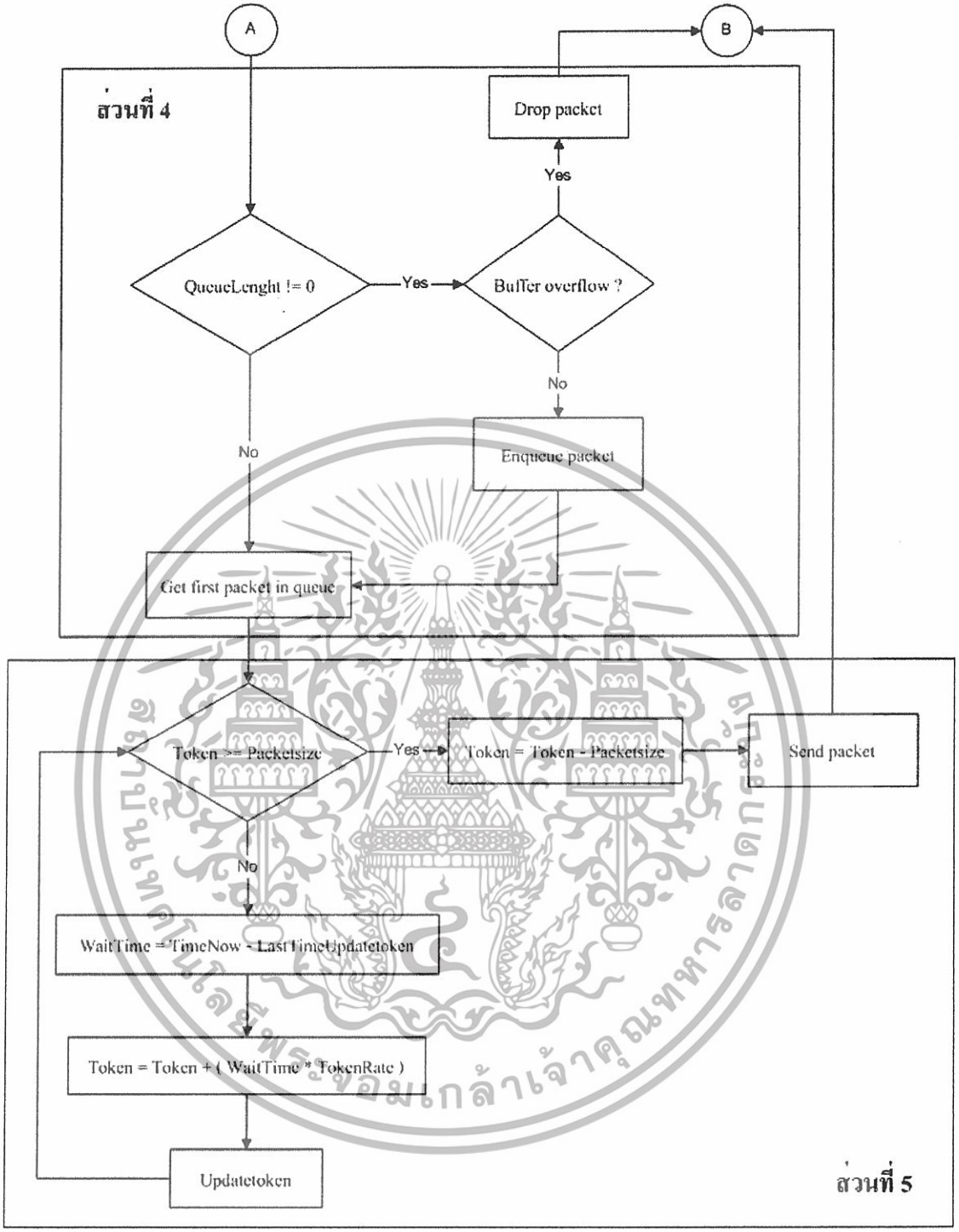
ส่วนที่ 3 ทำการตรวจสอบไม่ให้ค่า TokenRate มีค่าเกินค่า CR ของคลาส

ส่วนที่ 4 ทำการตรวจสอบบัฟเฟอร์ของคิว ถ้ามีแพคเก็ตอื่นอยู่ในคิวแล้วจะทำการนำแพคเก็ตที่เข้ามาใหม่เข้าสู่คิว และถ้าบัฟเฟอร์ของคิวเต็มแล้วจะเกิดการละทิ้งแพคเก็ต

ส่วนที่ 5 ทำการตรวจสอบจำนวนโทเคนที่มีอยู่ ถ้ามีจำนวนน้อยกว่าขนาดแพคเก็ตจะให้แพคเก็ตนั้นรออยู่ในบัฟเฟอร์ของคิวจนถึงเวลาที่โทเคนเพิ่มจำนวนเท่ากับขนาดแพคเก็ตแล้วจึงทำการส่งแพคเก็ตนั้นออกไปพร้อมกับลดจำนวนโทเคนลงเท่ากับขนาดแพคเก็ต



รูปที่ 3.6 แสดงผังการทำงานของกลไก HTB



รูปที่ 3.6 แสดงผังการทำงานของกลไก HTB (ต่อ)

3.2.3 การสร้างไฟล์การทำงาน

ทำการสร้างไฟล์ที่เป็นส่วนหัวโดยให้ชื่อว่า htb.h ดังนี้

```
#include "connector.h"
#include "timer-handler.h"
#include "queue.h"
#include "address.h"
#include <string>

class HTB;
class HTB_Node;

class HTB_Timer : public TimerHandler {
public:
        HTB_Timer(HTB *t) : TimerHandler() { htb_ = t;}
        HTB_Timer();
        string pktName;
protected:
        virtual void expire(Event *e);
        HTB *htb_;
};


class HTB : public Connector {
public:
        HTB();
        ~HTB();
        void timeout(string pktName);
protected:
        void recv(Packet *p, Handler *);
        HTB_Node *root , *tcp , *udp , *ftp , *http , *tftp , *audio , *exnode;
        HTB_Timer tcp_timer , udp_timer , ftp_timer , http_timer , tftp_timer , audio_timer , exnode_timer;
        int init_ , shape_by_source;

};

class HTB_Node {
public:
        HTB_Node();
        void getupdatedtokens();
        void getarrival(Packet*);
        void borrowbw();
        double min(double,double);
        double r , ar , cr , tokens , rate , quantum , bw , arrival , lastupdatetime;
        int bucket , qlen , sumByte , borrowActive;
        PacketQueue *q;
        HTB_Node *parent , *nfriend , *nfriend2;
};
```

ภายในไฟล์ htb.h จะมีการประกาศคลาสที่ใช้สำหรับการทำงานของ HTB ที่สำคัญ 3 คลาสคือ

1. HTB_Timer ทำหน้าที่จัดการตรวจสอบเวลาที่จะทำการเพิ่มโทเคน
2. HTB ทำหน้าที่จัดการกลไกการทำงานทั้งหมดรวมทั้งสร้างโครงสร้างลำดับชั้นของแต่ละคลาสของHTB
3. HTB_Node ทำหน้าที่เป็นคลาสของ HTB

จากนั้นทำการสร้างไฟล์ที่เป็นส่วนการทำงานโดยใช้ชื่อว่า htb.cc ซึ่งมีฟังก์ชั่นการทำงานส่วนต่างๆที่สำคัญดังนี้

ส่วนที่1

```
static class HTBClass : public TclClass {
public:
        HTBClass() : TclClass ("HTB") {}
        TclObject* create(int,const char*const*) {
          return (new HTB());
        }
}class_htb;
```

ส่วนที่ 1 สำหรับทำการสร้างชื่อ "HTB" เพื่อให้ OTCL ใช้อ้างอิงในการสร้างออบเจ็คในโปรแกรม NS-2

ส่วนที่2

```
HTB::HTB() :tcp_timer(this),udp_timer(this),ftp_timer(this),http_timer(this),tftp_timer(this),audio_timer(this),
exnode_timer(this),init_(1),shape_by_source(0)
{
        //สร้าง node สำหรับ HTB
        root = new HTB_Node();
        tcp = new HTB_Node();
        udp = new HTB_Node();
        ftp = new HTB_Node();
        http = new HTB_Node();
        tftp = new HTB_Node();
        audio = new HTB_Node();
        exnode = new HTB_Node();

        //กำหนด node แม่สำหรับแต่ละ node
        root->parent = root;
        tcp->parent = root;
        udp->parent = root;
        exnode->parent = root;
        ftp->parent = tcp;
        http->parent = tcp;
        tftp->parent = udp;
        audio->parent = udp;

        //กำหนด node ข้างเคียงสำหรับแต่ละ node
        root->nfriend = root;
        tcp->nfriend = udp;
        udp->nfriend = tcp;
        exnode->nfriend = tcp;
        ftp->nfriend = http;
        http->nfriend = ftp;
        tftp->nfriend = audio;
        audio->nfriend = tftp;

        //กำหนด node ข้างเคียงสำหรับแต่ละ node
        root->nfriend2 = root;
        tcp->nfriend2 = exnode;
        udp->nfriend2 = exnode;
        exnode->nfriend2 = udp;
        ftp->nfriend2 = exnode;
        http->nfriend2 = exnode;
        tftp->nfriend2 = exnode;
        audio->nfriend2 = exnode;
```

```
        //กำหนดตัวแปรสำหรับเลือกปรับให้ทำงานจาก node จาก tcl
        bind("shape_by_source",&shape_by_source);

        //กำหนดตัวแปรสำหรับ root
        bind_bw("root_CR",&root->cr);
        bind_bw("root_AR",&root->ar);
        bind("root_bucket",&root->bucket);
        bind("root_qlen",&root->qlen);

        //กำหนดตัวแปรสำหรับ tcp
        bind_bw("tcp_CR",&tcp->cr);
        bind_bw("tcp_AR",&tcp->ar);
        bind("tcp_bucket",&tcp->bucket);
        bind("tcp_qlen",&tcp->qlen);

        //กำหนดตัวแปรสำหรับ udp
        bind_bw("udp_CR",&udp->cr);
        bind_bw("udp_AR",&udp->ar);
        bind("udp_bucket",&udp->bucket);
        bind("udp_qlen",&udp->qlen);

        //กำหนดตัวแปรสำหรับ node1 ใช้ร่วมกับ tcp
        bind_bw("exnode1_CR",&tcp->cr);
        bind_bw("exnode1_AR",&tcp->ar);
        bind("exnode1_bucket",&tcp->bucket);
        bind("exnode1_qlen",&tcp->qlen);

        //กำหนดตัวแปรสำหรับ node2 ใช้ร่วมกับ udp
        bind_bw("exnode2_CR",&udp->cr);
        bind_bw("exnode2_AR",&udp->ar);
        bind("exnode2_bucket",&udp->bucket);
        bind("exnode2_qlen",&udp->qlen);

        //กำหนดตัวแปรสำหรับ node3
        bind_bw("exnode3_CR",&exnode->cr);
        bind_bw("exnode3_AR",&exnode->ar);
        bind("exnode3_bucket",&exnode->bucket);
        bind("exnode3_qlen",&exnode->qlen);

        //กำหนดตัวแปรสำหรับ ftp
        bind_bw("ftp_CR",&ftp->cr);
        bind_bw("ftp_AR",&ftp->ar);
        bind("ftp_bucket",&ftp->bucket);
        bind("ftp_qlen",&ftp->qlen);

        //กำหนดตัวแปรสำหรับ http
        bind_bw("http_CR",&http->cr);
        bind_bw("http_AR",&http->ar);
        bind("http_bucket",&http->bucket);
        bind("http_qlen",&http->qlen);

        //กำหนดตัวแปรสำหรับ tftp
        bind_bw("tftp_CR",&tftp->cr);
        bind_bw("tftp_AR",&tftp->ar);
        bind("tftp_bucket",&tftp->bucket);
        bind("tftp_qlen",&tftp->qlen);

        //กำหนดตัวแปรสำหรับ audio
        bind_bw("audio_CR",&audio->cr);
        bind_bw("audio_AR",&audio->ar);
        bind("audio_bucket",&audio->bucket);
        bind("audio_qlen",&audio->qlen);
}
```

ส่วนที่ 2 จะมีการทำงานเมื่อมีการสร้างออปเจ็คของ HTB ขึ้นมาใช้งานด้วยคำสั่ง new โดยจะทำการสร้างออปเจ็คของ HTB_Node ต่างๆขึ้นมาตามโครงสร้างลำดับชั้นของการแบ่งคลาสของ HTB ที่ได้กำหนดเอาไว้ในหัวข้อที่ 3.2.1 ด้วยการกำหนด parent และ nfriend จากนั้นมีการ

กำหนดค่าตัวแปรในภาษา C++ ที่เชื่อมต่อกับภาษา TCL สำหรับให้ทำการกำหนดค่าเริ่มต้นของคลาสต่างๆ ใน HTB ด้วยคำสั่ง bind

ส่วนที่3

```
void HTB::recv(Packet *p, Handler *)
{
        if (init_) {

                //กำหนดค่าเริ่มต้นให้กับ node ต่างๆ
                //โดยให้จำนวน token เริ่มมาเท่ากับขนาดของ bucket
                //ให้ token rate เท่ากับ assure rate ( ar )
                //ให้ quantum เท่ากับ ar/10
                //ให้เวลาสำหรับการอัพเดท token ล่าสุดเป็นปัจจุบัน
                //ให้ ActualRate ( r ) ของ root เท่ากับ AR หรือ CR

                tcp->tokens=tcp->bucket;
                tcp->rate=tcp->ar;
                tcp->quantum=tcp->ar/10.0;
                 tcp->borrowActive=0;
                tcp->lastupdatetime = Scheduler::instance().clock();
                udp->tokens=udp->bucket;
                udp->rate=udp->ar;
                udp->quantum=udp->ar/10.0;
                 udp->borrowActive=0;
                udp->lastupdatetime = Scheduler::instance().clock();
                exnode->tokens=exnode->bucket;
                exnode->rate=exnode->ar;
                exnode->quantum=exnode->ar/10.0;
                 exnode->borrowActive=0;
                exnode->lastupdatetime = Scheduler::instance().clock();
                ftp->tokens=ftp->bucket;
                ftp->rate=ftp->ar;
                ftp->quantum=ftp->ar/10.0;
                 ftp->borrowActive=0;
                ftp->lastupdatetime = Scheduler::instance().clock();
                http->tokens=http->bucket;
                http->rate=http->ar;
                http->quantum=http->ar/10.0;
                 http->borrowActive=0;
                http->lastupdatetime = Scheduler::instance().clock();
                tftp->tokens=tftp->bucket;
                tftp->rate=tftp->ar;
                tftp->quantum=tftp->ar/10.0;
                 tftp->borrowActive=0;
                tftp->lastupdatetime = Scheduler::instance().clock();
                audio->tokens=audio->bucket;
                audio->rate=audio->ar;
                audio->quantum=audio->ar/10.0;
                 audio->borrowActive=0;
                audio->lastupdatetime = Scheduler::instance().clock();

                root->r = min(root->cr,root->ar);

                init_=0;
        }

        //ทำการตรวจสอบชนิดของ packet ที่รับเข้ามา
        hdr_cmn* ch=hdr_cmn::access(p);
        packet_t pt = ch->ptype();
        p_info* info;
        const char* pkt = info->name(pt);
        string pktName = pkt;

        //ถ้าเป็น packet ชนิด tcp หรือ http จะให้ tcp ที่เป็น node แม่มีการทำงานก่อน
        if((pktName == "ftp")||(pktName == "http"))
        {
                //ทำการตรวจสอบว่ามีการยืม BW หรือไม่ และสามารถยืมได้ในปริมาณเท่าใด
```

```
                //จากนั้นทำการกำหนดค่า r สำหรับให้ node ลูกใช้ต่อไป
                tcp->getarrival(p);
                if(tcp->arrival <= tcp->ar) {
                        tcp->borrowActive = 0;
                }
                if(tcp->arrival > tcp->ar) {
                        tcp->borrowActive = 1;
                }
                tcp->borrowbw();
                tcp->r = tcp->min(tcp->cr,tcp->ar+tcp->bw);
        }


        //ถ้าเป็น packet ชนิด tftp หรือ audio จะให้ udp ที่เป็น node แม่มีการทำงานก่อน
        if((pktName == "tftp")||(pktName == "audio"))
        {
                //ทำการตรวจสอบว่ามีการยืม BW หรือไม่ และสามารถยืมได้ในปริมาณเท่าใด
                //จากนั้นทำการกำหนดค่า r สำหรับให้ node ลูกใช้ต่อไป
                udp->getarrival(p);
                if(udp->arrival <= udp->ar) {
                        udp->borrowActive = 0;
                }
                if(udp->arrival > udp->ar) {
                        udp->borrowActive = 1;
                }
                udp->borrowbw();
                udp->r = udp->min(udp->cr,udp->ar+udp->bw);
        }

        HTB_Node *node;

        //ทำการตรวจสอบชนิดของ packet และให้ class HTB_Node ทำงานที่ node ของ packet นั้น
        if((pktName == "tcp")||(pktName == "ack"))
                node = tcp;
        if((pktName == "udp")||(pktName == "cbr"))
                node = udp;
        if(pktName == "ftp")
                node = ftp;
        if(pktName == "http")
                node = http;
        if(pktName == "tftp")
                node = tftp;
        if(pktName == "audio")
                node = audio;

        //ถ้ากำหนดให้มีการ shape จาก source node
        //ให้ tcp ทำงานเป็น source 0 และ udp ทำงานเป็น source 1
        hdr_ip* ip = hdr_ip::access(p);
        if(shape_by_source) {
                if(ip->src_.addr_ == 0)
                        node = tcp;
                if(ip->src_.addr_ == 1)
                        node = udp;
                if(ip->src_.addr_ == 2)
                        node = exnode;
        }

        //ทำการตรวจสอบปริมาณข้อมูลที่เข้ามาเป็น byte/sec
        //ซึ่งถ้าปริมาณข้อมูลที่เข้ามาน้อยกว่าค่า ar จะไม่มีการยืม BW
        //แต่ถ้าปริมาณข้อมูลที่เข้ามามากกว่าค่า ar จะเริ่มมีการยืม BW
        node->getarrival(p);
        if(node->arrival <= node->ar) {
                node->borrowActive = 0;
        }
        if(node->arrival > node->ar) {
                node->borrowActive = 1;
        }

        //ทำการเพิ่มจำนวน token
        node->getupdatedtokens();
        //ทำการยืม BW
        node->borrowbw();
```

```
//กำหนดค่า r สำหรับใช้ใน node ลูกต่อไป
node->r = node->min(node->cr,node->ar+node->bw);

//ถ้าไม่มีการยืม BW ค่า token rate จะมีค่าเท่ากับ ar
//ถ้ามีการยืม BW ค่า token rate จะมีค่าเท่ากับ ar รวมกับค่า bw ที่ยืมมา
if(node->borrowActive == 0) {
        node->rate = node->ar;
}
if(node->borrowActive == 1) {
        node->rate = node->ar + node->bw;
        if(node->rate > node->cr)
                node->rate = node->cr;
}
//ตรวจสอบว่ามี packet อื่นอยู่ในคิวหรือไม่
//ถ้ามี packet อื่นอยู่ในคิวและคิวยังไม่เต็มให้นำ packet เข้าคิว
//แต่ถ้าคิวเต็มแล้วให้ทำการทิ้ง packet นั้น
if (node->q->length() !=0) {
        if (node->q->length() < node->qlen) {
                node->q->enque(p);
                return;
        }
        drop(p);
        return;
}
int pktsize = ch->size();

//ถ้าจำนวน token มากกว่าขนาด packet ให้ทำการส่ง packet นั้นออกไปและลดจำนวน token ลงเท่ากับขนาด packet นั้น
if (node->tokens >=pktsize) {
        target_->recv(p);
        node->tokens-=pktsize;
}
else {
        //ถ้าจำนวน token มีน้อยกว่าขนาด packet ให้มีการกำหนดเวลาครั้งต่อไปที่จะส่ง packet ออกไปได้
        //โดยใช้การตรวจสอบเวลาครั้งต่อไปที่จะมี token พอให้ส่ง packet นั้น
        //คำนวนเวลาจาก จำนวน token ที่ขาดอยู่ คูณกับ token rate
        node->q->enque(p);
        if(shape_by_source) {
                hdr_ip* ip = hdr_ip::access(p);
                if(ip->src_.addr_ == 0) {
                        tcp_timer.resched(abs((pktsize-node->tokens)/node->rate));
                        tcp_timer.pktName = "n1";
                }
                if(ip->src_.addr_ == 1) {
                        udp_timer.resched(abs((pktsize-node->tokens)/node->rate));
                        udp_timer.pktName = "n2";
                }
                if(ip->src_.addr_ == 2) {
                        exnode_timer.resched(abs((pktsize-node->tokens)/node->rate));
                        exnode_timer.pktName = "n3";
                }
        }
        else {
                if((pktName == "tcp")||(pktName == "ack"))
                {
                        tcp_timer.resched(abs((pktsize-node->tokens)/node->rate));
                        tcp_timer.pktName = "tcp";
                }
                if((pktName == "udp")||(pktName == "cbr"))
                {
                        udp_timer.resched(abs((pktsize-node->tokens)/node->rate));
                        udp_timer.pktName = "udp";
                }
                if(pktName == "ftp")
                {
                        ftp_timer.resched(abs((pktsize-node->tokens)/node->rate));
                        ftp_timer.pktName = "ftp";
                }
                if(pktName == "http")
                {
                        http_timer.resched(abs((pktsize-node->tokens)/node->rate));
                        http_timer.pktName = "http";
                }
```

```
                        if(pktName == "tftp")
                        {
                                tftp_timer.resched(abs((pktsize-node->tokens)/node->rate));
                                tftp_timer.pktName = "tftp";
                        }
                        if(pktName == "audio")
                        {
                                audio_timer.resched(abs((pktsize-node->tokens)/node->rate));
                                audio_timer.pktName = "audio";
                        }
                }
        }
}
```

ส่วนที่ 3 จะมีการทำงานเมื่อ HTB มีการรับแพคเก็ตเข้ามาโดยหลังจากที่กำหนดค่าเริ่มต้นสำหรับการทำงานแล้วจะมีการตรวจสอบชนิดของแพคเก็ตที่เข้ามาเพื่อให้คลาสที่กำหนดไว้ตรงกับชนิดแพคเก็ตนั้นทำการทำงานซึ่งถ้ามีการกำหนดให้มีการจำกัดทราฟฟิคแบบแบ่งตามลักษณะผู้ใช้จะให้คลาส TCP และ UDP ทำงานเป็น Source1 และ Source2 ตามรูปที่ 3.4 จากนั้นจะเป็นการทำงานตามหลักการของกลไก HTB โดยการตรวจสอบอัตราการเข้ามาของแพคเก็ตและคำนวณค่าแบนด์วิดท์ที่ยืมและตรวจสอบบัฟเฟอร์ของคิวและสุดท้ายจะมีการคำนวณเวลาที่จะเกิดเหตุการณ์ครั้งต่อไปสำหรับการเพิ่มจำนวนโทเคน

ส่วนที่4

```
void HTB_Timer::expire(Event* /*e*/)
{
        //ให้ HTB มีการทำงานเมื่อถึงเวลากำหนดส่ง packet ครั้งต่อไปที่ตรวจสอบเอาไว้แล้ว
        htb_->timeout(pktName);
}

void HTB::timeout(string pktName)
{
        HTB_Node *node;

        //ทำการตรวจสอบชนิดของ packet และให้ class HTB_Node ทำงานที่ node ของ packet นั้น
        if((pktName == "tcp") || (pktName == "n1"))
                node = tcp;
        if((pktName == "udp") || (pktName == "n2"))
                node = udp;
        if(pktName == "n3")
                node = exnode;
        if(pktName == "ftp")
                node = ftp;
        if(pktName == "http")
                node = http;
        if(pktName == "tftp")
                node = tftp;
        if(pktName == "audio")
                node = audio;

        Packet *p = node->q->head();

        //ทำการเพิ่มจำนวน token
        node->getupdatedtokens();

        hdr_cmn *ch=hdr_cmn::access(p);
        int pktsize = ch->size();

        //ส่ง packet ที่อยู่ลำดับแรกของคิวออกไปและลดจำนวน token ลงเท่ากับขนาด packet นั้น
        p = node->q->deque();
```

```
        target_->recv(p);
        node->tokens-=pktsize;

        //ถ้ายังมี packet อื่นอยู่ในคิวอีกให้นำออกมาตรวจสอบ
        if (node->q->length() !=0 ) {
                p=node->q->head();
                hdr_cmn *ch=hdr_cmn::access(p);
                pktsize = ch->size();

                //กำหนดเวลาครั้งต่อไปที่จะมีการส่ง packet ออกไปได้
                //โดยใช้การตรวจสอบเวลาครั้งต่อไปที่จะมี token พอให้ส่ง packet นั้น
                //คำนวนเวลาจาก จำนวน token ที่ขาดอยู่ คูณกับ token rate
                if(shape_by_source) {
                        hdr_ip* ip = hdr_ip::access(p);
                        if(ip->src_.addr_ == 0) {
                                tcp_timer.resched(abs((pktsize-node->tokens)/node->rate));
                                tcp_timer.pktName = "n1";
                        }
                        if(ip->src_.addr_ == 1) {
                                udp_timer.resched(abs((pktsize-node->tokens)/node->rate));
                        udp_timer.pktName = "n2";
                        }
                        if(ip->src_.addr_ == 2) {
                                exnode_timer.resched(abs((pktsize-node->tokens)/node->rate));
                                exnode_timer.pktName = "n3";
                        }
                }
                else {
                        if((pktName == "tcp")||(pktName == "ack"))
                        {
                                tcp_timer.resched(abs((pktsize-node->tokens)/node->rate));
                                tcp_timer.pktName = "tcp";
                        }
                        if((pktName == "udp")||(pktName == "cbr"))
                        {
                                udp_timer.resched(abs((pktsize-node->tokens)/node->rate));
                                udp_timer.pktName = "udp";
                        }
                        if(pktName == "ftp")
                        {
                                ftp_timer.resched(abs((pktsize-node->tokens)/node->rate));
                                ftp_timer.pktName = "ftp";
                        }
                        if(pktName == "http")
                        {
                                http_timer.resched(abs((pktsize-node->tokens)/node->rate));
                                http_timer.pktName = "http";
                        }
                        if(pktName == "tftp")
                        {
                                tftp_timer.resched(abs((pktsize-node->tokens)/node->rate));
                                tftp_timer.pktName = "tftp";
                        }
                        if(pktName == "audio")
                        {
                                audio_timer.resched(abs((pktsize-node->tokens)/node->rate));
                                audio_timer.pktName = "audio";
                        }
                }
        }
}
```

ส่วนที่ 4 จะมีการทำงานเมื่อถึงเวลาสำหรับเหตุการณ์เพิ่มโทเคนที่โปรแกรมได้มีการตรวจสอบเอาไว้ด้วย htb_timer โดยจะทำการเพิ่มจำนวนโทเคนตามช่วงเวลาที่ผ่านมาและทำการตรวจสอบแพคเก็ตที่อยู่ในคิวเพื่อทำการส่งออกไปแล้วคำนวณเวลาที่จะเกิดเหตุการณ์ครั้งต่อไป

ส่วนที่5

```
HTB_Node::HTB_Node()
{
        q = new PacketQueue();
}

void HTB_Node::getarrival(Packet *p)
{
        //คำนวนปริมาณข้อมูลที่เข้ามาโดยนับขนาด packet ทั้งหมดหารด้วยเวลาทั้งหมด
        double now=Scheduler::instance().clock();
        hdr_cmn *ch=hdr_cmn::access(p);
        int pktsize = ch->size();
        sumByte += pktsize;
        arrival = sumByte/now;
}

void HTB_Node::borrowbw()
{
        double r1;
        double r2;

        //ตรวจสอบว่า node ข้างเคียงมีการยืม BW หรือไม่
        if(nfriend->borrowActive == 1)
        {
                r1 = nfriend->ar;
        } else {
                r1 = nfriend->arrival;
        }
        if(nfriend2->borrowActive == 1)
        {
                r2 = nfriend2->ar;
        } else {
                r2 = nfriend2->arrival;
        }
        //คำนวน BW ที่เหลือให้ยืมได้
        double bwleft = parent->r - ar - r1 - r2;

        //ถ้ามีการยืม BW จะมีการกำหนดค่า BW ที่สามารถยืมได้จากค่า Quantum และค่า BW ที่เหลือให้ยืมได้
        if(borrowActive == 1)
        {
                bw = (quantum/(quantum+(nfriend->quantum*nfriend->borrowActive)+
                (nfriend2->quantum*nfriend2->borrowActive)))*bwleft;
        }
        else
        {
                bw = 0;
        }
}

void HTB_Node::getupdatedtokens()
{
        //ทำการเพิ่มจำนวน token ตามเวลาที่ผ่านไป
        //โดยคิดจากเวลาที่เพิ่ม token ครั้งล่าสุดจนถึงเวลาปัจจุบันคูณกับ token rate
        double now=Scheduler::instance().clock();
        tokens += (now-lastupdatetime)*rate;
        if (tokens > bucket)
                tokens=bucket;
        lastupdatetime = Scheduler::instance().clock();
}

double HTB_Node::min(double cr,double ar)
{
        if(cr <= ar)
                return cr;
        else
                return ar;
}
```

ส่วนที่ 5 เป็นฟังก์ชั่นการทำงานส่วนย่อยของการทำงานหลักของ HTB ซึ่งจะมีการคำนวณปริมาณข้อมูลที่เข้ามาและปริมาณแบนด์วิทด์ที่สามารถยืมได้รวมถึงมีการคำนวณจำนวนโทเคนที่เพิ่มขึ้นเมื่อเวลาผ่านไป

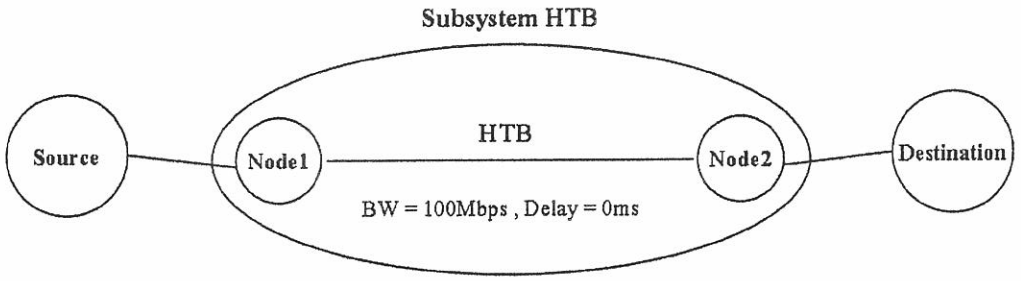
ต่อไปทำการเพิ่มคำสั่ง attach-shape เพื่อให้สามารถประยุกต์ใช้งาน HTB เข้ากับโปรแกรม NS-2 ผ่านทางภาษา TCL ได้โดยปรับปรุงไฟล์ lib.tcl ที่อยู่ใน ns-allinone-2.29\ns-2.29\tcl\lib ดังนี้

```
Simulator instproc attach-shape {shaper n1 n2} {
        $self instvar link_
        set sid [$n1 id]
        set did [$n2 id]
        set templink $link_($sid:$did)
        set linktarget [$templink get-target]
        $templink set-target $shaper
        $shaper target $linktarget
}

SimpleLink instproc set-target {tg} {
        $self instvar link_
        $link_ target $tg
}

SimpleLink instproc get-target {} {
        $self instvar link_
        set tg [$link_ target]
        return $tg
}
```

3.2.4 การนำไปใช้งาน

นำไฟล์ htb.h และ htb.cc ไปไว้ที่ ns-allinone-2.29\ns-2.29\queue และทำการเพิ่มเติมในไฟล์ makefile.in ในช่วง OBJ_CC = โดยทำการเพิ่ม queue/htb.o เข้าไปเพื่อเป็นการสั่งให้สร้างไฟล์ htb.o เพื่อใช้ในการทำงาน หลังจากนั้นทำการคอมไพล์ไฟล์ด้วยคำสั่ง make depend ที่ ns-allinone-2.29\ns-2.29 และตามด้วยคำสั่ง make

ในการนำกลไก HTB ไปใช้งานในโปรแกรม NS-2 นั้นจะต้องกำหนดให้ HTB ทำงานอยู่บนลิ้งก์ระหว่างสองโหนด โดยควรจะกำหนดให้แบนด์วิทด์บนลิ้งก์นั้นมีค่ามากๆเพื่อที่จะไม่ให้ค่าแบนด์วิทด์บนลิ้งจำกัดการผ่านของแพคเก็ตที่เข้าสู่การทำงานของ HTB เนื่องจากโมดูล HTB ที่สร้างขึ้นนี้จะมีการเริ่มทำงานหลังจากที่แพคเก็ตได้ผ่านเข้าลิ้งก์ระหว่างโหนดมาแล้ว และควรจะกำหนดค่าความหน่วงของลิ้งให้เป็นศูนย์เพื่อไม่ให้เกิดการหน่วงเวลาในขณะแพคเก็ตผ่านลิ้งก์นี้ โดยทำจะสามารถสร้างเป็นลักษณะของระบบย่อยได้ดังรูปที่ 3.7



รูปที่ 3.7 แสดงการสร้างระบบย่อย HTB ในโปรแกรม NS-2

ตัวอย่างการนำกลไก HTB ที่สร้างขึ้นไปใช้งานในภาษา TCL โดยให้มีการทำงานแบบแบ่งตามลักษณะแอพพลิเคชั่นที่เป็น TCP และ UDP ดังนี้

```
set ns [new Simulator]

set tf [open out.tr w]
$ns trace-all $tf

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]


$ns duplex-link $n0 $n2 1Mb 100ms DropTail
$ns duplex-link $n1 $n2 1Mb 100ms DropTail
$ns duplex-link $n2 $n3 100Mb 0ms DropTail
$ns duplex-link $n3 $n4 1Mb 100ms DropTail

set src [new Agent/UDP]
$ns attach-agent $n0 $src

set src2 [new Agent/TCP]
$src2 set packetSize_ 960
$ns attach-agent $n1 $src2

set ftp [new Application/FTP]
$ftp attach-agent $src2

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $src
$cbr set packetSize_ 1000
$cbr set rate_ 0.1MB
$cbr set random_ false

set null [new Agent/Null]
$ns attach-agent $n4 $null
set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink

$ns connect $src2 $sink
$ns connect $src $null

set htb [new HTB]

$htb set shape_by_source 0

$htb set root_CR 100k
```

```
$htb set root_AR 100k

$htb set tcp_CR 100k
$htb set tcp_AR 20k
$htb set tcp_bucket 100000
$htb set tcp_qlen 100

$htb set udp_CR 100k
$htb set udp_AR 80k
$htb set udp_bucket 100000
$htb set udp_qlen 100

$htb set exnode3_CR 0k
$htb set exnode3_AR 0k
$htb set exnode3_bucket 0
$htb set exnode3_qlen 0

$ns attach-shape $htb $n2 $n3

proc finish {} {
        global ns tf
        $ns flush-trace
        close $tf
        exit 0
}

$ns at 0.001 "$cbr start"
$ns at 0.001 "$ftp start"
$ns at 20 "finish"

$ns run
```

ในการใช้งานถ้าไม่มีการใช้คลาส Source3 ต้องทำการกำหนดให้ค่าเริ่มต้นของคลาส Source3 เป็นศูนย์ด้วยเพื่อไม่ให้มีผลกระทบต่อการทำงานของคลาสอื่นๆ

3.2.5 แนวทางการพัฒนาต่อในอนาคต

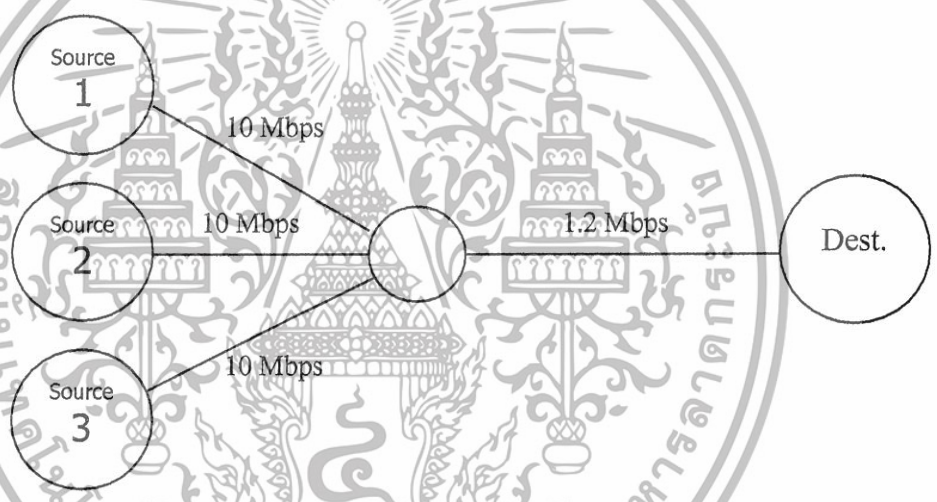
ในรายงานฉบับนี้ได้เขียนโค้ดสร้างโครงสร้างลำดับชั้นและการกำหนดคลาสของกลไก HTB ตามรูปที่ 3.6 โดยที่รองรับให้สามารถทำการปรับเปลี่ยนหรือเพิ่มเติมคลาสได้ตามความต้องการในอนาคต ซึ่งการสร้างคลาสเพิ่มเติมเข้าไปในโค้ดจะมีขั้นตอนดังต่อไปนี้

1. ประกาศคลาส HTB_Node เพิ่มเติมเป็นชื่อ newClass ทั้งใน htb.h และ htb.cc
2. ประกาศคลาส HTB_Timer เพิ่มเติมเป็นชื่อ newClassTimer ทั้งใน htb.h
3. ในฟังก์ชั่น HTB::HTB() ทำการสร้างออบเจ็คของ newClass ขึ้นมาจากนั้นทำการกำหนดคลาสที่เป็นแม่หรือคลาสข้างเคียง (โดยที่อาจสร้างตัวแปรสำหรับเป็นคลาสข้างเคียงเพิ่มขึ้นได้ในคลาส HTB_Node ในกรณีที่มีคลาสข้างเคียงมากกว่า 2)
4. ประกาศชื่อตัวแปรที่จะรับค่าจากภาษา TCL เชื่อมโยงกับตัวแปรในภาษา C++
5. ในฟังก์ชั่น HTB::recv() ประกาศค่าเริ่มต้นของ newClass และในส่วนการทำงานให้กำหนดชนิดของโปรโตคอลที่จะทำการแบ่งแยก
6. ถ้า newClass เป็นคลาสลูกจากคลาสใดๆที่ไม่ใช่คลาส Rootให้เพิ่มการทำงานของคลาสแม่ให้ทำงานก่อนเพื่อจัดการทราฟฟิคที่จะส่งต่อมายังคลาสลูก

บทที่ 4
การทดลองจำลองการควบคุมปริมาณทราฟฟิคด้วยกลไก HTB

ในหัวข้อนี้จะแสดงให้เห็นถึงกลไก HTB ที่ใช้ในการควบคุมปริมาณทราฟฟิค โดยจะมีการทดลองคือ การทดลองโมดูลของกลไก HTB ที่เพิ่มเข้าในโปรแกรม NS-2 ว่าสามารถทำงานได้ครบถ้วน และถูกต้องตามทฤษฎี

4.1 การทดลองเพื่อแสดงว่ากลไก HTB ที่เพิ่มเข้าไปในโปรแกรม NS-2 สามารถทำงานได้ถูกต้อง



รูปที่ 4.1 แสดงโทโพโลยีของการทดลอง 4.1

4.1.1 การทดลองเพื่อแสดงประสิทธิภาพของเครือข่ายเมื่อไม่มีการใช้กลไก HTB

วิธีการทดลอง

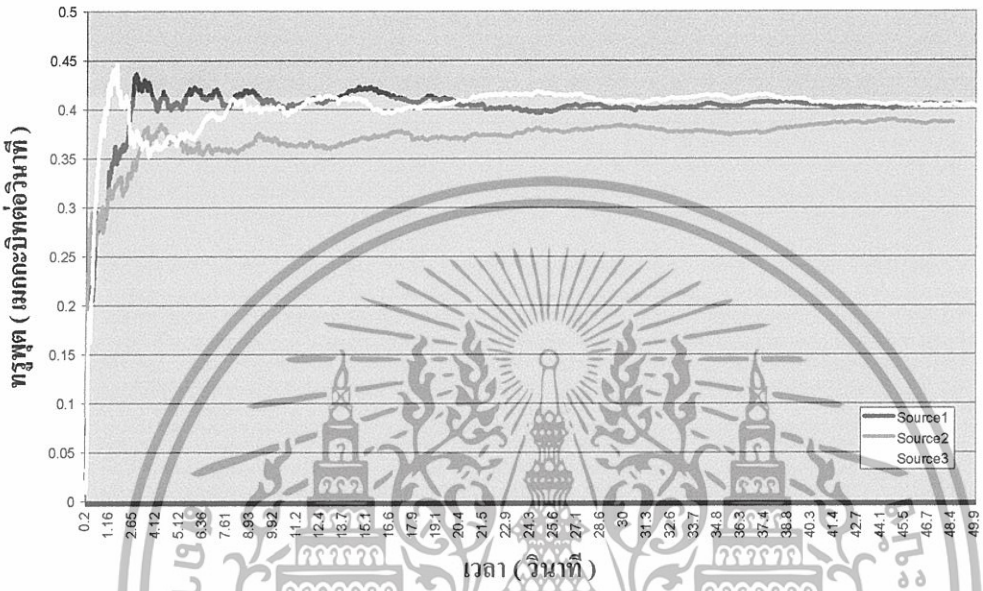
จากรูปที่ 4.1 ให้ Source1 , Source2 , Source3 ส่งแพคเก็ตชนิดยูดีพีด้วยอัตรา 2 เมกะบิตต่อวินาทีไปยังโหนด Dest. ที่เป็นปลายทางแล้วทำการวัดค่าความหน่วงเฉลี่ยของแพคเก็ตจากต้นทางไปยังปลายทาง และ วัดค่าทรูพุตเฉลี่ยของแต่ละ Source

ผลการทดลอง

ความหน่วงเฉลี่ยจาก Source1 ถึง Dest. คือ 19.96 วินาที

ความหน่วงเฉลี่ยจาก Source2 ถึง Dest. คือ 20.29 วินาที

ความหน่วงเฉลี่ยจาก Source3 ถึง Dest. คือ 19.52 วินาที

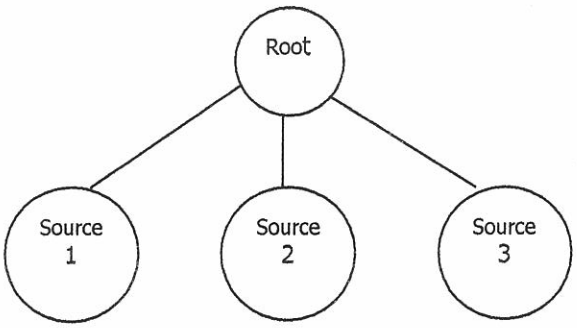


รูปที่ 4.2 กราฟแสดงค่าทรูพุตเมื่อไม่มีการใช้กลไก HTB

สรุปผลการทดลอง

เมื่อไม่มีการใช้กลไกใดๆเข้ามาจัดการควบคุมปริมาณทราฟฟิคทำให้แต่ละ Source ที่มีอัตราการส่งแพคเก็ตเท่ากันจะได้ค่าทรูพุตเฉลี่ยเท่าๆกันที่ค่าประมาณ 0.4 เมกะบิตต่อวินาที ซึ่งทำให้ค่าทรูพุตรวมเท่ากับแบนด์วิทด์ของสายคือ 1.2 เมกะบิตต่อวินาที จึงแสดงให้เห็นว่าปริมาณทราฟฟิคที่แต่ละ Source ส่งออกมาขณะนั้นจะผ่านไปถึงปลายทางได้ด้วยปริมาณที่รวมกันไม่เกินค่าแบนด์วิทด์ของสายที่รองรับได้

4.1.2 การทดลองเพื่อแสดงประสิทธิภาพของเครือข่ายเมื่อมีการใช้กลไก HTB โดยทำการจำกัดปริมาณทราฟฟิคตามลักษณะของผู้ใช้



รูปที่ 4.3 แสดงการแบ่งคลาสของกลไก HTB ตามลักษณะผู้ใช้

4.1.2.1 การทดลองเปรียบเทียบการจำกัดปริมาณทราฟฟิคด้วยกลไก HTB และไม่จำกัดทราฟฟิคด้วยกลไก HTB

4.1.2.1.1 การทดลองเมื่อมีการใช้กลไก HTB และกำหนดค่าอัตราการส่งแพคเก็ตให้มีค่าน้อยกว่าค่า AR

ตารางที่ 4.1 แสดงค่ากำหนดในการทดลองที่ 4.1.2.1.1

	Ceil Rate (Mbps)	Assure Rate (Mbps)
Root	1.2	1.2
Source 1	1.2	0.6
Source 2	1.2	0.4
Source 3	1.2	0.2

วิธีการทดลอง

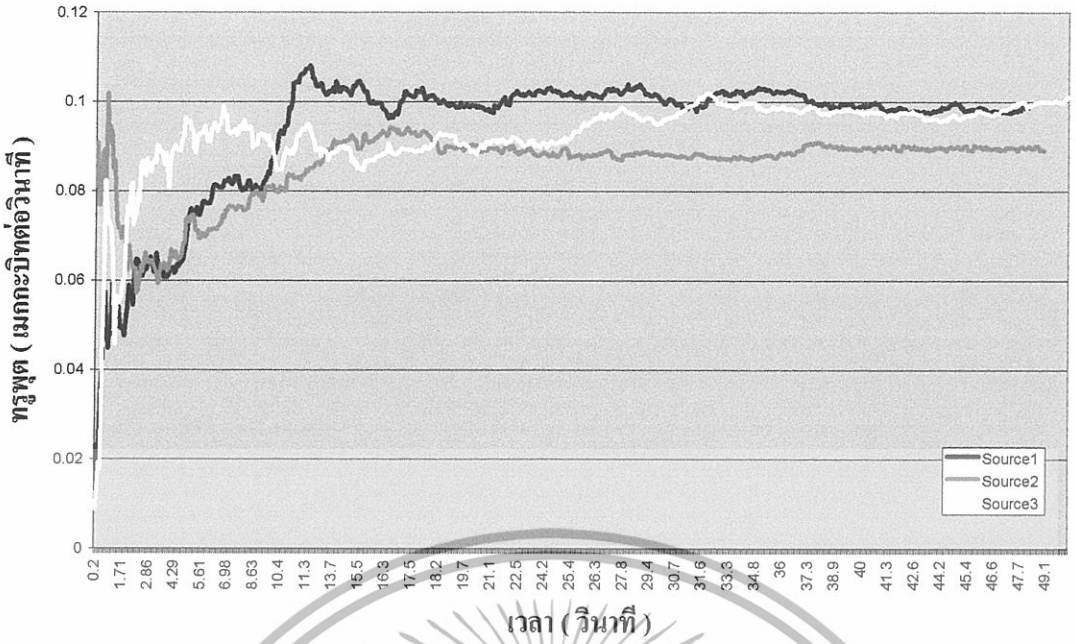
จากรูปที่ 4.1 ให้ Source ทั้งสามส่งแพคเก็ตชนิดยูดีพีด้วยอัตรา 0.1 เมกะบิตต่อวินาทีไปยังโหนด Dest. เพื่อที่จะไม่ให้เกิดการจำกัดทราฟฟิคโดยที่ใช้กลไก HTB ที่โหนดตรงกลางแล้วกำหนดค่า AR ของคลาส Source1 , Soure2 , Source3 เป็น 3:2:1 ตามลำดับจากนั้นทำการวัดค่าความหน่วงเฉลี่ยของแพคเก็ตจากต้นทางไปยังปลายทาง และ วัดค่าทรูพุตเฉลี่ยของแต่ละ Source

ผลการทดลอง

ความหน่วงเฉลี่ยจาก Source1 ถึง Dest. คือ 0.2051 วินาที

ความหน่วงเฉลี่ยจาก Source2 ถึง Dest. คือ 0.2048 วินาที

ความหน่วงเฉลี่ยจาก Source3 ถึง Dest. คือ 0.2046 วินาที



รูปที่ 4.4 กราฟแสดงค่าทรูพุตเมื่อมีการใช้กลไก HTBและมีอัตราส่งแพคเก็ต 0.1 เมกะบิตต่อวินาที

สรุปผลการทดลอง

เมื่อมีการใช้กลไก HTB เข้ามาจัดการควบคุมปริมาณทราฟฟิคแต่กำหนดให้แต่ละ Source มีอัตราการส่งแพคเก็ตที่น้อยกว่าค่า AR จะทำให้แต่ละคลาสของกลไก HTB ไม่มีการยืมแบนด์วิทด์เกิดขึ้น จึงทำให้แต่ละ Source มีค่าทรูพุตเฉลี่ยเท่าๆกันที่ค่าประมาณ 0.1 เมกะบิตต่อวินาทีซึ่งเท่ากับค่าอัตราการส่งแพคเก็ตที่กำหนด จึงแสดงให้เห็นว่าเมื่อปริมาณทราฟฟิคมีค่าน้อยกว่าค่า AR ของคลาสจะทำให้กลไก HTB ไม่มีการทำงานในการยืมแบนด์วิทด์เกิดขึ้น

4.1.2.1.2 การทดลองเมื่อมีการใช้กลไก HTB และกำหนดค่าอัตราการส่งแพคเก็ตให้มีค่ามากกว่าค่า AR

ตารางที่ 4.2 แสดงค่ากำหนดในการทดลองที่ 4.1.2.1.2

	Ceil Rate (Mbps)	Assure Rate (Mbps)
Root	**1.2**	**1.2**
Source 1	**1.2**	**0.6**
Source 2	**1.2**	**0.4**
Source 3	**1.2**	**0.2**

วิธีการทดลอง

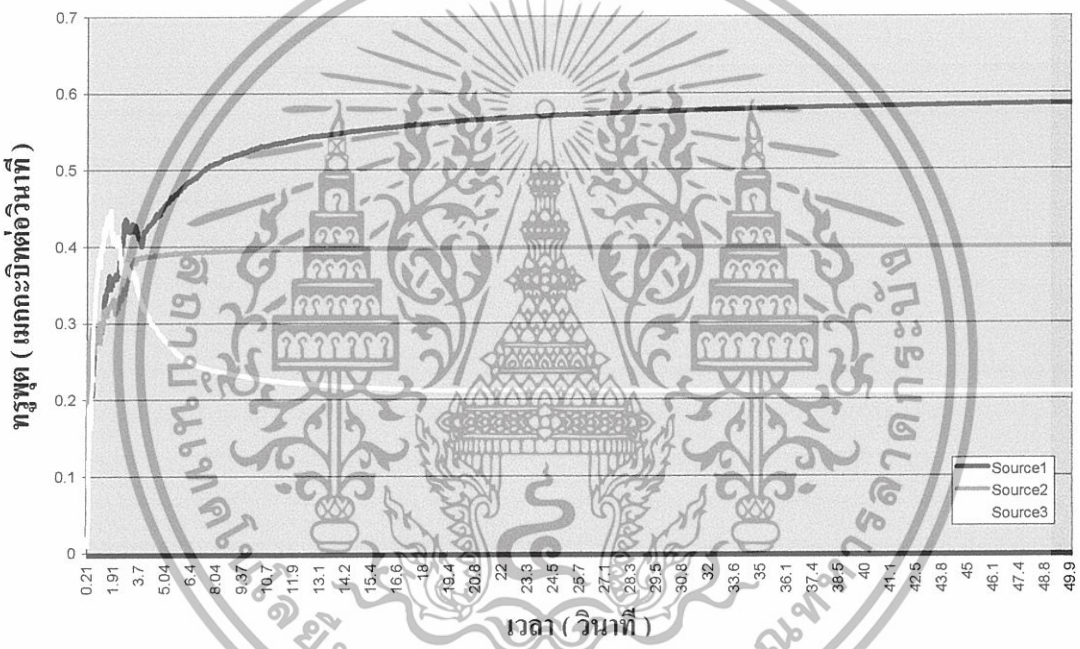
จากรูปที่ 4.1 ให้ Source ทั้งสามส่งแพคเก็ตชนิดยูดีพีด้วยอัตรา 2 เมกะบิตต่อวินาทีไปยังโหนด Dest. เพื่อที่จะให้เกิดการจำกัดทราฟฟิคโดยที่ใช้กลไก HTB ที่โหนดตรงกลางแล้วกำหนดค่า AR ของคลาส Source1 , Soure2 , Source3 เป็น 3:2:1 ตามลำดับจากนั้นทำการวัดค่าความหน่วงเฉลี่ยของแพคเก็ตจากต้นทางไปยังปลายทาง และ วัดค่าทรูพุตเฉลี่ยของแต่ละ Source

ผลการทดลอง

ความหน่วงเฉลี่ยจาก Source1 ถึง Dest. คือ 18.40 วินาที

ความหน่วงเฉลี่ยจาก Source2 ถึง Dest. คือ 19.69 วินาที

ความหน่วงเฉลี่ยจาก Source3 ถึง Dest. คือ 20.28 วินาที



รูปที่ 4.5 กราฟแสดงค่าทรูพุตเมื่อมีการใช้กลไก HTBและมีอัตราส่งแพคเก็ต 2 เมกะบิตต่อวินาที

สรุปผลการทดลอง

เมื่อมีการใช้กลไก HTB เข้ามาควบคุมปริมาณทราฟฟิคและเมื่อมีทราฟฟิคของแต่ละคลาสที่มากกว่าค่า AR ของคลาสจะเกิดการยืมแบนด์วิทด์ที่ยังคงเหลือจากผลต่างของทราฟฟิครวมในขณะนั้นกับค่า CR ของ Root และเมื่อทราฟฟิครวมมีค่าเกินค่า CR ของ Root จะทำให้ไม่มีแบนด์วิทด์เหลือให้ยืมจึงทำให้แบนด์วิทด์ที่ใช้ได้ของแต่ละคลาสถูกจำกัดไว้ที่ค่า AR ของแต่ละคลาสซึ่งจะทำให้ได้ทรูพุตของแต่ละ Source คือ 0.6 , 0.4 , 0.2 เมกกะบิตต่อวินาที เป็นอัตราส่วน 3:2:1 ตามที่กำหนดไว้และมีทรูพุตรวมเท่ากับ 1.2 เมกกะบิตต่อวินาทีเท่ากับค่าแบนด์วิทด์ของสายที่สามารถใช้ได้

4.1.2.2 การทดลองกำหนดค่า CR และค่า AR ของกลไก HTB ในลักษณะต่างๆ

4.1.2.2.1 การทดลองกำหนดค่า CR และค่า AR ให้แบนด์วิทด์รวมที่ทราฟฟิคทุกคลาสจะใช้ได้มีค่าเท่ากับแบนด์วิทด์ของสาย

ตารางที่ 4.3 แสดงค่ากำหนดในการทดลองที่ 4.1.2.2.1

	Ceil Rate (Mbps)	Assure Rate (Mbps)
Root Node	1.2	1.2
Source 1	1.2	0.6
Source 2	1.2	0.4
Source 3	1.2	0.2

วิธีการทดลอง

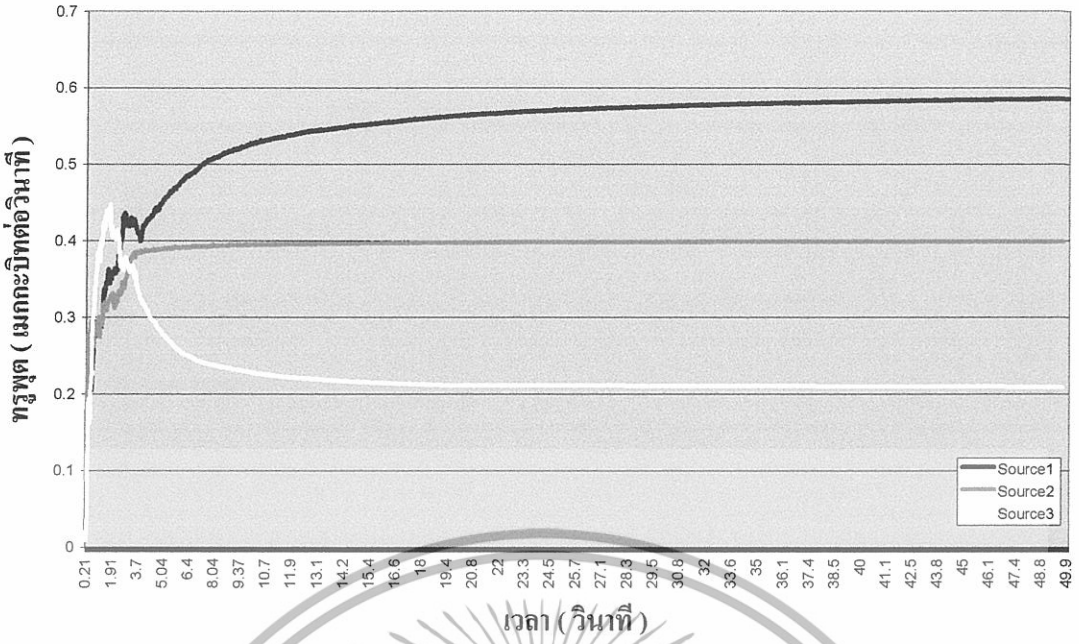
จากรูปที่ 4.1 ให้ Source ทั้งสามส่งแพคเก็ตชนิดยูดีพีด้วยอัตรา 2 เมกะบิตต่อวินาทีไปยังโหนด Dest. โดยที่ใช้กลไก HTB ที่โหนดตรงกลางแล้วกำหนดค่า AR ของคลาส Source1 , Soure2 , Source3 เป็น 3:2:1 ตามลำดับ โดยที่ให้มีค่า AR รวมกันทั้งสามคลาสได้ 1.2 เมกกะบิตต่อวินาที เท่ากับค่าแบนด์วิทด์ของสายที่สามารถใช้ได้ และกำหนดค่า AR , CR ของ Root ให้เท่ากับค่าแบนด์วิทด์ของสายเช่นเดียวกัน จากนั้นทำการวัดค่าความหน่วงเฉลี่ยของแพคเก็ตจากต้นทางไปยังปลายทาง และ วัดค่าทรูพุตเฉลี่ยของแต่ละ Source

ผลการทดลอง

ความหน่วงเฉลี่ยจาก Source1 ถึง Dest. คือ 18.40 วินาที

ความหน่วงเฉลี่ยจาก Source2 ถึง Dest. คือ 19.69 วินาที

ความหน่วงเฉลี่ยจาก Source3 ถึง Dest. คือ 20.28 วินาที



รูปที่ 4.6 กราฟแสดงค่าทรูพุตเมื่อให้แบนด์วิทด์รวมที่ทุกคลาสจะใช้ได้เท่ากับแบนด์วิทด์ของสาย

สรุปผลการทดลอง

เมื่อมีการใช้กลไก HTB เข้ามาควบคุมปริมาณทราฟฟิคโดยกำหนดให้ค่า AR และ CR ของคลาส Root เท่ากับค่าแบนด์วิดท์ของสายที่มีอยู่จริงคือ 1.2 เมกะบิทต่อวินาที และให้ AR ของคลาสลูกรวมกันทั้งหมดไม่เกินค่าของแบนด์วิทด์ของสาย จะพบว่าในช่วงแรกที่มีอัตราส่งแพคเก็ตเกิน 0.2 เมกะบิตต่อวินาทีทำให้ Source3 มีการยืมแบนด์วิดท์จนถึง ณ เวลาที่ทรูพุตมีค่า 0.4 เมกะบิตต่อวินาที ทำให้มีทราฟฟิคการส่งของแต่ละ Source รวมกันเกิน 1.2 เมกะบิทต่อวินาที จะทำให้แบนด์วิทด์ที่ยังคงเหลือจากผลต่างของทราฟฟิครวมในขณะนั้นกับค่า CR ของ Root ไม่มีค่าเหลือจะให้แต่ละคลาสยืมได้ กลไก HTB จะทำการควบคุมแบนด์วิทด์ที่แต่ละ Source จะใช้ได้มีค่าเท่ากับค่า AR ของคลาสเป็นอัตราส่วน 3:2:1 ตามที่ได้กำหนดไว้ จึงทำให้ทรูพุตรวมทั้งหมดจะมีค่าเท่ากับแบนด์วิดท์ของสายคือ 1.2 เมกะบิตต่อวินาที และเมื่อพิจารณาจากค่าความหน่วงเมื่อเปรียบเทียบกับการทดลองที่ 4.1.1 ที่ Source1 ไม่มีการจำกัดแบนด์วิทด์ทำให้มีค่าทรูพุต 0.4 เมกกะบิตต่อวินาที ส่วนในการทดลองนี้มีการจำกัดแบนด์วิทด์ทำให้มีค่าทรูพุต 0.6 เมกกะบิตต่อวินาที จึงเป็นผลให้ค่าความหน่วงมีค่าลดน้อยลงเมื่อมีทรูพุตมากขึ้น เป็นแนวทางเดียวกับ Source3 ที่มีค่าทรูพุตลดน้อยลงกว่าของการทดลองที่ 4.1.1 เนื่องจากการจำกัดแบนด์วิทด์จึงมีผลทำให้ค่าความหน่วงเพิ่มมากขึ้น

4.1.2.2.2 การทดลองกำหนดค่า CR และค่า AR ให้แบนด์วิทด์รวมที่ทราฟฟิคทุกคลาสจะใช้ได้มีค่ามากกว่าแบนด์วิทด์ของสาย

ตารางที่ 4.4 แสดงค่ากำหนดในการทดลองที่ 4.1.2.2.2

	Ceil Rate (Mbps)	Assure Rate (Mbps)
Root Node	12	12
Source 1	12	1
Source 2	12	0.66
Source 3	12	0.33

วิธีการทดลอง

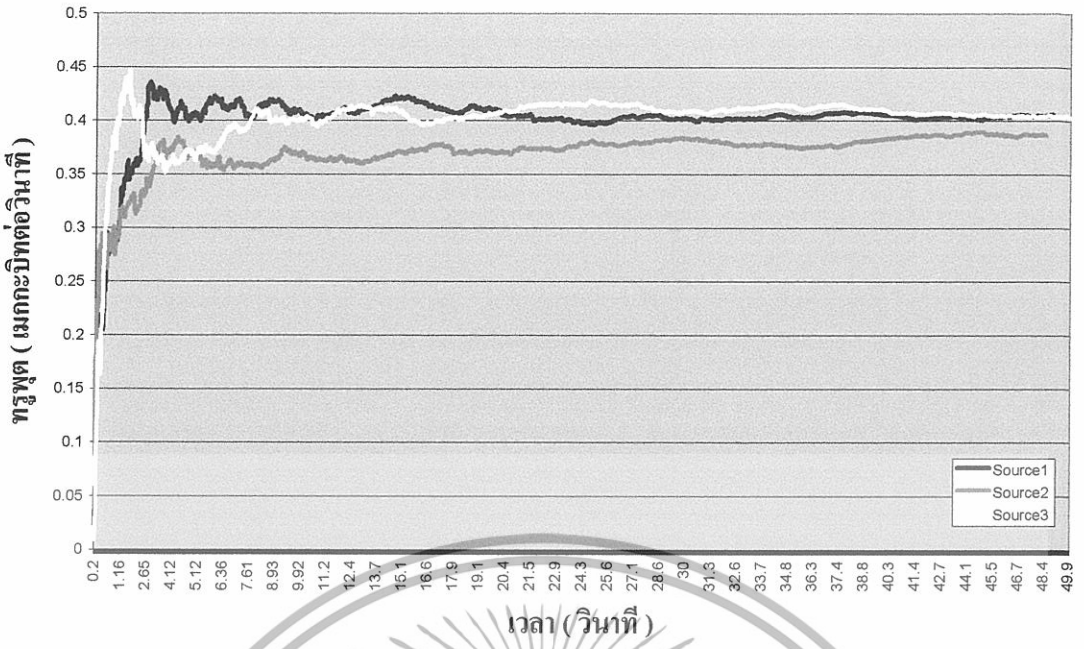
จากรูปที่ 4.1 ให้ Source ทั้งสามส่งแพคเก็ตชนิดยูดีพีด้วยอัตรา 2 เมกะบิตต่อวินาทีไปยังโหนด Dest. โดยที่ใช้กลไก HTB ที่โหนดตรงกลางแล้วกำหนดค่า AR ของคลาส Source1 , Soure2 , Source3 เป็น 3:2:1 ตามลำดับ โดยที่ให้มีค่า AR รวมกันทั้งสามคลาสได้ 2 เมกะบิตต่อวินาทีซึ่งมากกว่าค่าแบนด์วิทด์ของสายที่สามารถใช้ได้ และกำหนดค่า AR , CR ของ Root ให้มากกว่าค่าแบนด์วิทด์ของสายเช่นเดียวกัน เพื่อที่จะให้มีแบนด์วิทด์ที่จะให้แต่ละคลาสยืมได้มีมากเกินกว่าที่แบนด์วิทด์ของสายรองรับ จากนั้นทำการวัดค่าความหน่วงเฉลี่ยของแพคเก็ตจากต้นทางไปยังปลายทาง และ วัดค่าทรูพุตเฉลี่ยของแต่ละ Source

ผลการทดลอง

ความหน่วงเฉลี่ยจาก Source1 ถึง Dest. คือ 19.96 วินาที

ความหน่วงเฉลี่ยจาก Source2 ถึง Dest. คือ 20.29 วินาที

ความหน่วงเฉลี่ยจาก Source3 ถึง Dest. คือ 19.52 วินาที



รูปที่ 4.7 กราฟแสดงค่าทรูพุตเมื่อให้แบนด์วิทด์รวมที่ทุกคลาสจะใช้ได้มากกว่าแบนด์วิทด์ของสาย

สรุปผลการทดลอง

เมื่อมีการใช้กลไก HTB เข้ามาควบคุมปริมาณทราฟฟิคโดยกำหนดให้ค่า AR และ CR ของคลาส Root มากกว่าค่าแบนด์วิดท์ของสายที่มีอยู่จริงคือ 12 เมกะบิทต่อวินาที และให้ AR ของคลาสลูกรวมกันทั้งหมดเกินค่าของแบนด์วิทด์ของสาย จะพบว่าในช่วงแรกที่มีอัตราส่งแพคเก็ตเกิน 0.2 เมกะบิตต่อวินาทีทำให้ Source3 มีการยืมแบนด์วิดท์จนถึง ณ เวลาที่ทรูพุตมีค่า 0.4 เมกะบิตต่อวินาทีทำให้มีทราฟฟิคการส่งของแต่ละ Source รวมกันเกิน 1.2 เมกะบิทต่อวินาที จะทำให้แบนด์วิทด์ที่ยังคงเหลือจากผลต่างของทราฟฟิครวมในขณะนั้นกับค่า CR ของ Root มีค่าเหลือจะให้แต่ละคลาสยืมได้ แต่ว่าเมื่อกลไก HTB ให้แต่ละคลาสมีการยืมแบนด์วิทด์แล้วจะพบว่าค่าแบนด์วิทด์หลังจากยืมมาแล้วของแต่ละคลาสรวมกันมีมากกว่าแบนด์วิทด์ของสายที่สามารถใช้ได้ จึงทำให้ค่าทรูพุตรวมของทุก Source ถูกจำกัดไว้โดยแบนด์วิทด์ของสายที่ 1.2 เมกกะบิตต่อวินาทีโดยที่แต่ละ Source จะมีทรูพุตเฉลี่ยเท่าๆกันคือประมาณ 0.4 เมกกะบิตต่อวินาที ทำให้ค่าแบนด์วิทด์ที่แต่ละคลาสจะใช้ได้ไม่เป็นอัตราส่วน 3:2:1 ตามที่ได้กำหนดไว้ และเมื่อพิจารณาจากค่าความหน่วงเมื่อเปรียบเทียบกับการทดลองที่ 4.1.1 จะพบว่ามีค่าใกล้เคียงกันเนื่องจากในการทดลองนี้แต่ละ Source มีทรูพุตที่ถูกจำกัดโดยแบนด์วิทด์ของสายไว้ที่ 0.4 เมกกะบิตต่อวินาทีทำให้มีค่าความหน่วงใกล้เคียงกับการทดลองที่ 4.1.1 ที่ทรูพุตของแต่ละ Source มีค่าประมาณ 0.4 เมกกะบิตต่อวินาทีเช่นเดียวกัน และเมื่อเปรียบเทียบกับการทดลองที่ 4.1.2.2.1 จะพบว่าค่าความหน่วงของ Source3 มีค่าลดลงเนื่องจากมีค่าทรูพุตที่มากกว่าการทดลองที่ 4.1.2.2.1 ที่ถูกจำกัดแบนด์วิทด์ไว้

4.1.2.2.3 การทดลองกำหนดค่า CR และ AR ให้แบนด์วิทด์รวมที่ทราฟฟิคทุกคลาสจะใช้ได้มีค่าน้อยกว่าแบนด์วิทด์ของสาย

ตารางที่ 4.5 แสดงค่ากำหนดในการทดลองที่ 4.1.2.2.3

	Ceil Rate (Mbps)	Assure Rate (Mbps)
Root Node	**1.2**	**1.2**
Source 1	**0.5**	**0.3**
Source 2	**0.5**	**0.2**
Source 3	**0.5**	**0.1**

วิธีการทดลอง

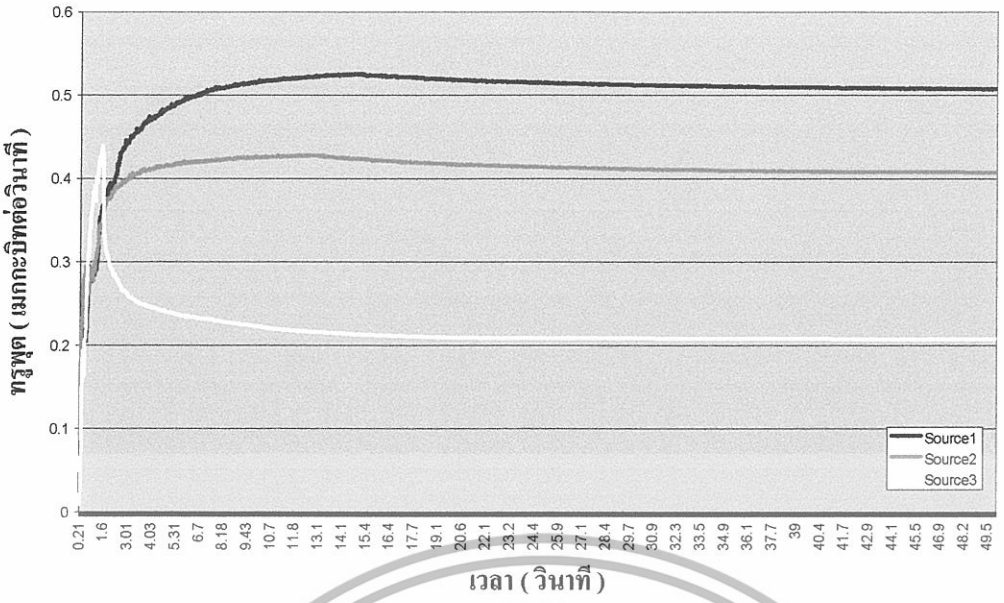
จากรูปที่ 4.1 ให้ Source ทั้งสามส่งแพคเก็ตชนิดยูดีพีด้วยอัตรา 2 เมกะบิตต่อวินาทีไปยังโหนด Dest. โดยที่ใช้กลไก HTB ที่โหนดตรงกลางแล้วกำหนดค่า AR ของคลาส Source1 , Soure2 , Source3 เป็น 3:2:1 ตามลำดับ โดยที่ให้มีค่า AR รวมกันทั้งสามคลาสได้ 0.6 เมกะบิตต่อวินาทีซึ่งน้อยกว่าค่าแบนด์วิทด์ของสายที่สามารถใช้ได้ และกำหนดค่า CR ของแต่ละคลาสให้น้อยกว่าค่าแบนด์วิทด์ของสายเช่นเดียวกัน เพื่อที่จะจำกัดแบนด์วิทด์มากที่สุดที่แต่ละคลาสจะสามารถใช้ได้ จากนั้นทำการวัดค่าความหน่วงเฉลี่ยของแพคเก็ตจากต้นทางไปยังปลายทาง และ วัดค่าทรูพุตเฉลี่ยของแต่ละ Source

ผลการทดลอง

ความหน่วงเฉลี่ยจาก Source1 ถึง Dest. คือ 18.51 วินาที

ความหน่วงเฉลี่ยจาก Source2 ถึง Dest. คือ 19.16 วินาที

ความหน่วงเฉลี่ยจาก Source3 ถึง Dest. คือ 20.66 วินาที

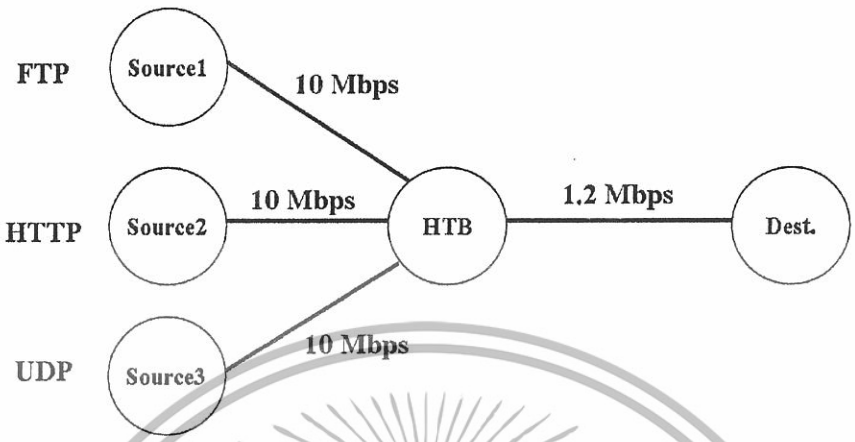


รูปที่ 4.8 กราฟแสดงค่าทรูพุตเมื่อให้แบนด์วิทด์รวมที่ทุกคลาสจะใช้ได้น้อยกว่าแบนด์วิทด์ของสาย

สรุปผลการทดลอง

เมื่อมีการใช้กลไก HTB เข้ามาควบคุมปริมาณทราฟฟิคโดยกำหนดให้ค่า AR และ CR ของคลาส Root มากกว่าค่าแบนด์วิดท์ของสายที่มีอยู่จริงคือ 1.2 เมกะบิทต่อวินาที และให้ค่า CR ของแต่ละคลาสลูกน้อยกว่าค่าแบนด์วิทด์ของสาย จะพบว่าในช่วงแรกที่มีอัตราส่งแพคเก็ตเกิน 0.2 เมกะบิตต่อวินาทีทำให้ Source3 มีการยืมแบนด์วิดท์จนถึง ณ เวลาที่ทรูพุตมีค่า 0.4 เมกะบิตต่อวินาที ทำให้มีทราฟฟิคการส่งของแต่ละ Source รวมกันเกิน 1.2 เมกะบิทต่อวินาที กลไก HTB จะมีการยืมแบนด์วิทด์เนื่องจากทราฟฟิคของแต่ละคลาสเกินค่า AR ของคลาสตามที่ได้กำหนดไว้ แต่คลาส Source1 หลังจากคำนวณค่าแบนด์วิทด์ที่ยืมแล้วจะสามารถยืมได้เป็น 0.6 เมกะบิตต่อวินาทีแต่มีการกำหนดค่า CR ไว้ 0.5 เมกะบิตต่อวินาทีจึงทำให้ทรูพุตของ Source1 มีค่าได้สูงสุดเท่ากับ 0.5 เมกะบิตต่อวินาทีเนื่องจากการจำกัดแบนด์วิทด์สูงสุดของค่า CR ที่กำหนด และทำให้แบนด์วิทด์รวมทั้งหมดจะมีค่าน้อยกว่าแบนด์วิดท์ของสายคือ 1.1 เมกะบิตต่อวินาที ซึ่งเป็นผลมาจากการถูกจำกัดแบนด์วิทด์สูงสุดของคลาส Source1 และเมื่อพิจารณาจากค่าความหน่วงเมื่อเปรียบเทียบกับการทดลองที่ 4.1.1 ที่ Source1 ไม่มีการจำกัดแบนด์วิทด์ทำให้มีค่าทรูพุต 0.4 เมกกะบิตต่อวินาที ส่วนในการทดลองนี้มีการจำกัดแบนด์วิทด์สูงสุดทำให้มีค่าทรูพุต 0.5 เมกกะบิตต่อวินาที จึงเป็นผลให้ค่าความหน่วงมีค่าลดน้อยลงเมื่อมีทรูพุตมากขึ้นและถ้าเทียบกับการทดลองที่ 4.1.2.2.1 ที่ Source1 มีทรูพุตเท่ากับ 0.6 เมกะบิตต่อวินาทีจะพบว่าในการทดลองนี้ Source1 มีค่าความหน่วงที่มากกว่าเล็กน้อยเนื่องมาจากการจำกัดแบนด์วิทด์ทำให้มีทรูพุตน้อยกว่าคือ 0.5 เมกะบิตต่อวินาที

4.1.3 การทดลองเพื่อแสดงประสิทธิภาพของเครือข่ายเมื่อมีการใช้กลไก HTB โดยทำการจำกัดปริมาณทราฟฟิคตามลักษณะแอพพลิเคชั่น



รูปที่ 4.9 แสดงโทโพโลยีของการทดลอง 4.1.3

4.1.3.1 การทดลองแสดงประสิทธิภาพโปรโตคอล TCP/IP เมื่อไม่มีการจำกัดปริมาณทราฟฟิคด้วยกลไก HTB

วิธีการทดลอง

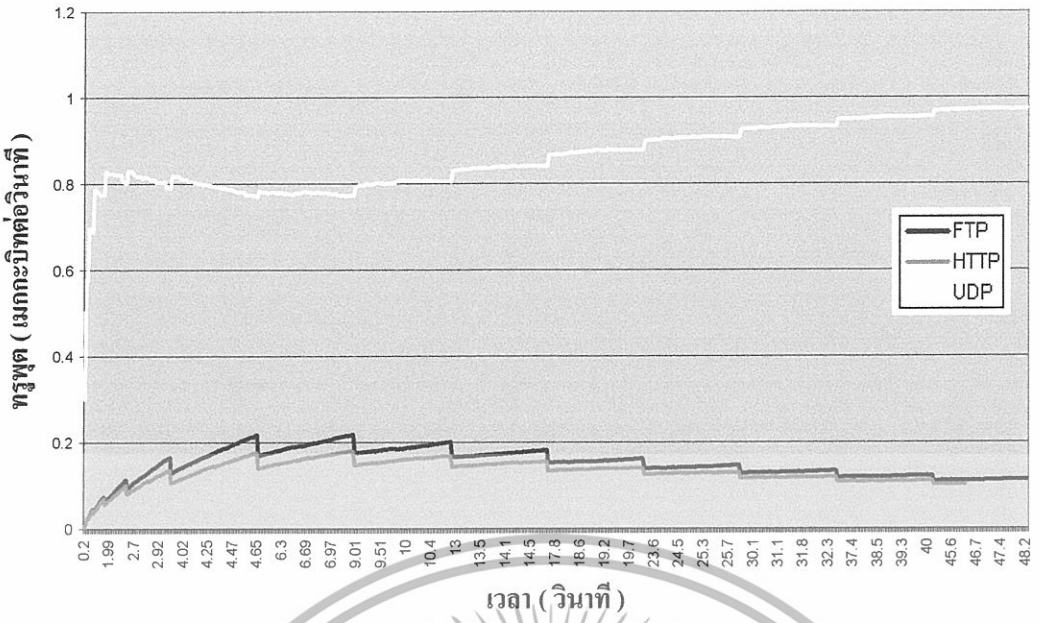
จากรูปที่ 4.9 กำหนดให้ Source1 ส่งแพคเก็ตชนิดเอฟทีพีด้วยอัตรา 1.2 เมกะบิตต่อวินาทีไปยัง Dest. และ Source2 ส่งแพคเก็ตชนิดเอชทีทีพีด้วยอัตรา 1.2 เมกะบิตต่อวินาทีไปยัง Dest. และ Source3 ส่งแพคเก็ตชนิดยูดีพีด้วยอัตรา 1.2 เมกะบิตต่อวินาทีไปยัง Dest. แล้วทำการวัดค่าความหน่วงเฉลี่ยของแพคเก็ตจากต้นทางไปยังปลายทาง และ วัดค่าทรูพุตเฉลี่ยของแต่ละ Source

ผลการทดลอง

ความหน่วงเฉลี่ยจาก Source1 ถึง Dest. คือ 4.31 วินาที

ความหน่วงเฉลี่ยจาก Source2 ถึง Dest. คือ 4.44 วินาที

ความหน่วงเฉลี่ยจาก Source3 ถึง Dest. คือ 5.37 วินาที

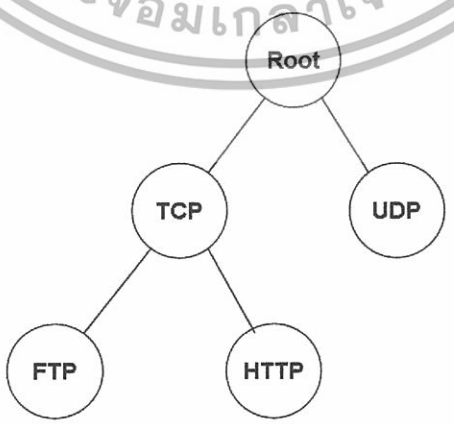


รูปที่ 4.10 แสดงค่าทรูพุตของประสิทธิภาพโปรโตคอล TCP/IP เมื่อไม่มีการใช้ HTB

สรุปผลการทดลอง

เมื่อไม่มีการใช้กลไก HTB เข้ามาควบคุมทราฟฟิคจะทำให้ทรูพุตที่ได้จากแพคเก็ตประเภทเอฟทีพีและเอชทีทีพีมีค่าที่ต่ำมากกว่าอัตราการส่งข้อมูลที่กำหนด เนื่องมาจากแพคเก็ตประเภทยูดีพีมีการใช้แบนด์วิทด์ในปริมาณมากส่งผลให้ทราฟฟิคที่เป็นโปรโตคอล TCP คือเอฟทีพีและเอชทีทีพีเกิดการรอคอยในบัฟเฟอร์ของคิวทำให้มีผลให้สไลด์ดิ้งวินโดว์ลดขนาดหน้าต่างการส่งลงทำให้ Source1 และ Source2 มีอัตราการส่งแพคเก็ตที่ลดลงจึงทำให้ค่าทรูพุตที่ได้จาก Source1 และ Source2 มีค่าที่ต่ำลง

4.1.3.2 การทดลองแสดงประสิทธิภาพโปรโตคอล TCP/IP เมื่อมีการจำกัดปริมาณทราฟฟิคด้วยกลไก HTB



รูปที่ 4.11 แสดงการแบ่งคลาสของกลไกHTBตามลักษณะแอพพลิเคชั่น

4.1.3.2.1 การทดลองแสดงประสิทธิภาพโปรโตคอล TCP/IP เมื่อกำหนดอัตราการส่งข้อมูลสูงสุดของคลาส UDP ให้เท่ากับแบนด์วิทด์ของสายที่สามารถใช้ได้

ตารางที่ 4.6 แสดงค่ากำหนดในการทดลองที่ 4.1.3.2.1

	Ceil Rate (Mbps)	Assure Rate (Mbps)
Root Node	1.2	1.2
TCP	1.2	1.0
UDP (source3)	1.2	0.2
FTP (source1)	1.0	0.6
HTTP (source2)	1.0	0.4

วิธีการทดลอง

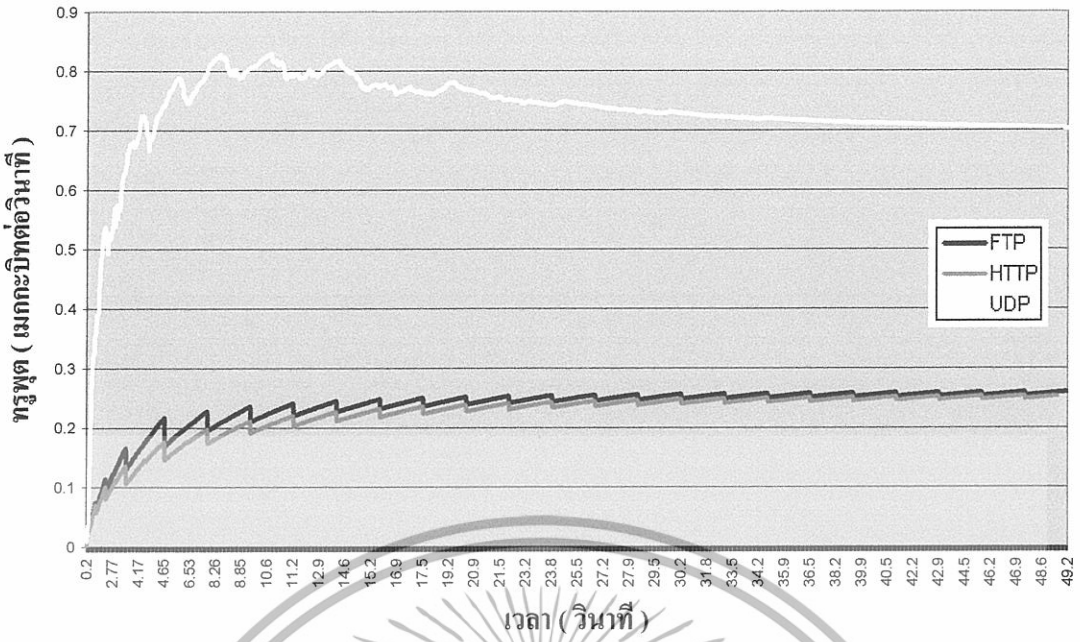
จากรูปที่ 4.10 กำหนดให้คลาส TCP มีค่า AR ที่มากกว่าคลาส UDP และคลาส FTP มีค่า AR ที่มากกว่าคลาส HTTP และกำหนดให้ค่า CR ของคลาส UDP มีค่าเป็น 1.2 เมกะบิตต่อวินาที เท่ากับค่าแบนด์วิทด์ของสายเพื่อให้คลาส UDP มีแบนด์วิทด์ที่สามารถใช้ได้เต็มที่ และจากรูปที่ 4.9 กำหนดให้ Source1 ส่งแพคเก็ตชนิดเอฟทีพีด้วยอัตรา 1.2 เมกะบิตต่อวินาทีไปยัง Dest. และ Source2 ส่งแพคเก็ตชนิดเอชทีทีพีด้วยอัตรา 1.2 เมกะบิตต่อวินาทีไปยัง Dest. และ Source3 ส่งแพคเก็ตชนิดยูดีพีด้วยอัตรา 1.2 เมกะบิตต่อวินาทีไปยัง Dest. โดยที่ใช้กลไก HTB ที่โหนดตรงกลาง แล้วทำการวัดค่าความหน่วงเฉลี่ยของแพคเก็ตจากต้นทางไปยังปลายทาง และ วัดค่าทรูพุตเฉลี่ยของแต่ละ Source

ผลการทดลอง

ความหน่วงเฉลี่ยจาก Source1 ถึง Dest. คือ 1.82 วินาที

ความหน่วงเฉลี่ยจาก Source2 ถึง Dest. คือ 1.85 วินาที

ความหน่วงเฉลี่ยจาก Source3 ถึง Dest. คือ 9.94 วินาที



รูปที่ 4.12 แสดงค่าทรูพุตของประสิทธิภาพโปรโตคอล TCP/IP เมื่อกำหนดอัตราการส่งข้อมูลสูงสุดของคลาส UDP ให้เท่ากับแบนด์วิทด์ของสายที่สามารถใช้ได้

สรุปผลการทดลอง

เมื่อมีการใช้กลไก HTB เข้ามาควบคุมทราฟฟิคจะพบว่าทรูพุตของ Source1 และ Source2 มีค่าเพิ่มมากขึ้นกว่าในการทดลองที่ 4.1.3.1 ที่ไม่ใช้กลไก HTB แต่ค่าทรูพุตที่ได้จะยังไม่ถึงค่า AR ที่เป็นค่าอัตราการส่งข้อมูลขั้นต่ำเนื่องจากคลาสยูดีพีมีการกำหนดแบนด์วิทด์สูงสุดที่สามารถใช้ได้ คือ 1.2 เมกกะบิตต่อวินาทีเท่ากับแบนด์วิทด์ของสาย ทำให้เมื่อกำหนดอัตราการส่งของแพคเก็ตประเภทยูดีพี 1.2 เมกะบิตต่อวินาที ส่งผลให้ทราฟฟิคประเภทยูดีพีมีการใช้แบนด์วิทด์ของสายในปริมาณมาก จึงเป็นผลให้ทราฟฟิคที่เป็นโปรโตคอล TCP คือเอฟทีพีและเอชทีทีพีเกิดการรอคอยในบัฟเฟอร์ของคิวทำให้สไลด์ดิ้งวินโดว์ของโปรโตคอล TCP ลดขนาดหน้าต่างการส่งลงจึงทำให้อัตราการส่งแพคเก็ตของ Source1 และ Source2 มีค่าลดลงน้อยกว่าค่า AR และมีผลให้ทรูพุตมีค่าลดต่ำลงด้วย และเมื่อพิจารณาจากค่าความหน่วงจะพบว่า Source1 และ Source2 จะมีค่าความหน่วงลดน้อยลงกว่าการทดลองที่ 4.1.3.1 เนื่องมาจากค่าทรูพุตที่เพิ่มมากขึ้นที่เป็นผลมาจากการกำหนดค่า AR ที่เป็นแบนด์วิทด์ขั้นต่ำที่สามารถใช้ได้ของคลาส FTP และ HTTP ส่วนค่าความหน่วงของ Source3 เพิ่มมากกว่าในการทดลองที่ 4.1.3.1 เนื่องจากค่าทรูพุตที่ลดน้อยลง

4.1.3.2.2 การทดลองแสดงประสิทธิภาพโปรโตคอล TCP/IP เมื่อกำหนดอัตราการส่งข้อมูลสูงสุดของคลาส UDP ให้น้อยกว่าแบนด์วิทด์ของสายที่สามารถใช้ได้

ตารางที่ 4.7 แสดงค่ากำหนดในการทดลองที่ 4.1.3.2.2

	Ceil Rate (Mbps)	Assure Rate (Mbps)
Root Node	**1.2**	**1.2**
TCP	**1.2**	**1.0**
UDP (source3)	**0.4**	**0.2**
FTP (source1)	**1.0**	**0.6**
HTTP (source2)	**1.0**	**0.4**

วิธีการทดลอง

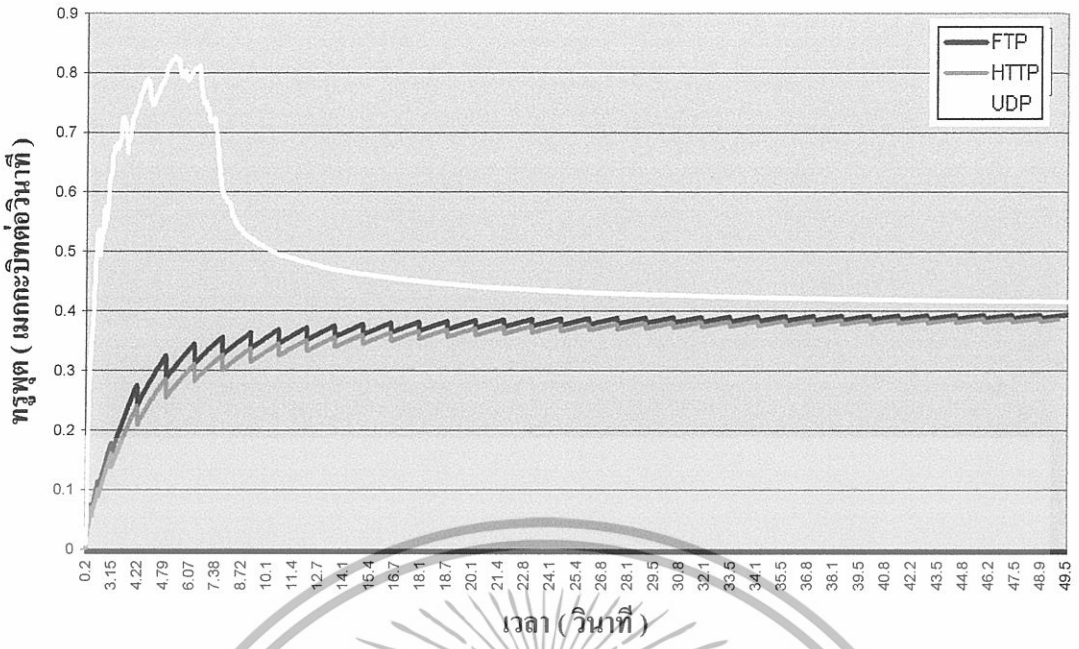
จากรูปที่ 4.10 กำหนดให้คลาส TCP มีค่า AR ที่มากกว่าคลาส UDP และคลาส FTP มีค่า AR ที่มากกว่าคลาส HTTP และกำหนดให้ค่า CR ของคลาส UDP มีค่าเป็น 0.4 เมกะบิตต่อวินาที น้อยกว่าค่าแบนด์วิทด์ของสายเพื่อให้คลาส UDP มีแบนด์วิทด์ที่สามารถใช้สูงสุดจำกัดที่ 0.4 เมกะบิตต่อวินาที และจากรูปที่ 4.9 กำหนดให้ Source1 ส่งแพคเก็ตชนิดเอฟทีพีด้วยอัตรา 1.2 เมกะบิตต่อวินาทีไปยัง Dest. และ Source2 ส่งแพคเก็ตชนิดเอชทีทีพีด้วยอัตรา 1.2 เมกะบิตต่อวินาทีไปยัง Dest. และ Source3 ส่งแพคเก็ตชนิดยูดีพีด้วยอัตรา 1.2 เมกะบิตต่อวินาทีไปยัง Dest. โดยที่ใช้กลไก HTB ที่โหนดตรงกลาง แล้วทำการวัดค่าความหน่วงเฉลี่ยของแพคเก็ตจากต้นทางไปยังปลายทาง และ วัดค่าทรูพุตเฉลี่ยของแต่ละ Source

ผลการทดลอง

ความหน่วงเฉลี่ยจาก Source1 ถึง Dest. คือ 1.14 วินาที

ความหน่วงเฉลี่ยจาก Source2 ถึง Dest. คือ 1.16 วินาที

ความหน่วงเฉลี่ยจาก Source3 ถึง Dest. คือ 15.32 วินาที



รูปที่ 4.13 แสดงค่าทรูพุตของประสิทธิภาพโปรโตคอล TCP/IP เมื่อมีการกำหนดอัตราการส่งข้อมูลสูงสุดของคลาส UDP ให้น้อยกว่าแบนด์วิทด์ของสายที่สามารถใช้ได้

สรุปผลการทดลอง

เมื่อมีการใช้กลไก HTB เข้ามาควบคุมทราฟฟิคและกำหนดค่า CR ของคลาส UDP ให้น้อยกว่าแบนด์วิทด์ของสายคือ 0.4 เมกะบิตต่อวินาที จะพบว่าในช่วงแรกทรูพุตของแพคเก็ตชนิด UDP จะมีค่าเกิน CR และค่อยๆลดต่ำลงมาเนื่องจากการกำหนดขนาดของตระกร้าเท่ากับค่า CR ของคลาสทำให้มีจำนวนโทเคนเหลือให้ใช้ได้ในตอนที่มีอัตราการส่งของแพคเก็ตเกิน 0.5 เมกะบิตต่อวินาทีและเมื่อจำนวนโทเคนหมดจากตระกร้าทำให้ทรูพุตมีค่าลดต่ำลงมาที่ 0.4 เมกบิตต่อวินาที และจากผลการทดลองจะพบว่าทรูพุตของ Source1 และ Source2 มีค่าเพิ่มมากขึ้นกว่าในการทดลองที่ 4.1.3.2.1 แต่ค่าทรูพุตที่ได้จะยังไม่ถึงค่า AR ที่เป็นค่าอัตราการส่งข้อมูลขั้นต่ำเนื่องจากคลาสยูดีพีที่มีการกำหนดอัตราการส่งของแพคเก็ตประเภทยูดีพี 1.2 เมกะบิตต่อวินาที ยังส่งผลให้ทราฟฟิคประเภทยูดีพีมีการใช้แบนด์วิทด์ของสายในปริมาณมาก จึงเป็นผลให้ทราฟฟิคที่เป็นโปรโตคอล TCP คือเอฟทีพีและเอชทีทีพีเกิดการรอคอยในบัฟเฟอร์ของคิวทำให้สไลด์ดิ้งวินโดว์ของโปรโตคอล TCP ลดขนาดหน้าต่างการส่งลงจึงทำให้อัตราการส่งแพคเก็ตของ Source1 และ Source2 มีค่าลดลง และมีผลให้ทรูพุตมีค่าลดต่ำลงด้วย ส่วนคลาส UDP จะมีการกำหนดค่า CR เท่ากับ 0.4 เมกะบิตต่อวินาทีทำให้ทรูพุตของ Source3 มีค่าสูงสุดที่ 0.4 เมกะบิตต่อวินาที และเมื่อพิจารณาจากค่าความหน่วงจะพบว่า Source1 และ Source2 จะมีค่าความหน่วงลดน้อยลงกว่าการทดลองที่ 4.1.3.2.1 เนื่องมาจากค่าทรูพุตที่เพิ่มมากขึ้น

บทที่ 5
สรุปผลการดำเนินโครงงาน และข้อเสนอแนะ

การวัดประสิทธิภาพของเน็ตเวิร์คในปัจจุบันส่วนใหญ่จะใช้การจำลอง (Simulation) เนื่องด้วยการจำลองสามารถจำลองระบบเครือข่ายอย่างง่าย จนถึงระบบเครือข่ายที่มีความซับซ้อน สามารถทำได้ง่ายและประหยัดค่าใช้จ่าย และสิ่งที่สำคัญคือ ผลการทดลองจากการเปรียบเทียบปัจจัยที่มีผลต่อประสิทธิภาพในเรื่องต่างๆของเครือข่ายในการจำลองเมื่อเปรียบเทียบกับโมเดลทางคณิตศาตร์แล้วควรมีแน้วโน้มไปในทิศทางเดียวกัน รวมทั้งสามารถเพิ่มความสามารถของโปรแกรมจำลองด้วยการศึกษาวิธีการสร้างโมดูลพร้อมทั้งศึกษาการเพิ่มเติมโมดูลบนโปรแกรมจำลองที่ได้นำมาจำลอง ดังนั้นจึงมีการเลือกใช้การจำลองซึ่งสามารถให้ผลลัพธ์ของการจำลองเครือข่ายที่เสมือนจริง และได้เลือกใช้โปรแกรม NS-2(Network Simulation version 2) ซึ่งเป็นโปรแกรมการจำลองระบบเครือข่ายชนิดหนึ่งซึ่งสามารถจำลองระบบเครือข่าย

ศึกษาทฤษฎีและหลักการทำงานของกลไกการทำงานบางส่วนของ HTB ซึ่งเป็นกลไกหนึ่งในเรื่องการควบคุมการจราจรบนเครือข่าย HTB ถูกพัฒนาจากแนวคิดของกลไกโทเคน/บัคเก็ต และการทำงานของซีบีคิว คือ มีการจำแนกประเภททราฟฟิกออกเป็นโครงสร้างแบบลำดับชั้นซึ่งแต่ละกลุ่มจะถูกเรียกว่าคลาส ซึ่งคลาสในระดับชั้นล่างสุดจะถูกจับคู่กับโทเคนเพื่อสามารถส่งทราฟฟิกออกจากระบบ อีกทั้ง HTB ยังมีความสามารถในจัดการแบนด์วิดท์ที่มีอย่างจำกัด โดยมีการกำหนดแบนด์วิดท์ขั้นต่ำเพื่อการันตีแบนด์วิดท์และแบนด์วิดท์สูงสุดของแต่ละคลาสได้รับ เป็นผลทำให้สามารถจัดสรรและสามารถควบคุมการจราจรบนเครือข่ายได้ตามความต้องการ

โครงงานนี้ได้ทำการการเพิ่มความสามารถให้กับโปรแกรม NS-2 ในเรื่องการควบคุมการจราจรบนเครือข่ายโดยลอกเลียนการทำงานของ HTB จำเป็นต้องศึกษาใน 2 ส่วน ในส่วนแรกศึกษาทฤษฎีและหลักการทำงานของ HTB ในส่วนที่สองศึกษาการสร้างโมดูลและการเพิ่มเติมโมดูลบนโปรแกรม NS-2 ซึ่งการสร้างโมดูลนั้นต้องมีความรู้ในการเขียนภาษา TCL และ C++ เนื่องจากโปรแกรม NS-2 สร้างจากทั้ง ภาษา TCL และ C++ เป็นผลให้ในการเพิ่มโมดูลบนโปรแกรม NS-2 จำเป็นต้องรู้วิธีเชื่อมต่อระหว่างภาษา TCL และ C++

จากการศึกษาทฤษฎีและหลักการทำงานของกลไกการทำงานของ HTBและได้เพิ่มเติมโมดูลการสร้างแบบจำลองการควบคุมการจราจรบนเครือข่ายโดยลอกเลียนการทำงานบางส่วนของ HTB ในโปรแกรม NS-2 พบว่ากลไกการทำงานของ HTB ในปัจจุบันมีการพัฒนาความสามารถอย่างต่อเนื่อง ซึ่งเป็นผลให้เราสามารถพัฒนากลไกการทำงานของ HTB ในโปรแกรม NS-2 ให้มีประสิทธิภาพมากยิ่งขึ้นเพื่อสามารถสร้างแบบจำลองการควบคุมการจราจรบนเครือข่ายในโปรแกรม NS-2 และได้ผลลัพธ์ที่เสมือนจริงมากยิ่งขึ้น

บรรณานุกรม

Altman, E. and Jimenez, T. 2003. **"NS Simulator for beginners".** In Lecture notes, 2003-2004 Univ. de Los Andes, Merida, Venezuela and ESSI, Sophia-Antipolis. France. n.p.(no place of publishing).

Brown.M.A 2002, **Traffic Control using tcng and HTB HOWTO** [Online]. Available : http://tldp.org/HOWTO/Traffic-Control-tcng-HTB-HOWTO/index.html

Computer Science of Worcester Polytechnic Institute. 2006. **NS by Example**. [Online]. Available : http://nile.wpi.edu/NS/

Fall, K. and Vardhan, K. 2006. **The *ns* Manual (formerly *ns* Notes and Documentation).** n.p.(no place of publishing).

Information Sciences Institute of University of Southern California. 2006. **The Network Simulator - ns – 2** . [Online]. Available : http://www.isi.edu/nsnam/ns/.

IvanČiĆ, D. , Hadjina, N. and Basch, D. 2005. Analysis of precision of the HTB packet scheduler. ***IEEE Conference Proceeding***, (1): 1-4, October 12-14

M. Devera, "Hierarchical Token Bucket Theory", 2003 Available : http://luxik.cdi.cz/~devik/qos/htb

Traffic Control HOWTO, **7. Classful Queuing Disciplines (qdiscs)** [Online]. Available : http://linux-ip.net/articles/Traffic-Control-HOWTO/classful-qdiscs.html

Wikipedia The Free Encyclopedia, **Token bucket**[Online]. Available : http://en.wikipedia.org/wiki/Token_bucket

ภาคผนวก ก.

ภาคผนวก ก.

Code ที่ใช้ในทดสอบการทำงานกลไก HTB

ก.1 การทดลองที่ 4.1.1 เพื่อแสดงประสิทธิภาพของเครือข่ายเมื่อไม่มีการใช้ HTB

ก.1.1 ไฟล์ htb1.tcl

```
set ns [new Simulator]

#Open the Trace files
set tf [open out.tr w]
$ns trace-all $tf

#Open the NAM trace file
#set namfile [open out.nam w]
#$ns namtrace-all $namfile

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns duplex-link $n0 $n3 100Mb 100ms DropTail #bit
$ns duplex-link $n1 $n3 100Mb 100ms DropTail
$ns duplex-link $n2 $n3 100Mb 100ms DropTail
$ns duplex-link $n3 $n4 100Mb 0ms DropTail
$ns duplex-link $n4 $n5 1.2Mb 100ms DropTail

$ns queue-limit $n0 $n3 10000000
$ns queue-limit $n1 $n3 10000000
$ns queue-limit $n2 $n3 10000000
$ns queue-limit $n3 $n4 10000000
$ns queue-limit $n4 $n5 10000000

# generate random interarrival times and packet sizes
set InterArrivalTime1 [new RandomVariable/Exponential]
$InterArrivalTime1 set avg_ [expr 1/250.0]
set InterArrivalTime2 [new RandomVariable/Exponential]
$InterArrivalTime2 set avg_ [expr 1/250.0]
set InterArrivalTime3 [new RandomVariable/Exponential]
$InterArrivalTime3 set avg_ [expr 1/250.0]
set pktSize [new RandomVariable/Exponential]
$pktSize set avg_ 1000

set src1 [new Agent/UDP]
$src1 set packetSize_ 10000 #byte
$ns attach-agent $n0 $src1

set src2 [new Agent/UDP]
$src2 set packetSize_ 10000
$ns attach-agent $n1 $src2

set src3 [new Agent/UDP]
$src3 set packetSize_ 10000
$ns attach-agent $n2 $src3

set null [new Agent/Null]
$ns attach-agent $n5 $null

$ns connect $src1 $null
$ns connect $src2 $null
$ns connect $src3 $null
```

```
proc finish {} {
        global ns tf namfile
        $ns flush-trace
        #close $namfile
        close $tf
        #exec nam out.nam &
        exit 0
}

proc sendpacket1 {} {
        global ns src1 InterArrivalTime1 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime1 value]] "sendpacket1"
        set bytes [expr round ([$pktSize value])]
        $src1 send $bytes
}

proc sendpacket2 {} {
        global ns src2 InterArrivalTime2 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime2 value]] "sendpacket2"
        set bytes [expr round ([$pktSize value])]
        $src2 send $bytes
}

proc sendpacket3 {} {
        global ns src3 InterArrivalTime3 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime3 value]] "sendpacket3"
        set bytes [expr round ([$pktSize value])]
        $src3 send $bytes
}

$ns at 0.001 "sendpacket1"
$ns at 0.001 "sendpacket2"
$ns at 0.001 "sendpacket3"
$ns at 50.0 "finish"
$ns run
```

ก.2 การทดลองที่ 4.1.2.1.1 เพื่อแสดงประสิทธิภาพของเครือข่ายเมื่อมีการใช้ HTB กำหนดค่ากำหนดค่าอัตราการส่งแพคเก็ตให้มีค่าน้อยกว่าค่ารับประกันอัตราการส่งข้อมูลขั้นต่ำ

ก.2.1 ไฟล์ htb2.tcl

```
set ns [new Simulator]

#Open the Trace files
set tf [open out.tr w]
$ns trace-all $tf

#Open the NAM trace file
#set namfile [open out.nam w]
#$ns namtrace-all $namfile

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns duplex-link $n0 $n3 100Mb 100ms DropTail #bit
$ns duplex-link $n1 $n3 100Mb 100ms DropTail
$ns duplex-link $n2 $n3 100Mb 100ms DropTail
$ns duplex-link $n3 $n4 100Mb 0ms DropTail
$ns duplex-link $n4 $n5 1.2Mb 100ms DropTail

$ns queue-limit $n0 $n3 10000000
```

```
$ns queue-limit $n1 $n3 10000000
$ns queue-limit $n2 $n3 10000000
$ns queue-limit $n3 $n4 10000000
$ns queue-limit $n4 $n5 10000000

# generate random interarrival times and packet sizes
set InterArrivalTime1 [new RandomVariable/Exponential]
$InterArrivalTime1 set avg_ [expr 1/12.5]
set InterArrivalTime2 [new RandomVariable/Exponential]
$InterArrivalTime2 set avg_ [expr 1/12.5]
set InterArrivalTime3 [new RandomVariable/Exponential]
$InterArrivalTime3 set avg_ [expr 1/12.5]
set pktSize [new RandomVariable/Exponential]
$pktSize set avg_ 1000

set src1 [new Agent/UDP]
$src1 set packetSize_ 10000 #byte
$ns attach-agent $n0 $src1

set src2 [new Agent/UDP]
$src2 set packetSize_ 10000
$ns attach-agent $n1 $src2

set src3 [new Agent/UDP]
$src3 set packetSize_ 10000
$ns attach-agent $n2 $src3

set null [new Agent/Null]
$ns attach-agent $n5 $null

set htb [new HTB]

$htb set shape_by_source 1

$htb set root_CR 150k # 1Mbps
$htb set root_AR 150k # byte

$htb set exnode1_CR 150k
$htb set exnode1_AR 75k # 0.6Mbps
$htb set exnode1_bucket 150000
$htb set exnode1_qlen 10000000

$htb set exnode2_CR 150k
$htb set exnode2_AR 50k # 0.4Mbps
$htb set exnode2_bucket 150000
$htb set exnode2_qlen 10000000

$htb set exnode3_CR 150k
$htb set exnode3_AR 25k # 0.2Mbps
$htb set exnode3_bucket 150000
$htb set exnode3_qlen 10000000

$ns attach-shape $htb $n3 $n4
$ns connect $src1 $null
$ns connect $src2 $null
$ns connect $src3 $null

proc finish {} {
        global ns tf namfile
        $ns flush-trace
        #close $namfile
        close $tf
        #exec nam out.nam &
        exit 0
}

proc sendpacket1 {} {
        global ns src1 InterArrivalTime1 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime1 value]] "sendpacket1"
        set bytes [expr round ([$pktSize value])]
```

```
        $src1 send $bytes
}

proc sendpacket2 {} {
        global ns src2 InterArrivalTime2 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime2 value]] "sendpacket2"
        set bytes [expr round ([$pktSize value])]
        $src2 send $bytes
}

proc sendpacket3 {} {
        global ns src3 InterArrivalTime3 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime3 value]] "sendpacket3"
        set bytes [expr round ([$pktSize value])]
        $src3 send $bytes
}

$ns at 0.001 "sendpacket1"
$ns at 0.001 "sendpacket2"
$ns at 0.001 "sendpacket3"
$ns at 50.0 "finish"
$ns run
```

ก.3 การทดลองที่ 4.1.2.1.2 เพื่อแสดงประสิทธิภาพของเครือข่ายเมื่อมีการใช้ HTB กำหนดค่ากำหนดค่าอัตราการส่งแพคเก็ตให้มีค่ามากกว่าค่ารับประกันอัตราการส่งข้อมูลขั้นต่ำ

ก.3.1 ไฟล์ htb3.tcl

```
set ns [new Simulator]

#Open the Trace files
set tf [open out.tr w]
$ns trace-all $tf

#Open the NAM trace file
#set namfile [open out.nam w]
#$ns namtrace-all $namfile

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns duplex-link $n0 $n3 100Mb 100ms DropTail #bit
$ns duplex-link $n1 $n3 100Mb 100ms DropTail
$ns duplex-link $n2 $n3 100Mb 100ms DropTail
$ns duplex-link $n3 $n4 100Mb 0ms DropTail
$ns duplex-link $n4 $n5 1.2Mb 100ms DropTail

$ns queue-limit $n0 $n3 10000000
$ns queue-limit $n1 $n3 10000000
$ns queue-limit $n2 $n3 10000000
$ns queue-limit $n3 $n4 10000000
$ns queue-limit $n4 $n5 10000000

# generate random interarrival times and packet sizes
set InterArrivalTime1 [new RandomVariable/Exponential]
$InterArrivalTime1 set avg_ [expr 1/250.0]
set InterArrivalTime2 [new RandomVariable/Exponential]
$InterArrivalTime2 set avg_ [expr 1/250.0]
set InterArrivalTime3 [new RandomVariable/Exponential]
$InterArrivalTime3 set avg_ [expr 1/250.0]
set pktSize [new RandomVariable/Exponential]
$pktSize set avg_ 1000
```

```
set src1 [new Agent/UDP]
$src1 set packetSize_ 10000 #byte
$ns attach-agent $n0 $src1

set src2 [new Agent/UDP]
$src2 set packetSize_ 10000
$ns attach-agent $n1 $src2

set src3 [new Agent/UDP]
$src3 set packetSize_ 10000
$ns attach-agent $n2 $src3

set null [new Agent/Null]
$ns attach-agent $n5 $null

set htb [new HTB]

$htb set shape_by_source 1

$htb set root_CR 150k # 1Mbps
$htb set root_AR 150k # byte

$htb set exnode1_CR 150k
$htb set exnode1_AR 75k # 0.6Mbps
$htb set exnode1_bucket 150000
$htb set exnode1_qlen 10000000

$htb set exnode2_CR 150k
$htb set exnode2_AR 50k # 0.4Mbps
$htb set exnode2_bucket 150000
$htb set exnode2_qlen 10000000

$htb set exnode3_CR 150k
$htb set exnode3_AR 25k # 0.2Mbps
$htb set exnode3_bucket 150000
$htb set exnode3_qlen 10000000

$ns attach-shape $htb $n3 $n4
$ns connect $src1 $null
$ns connect $src2 $null
$ns connect $src3 $null

proc finish {} {
        global ns tf namfile
        $ns flush-trace
        #close $namfile
        close $tf
        #exec nam out.nam &
        exit 0
}

proc sendpacket1 {} {
        global ns src1 InterArrivalTime1 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime1 value]] "sendpacket1"
        set bytes [expr round ([$pktSize value])]
        $src1 send $bytes
}

proc sendpacket2 {} {
        global ns src2 InterArrivalTime2 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime2 value]] "sendpacket2"
        set bytes [expr round ([$pktSize value])]
        $src2 send $bytes
}

proc sendpacket3 {} {
        global ns src3 InterArrivalTime3 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime3 value]] "sendpacket3"
```

```
        set bytes [expr round ([$pktSize value])]
        $src3 send $bytes
}

$ns at 0.001 "sendpacket1"
$ns at 0.001 "sendpacket2"
$ns at 0.001 "sendpacket3"
$ns at 50.0 "finish"
$ns run
```

ก.4 การทดลองที่ 4.1.2.2.1 เพื่อกำหนดค่าอัตราการส่งข้อมูลสูงสุดที่คลาสสามารถส่งได้ให้และค่ารับประกันอัตราการส่งข้อมูลขั้นต่ำ ให้เกิดเอาต์พุตทราฟฟิคที่ออกจากคลาสโดยค่าอัตราการส่งรวมของคลาสที่สามารถส่งได้จริงมีค่าเท่ากับแบนด์วิดท์ที่มีอยู่จริง

ก.4.1 ไฟล์ htb4.tcl

```
set ns [new Simulator]

#Open the Trace files
set tf [open out.tr w]
$ns trace-all $tf

#Open the NAM trace file
#set namfile [open out.nam w]
#$ns namtrace-all $namfile

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns duplex-link $n0 $n3 100Mb 100ms DropTail #bit
$ns duplex-link $n1 $n3 100Mb 100ms DropTail
$ns duplex-link $n2 $n3 100Mb 100ms DropTail
$ns duplex-link $n3 $n4 100Mb 0ms DropTail
$ns duplex-link $n4 $n5 1.2Mb 100ms DropTail

$ns queue-limit $n0 $n3 10000000
$ns queue-limit $n1 $n3 10000000
$ns queue-limit $n2 $n3 10000000
$ns queue-limit $n3 $n4 10000000
$ns queue-limit $n4 $n5 10000000

# generate random interarrival times and packet sizes
set InterArrivalTime1 [new RandomVariable/Exponential]
$InterArrivalTime1 set avg_ [expr 1/250.0]
set InterArrivalTime2 [new RandomVariable/Exponential]
$InterArrivalTime2 set avg_ [expr 1/250.0]
set InterArrivalTime3 [new RandomVariable/Exponential]
$InterArrivalTime3 set avg_ [expr 1/250.0]
set pktSize [new RandomVariable/Exponential]
$pktSize set avg_ 1000

set src1 [new Agent/UDP]
$src1 set packetSize_ 10000 #byte
$ns attach-agent $n0 $src1

set src2 [new Agent/UDP]
$src2 set packetSize_ 10000
$ns attach-agent $n1 $src2

set src3 [new Agent/UDP]
$src3 set packetSize_ 10000
```

```
$ns attach-agent $n2 $src3

set null [new Agent/Null]
$ns attach-agent $n5 $null

set htb [new HTB]

$htb set shape_by_source 1

$htb set root_CR 150k # 1Mbps
$htb set root_AR 150k # byte

$htb set exnode1_CR 150k
$htb set exnode1_AR 75k # 0.6Mbps
$htb set exnode1_bucket 150000
$htb set exnode1_qlen 10000000

$htb set exnode2_CR 150k
$htb set exnode2_AR 50k # 0.4Mbps
$htb set exnode2_bucket 150000
$htb set exnode2_qlen 10000000

$htb set exnode3_CR 150k
$htb set exnode3_AR 25k # 0.2Mbps
$htb set exnode3_bucket 150000
$htb set exnode3_qlen 10000000

$ns attach-shape $htb $n3 $n4
$ns connect $src1 $null
$ns connect $src2 $null
$ns connect $src3 $null

proc finish {} {
        global ns tf namfile
        $ns flush-trace
        #close $namfile
        close $tf
        #exec nam out.nam &
        exit 0
}

proc sendpacket1 {} {
        global ns src1 InterArrivalTime1 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime1 value]] "sendpacket1"
        set bytes [expr round ([$pktSize value])]
        $src1 send $bytes
}

proc sendpacket2 {} {
        global ns src2 InterArrivalTime2 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime2 value]] "sendpacket2"
        set bytes [expr round ([$pktSize value])]
        $src2 send $bytes
}

proc sendpacket3 {} {
        global ns src3 InterArrivalTime3 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime3 value]] "sendpacket3"
        set bytes [expr round ([$pktSize value])]
        $src3 send $bytes
}

$ns at 0.001 "sendpacket1"
$ns at 0.001 "sendpacket2"
$ns at 0.001 "sendpacket3"
$ns at 50.0 "finish"

$ns run
```

ก.5 การทดลองที่ 4.1.2.2.2 เพื่อกำหนดค่าอัตราการส่งข้อมูลสูงสุดที่คลาสสามารถส่งได้ให้และค่ารับประกันอัตราการส่งข้อมูลขั้นต่ำ ให้เกิดเอาต์พุตทราฟฟิคที่ออกจากคลาสโดยค่าอัตราการส่งรวมของคลาสที่สามารถส่งได้จริงมีค่ามากกว่าแบนด์วิดท์ที่มีอยู่จริง

ก.5.1 ไฟล์ htb5.tcl

```
set ns [new Simulator]

#Open the Trace files
set tf [open out.tr w]
$ns trace-all $tf

#Open the NAM trace file
#set namfile [open out.nam w]
#$ns namtrace-all $namfile

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns duplex-link $n0 $n3 100Mb 100ms DropTail #bit
$ns duplex-link $n1 $n3 100Mb 100ms DropTail
$ns duplex-link $n2 $n3 100Mb 100ms DropTail
$ns duplex-link $n3 $n4 100Mb 0ms DropTail
$ns duplex-link $n4 $n5 1.2Mb 100ms DropTail

$ns queue-limit $n0 $n3 10000000
$ns queue-limit $n1 $n3 10000000
$ns queue-limit $n2 $n3 10000000
$ns queue-limit $n3 $n4 10000000
$ns queue-limit $n4 $n5 10000000

# generate random interarrival times and packet sizes
set InterArrivalTime1 [new RandomVariable/Exponential]
$InterArrivalTime1 set avg_ [expr 1/250.0]
set InterArrivalTime2 [new RandomVariable/Exponential]
$InterArrivalTime2 set avg_ [expr 1/250.0]
set InterArrivalTime3 [new RandomVariable/Exponential]
$InterArrivalTime3 set avg_ [expr 1/250.0]
set pktSize [new RandomVariable/Exponential]
$pktSize set avg_ 1000

set src1 [new Agent/UDP]
$src1 set packetSize_ 10000 #byte
$ns attach-agent $n0 $src1

set src2 [new Agent/UDP]
$src2 set packetSize_ 10000
$ns attach-agent $n1 $src2

set src3 [new Agent/UDP]
$src3 set packetSize_ 10000
$ns attach-agent $n2 $src3

set null [new Agent/Null]
$ns attach-agent $n5 $null

set htb [new HTB]

$htb set shape_by_source 1

$htb set root_CR 1500k # 12Mbps
$htb set root_AR 1500k # byte

$htb set exnode1_CR 1500k
```

```
$htb set exnode1_AR 750k # 6Mbps
$htb set exnode1_bucket 1500000
$htb set exnode1_qlen 10000000

$htb set exnode2_CR 1500k
$htb set exnode2_AR 500k # 4Mbps
$htb set exnode2_bucket 1500000
$htb set exnode2_qlen 10000000

$htb set exnode3_CR 1500k
$htb set exnode3_AR 250k # 2Mbps
$htb set exnode3_bucket 1500000
$htb set exnode3_qlen 10000000

$ns attach-shape $htb $n3 $n4
$ns connect $src1 $null
$ns connect $src2 $null
$ns connect $src3 $null

proc finish {} {
        global ns tf namfile
        $ns flush-trace
        #close $namfile
        close $tf
        #exec nam out.nam &
        exit 0
}

proc sendpacket1 {} {
        global ns src1 InterArrivalTime1 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime1 value]] "sendpacket1"
        set bytes [expr round ([$pktSize value])]
        $src1 send $bytes
}

proc sendpacket2 {} {
        global ns src2 InterArrivalTime2 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime2 value]] "sendpacket2"
        set bytes [expr round ([$pktSize value])]
        $src2 send $bytes
}

proc sendpacket3 {} {
        global ns src3 InterArrivalTime3 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime3 value]] "sendpacket3"
        set bytes [expr round ([$pktSize value])]
        $src3 send $bytes
}

$ns at 0.001 "sendpacket1"
$ns at 0.001 "sendpacket2"
$ns at 0.001 "sendpacket3"
$ns at 50.0 "finish"

$ns run
```

ก.6 การทดลองที่ 4.1.2.2.3 เพื่อกำหนดค่าอัตราการส่งข้อมูลสูงสุดที่คลาสสามารถส่งได้ให้และค่ารับประกันอัตราการส่งข้อมูลขั้นต่ำ ให้เกิดเอาต์พุตทราฟฟิคที่ออกจากคลาสโดยค่าอัตราการส่งรวมของคลาสที่สามารถส่งได้จริงมีค่าน้อยกว่าแบนด์วิดท์ที่มีอยู่จริง

ก.6.1 ไฟล์ htb6.tcl

```
set ns [new Simulator]

#Open the Trace files
set tf [open out.tr w]
$ns trace-all $tf

#Open the NAM trace file
#set namfile [open out.nam w]
#$ns namtrace-all $namfile

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns duplex-link $n0 $n3 100Mb 100ms DropTail #bit
$ns duplex-link $n1 $n3 100Mb 100ms DropTail
$ns duplex-link $n2 $n3 100Mb 100ms DropTail
$ns duplex-link $n3 $n4 100Mb 0ms DropTail
$ns duplex-link $n4 $n5 1.2Mb 100ms DropTail

$ns queue-limit $n0 $n3 10000000
$ns queue-limit $n1 $n3 10000000
$ns queue-limit $n2 $n3 10000000
$ns queue-limit $n3 $n4 10000000
$ns queue-limit $n4 $n5 10000000

# generate random interarrival times and packet sizes
set InterArrivalTime1 [new RandomVariable/Exponential]
$InterArrivalTime1 set avg_ [expr 1/250.0]
set InterArrivalTime2 [new RandomVariable/Exponential]
$InterArrivalTime2 set avg_ [expr 1/250.0]
set InterArrivalTime3 [new RandomVariable/Exponential]
$InterArrivalTime3 set avg_ [expr 1/250.0]
set pktSize [new RandomVariable/Exponential]
$pktSize set avg_ 1000

set src1 [new Agent/UDP]
$src1 set packetSize_ 10000 #byte
$ns attach-agent $n0 $src1

set src2 [new Agent/UDP]
$src2 set packetSize_ 10000
$ns attach-agent $n1 $src2

set src3 [new Agent/UDP]
$src3 set packetSize_ 10000
$ns attach-agent $n2 $src3

set null [new Agent/Null]
$ns attach-agent $n5 $null

set htb [new HTB]

$htb set shape_by_source 1

$htb set root_CR 150k # 1Mbps
$htb set root_AR 150k # byte

$htb set exnode1_CR 62.5k
$htb set exnode1_AR 37.5k # 0.3Mbps
$htb set exnode1_bucket 62500
$htb set exnode1_qlen 10000000

$htb set exnode2_CR 62.5k
$htb set exnode2_AR 25k # 0.2Mbps
$htb set exnode2_bucket 62500
$htb set exnode2_qlen 10000000
```

```
$htb set exnode3_CR 62.5k
$htb set exnode3_AR 12.5k # 0.1Mbps
$htb set exnode3_bucket 62500
$htb set exnode3_qlen 10000000

$ns attach-shape $htb $n3 $n4
$ns connect $src1 $null
$ns connect $src2 $null
$ns connect $src3 $null

proc finish {} {
        global ns tf namfile
        $ns flush-trace
        #close $namfile
        close $tf
        #exec nam out.nam &
        exit 0
}

proc sendpacket1 {} {
        global ns src1 InterArrivalTime1 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime1 value]] "sendpacket1"
        set bytes [expr round ([$pktSize value])]
        $src1 send $bytes
}

proc sendpacket2 {} {
        global ns src2 InterArrivalTime2 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime2 value]] "sendpacket2"
        set bytes [expr round ([$pktSize value])]
        $src2 send $bytes
}

proc sendpacket3 {} {
        global ns src3 InterArrivalTime3 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime3 value]] "sendpacket3"
        set bytes [expr round ([$pktSize value])]
        $src3 send $bytes
}

$ns at 0.001 "sendpacket1"
$ns at 0.001 "sendpacket2"
$ns at 0.001 "sendpacket3"
$ns at 50.0 "finish"
$ns run
```

ก.7 การทดลองที่ 4.1.3.1 การทดลองแสดงประสิทธิภาพโปรโตคอล TCP/IP เมื่อไม่มีการจำกัดปริมาณทราฟฟิคด้วยกลไก HTB

ก.7.1 ไฟล์ htb7.tcl

```
set ns [new Simulator]

#Open the Trace files
set tf [open out.tr w]
$ns trace-all $tf

#Open the NAM trace file
#set namfile [open out.nam w]
#$ns namtrace-all $namfile

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
```

```
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns duplex-link $n0 $n3 100Mb 100ms DropTail #bit
$ns duplex-link $n1 $n3 100Mb 100ms DropTail
$ns duplex-link $n2 $n3 100Mb 100ms DropTail
$ns duplex-link $n3 $n4 100Mb 0ms DropTail
$ns duplex-link $n4 $n5 1.2Mb 100ms DropTail

$ns queue-limit $n0 $n3 10000000
$ns queue-limit $n1 $n3 10000000
$ns queue-limit $n2 $n3 10000000
$ns queue-limit $n3 $n4 10000000
$ns queue-limit $n4 $n5 10000000

set InterArrivalTime3 [new RandomVariable/Exponential]
$InterArrivalTime3 set avg_ [expr 1/150.0]
set pktSize [new RandomVariable/Exponential]
$pktSize set avg_ 1000

set src1 [new Agent/TCP]
$src1 set packetSize_ 960 #byte
$src1 set window_ 70
$src1 set_pkttype 27
$ns attach-agent $n0 $src1

set src2 [new Agent/TCP]
$src2 set packetSize_ 960
$src2 set window_ 70
$src2 set_pkttype 31
$ns attach-agent $n1 $src2

set src3 [new Agent/UDP]
$src3 set packetSize_ 10000
$ns attach-agent $n2 $src3

set ftp1 [new Application/FTP]
$ftp1 attach-agent $src1

set ftp2 [new Application/FTP]
$ftp2 attach-agent $src2

set sink1 [new Agent/TCPSink]
$ns attach-agent $n5 $sink1

set sink2 [new Agent/TCPSink]
$ns attach-agent $n5 $sink2

set null [new Agent/Null]
$ns attach-agent $n5 $null

$ns connect $src1 $sink1
$ns connect $src2 $sink2
$ns connect $src3 $null

proc finish {} {
        global ns tf namfile
        $ns flush-trace
        #close $namfile
        close $tf
        #exec nam out.nam &
        exit 0
}

proc sendpacket3 {} {
        global ns src3 InterArrivalTime3 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime3 value]] "sendpacket3"
        set bytes [expr round ([$pktSize value])]
        $src3 send $bytes
```

```
}

$ns at 0.001 "$ftp1 start"
$ns at 0.001 "$ftp2 start"
$ns at 0.001 "sendpacket3"
$ns at 50.0 "finish"
$ns run
```

ก.8 การทดลองที่ 4.1.3.2.1 การทดลองแสดงประสิทธิภาพโปรโตคอล TCP/IP เมื่อไม่มีการกำหนดอัตราการส่งข้อมูลสูงสุดของคลาส UDP

ก.8.1 ไฟล์ htb8.tcl

```
set ns [new Simulator]

#Open the Trace files
set tf [open out.tr w]
$ns trace-all $tf

#Open the NAM trace file
#set namfile [open out.nam w]
#$ns namtrace-all $namfile

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns duplex-link $n0 $n3 100Mb 100ms DropTail #bit
$ns duplex-link $n1 $n3 100Mb 100ms DropTail
$ns duplex-link $n2 $n3 100Mb 100ms DropTail
$ns duplex-link $n3 $n4 100Mb 0ms DropTail
$ns duplex-link $n4 $n5 1.2Mb 100ms DropTail

$ns queue-limit $n0 $n3 10000000
$ns queue-limit $n1 $n3 10000000
$ns queue-limit $n2 $n3 10000000
$ns queue-limit $n3 $n4 10000000
$ns queue-limit $n4 $n5 10000000

set InterArrivalTime3 [new RandomVariable/Exponential]
$InterArrivalTime3 set avg_ [expr 1/150.0]
set pktSize [new RandomVariable/Exponential]
$pktSize set avg_ 1000

set src1 [new Agent/TCP]
$src1 set packetSize_ 960 #byte
$src1 set window_ 70
$src1 set_pkttype 27
$ns attach-agent $n0 $src1

set src2 [new Agent/TCP]
$src2 set packetSize_ 960
$src2 set window_ 70
$src2 set_pkttype 31
$ns attach-agent $n1 $src2

set src3 [new Agent/UDP]
$src3 set packetSize_ 10000
$ns attach-agent $n2 $src3

set ftp1 [new Application/FTP]
$ftp1 attach-agent $src1

set ftp2 [new Application/FTP]
```

```
$ftp2 attach-agent $src2

set sink1 [new Agent/TCPSink]
$ns attach-agent $n5 $sink1

set sink2 [new Agent/TCPSink]
$ns attach-agent $n5 $sink2

set null [new Agent/Null]
$ns attach-agent $n5 $null

set htb [new HTB]

$htb set shape_by_source 0

$htb set root_CR 150k # 1.2Mbps
$htb set root_AR 150k # byte

$htb set exnode1_CR 150k
$htb set exnode1_AR 125k # 1Mbps
$htb set exnode1_bucket 150000
$htb set exnode1_qlen 10000000

$htb set exnode2_CR 150k
$htb set exnode2_AR 25k # 0.2Mbps
$htb set exnode2_bucket 150000
$htb set exnode2_qlen 10000000

$htb set exnode3_CR 0k
$htb set exnode3_AR 0k
$htb set exnode3_bucket 0
$htb set exnode3_qlen 0

$htb set ftp_CR 125k
$htb set ftp_AR 100k # 0.8Mbps
$htb set ftp_bucket 125000
$htb set ftp_qlen 10000000

$htb set http_CR 125k
$htb set http_AR 25k # 0.2Mbps
$htb set http_bucket 125000
$htb set http_qlen 10000000

$ns attach-shape $htb $n3 $n4

$ns connect $src1 $sink1
$ns connect $src2 $sink2
$ns connect $src3 $null

proc finish {} {
        global ns tf namfile
        $ns flush-trace
        #close $namfile
        close $tf
        #exec nam out.nam &
        exit 0
}

proc sendpacket3 {} {
        global ns src3 InterArrivalTime3 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime3 value]] "sendpacket3"
        set bytes [expr round ([$pktSize value])]
        $src3 send $bytes
}

$ns at 0.001 "$ftp1 start"
$ns at 0.001 "$ftp2 start"
$ns at 0.001 "sendpacket3"
$ns at 50.0 "finish"
$ns run
```

ก.9 การทดลองที่ 4.1.3.2.2 การทดลองแสดงประสิทธิภาพโปรโตคอล TCP/IP เมื่อมีการกำหนดอัตราการส่งข้อมูลสูงสุดของคลาส UDP

ก.9.1 ไฟล์ htb9.tcl

```
set ns [new Simulator]

#Open the Trace files
set tf [open out.tr w]
$ns trace-all $tf

#Open the NAM trace file
#set namfile [open out.nam w]
#$ns namtrace-all $namfile

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$ns duplex-link $n0 $n3 100Mb 100ms DropTail #bit
$ns duplex-link $n1 $n3 100Mb 100ms DropTail
$ns duplex-link $n2 $n3 100Mb 100ms DropTail
$ns duplex-link $n3 $n4 100Mb 0ms DropTail
$ns duplex-link $n4 $n5 1.2Mb 100ms DropTail

$ns queue-limit $n0 $n3 10000000
$ns queue-limit $n1 $n3 10000000
$ns queue-limit $n2 $n3 10000000
$ns queue-limit $n3 $n4 10000000
$ns queue-limit $n4 $n5 10000000

set InterArrivalTime3 [new RandomVariable/Exponential]
$InterArrivalTime3 set avg_ [expr 1/150.0]
set pktSize [new RandomVariable/Exponential]
$pktSize set avg_ 1000

set src1 [new Agent/TCP]
$src1 set packetSize_ 960 #byte
$src1 set window_ 70
$src1 set_pkttype 27
$ns attach-agent $n0 $src1

set src2 [new Agent/TCP]
$src2 set packetSize_ 960
$src2 set window_ 70
$src2 set_pkttype 31
$ns attach-agent $n1 $src2

set src3 [new Agent/UDP]
$src3 set packetSize_ 10000
$ns attach-agent $n2 $src3

set ftp1 [new Application/FTP]
$ftp1 attach-agent $src1

set ftp2 [new Application/FTP]
$ftp2 attach-agent $src2

set sink1 [new Agent/TCPSink]
$ns attach-agent $n5 $sink1

set sink2 [new Agent/TCPSink]
$ns attach-agent $n5 $sink2

set null [new Agent/Null]
$ns attach-agent $n5 $null
```

```
set htb [new HTB]

$htb set shape_by_source 0

$htb set root_CR 150k # 1.2Mbps
$htb set root_AR 150k # byte

$htb set exnode1_CR 150k
$htb set exnode1_AR 125k # 1Mbps
$htb set exnode1_bucket 150000
$htb set exnode1_qlen 10000000

$htb set exnode2_CR 50k
$htb set exnode2_AR 25k # 0.2Mbps
$htb set exnode2_bucket 150000
$htb set exnode2_qlen 10000000

$htb set exnode3_CR 0k
$htb set exnode3_AR 0k
$htb set exnode3_bucket 0
$htb set exnode3_qlen 0

$htb set ftp_CR 125k
$htb set ftp_AR 100k # 0.8Mbps
$htb set ftp_bucket 125000
$htb set ftp_qlen 10000000

$htb set http_CR 125k
$htb set http_AR 25k # 0.2Mbps
$htb set http_bucket 125000
$htb set http_qlen 10000000

$ns attach-shape $htb $n3 $n4

$ns connect $src1 $sink1
$ns connect $src2 $sink2
$ns connect $src3 $null

proc finish {} {
        global ns tf namfile
        $ns flush-trace
        #close $namfile
        close $tf
        #exec nam out.nam &
        exit 0
}

proc sendpacket3 {} {
        global ns src3 InterArrivalTime3 pktSize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime3 value]] "sendpacket3"
        set bytes [expr round ([$pktSize value])]
        $src3 send $bytes
}

$ns at 0.001 "$ftp1 start"
$ns at 0.001 "$ftp2 start"
$ns at 0.001 "sendpacket3"
$ns at 50.0 "finish"
$ns run
```

ประวัติผู้เขียน

ชื่อผู้เขียน	นายภควัต เชี่ยวเจริญวงศ์
วัน เดือน ปีเกิด	25 ตุลาคม 2527
สถานที่เกิด	กรุงเทพมหานคร
สถานที่การศึกษาปัจจุบัน	คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อผู้เขียน	นางสาวภัทรจิตร์ เปี่ยมด้วยธรรม
วัน เดือน ปีเกิด	5 มีนาคม 2528
สถานที่เกิด	กรุงเทพมหานคร
สถานที่การศึกษาปัจจุบัน	คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อผู้เขียน	นายอารยะ จิระภิญโญ
วัน เดือน ปีเกิด	22 สิงหาคม 2527
สถานที่เกิด	กรุงเทพมหานคร
สถานที่การศึกษาปัจจุบัน	คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง