การประมวลล่วงหน้าเพื่อหาเส้นทางที่รับประกันคุณภาพของการให้บริการ

QoS PATH PRECOMPUTING ALGORITHM

ฟองแก้ว แก้วยองยาง
PHONGKAEW KAEWYONGPHANG

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2545

ISBN 974-648-882-1

การประมวลล่วงหน้าเพื่อหาเส้นทางที่รับประกันคุณภาพของการให้บริการ

QoS PATH PRECOMPUTING ALGORITHM

ฟองแก้ว แก้วยองผาง

PHONGKAEW KAEWYONGPHANG

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2545

ISBN 974-648-882-1

QoS PATH PRECOMPUTING ALGORITHM

PHONGKAEW KAEWYONGPHANG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S OF TECHNOLOGY LADKRABANG

2002

ISBN 974-648-882-1

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การประมวลล่วงหน้าเพื่อหาเส้นทางที่รับประกันคุณภาพของการให้บริการ QoS PATH PRECOMPUTING ALGORITHM

ชื่อนักศึกษา นางสาวฟองแก้ว แก้วยองผาง

รหัสประจำตัว 40061066

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมไฟฟ้า

อาจารย์ผู้ควบคุมวิทยานิพนธ์ รศ.ดร.รัตติกร วรากูลศิริพันธุ์

อาจารย์ผู้ควบคุมวิทยานิพนธ์ร่วม ดร.โชติพัชร์ ภรณวลัย

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
รศ.ดร.มนัส	สังวรศิลป์	
รศ.บรรจง	ปิยธำรง	
ดร.สุพันธุ์	ตั้งจิตกุศลมั่น	
รศ.ดร.เอื้อน	ปิ่นเงิน	
รศ.ดร.รัตติกร	วรากูลศิริพันธุ์	

วัน/เดือน/ปี ที่สอบ 23 พฤษภาคม 2545 เวลา 10.30-12.30 น.

สถานที่สอบ ณ อาคาร 12 ชั้น ชั้น 4 (ห้อง E12-402)

บัณฑิตวิทยาลัยรับรองแล้ว

(รศ.ดร.บุญวัฒน์ อัติชู)

คณบดีบัณฑิตวิทยาลัย

วันที่ 19 **เดือน** มิถุนายน **พ.ศ.** 2545

หัวข้อวิทยานิพนธ์	การประมวลล่วงหน้าเพื่อหาเส้นทางที่รับประกันคุณภาพของการให้บริการ
นักศึกษา	นางสาวฟองแก้ว แก้วยองผาง
รหัสประจำตัว	40061066
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้า
พ.ศ.	2545
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร.รัตติกร วรากูลศิริพันธุ์
อาจารย์ผู้ควบคุมวิทยานิพนธ์ร่วม	ผศ.ดร.โชติพัชร์ ภรณวลัย

บทคัดย่อ

อัลกอริทึมการเลือกเส้นทางแบบเดิมนั้นถูกออกแบบมาสำหรับการส่งผ่านข้อมูลที่เป็นแบบตัวอักษร ไม่ได้ออกแบบมาเพื่อการส่งผ่านข้อมูลที่เป็นแอพพลิเคชั่นแบบสื่อผสม เนื่องจากการส่งผ่านข้อมูลที่เป็นแอพพลิเคชั่นแบบสื่อผสมนั้นจำเป็นที่จะต้องมีการรับประกันคุณภาพของการบริการ เช่น การรับประกันคุณภาพของการให้บริการด้าน แบนด์วิดธ์คอขวด เอนทูเอนดีเลย์ เอนทูเอนจิตเตอร์ดีเลย์ และอัตราการสูญเสีย เป็นต้น

ในวิทยานิพนธ์นี้ เราได้ทำการศึกษาอัลกอริทึมการเลือกเส้นทางที่มีการรับประกันคุณภาพของการให้บริการ ที่เรียกว่า "k-WSP_{EP}" สำหรับเส้นทางที่คำนวณไว้ล่วงหน้า อัลกอริทึมนี้ปรับปรุงมาจากอัลกอริทึมเบลแมนฟอร์ด โดยการใช้ระเบียบบริการการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม มากำหนด ขอบเขตบนของเอนทูเอนดีเลย์ เอนทูเอนจิตเตอร์ดีเลย์ และ ไม่มีอัตราการสูญเสียเกิดขึ้น ดังนั้นอัลกอริทึมนี้จึงถูกออกแบบมาเพื่อสนับสนุนการรับประกันคุณภาพของการให้บริการได้อย่างมีประสิทธิภาพ

สำหรับขนาดของตารางเส้นทางของอัลกอริทึมที่มีการคำนวณไว้ล่วงหน้ามีการรับประกันคุณภาพของการให้บริการ จะมีขนาดใหญ่มากเนื่องจากต้องเก็บทุกเส้นทาง ดังนั้นในวิทยานิพนธ์มีการประเมินอัลกอริทึมการคำนวณไว้ล่วงหน้าของเส้นทางที่รับประกันคุณภาพของการให้บริการ ผลการทดลองปรากฏว่าเราสามารถลดขนาดของตารางเส้นทางให้เล็กลงมาก เนื่องจากเส้นทางหลาย ๆ เส้นทางที่ตรงตามเงื่อนไขที่ได้กำหนดไว้นั้นเป็นเส้นทางที่ไม่จำเป็นจึงสามารถตัดทิ้งได้ ขณะที่ตารางเส้นทางยังคงมีประสิทธิภาพเนื่องจากมีเส้นทางที่ดีกว่าเก็บไว้ในตารางเรียบร้อยแล้ว

Thesis Title	QoS Path Precomputing Algorithm
Student	Miss. Phongkaew Kaewyongphang
Student ID.	40061066
Degree	Master of Engineering
Programme	Electrical Engineering
Year	2002
Thesis Advisor	Assoc. Prof. Dr. Ruttikorn Varakulsiripunth
Thesis Assoc. Advisor	Asst. Prof. Dr. Chotipat Pornavalai

ABSTRACT

Traditional routing algorithms were designed for data transmission applications but were not for multimedia applications. Because these applications need to support quality of service (QoS) such as bandwidth, end-to-end delay, end-to-end delay-jitter and loss rate.

In this thesis, we study a QoS routing algorithm called "k-WSP_{EP}" for precomputed paths. It is a modified version of Bellman- Ford algorithm which employed weighted fair queueing service discipline. It can be used to provide tight upper bound on end-to-end delay, end-to-end jitter and ensure that no packet loss. The algorithm was design for supporting performance guaranteed service.

For precomputation of QoS paths, the size of the routing table must be very large in order to store all QoS routes. However we have proposed that many paths are likely to be redundant as paths satisfying defined conditions could be neglected. Results from the simulation showed, that the size of routing table can be greatly reduced. Many redundant paths are deleted, while still achieving the optimal performance.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดี โดยได้รับการช่วยเหลือ และสนับสนุนจากบุคคลหลาย ๆ ฝ่าย ซึ่งผู้เขียนขอขอบพระคุณทุกท่าน ดังต่อไปนี้

คุณพ่อและคุณแม่ ผู้ซึ่งคอยอบรมสั่งสอน ให้การสนับสนุนในเรื่องการศึกษา และเป็นกำลังใจให้อย่างเสมอมา

รศ.ดร.รัตติกร วรากูลศิริพันธุ์ อาจารย์ที่ปรึกษา และ ผศ.ดร.โชติพัชร์ ภรณวลัย ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ร่วม ที่กรุณาให้คำปรึกษา ชี้แนะแนวทางและให้ข้อคิดเห็นที่เป็นประโยชน์ต่อการทำวิจัย รวมทั้งตรวจทาน แก้ไขวิทยานิพนธ์ฉบับนี้จนสำเร็จลุล่วงไปด้วยดี

อาจารย์ทุกท่าน ที่กรุณาให้การอบรมสั่งสอนและให้ความรู้

อาจารย์สุพจน์ จันทร์วิพัฒน์ คณะวิทยาลัยเทคโนโลยีอุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ ที่คอยให้กำลังใจ และช่วยเหลือในการเตรียมต้นฉบับ ตรวจทานแก้ไขข้อผิดพลาดในวิทยานิพนธ์ฉบับนี้ ให้สำเร็จสมบูรณ์ยิ่งขึ้น

ห้องปฏิบัติการเครือข่ายการสื่อสาร (Communication Network Lab.) สำนักวิจัยการสื่อสารและเทคโนโลยีสารสนเทศ (ReCCIT : Research Center for Communications and Information Technology) ที่สนับสนุนอุปกรณ์และสถานที่ทำการวิจัย

มูลนิธิเพื่อการศึกษาคอมพิวเตอร์ และการสื่อสาร ที่สนับสนุนงานวิจัย

บัณฑิตวิทยาลัย ที่ได้สนับสนุนทุนการศึกษาในการจัดทำวิทยานิพนธ์ครั้งนี้

และทุกท่านที่ไม่ได้กล่าวถึงในที่นี้ ที่ให้ความช่วยเหลือ และกำลังใจในการทำวิทยานิพนธ์ฉบับนี้จนสำเร็จ

คุณค่าและประโยชน์อันพึงมีจากวิทยานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

ฟองแก้ว แก้วยองผาง

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันนี้ มีการนำเอาระบบคอมพิวเตอร์มาใช้อย่างกว้างขวาง ทั้งภายในมหาวิทยาลัย วงการธุรกิจ และได้ถูกใช้งานแพร่หลายไปยังผู้ใช้ตามบ้านเรือนทั่วไปเพื่อใช้ท่องเที่ยวไปในโลกอินเทอร์เน็ต (Internet) สำหรับข้อมูลที่ส่งผ่านเครือข่ายก็มีความหลากหลายและมีการพัฒนามากยิ่งขึ้น มีทั้งเอกสาร เสียง ภาพนิ่ง ภาพเคลื่อนไหว ซึ่งเป็นแอพพลิเคชั่นแบบสื่อผสม (Multimedia Applications) จะมีใช้ในอนาคตอันใกล้นี้ ดังนั้นจึงต้องมีการพัฒนาเทคโนโลยีทางด้านการส่งผ่านและการคำนวณ (Transmission and Computing Technology) เช่นกันทั้งนี้เพื่อจะได้รองรับการส่งผ่านข้อมูลในเครือข่ายคอมพิวเตอร์ที่เป็นแอพพลิเคชั่นแบบสื่อผสม [1][2]

โดยปกติแล้วแอพพลิเคชั่นแบบสื่อผสม จะมีคุณสมบัติในการจราจรที่แตกต่างกัน และมีความหลากหลายในการต้องการการรับประกันคุณภาพของการให้บริการ (QoS : Quality of Service) เช่น แบนด์วิดธ์ (Bandwidth) เอนทูเอนดีเลย์ (End-to-End Delay) เอนทูเอนจิตเตอร์ดีเลย์ (End-to-End Delay-Jitter) และอัตราการสูญเสีย (Loss rate)

เครือข่ายสื่อสารข้อมูลแบบดั้งเดิมถูกออกแบบเพื่อการบริการที่เป็นแอพพลิเคชั่นแบบตัวอักษร เช่น เอฟทีพี (FTP) เทลเน็ต (Telnet) เป็นต้น ในอนาคตเครือข่ายบริการร่วมแพ็กเกตสวิตช์ชิ่ง ซึ่งประกอบด้วยอินเทอร์เน็ต จะสามารถรองรับการบริการการสื่อสารข้อมูลที่เป็นแอพพลิเคชั่นแบบสื่อผสมซึ่งเป็นแอพพลิเคชั่นที่เป็นแบบเรียลไทม์ (Real-time Communication) และเป็นแอพพลิเคชั่นที่ต้องการ การรับประกัน QoS ดังนั้นการพัฒนาการเลือกเส้นทางที่จะรองรับการส่งผ่านแอพพลิเคชั่นเหล่านี้ ต้องมีการรับประกัน QoS แต่ว่าอัลกอริทึมการเลือกเส้นทางที่มีการรับประกัน QoS หลาย ๆ พารามิเตอร์นั้น มีความซับซ้อนมาก ดังนั้นเพื่อให้อัลกอริทึมนั้นสามารถค้นหาเส้นทางได้ง่ายขึ้น จึงมีการออกแบบอัลกอริทึมโดยมีการประยุกต์ใช้ ระเบียบบริการ (Service Discipline) ใหม่ ๆ ในวิทยานิพนธ์นี้ได้มีการประยุกต์ใช้ ระเบียบการบริการการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม (WFQ : Weighted Fair Queuing) และโปรโตคอลการจองทรัพยากร (RSVP : Resource Reservation Protocal) เป็นต้น

1.2 วัตถุประสงค์ของวิทยานิพนธ์

วิทยานิพนธ์นี้มีวัตถุประสงค์ดังต่อนี้

1. เพื่อพัฒนาอัลกอริทึมเพื่อหาเส้นทางที่มีการรับประกันคุณภาพของการให้บริการ (QoS)
2. เพื่อลดขนาดของตารางเส้นทาง โดยยังคงมีประสิทธิภาพของการให้บริการ QoS

1.3 ทฤษฎีและหลักการที่ใช้ในวิทยานิพนธ์

วิทยานิพนธ์นี้ได้นำเอาหลักการและทฤษฎีที่เกี่ยวข้องมาประยุกต์ใช้ ดังต่อนี้

1. ทฤษฎีเครือข่ายแวกซ์แมน

เครือข่ายแวกซ์แมน คือกราฟที่มีทิศทาง (Directed Graph) โดยที่โหนดแทนเราเตอร์ หรือสวิตซ์โหนดต่าง ๆ ภายในเครือข่ายถูกสร้างขึ้นแบบสุ่ม ตามความน่าจะเป็นของแวกซ์แมน เครือข่ายนี้จะมีคุณลักษณะใกล้เคียงกับเครือข่ายที่มีใช้งานจริงอยู่ในปัจจุบัน

2. เทคนิคการค้นหาเส้นทาง WSP

เทคนิคการค้นหาเส้นทางโดยใช้ WSP เป็นการค้นหาเส้นทางเส้นโดยคำนึงถึงเส้นทางที่มีจำนวนฮอปที่น้อยที่สุด ถ้าหากว่ามีเส้นทางมากกว่าหนึ่งเส้นทางแล้วให้เลือกเส้นทางที่มีค่าแบนด์วิดธ์มากที่สุด และถ้าหากว่ายังมีเส้นทางมากกว่าหนึ่งเส้นทาง ก็จะใช้การสุ่มเพื่อเลือกเส้นทาง เพื่อเก็บไว้ในตารางเส้นทาง

3. เทคนิคการบริหารแพ็กเกตในแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม

ในการออกแบบอัลกอริทึมการเลือกเส้นทางที่มีการรับประกัน QoS ได้มีการนำเอาระเบียบบริการนี้ มากำหนดขอบเขตบนของพารามิเตอร์ต่าง ๆ ของการรับประกันคุณภาพ QoS ในวิทยานิพนธ์นี้ รายละเอียดดังแสดงในบทที่ 4

4. โปรโตคอลสำรองทรัพยากร

การสำรองทรัพยากร [1] เป็นกระบวนการหนึ่ง ที่ทำงานร่วมกับขั้นตอนการควบคุมปริมาณข้อมูลในเครือข่าย รายละเอียดในการขอจัดตั้งวงจรเสมือน จะบอกความต้องการ อัตราเร็วสูงสุดในการส่งข้อมูล ซึ่งระบบจะทำการตรวจสอบ และการจัดการสำรองทรัพยากรไว้สำหรับวงจรเสมือนนั้น ๆ อัตราความเร็วสูงสุดที่ผู้ใช้ ต้องการนี้ จะถูกนำไปพิจารณาในการค้นหาเส้นทางของข้อมูล ที่สามารถตอบสนองตามความต้องการได้

1.4 ขอบเขตของวิทยานิพนธ์

ขอบเขตของวิทยานิพนธ์ มีดังต่อไปนี้

1. วิทยานิพนธ์นี้มุ่งศึกษาการพัฒนา อัลกอริทึมการค้นหาเส้นทางที่มีการรับประกัน QoS โดยการปรับปรุงอัลกอริทึมการค้นหาเส้นทางแบบเบลแมนฟอร์ด

2. ศึกษาการนำเอาเทคนิคการบริหารแพ็กเกตในแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม มาใช้ในการพัฒนาอัลกอริทึมการค้นหาเส้นทางที่มีการรับประกัน QoS แบบเบลแมนฟอร์ด

3. การวิจัยนี้ได้ทำการซิมมูเลชั่น (Simmulation) บนเครื่องคอมพิวเตอร์ส่วนบุคคล Pentium II Processor 333 เมกกะเฮิร์ซ RAM 64 เมกกะไบต์ ซึ่งในการทำซิมมูเลชั่นเครือข่าย และการคำนวณหาเส้นทาง โดยใช้โปรแกรมจำลองทางคณิตศาสตร์ เวอร์ชั่น 5.1 (Mathlab version 5.1) ของบริษัทแมทซอร์ฟ จำกัด (Mathsoft Co., LTD)

4. การซิมมูเลชั่น ได้กำหนดเครือข่ายเป็นเครือข่าย ATM มีขนาดโดยประมาณเท่ากับขนาดของประเทศสหรัฐอเมริกา กล่าวคือมีขนาดประมาณ 4000 x 2400 ตารางกิโลเมตร โหนดทั้งหมดภายในเครือข่ายเชื่อมต่อกัน ด้วยดีกรีน้อยที่สุดเท่ากับ 2 และมีดีกรีเฉลี่ยเท่ากับ 3 เครือข่ายเหล่านี้ถูกสร้างมาแบบสุ่ม ซึ่งมีจำนวนโหนด ภายในเครือข่ายแบบสุ่มนี้ มีจำนวนเท่ากับ 15, 30, 45 และ 60 โหนด

1.5 ขั้นตอนการทำวิทยานิพนธ์

ขั้นตอนการทำวิทยานิพนธ์ มีดังต่อไปนี้

1. กำหนดหัวข้อ เป้าหมาย จุดประสงค์ และขอบเขตการทำวิทยานิพนธ์
2. ศึกษาทฤษฎี และหลักการพื้นฐานที่ใช้ในทำวิทยานิพนธ์
3. ศึกษา และทำการสร้างแบบจำลองเครือข่ายที่ใช้ในการทดลอง
4. ศึกษาพารามิเตอร์ต่าง ๆ ที่เกี่ยวกับการเลือกเส้นทางที่มีการรับประกัน QoS
5. ศึกษาอัลกอริทึมการเลือกเส้นทางที่มีการรับประกัน QoS
6. ทำการซิมมูเลชั่น สรุปผลการซิมมูเลชั่น และเสนอแนวทางในการวิจัยต่อไป
7. จัดทำเอกสารประกอบวิทยานิพนธ์

1.6 โครงสร้างของวิทยานิพนธ์

ในวิทยานิพนธ์ฉบับนี้เป็นการนำเสนอ การประมวลล่วงหน้าเพื่อหาเส้นทางที่มีการรับประกัน QoS โดยรายละเอียดต่าง ๆ ภายในวิทยานิพนธ์ฉบับนี้ได้จัดแบ่งเนื้อหาทั้งหมดออกเป็น 6 บท ซึ่งแต่ละบทจะมีหัวข้อและเนื้อหาดังต่อไปนี้

บทที่ 1 บทนำ

จะกล่าวถึงลักษณะทั่วไปของวิทยานิพนธ์ ตั้งแต่ความเป็นมา และความสำคัญของปัญหา วัตถุประสงค์ ทฤษฎีและหลักการต่าง ๆ ที่นำมาใช้ในงานวิจัย ขอบเขต วิธีการทำ และโครงสร้างของวิทยานิพนธ์

บทที่ 2 ปัญหาการเลือกเส้นทางและงานวิจัยที่เกี่ยวข้อง

อธิบายถึงปัญหาการเลือกเส้นทาง พารามิเตอร์ต่าง ๆ ที่เกี่ยวข้องกับการรับประกัน QoS และงานวิจัยที่เกี่ยวข้องกับการเลือกเส้นทางที่มีและไม่มีการรับประกัน QoS

บทที่ 3 ทฤษฎีการเลือกเส้นทางในระบบเครือข่ายคอมพิวเตอร์

อธิบายถึงทฤษฎีการเลือกเส้นทางในระบบเครือข่ายคอมพิวเตอร์ หลักการการจำแนกการเลือกเส้นทาง และตัวอย่างวิธีการเลือกเส้นทาง

บทที่ 4 อัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS

อธิบายถึงรายละเอียด ลักษณะของอัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS ของเบลแมนฟอร์ด นอกจากนี้ได้อธิบายอัลกอริทึม ประมวลผลล่วงหน้าเพื่อหาเส้นทางที่มีการรับประกัน QoS ซึ่งอัลกอริทึมนี้ปรับปรุงมาจากอัลกอริทึมทางเดินที่สั้นที่สุดของเบลแมนฟอร์ด ถึงแม้ว่าปัญหาการเลือกเส้นทางที่รับประกัน QoS หลาย ๆ พารามิเตอร์ จะเป็นปัญหา *NP* สัมบูรณ์ (Non-deterministic Polynomial) [3][4] แต่ในบทนี้มีการแสดงการใช้ระเบียบบริการการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม ซึ่งจะทำให้สามารถหาเส้นทางที่รับประกัน QoS ทั้งหมด 4 พารามิเตอร์ คือ แบนด์วิดธ์คอขวด เอนทูเอนดีเลย์ เอนทูเอนจิตเตอร์ดีเลย์ และขนาดของบัฟเฟอร์ (ไม่มีการสูญเสีย) ได้ ถ้าหากว่ามีเส้นทางนั้นอยู่ ภายในเครือข่าย

บทที่ 5 การซิมมูเลชั่น และผลการซิมมูเลชั่น

บทที่ 6 บทสรุปและแนวทางในการพัฒนา

อธิบายถึงบทสรุปและแนวทางในการพัฒนาในอนาคต

บทที่ 2
ปัญหาการเลือกเส้นทางและงานวิจัยที่เกี่ยวข้อง

2.1 ปัญหาของการเลือกเส้นทางที่มีการรับประกันคุณภาพของการให้บริการ (QoS)

ปัญหาการหาเส้นทางที่มีการรับประกัน QoS ถ้าหากว่ามีการรับประกัน QoS มากกว่าหนึ่งพารามิเตอร์ขึ้นไป จะทำให้เกิดเป็นปัญหา *NP* สัมบูรณ์ โดยปกติแล้วแอพพลิเคชั่นที่เป็นสื่อผสมต้องการการรับประกัน แบนด์วิดธ์คอขวด, ขอบเขตของเอนทูเอนดีเลย์, ขอบเขตของจิตเตอร์ดีเลย์ และมีการควบคุมอัตราการสูญเสียของข้อมูล ซึ่งพารามิเตอร์เหล่านี้ทำให้การแก้ปัญหาทำได้ยากขึ้น

2.1.1 พารามิเตอร์ ที่เกี่ยวข้องกับการรับประกัน QoS

ในวิทยานิพนธ์นี้ ได้พิจารณาพารามิเตอร์ ต่าง ๆ ของการรับประกัน QoS มีดังต่อไปนี้

2.1.1.1 จำนวนฮอป

จำนวนฮอป ในวิทยานิพนธ์นี้กำหนดให้แต่ละลิงค์ (Link) มีจำนวนฮอปเท่ากับ 1

$h(e) = h(u,v) = 1$ คือจำนวนฮอป ของลิงค์ e ที่เชื่อมระหว่างโหนด u และโหนด v

2.1.1.2 ค่าใช้จ่าย

ฟังก์ชั่นของค่าใช้จ่ายของลิงค์ e เป็นฟังก์ชั่นโหลด (Load) ของลิงค์ แสดงดังสมการที่ (2.1)

$$c(e) = \frac{1}{b(e)} \tag{2.1}$$

โดยที่

$c(e) = c(u,v)$ คือค่าใช้จ่าย ของลิงค์ e ที่เชื่อมระหว่างโหนด u และโหนด v

$b(e) = b(u,v)$ คือค่าแบนด์วิดธ์ที่ใช้ประโยชน์ได้ ของลิงค์ e ที่เชื่อมระหว่างโหนด u และโหนด v

2.1.1.3 ค่าแบนด์วิดธ์

ค่าแบนด์วิดธ์ คือ จำนวนสูงสุดของหน่วยข้อมูลที่สามารถส่งไปในช่องสัญญาณต่อวินาที สื่อในการส่งผ่านข้อมูลเครือข่ายที่จำลองขึ้นในวิทยานิพนธ์นี้ มีแบนด์วิดธ์ เท่ากับ 155.52 เมกกะบิตต่อวินาที (OC-3) พารามิเตอร์ที่เกี่ยวข้องกับการเลือกเส้นทางที่มีการรับประกัน QoS หลาย ๆ พารามิเตอร์ จะสัมพันธ์กับจำนวนของแบนด์วิดธ์ กล่าวคือถ้าหากว่าบนเส้นทางนั้นมีจำนวนแบนด์วิดธ์เพียงพอแล้ว พารามิเตอร์อื่น ๆ ก็จะมีแนวโน้มที่จะถูกรับประกัน QoS ด้วย รายละเอียดจะกล่าวในบทที่ 4

2.1.1.4 ค่าดีเลย์

ค่าดีเลย์ คือค่าเอนทูเอนดีเลย์ เป็นค่าความล่าช้าที่แพ็กเกตเดินทางจากโหนดต้นทางถึงโหนดปลายทาง

2.1.1.5 ค่าจิตเตอร์ดีเลย์

ค่าสัมบูรณ์ (Absolute Value) ของผลต่างกันระหว่างเวลาที่แพ็กเกตสองแพ็กเกต ที่มาถึงลิงค์ e ไล่เลี่ยกัน กับเวลาที่แพ็กเกตทั้งสองออกจากลิงค์ e (2.2)

$$\left| \left(t_{1arrival} - t_{1departure}\right) - \left(t_{2arrival} - t_{2departure}\right) \right| \qquad (2.2)$$

โดยที่ $t_{1arrival}, t_{2arrival}$ คือเวลาที่แพ็กเกตที่ 1 และ 2 มาถึงลิงค์ e ตามลำดับ

$t_{1departure}, t_{2departure}$ คือเวลาที่แพ็กเกตที่ 1 และ 2 ออกจากลิงค์ e ตามลำดับ

2.1.1.6 อัตราการสูญเสียแพ็กเกต

อัตราการสูญเสียแพ็กเกต คืออัตราส่วนระหว่างจำนวนแพ็กเกตที่สูญหายกับจำนวนแพ็กเกตทั้งหมดที่ถูกส่งออกมาจากผู้ส่ง โดยที่จำนวนแพ็กเกตที่สูญหายคำนวณได้จากจำนวนแพ็กเกตทั้งหมดที่ถูกส่งจากผู้ส่งถึงผู้รับที่ระบุไว้ ลบกับ จำนวนแพ็กเกตที่ผู้รับนั้นได้รับจริง

2.1.2 การอธิบายปัญหา

ถ้าเราสมมุติอย่างง่ายว่าเครือข่ายการสื่อสารมีรูปแบบเป็นกราฟที่เชื่อมต่อกันอย่างมีทิศทาง $G = (V, E)$ โดยที่

V แทน เซ็ตของโหนด ซึ่งจะเป็นเราเตอร์ เซิร์ฟเวอร์ หรือ สวิตซ์

E แทนเซ็ตของขอบ หรือลิงค์ ของเครือข่าย

$h(e)=h(u,v)=1$ คือจำนวนฮอปของลิงค์ e ที่เชื่อมระหว่างโหนด u และโหนด v

$c(e)=c(u,v)$ คือค่าใช้จ่ายของลิงค์ e ที่เชื่อมระหว่างโหนด u และโหนด v

$d(e)=d(u,v)$ คือค่าดีเลย์ ของลิงค์ e ที่เชื่อมระหว่างโหนด u และโหนด v

$b(e)=b(u,v)$ คือค่าแบนด์วิดธ์ที่ใช้ประโยชน์ได้ ของลิงค์ e ที่เชื่อมระหว่างโหนด u และโหนด v

$j(e)=j(u,v)$ คือจิตเตอร์ดีเลย์ ของลิงค์ e ที่เชื่อมระหว่างโหนด u และโหนด v

$l(e)=l(u,v)$ คือค่าฟังก์ชั่นการสูญเสียของลิงค์ e ที่เชื่อมระหว่างโหนด u และโหนด v

กำหนดให้โหนดต้นทางคือโหนด s และโหนดปลายทางคือโหนด z โดยที่ $(s,z \in V)$ ปัญหาคือการหาเส้นทาง (Route) หรือ (Path) จากโหนดต้นทาง s ถึงโหนดปลายทาง z , $P(s,z)$ หรือ $P = (s,j,k,...,l,z)$ จะต้องเป็นเส้นทางที่มีคุณสมบัติดังต่อไปนี้

1. เส้นทางที่ใช้ทรัพยากรให้น้อยที่สุดโดยการเลือกเส้นทางที่มีจำนวนฮอปน้อยที่สุด ดังสมการที่ (2.3)

2. เส้นทางที่มีการจัดสรรการจราจร หรือสมดุลย์ของโหลดให้ทั่วทั้งเครือข่ายโดยการเลือกเส้นทางที่มีค่าใช้จ่ายน้อยที่สุด (โหลดน้อยที่สุด) ดังสมการที่ (2.4)

3. เส้นทางที่เป็นไปตามการบังคับความต้องการคุณภาพของการบริการ QoS ดังสมการที่ (2.5), (2.6), (2.7), (2.8)

$$H(P) = \underset{P_i \in P'(s,z)}{MIN} \sum_{e \in P_i} h(e) \qquad (2.3)$$

$$C(P) = \underset{P_i \in P'(s,z)}{MIN} \sum_{e \in P_i} C(e) \qquad (2.4)$$

$$B(P) = \underset{e \in P}{MIN} [b(e)] \quad \text{และ} \quad B(P) \geq \Delta_{bandwidth} \qquad (2.5)$$

$$D(P) = \sum_{e \in P} d(e) \quad \text{และ} \quad D(P) \leq \Delta_{delay} \qquad (2.6)$$

$$J(P) = \sum_{e \in P} j(e) \quad \text{และ} \quad J(P) \leq \Delta_{jitter} \qquad (2.7)$$

$$L'(P) = \prod_{e \in P} l'(e) \quad \text{และ} \quad L'(P) \geq (1 - \Delta_{loss}) \qquad (2.8)$$

โดยที่ $H(P)$, $C(P)$, $B(P)$, $D(P)$, $J(P)$ และ $L(P)$ เป็นจำนวนฮอป ค่าใช้จ่าย แบนด์วิดธ์คอขวด ดีเลย์ จิตเตอร์ และความสูญเสียบนเส้นทาง $P(s,z)$ ตามลำดับ $P'(s,z)$ เป็นเซ็ตของเส้นทางที่เป็นไปได้ทุกเส้นทางจาก s ถึง z ซึ่งเป็นไปตามการบังคับพารามิเตอร์ต่าง ๆ ของ QoS ซึ่งถูกนิยามดังสมการที่ (2.5) (2.6) (2.7) และ (2.8) การบังคับพารามิเตอร์ต่าง ๆ ของคุณภาพของการบริการจะใช้สัญลักษณ์ Δ กับความสัมพันธ์ของตัวห้อย ความน่าจะเป็นของการสูญเสียจะมีสัญลักษณ์คือ $l(e)$ ดังนั้นความน่าจะเป็นของการส่งข้อมูลที่ประสบความสำเร็จคือ $l'(e) = 1-l(e)$ สำหรับเส้นทางที่สมบูรณ์ จะมีความน่าจะเป็นเท่ากับ $L'(P) = 1-L(P)$ สมการที่ (2.3) และ (2.4) เป็นฟังก์ชั่นที่ทำให้มีประสิทธิภาพ ทำให้ดีที่สุดเท่าที่จะทำได้ ของปัญหาการเลือกเส้นทางบนพื้นฐานการรับประกัน QoS ฟังก์ชั่นทั้งสอง บางครั้งอาจจะขัดแย้งกันกล่าวคือ เส้นทางที่มีจำนวนฮอปน้อยที่สุด อาจจะไม่จำเป็นต้องมีค่าใช้จ่ายต่ำที่สุด โดยทั่วไปแล้วเส้นทางที่มีจำนวนฮอปน้อยที่สุด ควรจะเป็นหลักการแรกของการเลือกเส้นทาง ถ้าหากว่ามีเส้นทางที่มี จำนวนฮอปน้อยที่สุดมากกว่าหนึ่งเส้นทาง แล้วเส้นทางที่มีค่าใช้จ่ายต่ำที่สุดจะถูกเลือก

ในวิทยานิพนธ์นี้ เราพิจารณากลุ่มบริการที่ได้รับการประกันของรูปแบบการบริการร่วมในเครือข่ายคอมพิวเตอร์ ซึ่งเป็นบริการที่ได้รับการประกันที่อธิบายในเทอมของ ความน่าจะเป็นในการสูญเสียเป็นศูนย์ เช่น เราจำเป็นที่จะแน่ใจว่าไม่มีการสูญเสียในคิว (Queueing) ซึ่งเป็นการบังคับให้ไม่มีการสูญเสียเกิดขึ้น ด้วยเหตุนี้เรากำหนดอีกครั้งว่า การบังคับความน่าจะเป็นในการสูญเสียดังสมการที่ (2.8) เป็นการบังคับช่องว่างของบัฟเฟอร์ (Buffer Space) ดังสมการที่ (2.9) อีกอย่างหนึ่งทุกโหนดหรือสวิตซ์ตามเส้นทางที่ถูกเลือกจะต้องมีช่องว่างของบัฟเฟอร์ที่เพียงพอ ดังนั้นอาจจะไม่มีการสูญเสียคิวอิ่งของแพ็กเกต (Queueing Packet) ที่เกิดขึ้นที่โหนดนั้น

$$\forall e = (u,v) \in P,\ f(e) \geq \Delta^{e}_{buffer} \qquad (2.9)$$

โดยที่ $f(e) = f(u,v)$ สำหรับ $f(e)$ และ Δ^{e}_{buffer} เป็นจำนวนของบัฟเฟอร์ที่ใช้การได้ และช่องว่างของบัฟเฟอร์ที่ต้องการ สำหรับการไม่มีการสูญเสียคิวอิ่งที่โหนด v ที่มาจากโหนด u โดยผ่านลิงค์ e ตามลำดับ

การบังคับแบนด์วิดธ์ ดังสมการที่ (1.4) เป็นการบังคับแบบเว้าเข้า (Concave Constraint) หรือการบังคับลิงค์ (Link Constraint) การบังคับดีเลย์ (Delay Constraint) ดังสมการที่ (2.6) และการบังคับจิตเตอร์ (Jitter Constraint) ดังสมการที่ (2.7) เป็นการบังคับที่เพิ่มเข้ามา ท้ายที่สุดนี้ การบังคับความน่าจะเป็นในการสูญเสีย (Loss Constraint) ดังสมการที่ (2.8) เป็นการบังคับที่มีแนวโน้มจะเพิ่มขึ้น (Multiplicative Constraint) การบังคับดีเลย์ จิตเตอร์ และความน่าจะเป็นในการสูญเสีย เรียกว่าการบังคับเส้นทาง (Path Constraints)

2.2 ผลงานวิจัยที่เกี่ยวข้อง

ได้มีการศึกษาและวิจัยที่เกี่ยวข้องเพื่อแก้ไขความซับซ้อนของปัญหาโดยจัดกระบวนการความสำคัญของปัจจัยที่กำหนด การบริการที่รับประกัน QoS ส่วนมากเกี่ยวกับ ดีเลย์ จำนวนฮอป แบนด์วิดธ์ โดยใช้อัลกอริทึมทางเดินที่สั้นที่สุด (Shortest Path Algorithm) อย่างง่ายเพื่อค้นหาเส้นทาง ตัวอย่างเช่น แวง และโครวครอป ได้เสนอแนวคิดโดยใช้ SWP (Shortest Widest Path) โดยที่อัลกอริทึม SWP คือ การค้นหาเส้นทางที่มีค่าแบนด์วิดธ์ที่เป็นคอขวดที่มีค่ามากที่สุด ถ้าหากว่ามีมากกว่าหนึ่งเส้นทางแล้ว เส้นทางที่มีค่าดีเลย์น้อยที่สุดจะถูกเลือก

อัลกอริทึมที่เกี่ยวกับการเลือกเส้นทางที่รับประกันคุณภาพ ซาลามา (Salama) [5] ได้เสนออัลกอริทึมการเลือกเส้นทางยูนิคาสต์ที่บังคับดีเลย์แบบกระจาย (Distributed Delay Constrainted Unicast Routing) แม้ว่าการดำเนินการของอัลกอริทึมนี้จะง่าย แต่ทว่ามีลูปที่ไม่เสถียร (Transient Loops) เกิดขึ้น และจำเป็นที่จะต้องถูกนำออกไป ดังนั้นการดำเนินการจึงเพิ่มขึ้นตามจำนวนของข้อมูลที่ส่ง และใช้เวลานานเพื่อกำหนดการเชื่อมโยง กรณีที่แย่ที่สุดความซับซ้อนจะเป็น $O(|V|^3)$ ใน [6] อัลกอริทึมในการเลือกเส้นทางที่มีการรับประกัน QoS สำหรับการคำนวณไว้ล่วงหน้า โดยพิจารณาค่าเอนทูเอนดีเลย์ที่ต่ำที่สุด แต่ไม่ได้มีการพิจารณาการบังคับค่าใช้จ่าย จิตเตอร์ดีเลย์ และระเบียบการให้บริการ

อัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS ที่มีอยู่ตามที่กล่าวมาแล้วนั้น ไม่ได้พิจารณาการแสดงขอบเขตการรับประกัน QoS ซึ่งได้มาจากการนำระเบียบวิธีการให้บริการมาใช้ อัลกอริทึมในการเลือกเส้นทางถูกออกแบบการพิจารณาระเบียบวิธีการให้บริการ การเลือกเส้นทางที่รับประกัน QoS ที่ใช้ระเบียบวิธีการให้บริการแบบ RCSP [7] ขณะที่ [2] [8] ใช้ระเบียบวิธีการการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม ซึ่งความแตกต่างจากงานนี้ถูกระบุดังข้างล่าง ใน [2] อัลกอริทึมทางเดิน SWP สำหรับทางเดินที่ถูกคำนวณไว้ล่วงหน้า ถูกนำเสนอ อัลกอริทึมนี้จะค้นหาเส้นทางที่มีจำนวนฮอปน้อยที่สุด ที่มีแบนด์วิดธ์คอขวดที่มากที่สุด ถ้าเส้นทางนั้นไม่เป็นไปตามเงื่อนไขการบังคับดีเลย์ จำนวนฮอป แล้วอัลกอริทึมนี้ก็จะพยายามหาเส้นทางที่มีจำนวนฮอปที่มากกว่านั้น อัลกอริทึมนี้จะทำซ้ำ ๆ จนกระทั่งถึงจำนวนฮอปมากที่สุดที่กำหนดไว้ล่วงหน้า อย่างไรก็ตามในงานวิจัยนี้ไม่ได้พิจารณา พรอพพิเกชั่นและทรานสมิทชั่นดีเลย์ (Propagation and Transmission Delay) เนื่องจากค่าดีเลย์เหล่านี้ไม่สามารถกำหนดค่าได้ กล่าวคือ ค่าเหล่านี้ไม่จำเป็นที่จะต้องเท่ากันทั้งหมดทุกลิงค์

ใน [2] นโยบายในการตัดลิงค์ที่มีค่าพรอพพิเกชั่นดีเลย์สูง เช่น ลิงค์ของดาวเทียม ถูกนำเสนอในอัลกอริทึมของเรา $k-WSP$ ลิงค์ทั้งหลายเหล่านี้จะถูกละทิ้งโดยอัตโนมัติ ถ้าหากว่าเป็นไปตามเงื่อนไขที่ได้กำหนดไว้ รายละเอียดจะอธิบายในบทที่ 4 เนื่องจากว่าอัลกอริทึมเส้นทาง

WSP จะพิจารณาเฉพาะเส้นทางที่มีค่าแบนด์วิดธ์มากที่สุดเท่านั้น ดังนั้นอัลกอริทึมเส้นทาง *WSP* จึงไม่สามารถรับประกันในการหาเส้นทางที่เป็นไปตามการบังคับดีเลย์ ถึงแม้ว่าจะมีเส้นทางนั้นภายในเครือข่าย อีกอย่างหนึ่งคือนิยามของ ดีเลย์ของลิงค์ ก็แตกต่างจากของเราด้วย เนื่องจากค่าดีเลย์ของลิงค์ในงานวิจัยของพวกเขา เป็นผลรวมของดีเลย์ของลิงค์ที่เป็นทั้งแบบสถิต (Static Link Delay) และพลวัต ขณะที่ในวิทยานิพนธ์นี้ค่าดีเลย์ของลิงค์เป็นแบบสถิตเท่านั้น นอกจากนั้นผลการทดลองของเราได้แสดงให้เห็นว่า ถ้าหากมีเส้นทางเป็นไปตามเงื่อนไขที่ได้นิยามไว้ เส้นทางนั้นไม่จำที่จะต้องถูกเก็บไว้ในตารางเส้นทาง เนื่องจากว่ามีเส้นทางที่ดีกว่าถูกเก็บไว้เรียบร้อยแล้ว ด้วยเหตุนี้ขนาดของตารางเส้นทางจึงมีขนาดลดลง ใน [8] อัลกอริทึมเส้นทางที่สั้นที่สุดวิธีของเบลแมนฟอร์ดถูกนำมาใช้ สำหรับหาแบนด์วิดธ์ที่เหลืออยู่ที่เป็นไปได้ทั้งหมด ในที่นี้คือค่า k เมื่อไม่ทราบล่วงหน้าของจำนวนของแบนด์วิดธ์ที่ถูกจองไว้ ด้วยเหตุนี้เส้นทางนั้นสามารถเป็นไปตามการบังคับแบนด์วิดธ์และดีเลย์ จะถูกค้นพบเสมอถ้าหากว่ามีเส้นทางนั้น ถ้าอัลกอริทึมของเบลแมนฟอร์ด ถูกใช้เป็นอัลกอริทึมเส้นทางที่สั้นที่สุด ความซับซ้อนของเวลามีค่าเท่ากับ $O(K \cdot H_{\max} \cdot |E|)$ อย่างไรก็ตามในงานวิจัยของพวกเขาไม่ได้พิจารณา จิตเตอร์ดีเลย์ และช่องว่างของบัฟเฟอร์

บทที่ 3
ทฤษฎีการเลือกเส้นทางในระบบเครือข่ายคอมพิวเตอร์

การเลือกเส้นทางจากโหนดต้นทางไปยังโหนดปลายทางเป็นกิจกรรมหนึ่งที่จะมีขึ้นภายใต้กติกาการสื่อสารมาตราฐาน OSI (Open System Interconnection) ในเลเยอร์ที่ 3 หรือชั้นควบคุมเครือข่าย (Network Layer) ซึ่งภายในชั้นนี้จะมีโครงสร้างและการทำงานดังต่อไปนี้

ในโปรแกรมการสื่อสารชั้นควบคุมเครือข่ายมีหน้าที่ในการควบคุมดูแลจัดส่งข้อมูลแต่ละชิ้นจากผู้ส่งไปถึงจุดหมาย หรือผู้รับที่กำหนด ถ้าการจัดส่งดังกล่าวเป็นเพียงการจัดส่งข้อมูลภายในกลุ่มเดียวกัน ผู้ส่งจะมองเห็นหรือติดต่อกับผู้รับได้โดยตรง แต่ถ้าเป็นการส่งข้อมูลระหว่างสมาชิกของกลุ่มที่อยู่ติดกันนั้นโหนดในกลุ่มของผู้ส่งก็ยังสามารถมองเห็น หรือติดต่อได้โดยตรงกับโหนดของกลุ่มผู้รับ ส่วนการส่งข้อมูลระหว่างผู้รับและผู้ส่งที่อยู่ห่างไกลกันโดยไม่มีการเชื่อมต่อตรงด้วยวิธีการใด ๆ ในกรณีนี้โหนดของกลุ่มผู้ส่งจำเป็นจะต้อง "ฝาก" ข้อมูลไปยังโหนดกลุ่มที่อยู่ติดกันซึ่งก็จะ "ฝาก" ข้อมูลนั้นต่อไปเรื่อย ๆ จนถึงโหนดของกลุ่มผู้รับ ซึ่งจะจัดส่งข้อมูลไปยังผู้รับตัวจริงได้

เพื่อให้บรรลุวัตถุประสงค์ดังกล่าว "การรับ-ส่งข้อมูล" จึงได้รับการพัฒนาขึ้นมาใช้งานในชั้นควบคุมเครือข่าย เมื่อรับข้อมูลที่ถูกส่งมาจากผู้ใดก็ตาม โหนดต่าง ๆ ที่เข้าใจกฎเกณฑ์นี้ก็จะตรวจสอบได้ว่าข้อมูลนั้นเป็นของลูกข่ายตนเองหรือว่าจะต้องส่งต่อไปให้ใคร ในกรณีที่มีทางเลือกในการส่งข้อมูลหลายทาง โหนดก็ยังต้องมีความสามารถในการเลือกเส้นทางที่ดีที่สุด คือจะไม่ส่งข้อมูลไปยังเครื่องที่มีงานล้นอยู่ โดยปล่อยให้อีกเครื่องหนึ่งไม่มีงานทำ นอกจากนี้ยังต้องสามารถแก้ไขปัญหาในการส่งข้อมูลผ่านระบบที่แตกต่างกันได้ด้วย ซึ่งการจัดการในส่วนต่าง ๆ นั้น จะมีรายละเอียดดังต่อไปนี้

3.1 การออกแบบชั้นควบคุมเครือข่าย (Network Layer Design Issue)

ในส่วนนี้จะกล่าวถึงข้อพิจารณาเบื้องต้นสำหรับการออกแบบชั้นควบคุมเครือข่าย รวมถึงบริการต่าง ๆ ที่จะต้องจัดเตรียมไว้สำหรับให้ชั้นนำส่งข้อมูลเรียกใช้ได้โดยสะดวก

3.1.1 บริการสำหรับชั้นนำส่งข้อมูล (Service Provided to the Transport Layer)

ชั้นควบคุมเครือข่ายให้บริการแก่โปรแกรมในชั้นนำส่งข้อมูลผ่านทางส่วนติดต่อ (Interface) ระหว่างชั้นทั้งสอง ส่วนติดต่อนี้ยังมีความสำคัญอีกประการหนึ่งคือ เป็นการเชื่อมต่อระหว่างผู้นำส่งข้อมูล และผู้รับช่วงข้อมูล ซึ่งอาจจะเป็นผู้รับข้อมูลจริงหรือเป็นเพียงตัวกลางที่จะ

ต้องนำข้อมูลนั้นส่งต่อไปอีกก็ได้ ในกรณีนี้ส่วนติดต่อจะทำงานที่เขตติดต่อระหว่างเครือข่ายย่อย คือผู้นำส่งข้อมูลจะอยู่ในระบบเครือข่ายย่อยหนึ่ง ส่วนผู้รับช่วงข้อมูลอยู่ในระบบเครือข่ายย่อยอื่นที่อยู่ติดกัน โดยปกติผู้นำส่งข้อมูลมักจะเป็นผู้ควบคุมโพรโตคอลตั้งแต่ระดับล่างขึ้นไปจนถึงส่วนติดต่อในชั้นควบคุมเครือข่าย ซึ่งมีความรับผิดชอบในการนำแพ็กเกตไปส่งให้ได้ ส่วนติดต่อจึงต้องได้รับการออกแบบเป็นอย่างดี

บริการในชั้นควบคุมเครือข่ายมีวัตถุประสงค์ในการออกแบบ คือ

1. บริการที่มีให้แก่ชั้นนำส่งข้อมูลจะต้องเป็นอิสระจากเทคโนโลยีที่ใช้ในระบบเครือข่ายย่อย
2. ชั้นนำส่งข้อมูลจะต้องไม่เข้ามาเกี่ยวข้องกับโครงสร้างของระบบเครือข่ายย่อย
3. วิธีการกำหนดที่อยู่ในเครือข่ายที่ส่งให้ชั้นนำส่งข้อมูลจะต้องอยู่ในรูปแบบมาตราฐานที่เป็นที่เข้าใจได้แม้ในระบบเครือข่ายต่างแบบกัน

3.1.2 การจัดโครงสร้างภายในชั้นควบคุมเครือข่าย

การจัดโครงสร้างระบบเครือข่ายย่อยแบ่งออกเป็นสองแบบ คือ โครงสร้างที่ใช้การสื่อสารแบบต่อเนื่อง และโครงสร้างที่ใช้การสื่อสารแบบไม่ต่อเนื่อง ช่องสื่อสารสำหรับการเชื่อมต่อภายในระบบเครือข่ายย่อยแบบต่อเนื่องเรียกว่าวงจรชั่วคราว หรือวงจรเสมือน (Virtual Circuit) ซึ่งเปรียบเทียบได้กับการเชื่อมต่อสัญญาณที่ใช้ในระบบโทรศัพท์ ส่วนแพ็กเกตที่ใช้ในโครงสร้างการสื่อสารแบบไม่ต่อเนื่องเรียกว่าดาต้าแกรม (Datagram) เปรียบเทียบได้กับข้อความที่นำส่งในระบบโทรเลข

แนวความคิดหลักของการใช้วงจรเสมือนก็เพื่อต้องการหลีกเลี่ยงการค้นหาเส้นทางเดินข้อมูลในทุก ๆ ครั้งที่ส่งแพ็กเกต ดังนั้นตั้งแต่เริ่มต้นการติดต่อ โปรแกรมในชั้นควบคุมเครือข่ายจะต้องจัดการกำหนดเส้นทางเดินสำหรับการรับ-ส่งข้อมูลจากผู้ส่งไปยังผู้รับขึ้นมาใช้งาน จากนั้นข้อมูลทั้งหมดจะถูกส่งจากผู้ส่งไปยังผู้รับโดยใช้เส้นทางนี้ตลอดช่วงของการสื่อสาร เมื่อการติดต่อสิ้นสุดลงเส้นทางนี้จะถูกยกเลิก การกำหนดเส้นทางเดินข้อมูลจึงเป็นการกำหนดขึ้นมาใช้งานเป็นการชั่วคราวซึ่งมีลักษณะการทำงานแบบเดียวกันกับวิธีการที่ใช้ในระบบโทรศัพท์ ส่วนที่เรียกว่าเป็นเส้นทางเสมือนนั้นก็เนื่องจากว่าเส้นทางสื่อสารชั่วคราวนี้จะไม่มีการกำหนดเป็นที่แน่นอน การกำหนดเส้นทางเดินข้อมูลแต่ละครั้งจึงไม่มีการรับประกันว่าจะเป็นเส้นทางเดิมที่เคยใช้ในการติดต่อครั้งที่แล้วมา

ในทางตรงกันข้าม ดาต้าแกรมแต่ละตัวจะไม่มีการกำหนดเส้นทางเดินของข้อมูลเอาไว้ล่วงหน้าแม้ว่าจะใช้การสื่อสารแบบต่อเนื่องก็ตาม แพ็กเกตแต่ละตัวจะถูกส่งไปยังผู้รับโดยใช้เส้นทางเดินข้อมูลที่เป็นอิสระต่อแพ็กเกตอื่น ๆ ดังนั้นแม้ว่าแพ็กเกตจะถูกส่งออกมาอย่างต่อเนื่องกันไปก็ไม่มีการรับประกันใด ๆ ว่าแพ็กเกตทั้งหมดจะเดินทางไปถึงผู้รับด้วยเส้นทางเดียวกัน อย่างไรก็ตาม

การส่งข้อมูลด้วยวิธีนี้ทำให้เกิดความอ่อนตัวในการส่งข้อมูล โดยเฉพาะในกรณีที่เกิดความคับคั่งของเส้นทางเดินข้อมูลบางส่วน หรือเส้นทางเดินข้อมูลบางส่วนเกิดเสียหายจนใช้การไม่ได้ ดาต้าแกรมจะสามารถเลือกเส้นทางเดินอื่นเพื่อไปยังผู้รับได้โดยอัตโนมัติ

เมื่อกำหนดเส้นทางวงจรเสมือนขึ้นมาใช้งาน โหนดแต่ละตัวที่อยู่ในเส้นทางจะต้องบันทึกข้อมูลไว้ในตารางภายในของโหนดว่าจะต้องส่งแพ็กเกตของแต่ละวงจรเสมือนไปยังโหนดตัวใด ตารางนี้จะต้องทำการบันทึกข้อมูลสำหรับทุกวงจรเสมือนที่ถูกกำหนดขึ้นใช้งานดังที่กล่าวไว้ข้างต้น เมื่อมีการกำหนดวงจรเสมือนก็จะต้องกำหนดเลขหมายพิเศษกำกับไว้เสมอ ในแต่ละแพ็กเกตจึงต้องบันทึกเลขหมายพิเศษของวงจรเสมือนไว้แทนที่อยู่ของผู้รับและผู้ส่ง โหนดจะทำการตรวจสอบหมายเลขเหล่านี้กับตารางที่เก็บไว้ก็จะทราบว่าจะต้องส่งแพ็กเกตไปยังเส้นทางใด

วงจรเสมือนแบ่งออกตามขอบเขตของระบบเครือข่ายเป็น 2 ประเภท คือ วงจรเสมือนภายใน และวงจรเสมือนภายนอก วงจรเสมือนภายในนั้นเป็นการกำหนดเส้นทางของข้อมูลภายในระบบเครือข่ายย่อยระบบหนึ่งอย่างเป็นอิสระต่อวิธีการที่ใช้ในระบบเครือข่ายย่อยอื่น ตั้งแต่โหนดตัวแรกที่รับแพ็กเกตจากระบบเครือข่ายย่อยอื่นเข้ามา (หรือเป็นผู้ส่งข้อมูลเอง) ไปจนถึงโหนดตัวสุดท้ายก่อนที่จะส่งแพ็กเกตนั้นออกไปภายนอก วงจรเสมือนแบบภายนอกนั้นเปรียบได้กับการกำหนดเส้นทางวงจรเสมือนในระดับเครือข่ายย่อยเท่านั้น คือจากเครือข่ายย่อยของผู้ส่งจะต้องผ่านเครือข่ายย่อยใดบ้างไปจนถึงเครือข่ายย่อยของผู้รับ

3.2 อัลกอริทึมสำหรับเลือกทางเดินข้อมูล (Routing Algorithm)

หน้าที่หลักของชั้นควบคุมเครือข่าย คือการจัดส่งแพ็กเกตจากเครื่องผู้ส่งไปยังเครื่องผู้รับ ในระบบเครือข่ายย่อยส่วนมากแพ็กเกตจะต้องถูก “รับ-แล้ว-ส่งต่อ (Hop)” หลายครั้งกว่าจะถึงจุดหมายปลายทาง ยกเว้นชั้นควบคุมเครือข่ายของระบบที่ใช้การส่งข้อมูลแบบกระจายที่ไม่ต้องจัดการปัญหานี้ แต่ก็ยังคงเกิดปัญหาขึ้นได้ ถ้าผู้ส่งและผู้รับข้อมูลไม่ได้อยู่ในเครือข่ายเดียวกัน ดังนั้นอัลกอริทึมและโครงสร้างข้อมูลที่ทำหน้าที่เลือกเส้นทางของข้อมูล จึงเป็นส่วนประกอบหลักในการออกแบบชั้นควบคุมเครือข่าย

อัลกอริทึมสำหรับเลือกเส้นทางของข้อมูล ทำหน้าที่ตัดสินใจให้โหนดว่าแพ็กเกตที่รับเข้ามาควรจะส่งต่อไปให้โหนดตัวใดเป็นลำดับต่อไป ถ้าเครือข่ายย่อยใช้ดาต้าแกรมในการส่งข้อมูลภายในแล้ว การตกลงใจของแต่ละโหนดจะเกิดขึ้นทุกครั้งที่รับดาต้าแกรมตัวใหม่เข้ามา เพราะทางเลือกที่ดีที่สุด (สำหรับปลายทางเดิม) อาจเปลี่ยนแปลงไปแล้ว ถ้าเครือข่ายย่อยใช้วงจรเสมือนในการส่งข้อมูล การเลือกเส้นทางเดินข้อมูลตลอดทั้งเครือข่ายจะเกิดขึ้นเพียงครั้งเดียวในช่วงการจัดตั้งช่องสื่อสาร แพ็กเกตที่ถูกส่งมาในลำดับหลังจะถูกส่งไปยังโหนดที่แพ็กเกตแรก ๆ ถูกส่งไปเสมอ

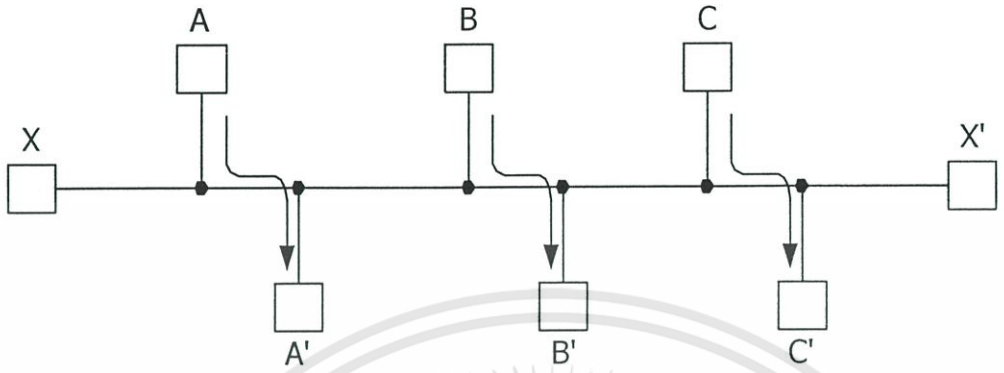
วิธีการเลือกเส้นทางเดินแบบนี้บางครั้งเรียกว่า "เซสชั่นเราติ้ง (Session Routing)" เนื่องจากเส้นทางจะคงเดิมตลอดช่วงเวลาสื่อสารที่กำหนดขึ้น

ไม่ว่าเส้นทางเดินข้อมูลจะได้รับการกำหนดขึ้นมาด้วยอัลกอริทึมใดก็ตาม ควรจะต้องมีคุณลักษณะ 6 ประการ คือ ความถูกต้อง (Correctness), ความง่าย (Simplicity), ความคงทน (Robustness), ความแน่นอน (Stability), ความเป็นธรรม (Fairness), และความเหมาะสม (Optimality) ความถูกต้องและความง่าย เป็นคุณสมบัติพื้นฐานที่สำคัญสองข้อแรกที่ทุกอัลกอริทึมจะต้องมี ความคงทนของอัลกอริทึมสามารถพิจารณาได้จากการที่เครือข่ายใหม่ที่เพิ่งได้รับการจัดตั้งขึ้นมาใช้งานย่อมมีความต้องการที่ให้บริการคงอยู่ โดยไม่ประสบปัญหาความล้มเหลวในระดับเครือข่าย แต่ความล้มเหลวของอุปกรณ์หรือโปรแกรมบางส่วนเป็นเหตุการณ์ที่เกิดขึ้นได้ตลอดเวลาสำหรับทุกระบบ และไม่สามารถหลีกเลี่ยงได้เช่น โฮสต์ เราเตอร์ และสายสื่อสารอาจเกิดการเสียหายเมื่อใดก็ได้ ซึ่งจะต้องได้รับการซ่อมแซมอยู่เสมอ หรือการจัดโครงสร้างของเครือข่าย (Network Topology) อาจมีการเปลี่ยนแปลงได้เช่นกัน อัลกอริทึมสำหรับเลือกทางเดินข้อมูลจำเป็นจะต้องแก้ไข หรือสามารถปรับตัวให้เข้ากับสภาพปัญหา และการเปลี่ยนแปลงเหล่านี้ได้ เมื่อเกิดปัญหาขึ้นโฮสต์ส่วนใหญ่จะต้องสามารถทำงานต่อไปได้ โดยไม่ต้องเข้ามามีส่วนร่วมในการแก้ไข หรือจัดการกับปัญหาที่เกิดขึ้น

ความแน่นอนก็เป็นส่วนสำคัญมากเช่นกัน อัลกอริทึมสำหรับเลือกเส้นทางของข้อมูลจำนวนไม่น้อยที่ไม่สามารถจัดการให้เครือข่ายเข้าสู่สถานะความสมดุลในการกระจายปริมาณข้อมูล ทำให้ระบบอยู่ในสภาวะที่เกิดความไม่แน่นอนในการสื่อสารข้อมูลในระดับสูงซึ่งเป็นสิ่งที่ไม่ต้องการ ความเป็นธรรม และความเหมาะสมเป็นสิ่งที่ผู้ใช้ทุกคนในระบบต้องการอย่างไม่ต้องสงสัยแต่ไม่ว่าจะทำอย่างไรก็ตามในทางปฏิบัติจะต้องมีผู้ที่ไม่ได้รับความเป็นธรรมอยู่บ้าง ในรูปที่ 3.1 แสดงระบบเครือข่ายหนึ่งที่สมมติให้ปริมาณข้อมูลที่ไหลจาก A ไป A' จาก B ไป B' และจาก C ไป C'มีปริมาณมากพอที่จะทำให้การส่งข้อมูลตามแนวราบอยู่ในสภาวะอิ่มตัว ถ้าต้องการให้ระบบนี้มีประสิทธิภาพโดยรวมสูงสุดแล้วการสื่อสารระหว่าง X และ X' ควรจะต้องถูกระงับไว้ก่อนมิฉะนั้นจะทำให้การสื่อสารในส่วนที่เหลือทั้งหมดต้องหยุดชะงัก ผลคือทำให้ประสิทธิภาพโดยรวมของระบบต่ำลงไป การตัดสินใจเช่นนี้เป็นการขัดต่อความเป็นธรรมที่จะพึงให้แก่สมาชิกทุกคนของระบบอย่างเท่าเทียมกัน ดังนั้นจึงต้องมีการประนีประนอมระหว่างความเหมาะสมกับความเป็นธรรม

ความเหมาะสมประการหนึ่งที่ทุกระบบต้องการ คือ การลดเวลารอคอยโดยเฉลี่ยของแพ็กเกตในขณะที่การเพิ่มอัตราสัมฤทธิ์ผล (Throughput) ของระบบเครือข่ายก็เป็นสิ่งที่พึงประสงค์เช่นกัน ความต้องการทั้งสองข้อนี้ขัดแย้งซึ่งกันและกัน เนื่องจากการเพิ่มอัตราสัมฤทธิ์ผลของระบบต้องการให้แพ็กเกตข้อมูลมารอคอยอยู่ในโหนดให้มากที่สุด เพื่อจะได้รักษาความต่อเนื่องในการส่ง

ข้อมูลให้อยู่ในระดับสูงสุดตลอดเวลา แต่การทำงานนี้กลับทำให้การรอคอยโดยเฉลี่ยของแต่ละแพ็กเกตสูงขึ้น เพื่อลดความขัดแย้งดังกล่าวเครือข่ายหลายระบบได้พยายามลดจำนวนโหนดที่ใช้



รูปที่ 3.1 ความขัดแย้งระหว่างความเหมาะสมและความเป็นธรรม

เป็นตัวกลางลงไปให้มากที่สุด ทั้งนี้จะทำให้เวลารอคอยของแพ็กเกตลดลงในขณะที่ช่วยเพิ่มอัตราสัมฤทธิ์ผลของระบบได้ในเวลาเดียวกัน

3.3 หลักการการจำแนกประเภทของการเลือกเส้นทาง

อัลกอลิทึมการเลือกเส้นทางที่ใช้ในการค้นหาเส้นทางที่เชื่อมต่อจากโหนดต้นทางถึงกลุ่มผู้รับของเซสชั่นนั้น โดยปกติแล้วอัลกอริทึมเส้นทางที่สั้นที่สุดจะเลือกเส้นทางที่มีดีเลย์ต่ำที่สุดหรือเส้นทางที่มีจำนวนฮอปน้อยที่สุด ในเครือข่ายคอมพิวเตอร์ มีหลาย ๆ อัลกอริทึมการเลือกเส้นทาง ซึ่งเราสามารถจัดประเภทของอัลกอริทึมการเลือกเส้นทางโดยการใช้หลักการดังต่อไปนี้

3.3.1 จำนวนของโหนดปลายทาง

พิจารณาตามจำนวนของโหนดปลายทาง ประเภทของอัลกอริทึมการเลือกเส้นทางมี 3 ประเภท ดังต่อไปนี้

3.3.1.1 ยูนิคาสต์ (Unicast) เป็นการสื่อสารจากโหนดต้นทางหนึ่งโหนดถึงโหนดปลายทางหนึ่งโหนด

3.3.1.2 มัลติคาสต์ (Multicast) เป็นการสื่อสารจากโหนดต้นทางหนึ่งโหนดถึงโหนดปลายทางจำนวนหนึ่ง

3.3.1.3 บรอดคาสต์ (Broadcast) เป็นการสื่อสารจากโหนดต้นทางหนึ่งโหนดถึง

โหนดปลายทางทุกโหนด

แอพพลิเคชั่นการสื่อสารหลายจุดที่เป็นแบบสื่อผสม เช่นการประชุมผ่านระบบเครือข่าย โดยปกติแล้วต้องการการเลือกเส้นทางที่เป็นมัลติคาสต์ เพื่อรองรับการเลือกเส้นทางถึงผู้ส่งและผู้รับทุกโหนด อัลกอริทึมการเลือกเส้นทางมัลติคาสต์ที่มีประสิทธิภาพจะทำให้การจัดการทรัพยากรที่มีประสิทธิภาพ

3.3.2 อัลกอริทึมการกระจายและพื้นฐานของโหนดต้นทาง (Source based and distributed algorithms)

อัลกอริทึมบนรากฐานของโหนดต้นทาง โหนดต้นทางจะตัดสินใจเลือกเส้นทางทั้งหมดจากโหนดต้นทางถึงโหนดปลายทาง แต่สำหรับอัลกอริทึมแบบกระจาย แต่ละโหนดจะมีข่าวสารในระดับท้องถิ่น แต่ละโหนดเลือกโหนดที่อยู่ข้างเคียงเท่านั้นเพื่อที่จะส่งแพ็กเกตไปถึงโหนดที่อยู่ปลายทาง

3.3.3 อัลกอริทึมที่คำนวณไว้ล่วงหน้าและขึ้นอยู่กับความต้องการ (Precomputed and On-demand algorithms) มีรายละเอียดดังต่อไปนี้

3.3.3.1 การเลือกเส้นทางแบบคำนวณไว้ล่วงหน้า

เลือกเส้นทางแบบนี้ผู้ดูแลระบบ เป็นผู้พิจารณาและคำนวณหาเส้นทางทั้งหมดที่ต้องกำหนดให้เราเตอร์ทุกตัวในเครือข่ายที่ดูแลอยู่ และป้อนตารางเส้นทางให้กับเราเตอร์ ตารางเลือกเส้นทางจะมีค่าตายตัวไปตลอดจนกว่าผู้ดูแลระบบจะเปลี่ยนแปลงใหม่ บางครั้งเรียกว่าการเลือกเส้นทางแบบสถิตย์ (Static Routing Algorithms) [9] ข้อดีของการเลือกเส้นทางแบบคำนวณไว้ล่วงหน้า คือ

1. สะดวกต่อการใช้งานกับเครือข่ายขนาดเล็ก
2. ไม่ต้องใช้ซอฟท์แวร์เลือกเส้นทาง เราเตอร์ไม่จำเป็นต้องมีซีพียูสมรรถนะสูง
3. ประหยัดแบนด์วิดธ์เครือข่าย เนื่องจากไม่ต้องแลกเปลี่ยนข้อมูลระหว่างเราเตอร์

การเลือกเส้นทางแบบคำนวณไว้ล่วงหน้านิยมใช้กับการเชื่อมโยงแบบจุดต่อจุดระหว่างเราเตอร์สองตัว ตัวอย่างเช่นเครือข่ายที่มีทางออกไปสู่ภายนอก หรืออินเตอร์เน็ตเพียงช่องทางเดียว การกำหนดเส้นทางจะเป็นแบบคำนวณไว้ล่วงหน้า ถึงแม้ว่าการเลือกเส้นทางแบบคำนวณไว้ล่วงหน้าจะมีข้อดีหลายประการ แต่ก็มีข้อเสีย คือ

1. ไม่เหมาะกับเครือข่ายขนาดใหญ่ ผู้ดูแลระบบสามารถคำนวณหาเส้นทางในเครือข่ายที่มีขนาดเล็กได้ไม่ยากนัก แต่ในเครือข่ายขนาดใหญ่ที่มีเราเตอร์เป็นจำนวนมากการคำนวณและป้อนค่าเข้าสู่เราเตอร์โดยตรงเป็นสิ่งที่เกินวิสัย
2. ไม่สะดวกต่อการเปลี่ยนโทโปโลยี เพราะต้องคำนวณและป้อนเส้นทางใหม่
3. ตารางเส้นทางคงตัวและไม่สามารถเปลี่ยนแปลงเองได้ หากเส้นทางใดถูกตัดขาด ผู้ดูแลระบบต้องคอยตรวจสอบและแก้ปัญหาเอง

3.3.3.2 การเลือกเส้นทางแบบขึ้นอยู่กับความต้องการ

การเลือกเส้นทางแบบนี้ใช้ซอฟท์แวร์ทำหน้าที่แลกเปลี่ยนข้อมูลการเลือกเส้นทางระหว่างเราเตอร์ด้วยกันโดยใช้ โปรโตคอลเลือกเส้นทาง (Routing Protocol) เราเตอร์จะสร้างตารางเลือกเส้นทางจากสภาพเครือข่ายขณะนั้น หากเครือข่ายเปลี่ยนแปลงไปตารางเลือกเส้นทางจะเปลี่ยนไปด้วย บางครั้งเรียกว่า การเลือกเส้นทางแบบพลวัตร (Dynamic Routing Algorithms) ข้อดีของการเลือกเส้นทางแบบขึ้นอยู่กับความต้องการ คือ

1. รองรับขนาดเครือข่ายที่ขยายขึ้นเป็นลำดับได้
2. ตารางเส้นทางเปลี่ยนแปลงค่าเองตามการทำงานของซอฟท์แวร์เส้นทางใดที่ถูกตัดขาดจะมีการหาเส้นทางใหม่ทดแทน

การเลือกเส้นทางแบบขึ้นอยู่กับความต้องการนี้ ต้องอาศัยการแลกเปลี่ยนค่าเส้นทางระหว่างเราเตอร์ และใช้ซีพียูในเราเตอร์เพื่อสร้างตารางเส้นทาง เราเตอร์ประเภทนี้จึงมักมีราคาสูงกว่าเราเตอร์ที่มีโปรโตคอลแบบคำนวณไว้ล่วงหน้าอย่างเดียว เพราะต้องออกแบบซอฟท์แวร์ทำงานตามโปรโตคอลเลือกเส้นทาง และต้องมีซีพียูสมรรถนะสูงเพียงพอในการคำนวณตารางเส้นทาง สำหรับอัลกอลริทึมที่ขึ้นอยู่กับความต้องการเป็นการเลือกเส้นทางที่ค้นหาเส้นทางทุกครั้งที่มีการร้องขอการเชื่อมต่อเซสชั่นที่มาถึง สำหรับอัลกอริทึมการคำนวณ เป็นการเลือกเส้นทางที่มีการค้นหา (การคำนวณล่วงหน้า) และเก็บไว้ในตารางเส้นทางตารางเส้นทางจะมีการปรับปรุง เมื่อจำเป็น เมื่อมีการร้องขอเส้นทางที่ใช้การได้หนึ่งเส้นทางจะถูกเลือกและมีการติดตั้งการเชื่อมต่อเส้นทาง เนื่องจากแต่ละการร้องขอมีข้อกำหนดของการไหลหรือลักษณะการจราจร และความต้องการการรับประกันคุณภาพของการบริการที่แตกต่างกัน อัลกอริทึมขึ้นอยู่กับความต้องการสามารถหาเส้นทางที่เหมาะสมได้อย่างถูกต้องกับความต้องการ แต่จะไม่เหมือนกับอัลกอริทึมคำนวณไว้ล่วงหน้า กล่าวคือจะมีการคำนวณทุกครั้งที่มีการร้องขอมาถึง ดังนั้นอัลกอริทึมคำนวณไว้ล่วงหน้าจะเร็วกว่าอัลกอริทึมที่ขึ้นอยู่กับความต้องการ

3.4 ตัวอย่างอัลกอริทึมสำหรับเลือกทางเดินของข้อมูล

3.4.1 หลักการเลือกเส้นทางที่เหมาะสมที่สุด

หลักการพื้นฐานของการเลือกเส้นทางที่เหมาะสมที่สุด (Optimality Principle) ที่ไม่ขึ้นอยู่กับโครงสร้างของระบบแบบใด กล่าวว่าถ้าโหนดหนึ่ง (จุด B) อยู่บนเส้นทางที่เหมาะสมที่สุด ระหว่างผู้ส่งข้อมูล (จุด A) และผู้รับข้อมูล (จุด C) แล้ว เส้นทางนั้นจะเป็นเส้นทางที่เหมาะสมที่สุดระหว่างโหนดนั้น (จุด B) กับผู้รับข้อมูล (จุด C) ด้วย ข้อความนี้สามารถพิสูจน์ได้โดยใช้หลักการตรรกศาสตร์ธรรมดา คือ สมมติให้ r1, r2 คือเส้นทางที่ดีที่สุดระหว่างจุด A และจุด C ถ้า r1 คือเส้นทางที่เหมาะสมที่สุดระหว่างจุด A กับจุด B แล้ว ให้ r2 เป็นเส้นทางระหว่างจุด B กับจุด C ถ้ามีเส้นทางอื่นที่เหมาะสมกว่าเส้นทาง r2 ก็ควรจะนำมาใช้เป็นเส้นทางที่ดีที่สุดระหว่างจุด A กับจุด C ซึ่งขัดกับข้อสมมติฐานที่ว่า r1, r2 คือเส้นทางที่ดีที่สุด ดังนั้นเส้นทาง r2 จึงต้องเป็นเส้นทางที่ดีที่สุดระหว่างจุด B และ จุด C ด้วย

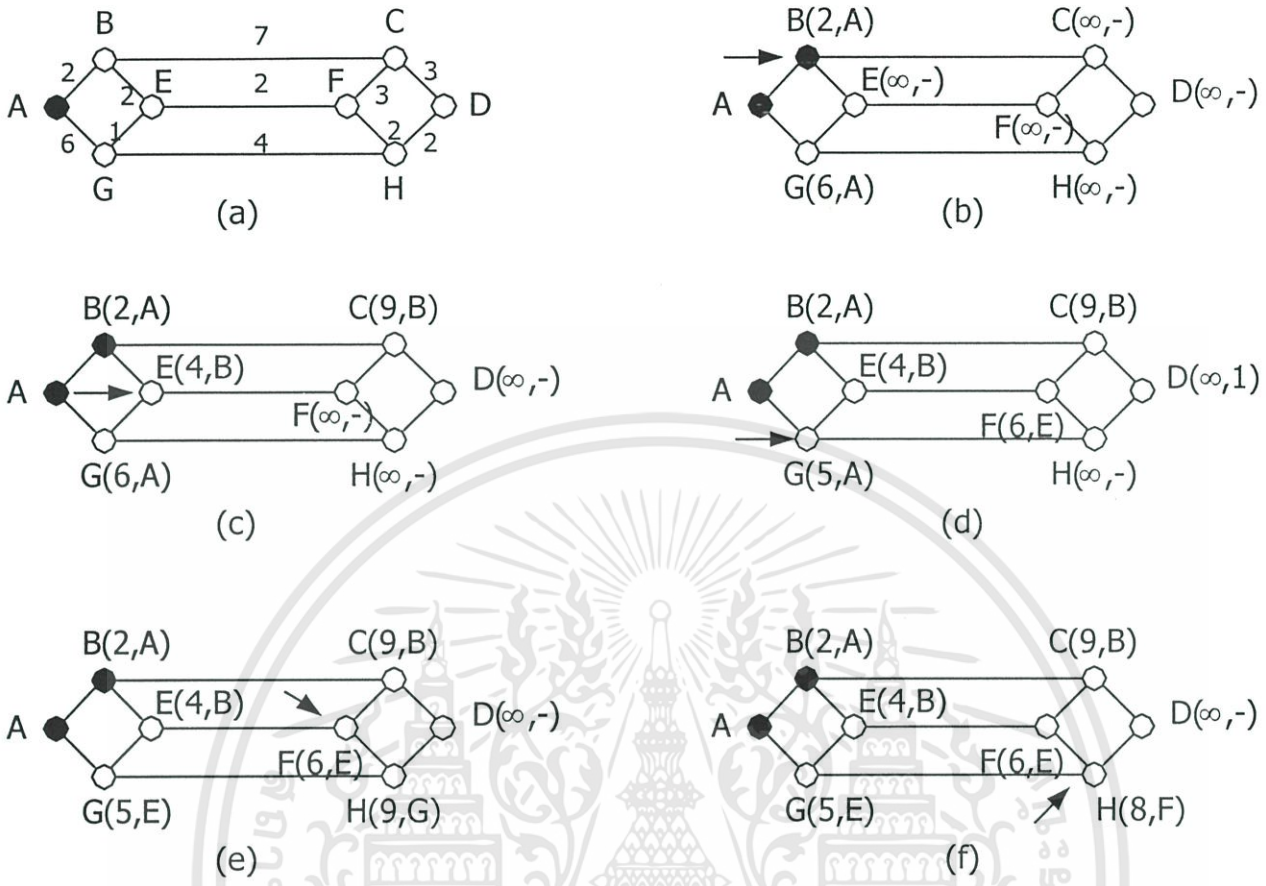
3.4.2 การเลือกเส้นทางที่สั้นที่สุด

การเลือกเส้นทางที่สั้นที่สุดเป็นวิธีการที่ถูกนำไปใช้มากที่สุดแบบหนึ่ง หลักการทำงานเริ่มต้นด้วยการสร้างรูปกราฟของระบบเครือข่ายย่อย โดยให้แต่ละโหนดในรูปกราฟแทนเราเตอร์แต่ละตัวในเครือข่าย และให้เส้นเชื่อมโยงโหนด แทนสายสื่อสารที่เชื่อมต่อระหว่างเราเตอร์ การเลือกเส้นทางเดินระหว่างเราเตอร์คู่หนึ่งทำได้โดยการค้นหาเส้นทางที่สั้นที่สุดในรูปกราฟ

นิยามของคำว่าเส้นทางที่สั้นที่สุดอาจมีได้หลายความหมาย เช่น การใช้จำนวนครั้งของการรับส่งข้อมูลหรือจำนวนเส้นเชื่อมโยงในรูปกราฟเป็นหลักพิจารณา เส้นทาง ABCF และ ABEF ในรูปที่ 3.2 จะถือว่ายาวเท่ากัน แต่ถ้าใช้ระยะทาง (เช่น เป็นกิโลเมตร) มาพิจารณาแล้ว เส้นทาง ABEF ย่อมสั้นกว่า

อีกวิธีการหนึ่งที่สามารถนำมาใช้แทนการนับจำนวนเส้นเชื่อมหรือการวัดระยะทาง คือการกำหนดให้ตัวเลขบนเส้นเชื่อมแต่ละเส้นเป็นตัวเลขที่บอกจำนวนแพ็กเกตที่รอการจัดส่ง และเวลารอคอยโดยเฉลี่ยที่คำนวณมาจากการรับ-ส่งแพ็กเกตมาตราฐาน ด้วยวิธีการนี้เส้นทางที่สั้นที่สุดจึงหมายถึงเส้นทางที่ส่งข้อมูลได้เร็วที่สุด

วิธีการที่นำมาใช้กับกรณีทั่วไปนั้น จะกำหนดให้ตัวเลขบนเส้นเชื่อมคือ ผลลัพธ์ที่ได้จากการคำนวณโดยมีระยะทาง ความเร็วในการส่งข้อมูล ปริมาณข้อมูลโดยเฉลี่ย ค่าใช้จ่าย ค่าเฉลี่ยมาตราฐานของจำนวนแพ็กเกตที่รอการจัดส่ง ระยะเวลารอคอย และอื่น ๆ เป็นตัวประกอบการเปลี่ยนค่าความสำคัญของตัวประกอบเหล่านี้ ทำให้แต่ละเราเตอร์สามารถใช้อัลกอริทึมเดียวกันในการคำนวณ แต่ให้ความสำคัญกับตัวประกอบไม่เหมือนกันได้



รูปที่ 3.2 ขั้นตอนในการคำนวณเส้นทางเดินที่สั้นที่สุด

3.4.2.1 อัลกอริทึมการเลือกเส้นทางวิธีของเบลแมนฟอร์ด (Bellman-Ford Routing)

อัลกอริทึมการเลือกเส้นทางวิธีของเบลแมนฟอร์ด [10] เป็นอัลกอริทึมในการหาเส้นทางที่สั้นที่สุดซึ่งใช้หลักการของการประมวลผลกราฟ เริ่มต้นด้วยการสร้างรูปกราฟของระบบเครือข่าย โดยให้แต่ละโหนดในรูปกราฟ แทนเราเตอร์ หรือสวิตซ์ แต่ละตัวในเครือข่าย และสายสื่อสารที่เชื่อมระหว่างโหนดก็จะแทนด้วยเส้นกราฟ นอกจากนั้นเส้นกราฟระหว่างโหนด จะถูกเลเบล (Label) ด้วยค่าน้ำหนัก ซึ่งเป็นค่าของระยะทาง อัตราการส่งข้อมูลของสายสื่อสาร ค่าใช้จ่ายของการสื่อสาร ความยาวของแถวรอคอยของข้อมูล ความล่าช้าของการส่งข้อมูลของสายสื่อสารเป็นต้น โดยการให้น้ำหนักแก่เส้นกราฟนี้ การหาเส้นทางที่สั้นที่สุดระหว่าง 2 โหนดในกราฟก็จะเป็นการหาเส้นทางที่สั้นที่สุดระหว่างโหนดทั้งสองในเครือข่าย

วิธีการของอัลกอริทึมนี้คือจะหาเส้นทางที่สั้นที่สุดจากโหนดต้นทางที่ถูกกำหนดขึ้นอยู่กับ การบีบบังคับเส้นทางซึ่งมีลิงค์มากที่สุดหนึ่งลิงค์ แล้วหาเส้นทางที่สั้นที่สุดกับการบีบบังคับของเส้น

ทางซึ่งมีลิงค์มากที่สุดสองลิงค์ และอื่น ๆ ทำนองนี้ อัลกอริทึมนี้ดำเนินการอย่างเป็นขั้นตอนด้วยการอธิบายอย่างเป็นทางการของอัลกอริทึมการเลือกเส้นทางที่สั้นที่สุดวิธีของเบลแมนฟอร์ด ได้ดังต่อไปนี้ ให้นิยาม

s คือโหนดเริ่มต้น

d_{ij} คือค่าใช้จ่ายของลิงค์จากโหนด i ถึงโหนด j; $d_{ii} = 0$; $d_{ij} = \infty$ ถ้าโหนด i และโหนด j ไม่ได้เชื่อมโยงกันโดยตรง ; $d_{ij} \geq 0$ ถ้าโหนด i และโหนด j เชื่อมโยงกันโดยตรง

h คือจำนวนมากที่สุดของลิงค์ในเส้นทางที่ขั้นตอนนั้นของอัลกอริทึม

$D_n^{(h)}$ คือค่าใช้จ่ายของเส้นทางที่มีค่าใช้จ่ายต่ำที่สุดจากโหนด s ถึงโหนด n ภายใต้การบังคับของการไม่มีลิงค์ที่มากกว่า h ลิงค์

อัลกอริทึมนี้มีขั้นตอนดังต่อไปนี้ ขั้นตอนที่สองจะทำซ้ำจนกระทั่งค่าใช้จ่าย จะไม่เปลี่ยนแปลง

ขั้นที่ 1 เริ่มต้น :

$D_n^{(0)} = \infty$ สำหรับ ทุก ๆ โหนด $n \neq s$

$D_s^{(h)} = 0$ สำหรับ ทุก ๆ h

ขั้นที่ 2 สำหรับแต่ละลำดับของ $h \geq 0$:

$$D_n^{(h+1)} = \min_j \left[D_j^{(h)} + d_{jn} \right]$$

เส้นทางจากโหนด s ถึงโหนด i สิ้นสุดลงกับลิงค์จาก j ถึง i

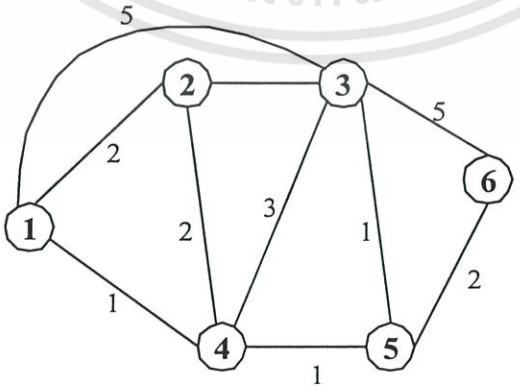
ตารางที่ 3.1 แสดงอัลกอริทึมของเบลแมนฟอร์ด เมื่อ $s = 1$

h	$D_2^{(h)}$	*Path*	$D_3^{(h)}$	*Path*	$D_4^{(h)}$	*Path*	$D_5^{(h)}$	*Path*	$D_6^{(h)}$	*Path*
0	∞	-	∞	-	∞	-	∞	-	∞	-
1	2	1-2	5	1-3	1	1-4	∞	-	∞	-
2	2	1-2	4	1-4-3	1	1-4	2	1-4-5	10	1-3-6
3	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
4	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6

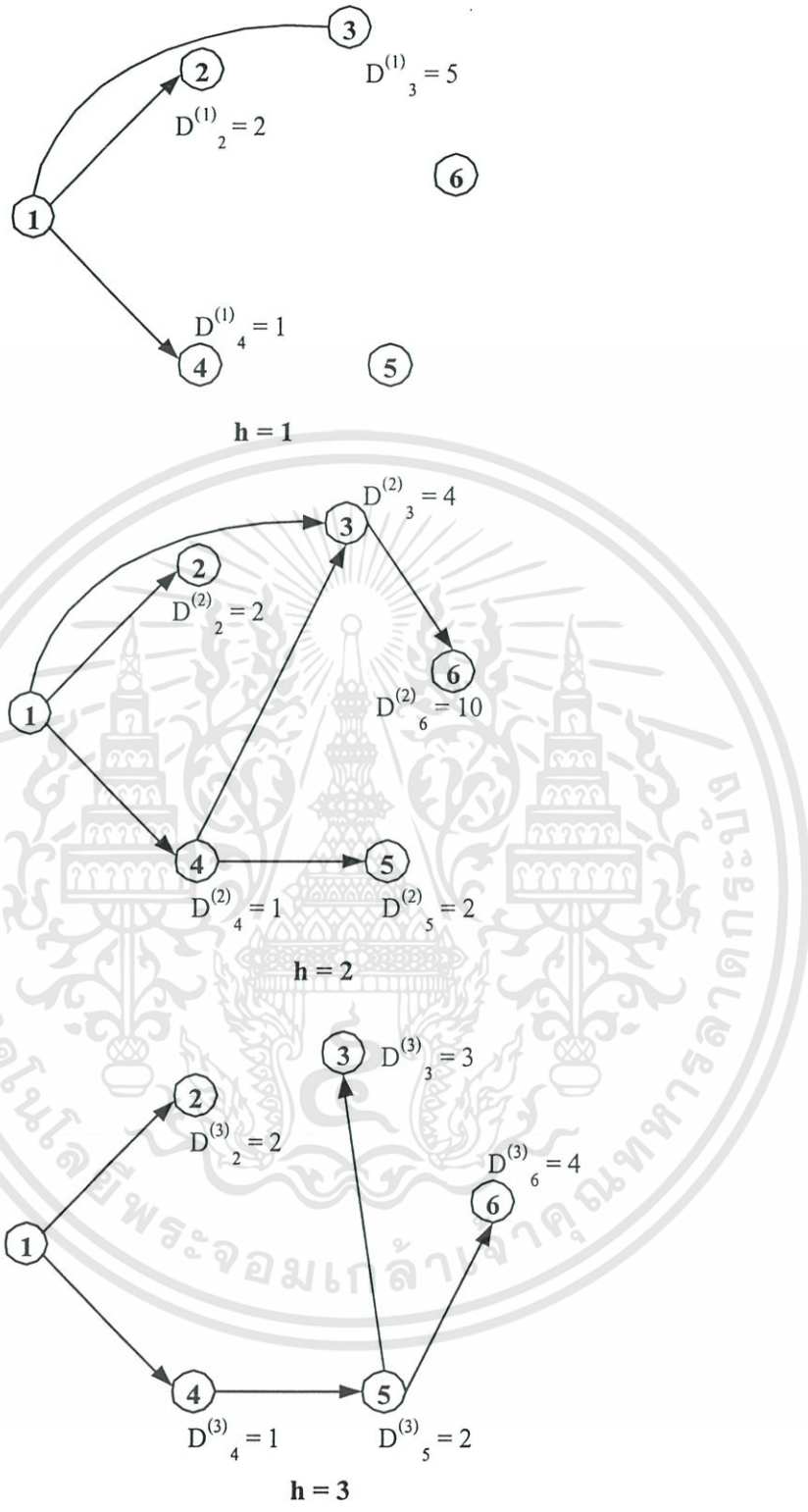
สำหรับอิเทอเรชั่น (Iteration) ของขั้นตอนที่สองเมื่อ $h = K$ และสำหรับแต่ละโหนดปลายทาง n อัลกอริทึมวิธีของเบลแมนฟอร์ด เปรียบเทียบเส้นทางทั้งหลายที่มีศักยภาพ (Potential Paths) จากโหนดต้นทาง s ถึงโหนด n ที่มีความยาว $K+1$ กับเส้นทางที่มีอยู่เส้นทางสุดท้ายของอิเทอเรชั่นก่อนหน้านี้ ถ้าเส้นทางก่อนหน้านี้เป็นเส้นทางที่สั้นที่สุดที่มีค่าใช้จ่ายที่ต่ำกว่า แล้วเส้นทางนั้นจะถูกรักษาไว้ ในทางตรงกันข้าม ถ้าเส้นทางใหม่ที่มีความยาว $K+1$ เป็นเส้นทางจากโหนดต้นทาง s ถึงโหนด n โดยที่เส้นทางนี้เกิดจากเส้นทางที่มีความยาว K จากโหนดต้นทาง s ถึงโหนดบางโหนด ในที่นี้สมมุติว่าคือโหนด j รวมกับฮอปที่เชื่อมโดยตรงถึงโหนด n ในกรณีนี้เส้นทางจากโหนดต้นทาง s ถึงโหนด j ซึ่งเป็นเส้นทางที่มีจำนวนฮอป K ฮอปในอิเทอเรชั่นก่อนหน้านี้

ตารางที่ 3.1 แสดงผลลัพธ์ของการใช้อัลกอริทึมของเบลแมนฟอร์ด โดยพิจารณาจากรูปที่ 3.3 และ 3.4 โดยให้โหนดต้นทาง $s = 1$ ที่แต่ละขั้นตอน เส้นทางที่มีค่าใช้จ่ายต่ำที่สุดจะมีจำนวนลิงค์มากที่สุดเท่ากับค่า h ที่ถูกพบ หลังอิเทอเรชั่นสุดท้าย เส้นทางที่มีค่าใช้จ่ายต่ำที่สุดถึงแต่ละโหนด และค่าใช้จ่ายที่ต่ำที่สุดของเส้นทางนั้นจะถูกพัฒนาด้วย การปฏิบัติการแบบเดียวกันนี้สามารถกับโหนดต้นทางโหนดที่สองได้เหมือนกัน และอื่น ๆ ทำนองนี้

ข่าวสารที่เก็บรวบรวมโดยอัลกอริทึมวิธีของเบลแมนฟอร์ด พิจารณาจาก ขั้นที่ 2 ของอัลกอริทึมจะพบว่าการคำนวณสำหรับโหนด n เกี่ยวพันกับความรู้ค่าใช้จ่ายของลิงค์ของโหนดข้างเคียงทุกโหนดถึงโหนด $n(d_{jn})$ รวมกับค่าใช้จ่ายทั้งหมดของเส้นทาง ถึงแต่ละโหนดข้างเคียงทั้งหลายเหล่านั้นจากโหนดต้นทางที่ถูกกำหนดไว้ $s\left(D_j^{(h)}\right)$ แต่ละโหนดสามารถเก็บรักษาเซ็ตของค่าใช้จ่าย และเส้นทางที่ร่วมกันสำหรับโหนดอื่น ๆ ในเครือข่าย และแลกเปลี่ยนข่าวสารนี้กับโหนดที่อยู่ข้างเคียงโดยตรงเป็นระยะ ๆ แต่ละโหนดสามารถเป็นดังนี้สมการ ขั้นที่ 2 ของอัลกอริทึมวิธีของเบลแมนฟอร์ด บนพื้นฐานของข่าวสารจากโหนดข้างเคียงของมัน และความรู้ค่าใช้จ่ายของลิงค์ของมันเองเท่านั้น เพื่อใช้ในการปรับปรุงค่าใช้จ่ายและเส้นทางของมัน



รูปที่ 3.3 แสดงเครือข่ายตัวอย่าง



รูปที่ 3.4 ตัวอย่างของการหาเส้นทางของอัลกอริทึมของเบลแมนฟอร์ด

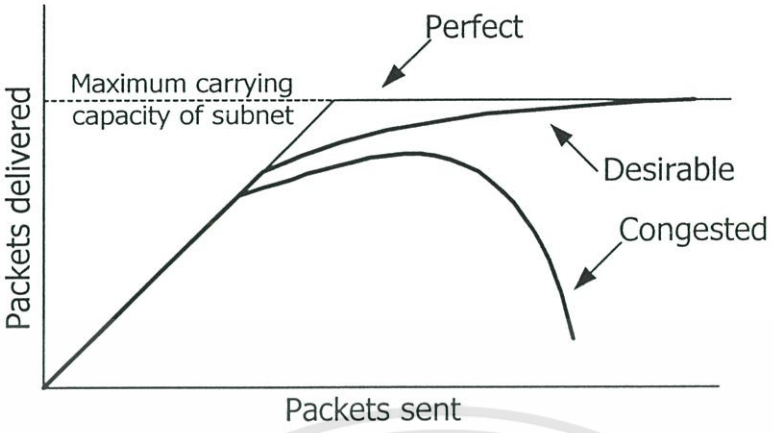
3.4.3 การเลือกเส้นทางแบบฟลัดดิ้ง

วิธีการเลือกเส้นทางข้อมูลแบบฟลัดดิ้ง (Flooding) เป็นวิธีการที่ไม่ปรับตัวเข้ากับสภาพแวดล้อมของระบบ โหนดจะส่งทุกแพ็กเกตที่รับเข้ามาออกไปทุกทิศทางที่มีการเชื่อมต่อกับโหนดตัวอื่นยกเว้นโหนดที่เป็นผู้ส่งเข้ามา วิธีการนี้จะเพิ่มปริมาณข้อมูลในเครือข่ายขึ้นมาอย่างมาก ซึ่งส่วนมากจะเป็นข้อมูลที่ซ้ำกัน ถ้าไม่ใช้กรรมวิธีอื่นเข้ามาช่วยแล้วจะทำให้เกิดข้อมูลจำนวนมหาศาลเลยทีเดียว หนทางหนึ่งที่นำมาใช้คือการใส่ตัวเลขเพื่อนับจำนวนโหนด เข้าไปในข้อมูลส่วนหัวของแต่ละแพ็กเกต ทุกครั้งที่โหนดรับแพ็กเกตเข้ามาก็จะตรวจสอบค่าตัวเลขนี้ ถ้าหากมีค่าเป็นศูนย์ก็จะลบแพ็กเกตทิ้ง มิฉะนั้นก็จะลดค่าจำนวนนับลงไปหนึ่งหน่วยก่อนที่จะส่งต่อไป เลขนับจำนวนโหนดนี้ควรจะมีค่าเท่ากับจำนวนโหนดที่จะต้องเดินทางผ่านจากผู้ส่งไปยังผู้รับ ถ้าผู้ส่งไม่ทราบจำนวนที่แน่นอนก็จะใช้ค่าสมมติ ซึ่งเป็นค่าสูงสุดของเครือข่ายนั้น ๆ แทน

วิธีการฟลัดดิ้งเหมาะกับงานบางประเภท เช่น ประเภทแรก นำไปใช้ในกิจการทางทหารซึ่งในยามสงครามโหนดแต่ละตัวอาจจะถูกข้าศึกทำลายได้ทุกเมื่อ ดังนั้นการที่มีสำเนาของข้อมูลค้างอยู่ในระบบเป็นจำนวนมาก จะทำให้มีความมั่นใจได้ว่าข่าวสารจะเดินทางไปถึงผู้รับ ประการต่อมาในระบบฐานข้อมูลแบบกระจายนั้น สามารถใช้วิธีการฟลัดดิ้งเข้ามาช่วยในการปรับปรุงข้อมูลในทุกฐานข้อมูลพร้อมกัน ประการที่สาม คือการใช้ฟลัดดิ้งเพื่อการเปรียบเทียบอัลกอริทึมแบบต่าง ๆ เนื่องจากวิธีการฟลัดดิ้งจะส่งข้อมูลออกไปทุกทิศทาง ทำให้สามารถหาเส้นทางที่สั้นที่สุดได้เสมอ ยิ่งกว่านั้นไม่มีอัลกอริทึมใดที่สามารถส่งข้อมูลโดยใช้เวลาในการรอคอยต่ำกว่านี้ได้

3.5 อัลกอริทึมควบคุมความคับคั่งของข้อมูล

ความคับคั่งของข้อมูล (Congestion) คือเหตุการณ์ที่เกิดขึ้นในบางส่วนของระบบเครือข่ายเมื่อมีจำนวนแพ็กเกตรอคอยการนำส่งมากเกินไป อันจะทำให้ประสิทธิภาพการทำงานของระบบลดลง กราฟในรูปที่ 3.5 แสดงให้เห็นถึงประสิทธิภาพการทำงานของระบบเมื่อเปรียบเทียบระหว่างจำนวนแพ็กเกตที่ถูกส่งเข้าระบบ (Packets Sent) กับจำนวนแพ็กเกตที่ส่งถึงตัวผู้รับแล้ว (Packets Delivered) ในภาวะปกติ โฮสต์ต่าง ๆ จะส่งแพ็กเกตเข้าไปในระบบ ซึ่งแพ็กเกตทั้งหมดก็จะถูกส่งไปถึงผู้รับได้ (ยกเว้นบางแพ็กเกตที่อาจเสียหาย หรือสูญหายระหว่างการนำส่ง) แสดงว่าจำนวนแพ็กเกตที่ถูกส่งนั้นเป็นอัตราส่วนที่พอเหมาะกับจำนวนแพ็กเกตที่ส่งถึงผู้รับ อย่างไรก็ตามเมื่ออัตราการส่งแพ็กเกตเข้าสู่ระบบเริ่มมีปริมาณมากขึ้นๆ โหนดเริ่มไม่สามารถจะจัดการส่งข้อมูลได้ทัน แพ็กเกตก็จะเกิดการสูญหายมากขึ้น และมีแนวโน้มที่จะแย่ลง ในที่สุดโหนดนั้นจะไม่สามารถจัดการนำส่งข้อมูลได้อีกต่อไป สมรรถนะในการทำงานของระบบก็จะพังลงอย่างราบคาบ และไม่มีแพ็กเกตส่งถึงผู้รับเลย



รูปที่ 3.5 แสดงเมื่อปริมาณข้อมูลในระบบเครือข่ายมีมากเกินไป จะเกิดความคับคั่งทำให้ประสิทธิภาพลดลงอย่างรวดเร็ว [10]

โดยปกติแล้วในระบบเครือข่ายความเร็วต่ำจะรอให้เกิดความคับคั่งขึ้นในระบบเสียก่อนแล้วจึงเริ่มแก้ไข โดยการแจ้งข่าวสารให้ผู้ส่งข้อมูลชะลอ หรือลดความเร็วในการส่งข้อมูลลง แนวทางนี้ไม่สามารถนำมาใช้ในเครือข่ายความเร็วสูงได้ เนื่องจากช่วงเวลาโหนดตรวจพบความคับคั่งส่งข่าวสารไปบอกให้ผู้ส่งข้อมูลนั้นเป็นระยะเวลานานมากพอที่จะทำให้แพ็กเกตอีกจำนวนมากถูกส่งเข้ามาในระบบ ซึ่งทำให้ระบบอยู่ในสถานะเลวร้ายลงไปอีกยิ่งระบบที่มีการให้บริการแบบที่จำกัดเวลาในการตอบสนองแอพพลิเคชั่นที่เป็นแบบเรียลไทม์ ให้แก่ผู้ใช้บริการ เช่น การส่งสัญญาณภาพเคลื่อนไหว การแจ้งให้ผู้ส่งข้อมูลประเภทนี้ทำการส่งข้อมูลช้าลง จึงเป็นไปไม่ได้

เครือข่ายความเร็วสูง จึงใช้แนวทางในการป้องกันไม่ให้เกิดความคับคั่งมากกว่าการแก้ไขภายหลัง การป้องกันเสียแต่เนิ่น ๆ จึงเป็นผลดีมากกว่า การควบคุมปริมาณข้อมูลในเครือข่าย เป็นแนวทางที่ได้ผลดีมาก เมื่อโหนดต้องการส่งข้อมูลเข้ามาในระบบจะต้องจัดตั้งวงจรเสมือนให้เสียก่อน ซึ่งในระหว่างการจัดตั้งรายละเอียดเกี่ยวกับข้อมูลที่โหนดจะส่งเข้ามาในระบบจะได้รับการตรวจสอบ ถ้าหากว่าระบบมีทรัพยากรและความสามารถเพียงพอที่จะใช้ในการรับ-ส่งข้อมูลของโหนด วงจรเสมือนที่ขอมาก็จะได้รับอนุญาตให้จัดตั้งขึ้นได้ มิฉะนั้นจะถูกปฏิเสธ

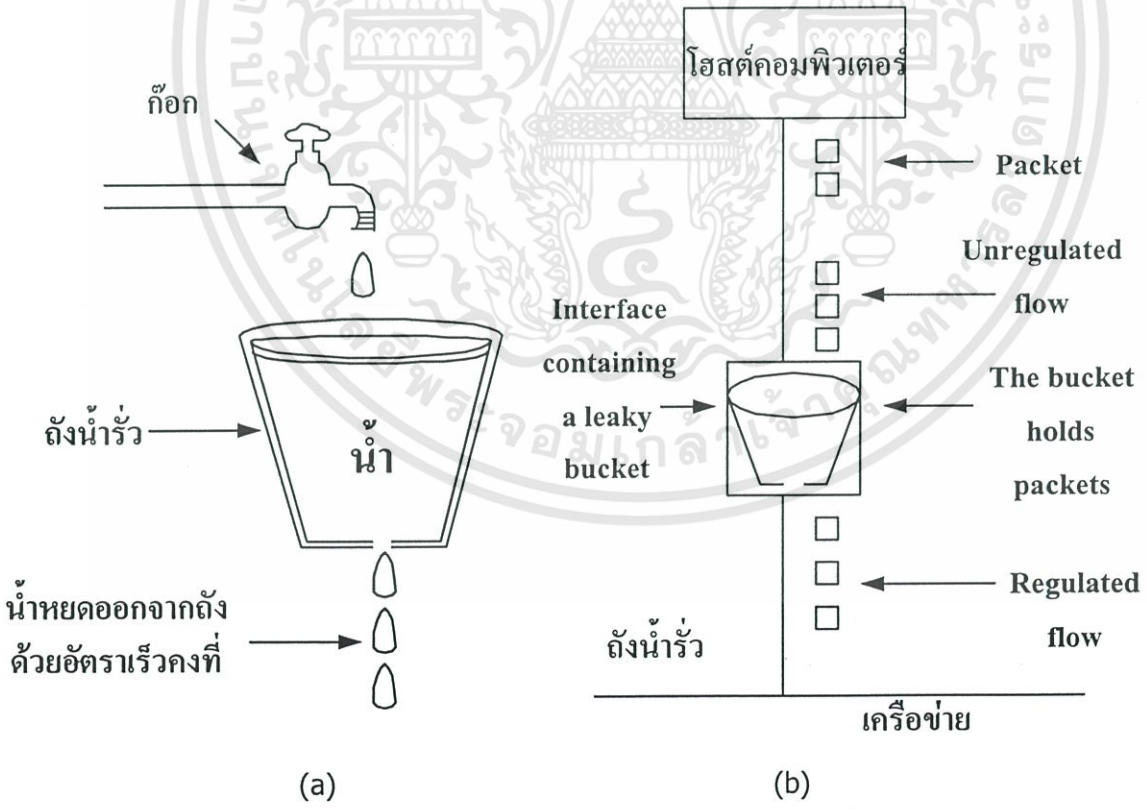
3.5.1 การควบคุมรูปแบบการจราจร

มูลเหตุหลักของการเกิดความคับคั่งของข้อมูลคือ ส่วนมากโหนดจะส่งข้อมูลเป็นช่วง ๆ ด้วยความเร็วสูงมากภายในระยะเวลาสั้น ๆ ที่เรียกว่า "เบิสท์ (Burst)" สลับกับการหยุดนิ่ง ถ้าโหนดสามารถส่งข้อมูลในอัตราเร็วสม่ำเสมอ โอกาสที่จะเกิดความคับคั่งของข้อมูลก็จะลดลง การบังคับให้แพ็กเกตถูกส่งออกมาในอัตราเร็วที่คาดเดาได้หรือคำนวณได้เป็นอีกวิธีหนึ่งของการแก้ปัญหาการคับคั่งของข้อมูล

3.5.1.1 อัลกอรทึมถังน้ำรั่ว

สมมติว่าให้ถังใส่น้ำใบหนึ่งซึ่งมีรูรั่วเล็ก ๆ ที่ก้นถังไปรองน้ำที่กำลังไหลอยู่ ดังแสดงในรูปที่ 3.6 (a) ไม่ว่าอัตราการไหลเข้าของน้ำจะเป็นเท่าใดก็ตามอัตราการไหลออกของน้ำจะมีค่าคงที่อยู่สองค่าเสมอ คือ หนึ่งเป็นค่าตัวเลขเมื่อมีน้ำอยู่ในถัง และสองมีค่าเป็นศูนย์เมื่อไม่มีน้ำอยู่ในถังเลย แต่ถ้ามีน้ำเต็มถังและก็ยังคงใส่น้ำลงไปอีก น้ำก็จะไหลล้นออกทางด้านข้างซึ่งถือว่าเป็นส่วนที่สูญเสีย (ไม่นำมาพิจารณา)

การจัดส่งแพ็กเกตข้อมูลก็มีสภาพคล้ายกัน ดังแสดงในรูปที่ 3.6 (b) หลักการมีอยู่ว่าโหนดแต่ละตัวจะเชื่อมต่อกับระบบเครือข่ายผ่านตัวกลาง (เราเตอร์) ที่ทำงานแบบถังน้ำรั่ว ซึ่งก็คือการใช้แถวคอยที่มีขนาดจำกัด ถ้าแพ็กเกตมาถึงแถวคอยในขณะที่แถวคอยเต็ม แพ็กเกตนั้นก็จะถูกลบทิ้งไปเหมือนกับน้ำที่ไหลล้นออกจากถังที่มีน้ำอยู่เต็ม หรือกล่าวอีกอย่างว่าถ้าโพรเซสบางตัวหรือหลาย ๆ ตัวของโหนดพยายามที่จะส่งแพ็กเกตใหม่เข้ามา ในขณะที่แถวคอยเต็มแล้ว แพ็กเกตตัวที่มาถึงทีหลังก็จะถูกลบทิ้งไป ขบวนการทำงานนี้อาจจะกำหนดให้ตัวอุปกรณ์การสื่อสาร (Communication Devices) เป็นผู้จัดการ หรือจะให้ระบบปฏิบัติการของโหนดทำการจำลองการ



รูปที่ 3.6 (a) ถังน้ำรั่วที่มีน้ำอยู่ภายใน
(b) อัลกอริทึมถังน้ำรั่ว

ทำงานนี้ก็ได้ Turner เป็นผู้นำเสนออัลกอริทึมนี้ในปี ค.ศ. 1986 โดยเรียกว่า “อัลกอริทึมถังน้ำรั่ว (Leaky Bucket Algorithm)” ซึ่งในเนื้อแท้แล้วกระบวนการนี้ก็คือ ระบบแถวคอยของผู้ให้บริการเดี่ยวที่มีระยะเวลาการให้บริการคงที่ (Single Server Queuing System with Constant Service Time) นั่นเอง

โหนดแต่ละตัวได้รับอนุญาตให้ส่งแพ็กเกตเข้าไปในเครือข่ายได้ 1 ตัวต่อ 1 จังหวะสัญญาณนาฬิกา (Clock Tick ของเครื่องคอมพิวเตอร์) ซึ่งจะถูกควบคุมโดยตัวอุปกรณ์สื่อสาร หรือระบบปฏิบัติการของโหนดเองก็ได้กลไกนี้จะทำการปรับปริมาณการไหลของแพ็กเกตข้อมูลของผู้ใช้งานทางฝั่งของโหนดที่มีอัตราความเร็วไม่แน่นอนให้กลายเป็นการไหลของแพ็กเกตเข้าสู่ระบบเครือข่ายที่มีอัตราความเร็วสม่ำเสมอ จึงเป็นการลดโอกาสที่จะทำให้เกิดความคับคั่งของข้อมูลลงได้

ในกรณีที่แพ็กเกตทั้งหมดมีขนาดเท่า ๆ กัน อัลกอริทึมนี้จะถูกนำมาใช้ได้ตามขั้นตอนที่กล่าวข้างต้น อย่างไรก็ตาม ถ้าแพ็กเกตที่มีขนาดไม่เท่ากันถูกส่งเข้ามาในระบบ วิธีการนับจำนวนแพ็กเกตจะเปลี่ยนไปเป็นการนับปริมาณข้อมูลแทน เช่นกำหนดให้ส่งข้อมูล 1024 ไบต์ต่อหนึ่งสัญญาณนาฬิกา ดังนั้นในหนึ่งสัญญาณนาฬิกาจะส่งแพ็กเกตขนาด 1024 ไบต์ได้ 1 ตัว หรือส่งแพ็กเกตขนาด 512 ไบต์ได้ 2 ตัวหรือส่งแพ็กเกตขนาด 256 ไบต์ได้ 4 ตัวเป็นต้น แพ็กเกตที่มีขนาดใหญ่กว่าจำนวนไบต์ที่เหลืออยู่จะถูกส่งในรอบสัญญาณนาฬิกาครั้งต่อไป

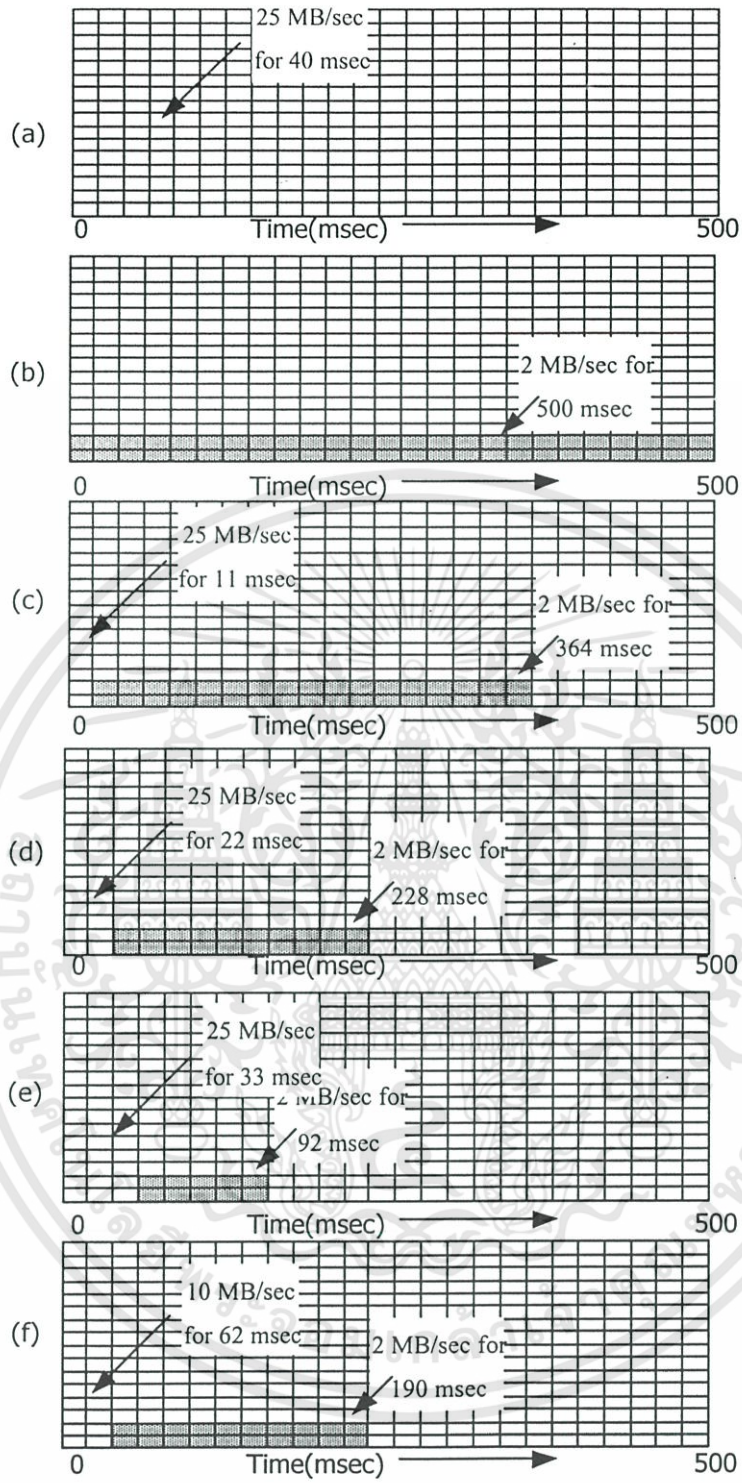
อัลกอริทึมถังน้ำรั่วประกอบด้วยแถวคอยขนาดจำกัดแถวหนึ่ง เมื่อแพ็กเกตเดินทางมาถึงถ้ามีที่ว่างในแถวคอย แพ็กเกตก็จะถูกนำมาเก็บไว้ในที่ว่าง ถ้าไม่มีที่ว่างก็จะทิ้งแพ็กเกตนั้นไป ทุก ๆ จังหวะสัญญาณนาฬิกาจะมีแพ็กเกตถูกส่งเข้าไปในระบบ 1 ตัว อัลกอริทึมที่ใช้การวัดปริมาณข้อมูลก็ใช้หลักการทำงานแนวเดียวกัน ในทุก ๆ สัญญาณนาฬิกาตัวนับปริมาณข้อมูล (Counter) จะเริ่มต้นกำหนดค่าไว้จำนวนหนึ่ง ถ้าแพ็กเกตแรกเข้ามาในแถวคอยมีปริมาณข้อมูลน้อยกว่าค่าที่ตั้งไว้แพ็กเกตนั้นจะถูกส่งออกไป และตัวนับปริมาณข้อมูลจะทำการลดค่าลงไปเท่ากับจำนวนไบต์ของแพ็กเกตที่เพิ่งส่งออกไป แพ็กเกตตัวอื่นอาจถูกส่งออกไปอีกก็ได้ตราบเท่าที่ขนาดของแพ็กเกตเล็กกว่าหรือเท่ากับค่าของตัวนับปริมาณข้อมูลที่เหลืออยู่ มิฉะนั้นแล้วโหนดจะหยุดส่งข้อมูลเพื่อรอสัญญาณนาฬิกาครั้งต่อไป ซึ่งจะกำหนดค่าตัวนับปริมาณข้อมูลใหม่โดยไม่สนใจว่าค่าเดิมจะเป็นอะไรก็ตาม

สมมุติให้คอมพิวเตอร์เครื่องหนึ่งสามารถส่งข้อมูลได้ด้วยความเร็ว 25 ล้านไบต์ต่อวินาที (เท่ากับ 200 ล้านบิตต่อวินาที) ซึ่งสายสื่อสารในระบบก็สามารถสนับสนุนการส่งข้อมูลที่ความเร็วนี้ได้ อย่างไรก็ตาม โหนดสามารถทำงานได้ที่ความเร็วขนาดนี้เพียงชั่วระยะเวลาสั้น ๆ ถ้าเป็นการส่งข้อมูลในช่วงระยะเวลายาวนานต่อเนื่องแล้ว โหนดสามารถทำงานได้ที่ความเร็ว 2 ล้านไบต์ต่อวินาที สมมุติว่าข้อมูลถูกส่งเข้ามา 1 ล้านไบต์ในเวลา 40 มิลลิวินาที (เทียบเท่ากับความเร็ว 25 ล้านไบต์ต่อวินาที) โหนดสามารถลดอัตราความเร็วในการส่งข้อมูลให้ลงมาอยู่ที่ 2 ล้านไบต์ต่อวินาทีได้โดยการรับข้อมูลทั้งหมดไว้ในตัวเองพร้อม ๆ กับการปล่อยข้อมูลเข้าไปในระบบที่ความเร็ว 2

ล้านไบต์ต่อวินาที เป็นเวลา 500 มิลลิวินาที ในรูปที่ 3.7 (a) แสดงการไหลของข้อมูลเข้ามายังโหนดที่ทำงานด้วยความเร็ว 25 ล้านไบต์ต่อวินาทีเป็นเวลา 40 มิลลิวินาที รูปที่ 3.7 (b) โหนดส่งข้อมูลเข้าสู่ระบบด้วยความเร็ว 2 ล้านไบต์ต่อวินาทีเป็นระยะเวลา 500 มิลลิวินาที

3.5.1.2 อัลกอริทึมโทคเก้นบัคเก็ต

อัลกอริทึมถังน้ำรั่วจัดการส่งข้อมูลเข้าสู่ระบบด้วยอัตราเร็วคงที่ไม่ว่าข้อมูลจะถูกส่งออกจากโหนดต้นทางด้วยความเร็วเท่าใดก็ตาม สำหรับโปรแกรมประยุกต์บางประเภท การยอมให้ส่งข้อมูลที่ความเร็วสูงกว่าปกติในช่วงเวลาสั้น ๆ เพื่อให้สอดคล้องสภาพที่ข้อมูลส่งออกจากโหนดต้นทางเป็นสิ่งที่ทำได้ อัลกอริทึมที่นำมาใช้จึงต้องปรับปรุงให้มีความอ่อนตัวมากขึ้น กลายเป็นวิธีที่เรียกว่า "อัลกอริทึมโทเก้นบัคเก็ต (Token Bucket Algorithm)" ซึ่งได้เพิ่มข้อกำหนดให้ถังน้ำ หรือบัคเก็ตต้องเก็บรักษาโทคเก้น (ซึ่งทำหน้าที่เหมือนใบกำกับสินค้า) ที่สร้างขึ้นตามสัญญาณนาฬิกา เช่น เพิ่มโทคเก้นหนึ่งตัวทุก ๆ 500 มิลลิวินาที



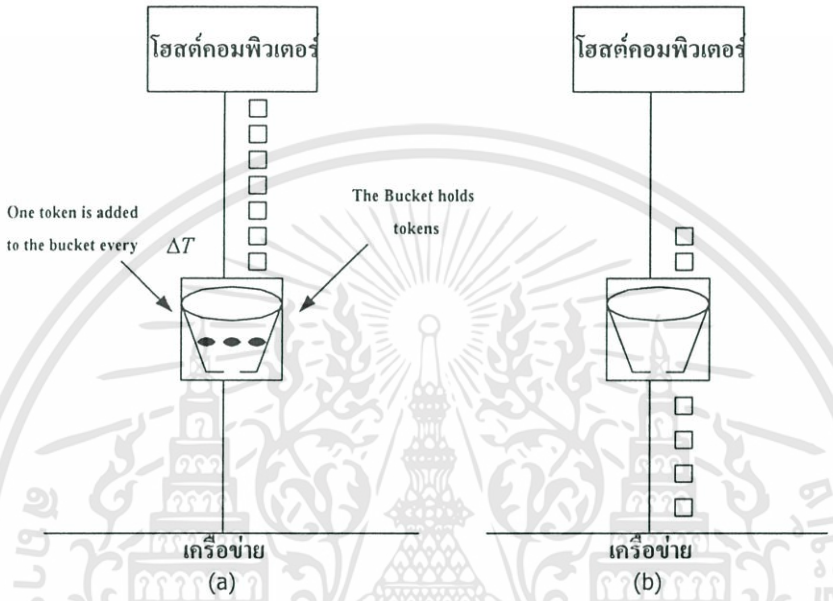
รูปที่ 3.7 (a) ข้อมูลที่ถูกส่งมา

(b) ข้อมูลที่ส่งออกมา

(c) ถึง (e) ข้อมูลที่ถูกส่งออกด้วยความเร็ว 250 KB, 500 KB และ 750KB ตามลำดับ

(f) ข้อมูลที่ถูกส่งออกจากบัคเก็ตขนาด 500 KB ด้วยความเร็ว 10 ล้านไบต์ต่อวินาที

ในรูป 3.8 (a) ถังน้ำ หรือบัคเก็ตมีโทคเก้นอยู่ 3 ตัว โดยมีแพ็กเกตรอการนำส่งอยู่ 5 ตัว ทุกแพ็กเกตที่ถูกส่งออกไปจะต้องใช้โทคเก้น หนึ่งตัว ดังนั้นจึงสามารถส่งแพ็กเกตได้ 3 ตัวทันที ที่เหลืออีก 2 ตัว จะต้องรอจนกว่าโทคเก้นใหม่จะเกิดขึ้น ถ้าแพ็กเกตข้อมูลเป็นสินค้า การส่งสินค้าก็จะต้องมีใบกำกับสินค้าติดไปด้วยทุกครั้งแต่โทคเก้นในที่ไม่ได้นำไปใช้ประโยชน์อย่างอื่นใดเลย การใช้โทคเก้นในทางปฏิบัติจึงเป็นการลบโทคเก้นหนึ่งตัวทิ้งต่อการส่งข้อมูลหนึ่งแพ็กเกตเท่านั้น



รูปที่ 3.8 อัลกอริทึม Token Bucket (a) ก่อนส่งข้อมูล
(b) หลังจากส่งข้อมูล

อัลกอริทึมโทคเก้นบัคเก็ตมีวิธีจัดการนำส่งแพ็กเกตข้อมูลที่เข้ามาแบบเบิสท์ ต่างไปจากวิธีเดิม อัลกอริทึมถังน้ำรั่วจะไม่ยอมให้โหนดมีการเก็บสะสมเวลาที่ไม่ได้ใช้งาน (Idle Time) เพื่อใช้สำหรับการส่งแพ็กเกตจำนวนมากภายหลัง แต่โทคเก้นบัคเก็ตยอมให้ยอมให้โทคเก้นได้ถึงขีดจำกัดอันหนึ่งที่กำหนดไว้ล่วงหน้า หมายความว่าโทคเก้นบัคเก็ตสามารถที่จะส่งแพ็กเกตติดต่อกันในทันทีทันใดได้เท่าจำนวนสูงสุดของโทคเก้นที่มีอยู่ในบัคเก็ต ทำให้รองรับการส่งข้อมูลที่มีความเร็วสูงมากได้เป็นบางครั้ง ซึ่งจะมีผลตอบสนองโดยตรงต่อข้อมูลที่เข้ามาแบบเบิสท์ ได้เป็นอย่างดี ข้อแตกต่างอีกประการหนึ่ง คือโทคเก้นบัคเก็ตไม่มีขีดจำกัดของแถวคอยเข้ามาเกี่ยวข้อง จึงไม่มีการลบแพ็กเกตของข้อมูลทิ้ง

ระบบเครือข่ายย่อยที่ใช้วิธีกำหนดปริมาณข้อมูลในการส่งแทนการนับจำนวนแพ็กเกต โทคเก้นแต่ละตัวจะใช้แทนจำนวนไบต์ของข้อมูล แพ็กเกตจะถูกนำส่งได้ถ้าจำนวนไบต์ในโทคเก้น

ที่มีอยู่ มีขนาดไม่น้อยไปกว่าจำนวนไบต์ของแพ็กเกตที่ต้องการส่ง เศษที่เหลืออยู่ของโทคเก้นสามารถนำไปรวมกับโทคเก้นอื่นได้

ทั้งวิธีถังน้ำรั่ว และวิธีโทคเก้นบัคเก็ต สามารถนำมาใช้ในการช่วยการจราจรระหว่างโหนดทั้งสองตัวให้ราบรื่นขึ้น และยังช่วยจัดการส่งข้อมูลของโหนดได้ อย่างไรก็ตามโทคเก้นบัคเก็ตสามารถบังคับให้โหนดหยุดส่งข้อมูลในขณะที่ข้อมูลจากแหล่งกำเหนิดยังคงถูกส่งเข้ามาอย่างไม่ขาดสาย อันจะทำให้ข้อมูลสูญหายได้อัลกอริทึม

รูปที่ 3.7 (c) สมมุติว่าเป็นโทคเก้นบัคเก็ตที่มีความจุของบัคเก็ต 250 กิโลไบต์ โทคเก้นถูกสร้างขึ้นในอัตราที่ทำให้สามารถส่งข้อมูลได้ 2 ล้านไบต์ต่อวินาที ถ้าบัคเก็ตมีโทคเก้นสะสมอยู่เต็มในขณะที่ข้อมูลขนาด 1 ล้านไบต์ถูกส่งเข้ามาแบบเบิสท์ (Burst-in) โหนดจะสามารถส่งข้อมูลเข้าสู่ระบบได้ที่ความเร็วสูงสุด25 ล้านไบต์ต่อวินาทีเป็นเวลา 11 มิลลิวินาที (Burst-out) จากนั้น เมื่อใช้โทคเก้นสะสมหมดแล้ว จึงส่งข้อมูลที่ระดับความเร็วปกติ

การคำนวณหาระยะเบิสท์ขาออก (Burst-out) ที่โหนดสามารถส่งข้อมูลด้วยอัตราความเร็วสูงสุดเป็นเรื่องซับซ้อนเล็กน้อย เนื่องจากในขณะที่ส่งข้อมูลอยู่นั้น โทเก้นใหม่ก็จะถูกสร้างขึ้นมาตลอดเวลา ถ้ากำหนดให้ระยะเวลาเบิสท์เป็น S วินาที, โทคเก้นถูกนำมาใช้ในการส่งข้อมูลขนาด C ไบต์, อัตราการสร้างโทคเก้นซึ่งทำให้เกิดเป็นอัตราการส่งข้อมูลปกติเป็น p ไบต์ต่อวินาที, และความเร็วสูงสุดในการส่งข้อมูลเป็น M ไบต์ต่อวินาทีแล้ว ปริมาณการส่งข้อมูลออกในช่วงเบิสท์จะเท่ากับ $C + pS$ ไบต์ ซึ่งจะต้องเท่ากับปริมาณข้อมูลที่ถูกส่งด้วยความเร็วสูงสุดในระยะเวลาเดียวกันนั่นคือ MS ดังนั้นจะได้ความสัมพันธ์ ดังสมการที่ (3.1) และ (3.2)

$$C + pS = MS \tag{3.1}$$

นั่นคือ

$$S = \frac{C}{(M - p)} \tag{3.2}$$

จากตัวอย่างในรูปที่ 3.7 (c) $C = 250Kbyte$, $M = 25Mbyte\ /\ Sec.$, $p = 2Mbyte\ /\ Sec$. เมื่อนำไปแทนค่าในสมการข้างบน จะได้ค่าระยะเวลาเบิสท์ประมาณ 11 มิลลิวินาที รูปที่ 3.7 (d) และ (e) แสดงตัวอย่างบัคเก็ตขนาด 500 กิโลไบต์ และ 750 กิโลไบต์ตามลำดับ

ปัญหาหลักของวิธีการโทคเก้นบัคเก็ตก็คือ การที่วิธีการนี้สนับสนุนการส่งข้อมูลแบบเบิสท์ แม้ว่าจะสามารถควบคุมปริมาณและความถี่ได้ด้วยการกำหนดค่าอัตราการสร้างโทคเก้น และความเร็วสูงสุดในการส่งข้อมูลแล้วก็ตาม ระบบทั่วไปก็พยายามจะจำกัดจำนวนครั้ง และความถี่ของการส่งข้อมูลที่ความเร็วสูงสุด ในขณะที่ไม่ต้องการส่งข้อมูลที่ความเร็วต่ำเสมอไป ดังที่เกิดขึ้นในอัลกอริทึมแบบถังน้ำรั่ว

วิธีการแก้ไขปัญหาอีกวิธีหนึ่งคือ การใช้วิธีถังน้ำรั่วเป็นตัวรับข้อมูลที่ส่งมาด้วยวิธีโทคเก้นบัคเก็ต ความเร็วในการส่งข้อมูลของวิธีถังน้ำรั่วควรจะสูงกว่าอัตราการสร้างโทคเก้น (p) แต่จะต้องต่ำกว่าความเร็วสูงสุดในการส่งข้อมูลของเครือข่าย รูปที่ 3.7 (f) แสดงตัวอย่างของการใช้โทคเก้นบัคเก็ตขนาด 500 กิโลไบต์ ตามด้วยวิธีถังน้ำรั่วที่มีความเร็ว 10 ล้านไบต์ต่อวินาที

3.5.2 ข้อกำหนดการไหลของข้อมูล

การส่งผ่านข้อมูลจะมีประสิทธิภาพสูงสุดเมื่อ ผู้ส่ง ผู้รับ และเครือข่ายย่อยสามารถทำความตกลงกันได้ เพื่อที่จะทำความตกลงกันอย่างรอบคอบในรายละเอียดของรูปแบบการจราจรของข้อมูล ซึ่งเรียกว่า การจัดทำข้อกำหนดของการไหลของข้อมูล (Flow Specification) [11] ประกอบด้วยโครงสร้างข้อมูลที่ใช้ในการส่งข้อมูลเข้าสู่ระบบ และ QoS ที่โปรแกรมประยุกต์ต้องการ ข้อกำหนดการไหลของข้อมูลสามารถนำไปใช้ได้กับระบบเครือข่ายที่ส่งแพ็กเกตผ่านวงจรเสมือนหรือใช้ดาต้าแกรม หรือแม้กระทั่งการส่งแบบหลายเป้าหมายก็ได้

ตารางที่ 3.2 ข้อกำหนดการไหลของข้อมูล

คุณลักษณะของข้อมูล	บริการที่ต้องการให้รับประกัน
ขนาดแพ็กเกตสูงสุด (ไบต์)	อัตราการสูญหายของข้อมูล (ไบต์)
อัตราความเร็วของโทเก้นบัคเก็ต	ช่วงเวลาที่สูญเสียข้อมูล (ไมโครวินาที)
ขนาดของโทคเก้นบัคเก็ต (ไบต์)	อัตราการสูญหายของข้อมูลในช่วง Burst (แพ็กเกต)
ความเร็วสูงสุดในการส่งข้อมูล (ไบต์/วินาที)	ช่วงการรอคอยที่ต่ำที่สุด (ไมโครวินาที) ความแปรปรวนของช่วงการรอคอยที่สูงที่สุด (ไมโครวินาที) ระดับการรับประกัน

ในส่วนถัดไปจะเป็นการอธิบายถึงข้อกำหนดการไหลของข้อมูลที่ออกแบบโดย แพททริก (Partridge 1992) ดังแสดงในตารางที่ 3.2 แนวความคิดหลักคือ ก่อนที่จะมีการจัดตั้งวงจรเสมือนหรือก่อนที่จะเริ่มส่งชุดดาต้าแกรม ผู้ส่งข้อมูลจะส่งข้อกำหนดการไหลของข้อมูลไปให้กับเครือข่ายย่อยเพื่อให้พิจารณา เครือข่ายย่อยอาจจะยอมรับหรือปฏิเสธก็ได้ หรืออาจจะแก้ไขเพียงบางส่วนแล้วส่งกลับมาให้ผู้ส่ง เมื่อทั้งสองฝ่ายตกลงกันได้แล้ว ผู้ส่งจึงจะเริ่มต่อรองกับผู้รับข้อมูล ในที่สุดทั้งสามฝ่ายก็จะได้ข้อกำหนดการไหลของข้อมูลที่ยอมรับได้ การสื่อสารจึงจะเริ่มต้นขึ้น

จากข้อกำหนดการไหลของข้อมูลในตารางที่ 3.2 เริ่มจากคอลัมน์ทางซ้าย " ขนาดแพ็กเกตสูงสุด (Maximum Packet Size)" หมายถึงขนาดของแพ็กเกตที่ใหญ่ที่สุดที่สามารถนำมาใช้งานได้ คุณลักษณะสองข้อถัดมาใช้สมมุติฐานว่าเป็นการส่งข้อมูลโดยใช้อัลกอริทึมโทคเก้นบัคเก็ต ซึ่งบอกอัตราการเกิดของโทคเก้นบัคเก็ต (Token Bucket Rate) มีหน่วยเป็นไบต์ต่อวินาที ซึ่งหมายถึงอัตราการส่งข้อมูลเข้ามาในระบบ ตามด้วยขนาดของบัคเก็ต (Token Bucket Size) ถ้าให้อัตราการเกิดของโทคเก้นเป็น r ไบต์ต่อวินาที และขนาดของบัคเก็ตเป็น b ไบต์แล้ว ในช่วงเวลา Δt ใด ๆ ปริมาณข้อมูลที่สามารถส่งได้สูงสุดเท่ากับ $b + r\Delta t$ ในที่นี้ b จะหมายถึงปริมาณข้อมูลที่บรรจุอยู่ในบัคเก็ตเมื่อเริ่มต้นการส่งข้อมูล และ $r\Delta t$ หมายถึงปริมาณข้อมูลที่เกิดการถ่ายทอดในช่วงเวลาต่อมาจนจบช่วงเวลานั้น ๆ "ความเร็วสูงสุดในการส่งข้อมูล (Maximum Transmission Rate)" คือความเร็วในการส่งข้อมูลในระดับสูงสุดที่โฮสต์จะสามารถทำได้ภายใต้สภาวะทั่วไป ซึ่งสามารถนำไปคำนวณหาเวลาที่ต้องใช้น้อยที่สุดในการส่งข้อมูลทั้งหมดที่มีอยู่ในบัคเก็ตได้

ส่วนคอลัมน์ทางขวามือนั้น คือบริการที่โปรแกรมประยุกต์ต้องการการรับประกันจากเครือข่ายย่อยสองบรรทัดแรก คือค่าตัวแปรที่ใช้คำนวณหาค่าการสูญเสียของแพ็กเกตที่สามารถยอมรับได้ เช่น มีอัตราการสูญหายของข้อมูล (Loss Sensitivity) เป็น 1 ไบต์ต่อชั่วโมง หรือช่วงเวลาที่สูญเสียข้อมูล (Loss Interval) เป็น 10 ไมโครวินาที เป็นต้น อัตราการสูญหายของข้อมูลในช่วงเบิสท์ (Burst Loss Sensitivity) บอกจำนวนแพ็กเกตที่สูญหายติดต่อกันในช่วงเบิสท์ที่ยอมรับได้ (ซึ่งจะต้องมีการแก้ไข และจะยังคงทำการส่งข้อมูลต่อไป)

สองบรรทัดต่อมาอธิบายเกี่ยวกับเวลาในการรอคอย "ช่วงการรอคอยที่ต่ำที่สุด (Minimum Delay Noticed)" หมายถึงช่วงเวลาการรอคอยของแพ็กเกตที่เกิดขึ้นระหว่างการเดินทางในเครือข่าย แต่โปรแกรมประยุกต์จะไม่ถือว่าช่วงเวลานี้เป็นช่วงเวลาที่เสียไปกับการรอคอย เช่น การคัดลอกสำเนาแฟ้มข้อมูลอาจกำหนดให้มีเวลารอคอยที่ต่ำที่สุดเป็น 1 วินาที หากข้อมูลใช้เวลาเดินทางภายใน 1 วินาที ก็ยังถือเสมือนว่าไม่มีปัญหาใด ๆ เกิดขึ้น แต่โปรแกรมประยุกต์บางประเภท เช่น การส่งสัญญาณเสียง อาจกำหนดให้มีเวลารอคอยที่ต่ำที่สุดเพียง 3 มิลลิวินาทีก็ได้ ดังนั้นผู้รับข้อมูลจะต้องเตรียมการแก้ไข ถ้าข้อมูลขาดช่วงไปนานเกินกว่า 3 มิลิวินาที "ความแปรปรวนของช่วงการรอคอยที่สูงที่สุด (Maximum Delay Variation)" หมายถึงช่วงเวลาที่แตกต่างกันของเวลารอคอย เช่น ถ้าให้เวลารอคอยแพ็กเกตต่ำสุดเป็น 5 วินาที และสูงสุดเป็น 8 วินาที ค่าของความแปรปรวนของช่วงการรอคอยที่สูงที่สุด คือ 3 วินาที

ส่วน "ระดับการรับประกัน (Quality of Guarantee)" บอกระดับความต้องการบริการต่าง ๆ ของโปรแกรมประยุกต์ ตัวอย่างเช่น อัตราการสูญหายของข้อมูล และช่วงการรอคอยที่ต่ำที่สุดของข้อมูลอาจเป็นวัตถุประสงค์ที่ต้องการ แต่โปรแกรมประยุกต์บางส่วนก็ไม่ได้ให้ความสนใจมากนักถ้าเรื่องดังกล่าวไม่เป็นไปตามที่ตกลงไว้ ในขณะที่โปรแกรมประยุกต์อีกบางส่วนอาจจะต้องเลิก

การทำงานไปในทันทีที่ความต้องการดังกล่าวไม่ได้รับการตอบสนอง โปรแกรมอื่น ๆ ก็อาจมีพฤติกรรมที่แตกต่างกันออกไปได้

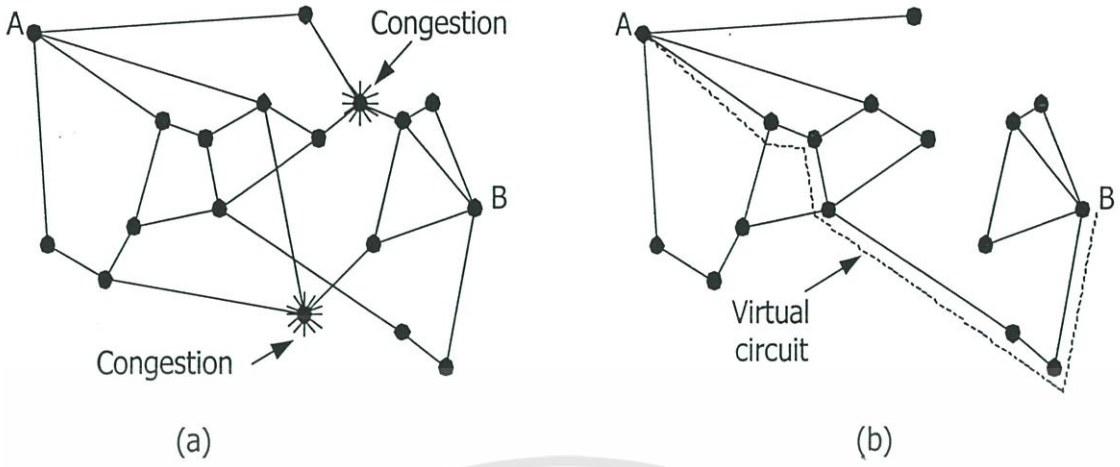
3.5.3 การควบคุมความคับคั่งของข้อมูลในเครือข่ายย่อยที่ใช้วงจรเสมือน

ในส่วนนี้จะเป็นการกล่าวถึงวิธีการควบคุมความคับคั่งของข้อมูลแบบ Dynamic บนเครือข่ายย่อย วิธีการแบบนี้จะแตกต่างจากวิธีที่กล่าวมาข้างต้น เนื่องจากจะเป็นการพิจารณาสภาวะแวดล้อมของระบบในขณะนั้น เพื่อหาทางแก้ไขปัญหาความคับคั่งของข้อมูลที่กำลังเกิดขึ้น โดยวิธีการแรกจะนำไปใช้แก้ปัญหาในระบบเครือข่ายที่ใช้วงจรเสมือนในการส่งข้อมูล และอีกสองวิธีที่เหลือจะใช้ได้ทั่วไป

เทคนิคการควบคุมจำนวนผู้ใช้ (Admission Control) สามารถนำมาใช้ในเครือข่ายย่อยที่เกิดความคับคั่งขึ้นแล้ว โดยจะนำมาควบคุมไม่ให้ความคับคั่งเพิ่มขนาดหรือขยายตัวออกไปอีก มีหลักการทำงานคือ เมื่อเกิดความคับคั่งขึ้นแล้วจะไม่ยอมให้มีการจัดตั้งวงจรเสมือนขึ้นมาอีกจนกว่าความคับคั่งนั้นจะหายไป นั่นคือเครือข่ายจะขัดขวางการติดต่อในชั้นสื่อสารนำส่งข้อมูล ซึ่งเป็นผู้จัดตั้งวงจรเสมือน แม้ว่าวิธีการนี้ค่อนข้างที่จะเด็ดขาดมากเกินไป แต่ก็เป็นวิธีการที่ง่ายแก่การนำไปใช้และมักจะได้ผลดี ในระบบโทรศัพท์ที่มีจำนวนผู้ใช้งานสูงมากเกินไป ทางชุมสายโทรศัพท์จะงดส่งสัญญาณ (Tone Signal) ไปให้แก่ผู้ใช้คนอื่น (ที่ยังไม่ได้ใช้โทรศัพท์ในขณะนั้น) เป็นการชั่วคราวจนกว่าจำนวนผู้ใช้จะลดลงสู่ระดับปกติ จะเห็นได้ว่าแม้วิธีการจะแตกต่างกันแต่ก็ใช้หลักการในทำนองเดียวกัน

วิธีประยุกต์ทางหนึ่งเพื่อลดความเข้มงวดลงบ้างคือ การยอมให้ผู้ใช้อื่นสามารถจัดตั้งวงจรเสมือนขึ้นมาใช้งานได้ แต่ก็จะจัดการให้วงจรที่สร้างใหม่นี้หลบไปใช้เส้นทางอื่นที่ไม่มีผลต่อความคับคั่งที่กำลังดำเนินอยู่ พิจารณาจากรูปที่ 3.9 สมมุติว่าโฮสต์ที่เชื่อมต่อกับเราเตอร์ A ต้องการสร้างวงจรเสมือนเพื่อติดต่อกับโฮสต์ที่เชื่อมต่อกับเราเตอร์ B แต่รูปที่ 3.9 (a) แสดงให้เห็นว่าเราเตอร์ทั้งสองตัว (A และ B) กำลังมีปัญหาความคับคั่งข้อมูลเกิดขึ้น โดยปกติวงจรเสมือนระหว่างโฮสต์ทั้งสองนี้จะใช้เป็นเส้นทางที่ผ่านเราเตอร์ที่กำลังมีปัญหา รูปที่ 3.9 (b) แสดงโครงสร้างของระบบเดิมที่ปรับปรุงโดยการลบเราเตอร์ A และ B รวมทั้งสายสื่อสารที่เชื่อมต่อเราเตอร์ทั้งสองตัวออกไปทั้งหมด แล้วเลือกใช้เส้นทางที่เขียนด้วยเส้นประแทน เพื่อหลีกเลี่ยงจุดที่มีปัญหา

อีกวิธีการหนึ่งที่นำมาใช้คือ การที่โฮสต์ทำการต่อรองกับเครือข่ายย่อยในขณะจัดตั้งวงจรเสมือน ข้อตกลงส่วนใหญ่จะเกี่ยวข้องกับปริมาณ และรูปแบบการส่งข้อมูลคุณภาพของการให้บริการ และอื่น ๆ เครือข่ายย่อยจะทำการสำรองทรัพยากรที่ต้องใช้ไว้ตลอดเส้นทางที่วงจรเสมือนถูกกำหนดขึ้น ทรัพยากรเหล่านี้ได้แก่ ตารางข้อมูล หน่วยความจำและความกว้างของช่องสัญญาณสื่อสาร ด้วยวิธีการดังกล่าวโอกาสที่จะเกิดความคับคั่งของข้อมูลนั้นเกือบจะเป็นไปไม่ได้เลย เนื่องจากทรัพยากรที่จำเป็นต้องใช้ ได้ถูกจองไว้ให้แล้ว



รูปที่ 3.9 (a) เครือข่ายที่มีความคับคั่งเกิดขึ้น
(b) รูปเครือข่ายที่สร้างขึ้นใหม่ด้วยการตัดส่วนที่มีความคับคั่งออกไป

3.5.4 โช้คแพ็กเกต

วิธีการโช้คแพ็กเกต (Choke Packet) ที่จะกล่าวถึงต่อไปนี้ สามารถนำไปใช้ได้กับเครือข่ายย่อยที่ใช้การส่งข้อมูลทั้งแบบวงจรเสมือน และแบบดาต้าแกรม โหนดแต่ละตัวสามารถตรวจสอบประสิทธิภาพการทำงานของสายสื่อสาร (ขาเข้า และขาออก) ที่เชื่อมต่อกับตนเองและทรัพยากรอื่น ๆ ได้เช่น โหนดสามารถคำนวณค่าประสิทธิภาพการทำงาน (u) ของสายสื่อสารขาออกแต่ละเส้นซึ่งมีค่าอยู่ระหว่าง 0.0 ถึง 1.0 ได้ด้วยการสุ่มตัวอย่างค่าประสิทธิภาพการทำงานของสายสื่อสาร (f) ซึ่งสามารถทำได้เป็นประจำมาคำนวณหาค่า u ให้ใกล้เคียงกับความเป็นจริง ได้ดังสมการที่ (3.3)

$$u_{new} = au_{old} + (1-a)f \tag{3.3}$$

เมื่อ a เป็นค่าคงที่ค่าหนึ่งที่ใช้เป็นน้ำหนักบอกความสำคัญของข้อมูลเดิม (U_{old})

การประมาณค่าประสิทธิภาพการทำงานของสายสื่อสารขาออก (u) เป็นตัวบอกปริมาณข้อมูลที่ส่งผ่านสายสื่อสารเส้นหนึ่ง หรือโดยอุปนัย ค่า u ก็คือตัววัดระดับความคับคั่งของข้อมูลในสายสื่อสารนั่นเอง เมื่อใดก็ตามที่ u มีค่ามากกว่าค่าคงที่ที่กำหนดไว้ล่วงหน้าสายสื่อสารขาออกเส้นนั้นถือได้ว่าอยู่ในระดับ "เตือนภัย" โหนดทุกตัวจะต้องพิจารณาค่า u ก่อนทำการส่งข้อมูลเสมอ ถ้าค่า u อยู่ในระดับ "เตือนภัย" โหนดจะต้องส่งแพ็กเกตพิเศษเรียกว่า "โช้คแพ็กเกต (Choke Packet)" ไปยังโหนดที่เป็นผู้ส่งข้อมูลเพื่อบอกสถานะความคับคั่งที่อาจเกิดขึ้นให้ทราบ แพ็กเกตนั้นจะได้รับการใส่รหัส (Tagged) เพื่อที่จะได้ไม่เกิดการส่งโช้คแพ็กเกตซ้ำอีก ก่อนที่จะได้

รับการนำส่งตามปกติ โหนดที่ได้รับโช้คแพ็กเกต จะต้องลดปริมาณหรือความถี่ในการส่งข้อมูลเข้าสู่ระบบลงส่วนหนึ่ง (กำหนดไว้ล่วงหน้า เช่น ลดลง 10 เปอร์เซ็นต์) แม้ว่าจะได้แจ้งปัญหาความคับคั่งไปแล้ว แต่แพ็กเกตอื่น ๆ ที่ส่งตามกันมาก็ยังอาจตกอยู่ในสภาวะเดียวกัน ดังนั้นจะมีโช้คแพ็กเกตจำนวนหนึ่งทยอยเข้ามาที่โหนดเพื่อบอกข่าวสารเดียวกัน โหนดจึงต้องเพิกเฉยโช้คแพ็กเกตที่เข้ามาติดต่อกันในช่วงระยะเวลาหนึ่ง หลังจากนั้นจึงให้ความสนใจกับโช้คแพ็กเกตตัวใหม่ ถ้าข่าวสารที่ได้ทำให้ทราบว่าความคับคั่งของข้อมูลยังคงเป็นเช่นเดิม (หรืออาจแย่กว่าเดิม) โหนดจะต้องลดปริมาณหรือความถี่ในการส่งข้อมูลลงไปอีก แต่ถ้าไม่ได้รับข่าวสารจากโช้คแพ็กเกตตัวใหม่ แสดงว่าความคับคั่งของข้อมูลได้หายไปแล้ว โหนดก็สามารถเพิ่มปริมาณหรือความถี่ในการส่งข้อมูลให้สูงขึ้นกว่าเดิมได้

โหนดสามารถปรับอัตราการส่งข้อมูลได้โดยการปรับค่าตัวแปรต่า ง ๆ เช่น อัตราการส่งข้อมูลในอัลกอริทึมถังน้ำรั่ว โดยทั่วไปโช้คแพ็กเกตตัวแรกที่มาถึงโหนดนั้น จะทำให้โหนดลดอัตราการส่งข้อมูลลงไปครึ่งหนึ่งของอัตราปัจจุบัน ครั้งต่อไปก็จะลดลงเหลือครึ่งหนึ่งของอัตราการส่งในขณะนั้น หรือเท่ากับหนึ่งในสี่ของอัตราเริ่มต้น ครั้งต่อ ๆ ไปก็จะลดลงไปครึ่งหนึ่งเสมอ ส่วนการเพิ่มอัตราก็จะทำแบบค่อยเป็นค่อยไปเพื่อไม่ทำให้เกิดความคับคั่งของข้อมูล

การปรับปรุงในด้านอื่น ๆ สำหรับอัลกอริทึมนี้ เช่น ให้โหนดเก็บค่าคงที่สำหรับการ “เตือนภัย” ได้หลายค่า โหนดก็มีโอกาสที่จะเลือกระดับการเตือนภัยให้เหมาะสม และโหนดก็สามารถเลือกวิธีการตอบสนองได้ดียิ่งขึ้น เช่น การเตือนภัยระดับต่ำ การเตือนภัยระดับกลาง และการเตือนภัยระดับสูงสุด การปรับปรุงอีกวิธีหนึ่งคือ การดูปริมาณงานที่ค้างอยู่ในแถวคอยสำหรับส่งแพ็กเกตหรือประสิทธิภาพการใช้งานหน่วยความจำ ก็สามารถนำมาใช้ในการตรวจสอบความคับคั่งของข้อมูลได้เช่นกัน

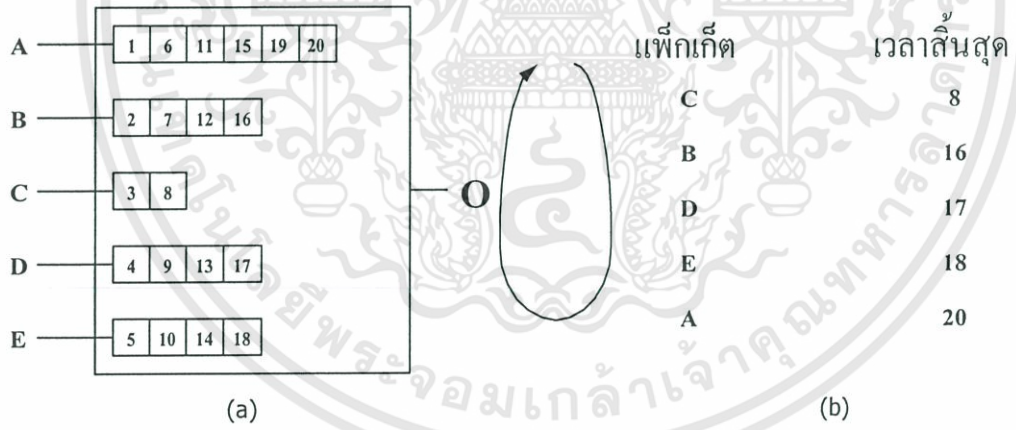
3.5.4.1 ระเบียบการบริการการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม

ปัญหาของการใช้โช้คแพ็กเกตคือ โหนดแต่ละตัว (ซึ่งมีส่วนเกี่ยวข้องโดยตรงกับปัญหาที่เกิดขึ้น) มีอิสระอย่างเต็มที่ในการตอบสนองคำเตือน สมมุติว่าโหนดตัวหนึ่งกำลังเกิดปัญหาความคับคั่งต่อเนื่องจากข้อมูลที่ส่งมาจากโหนดสี่ตัว จึงส่งโช้คแพ็กเกตไปยังโหนดทั้งสี่นั้น โหนดตัวหนึ่งตัดสินใจลดปริมาณการส่งข้อมูลลงครึ่งหนึ่งตามที่ควรจะทำ แต่โหนดตัวอื่น ๆ กลับทำเฉย โหนดตัวที่ซื่อสัตย์ก็เลยส่งข้อมูลช้าลงแต่เพียงโหนดเดียว ถ้าสถานการณ์ไม่คลี่คลาย โหนดที่ซื่อสัตย์นี้ก็ต้องลดความเร็วในการส่งข้อมูลลงไปเรื่อย ๆ จนอาจจะต้องหยุดไปในที่สุด ในเวลาเดียวกันโหนดอีกสามตัวที่เหลือก็ยังคงส่งข้อมูลที่มีความเร็วปกติต่อไป

เพื่อเป็นการแก้ปัญหาดังกล่าว เนเกิล ได้เสนอแนวทางแก้ไขโดยอัลกอริทึมการบริหารแพ็กเกตอย่างเป็นธรรม (Fair Queueing Algorithm) วิธีการนี้โหนดจะต้องสร้างแถวคอยแต่ละสายสื่อสารขาออก จำนวนแถวคอยในแต่ละสายจะเท่ากับจำนวนโหนดส่งข้อมูลออกทางสายเส้นนั้น เมื่อ

สายเส้นนั้นว่างลง โหนดก็จะเลือกแพ็กเกตขึ้นมาจากแถวคอย ทั้งหมดตามลำดับการหมุนเวียนอย่างเป็นธรรม ด้วยวิธีการนี้แพ็กเกตที่มาจากโหนดทุกตัวมีโอกาสที่จะได้รับการนำส่งเท่ากันหมด แม้ว่าจำนวนแพ็กเกตในแถวคอยจะไม่เท่ากันก็ตาม วิธีการนี้ได้นำไปใช้ในการส่งข้อมูลในเครือข่ายคอมพิวเตอร์

ด้วยลักษณะการทำงานของอัลกอริทึมนี้ แม้ว่าจะให้โอกาส (จำนวนครั้งในการส่ง) แก่โหนดต่าง ๆ เท่ากัน แต่แพ็กเกตที่มีขนาดใหญ่จะได้รับส่วนแบ่งเวลามากกว่าแพ็กเกตที่มีขนาดเล็กกว่าเสมอ ดีเมอร์และคณะ ได้เสนอแนวทางแก้ไขโดยการปรับปรุงให้ใช้การวัดปริมาณข้อมูลแทนการนับจำนวนแพ็กเกต โดยให้ตรวจข้อมูลในทุกแถวคอยหมุนเวียนกันไปแบบเดิมแต่เป็นการตรวจขนาดข้อมูลไม่ใช่จำนวนแพ็กเกต หรืออีกในหนึ่งเป็นการสมมุติว่าทุกแถวคอยใช้แพ็กเกตขนาดเดียวกันหมดในการตรวจนี้จะใส่หมายเลขกำกับสำหรับทุก ๆ แพ็กเกตสมมุติ เรียงตามลำดับที่ควรจะนำส่ง เมื่อหมุนเวียนครบทุกแถวคอยจะได้หมายเลขสุดท้ายซึ่งเป็นหมายเลขสุดท้ายซึ่งเป็นของตนเอง จากนั้นจึงจัดเรียงแถวคอยทั้งหมดตามลำดับหมายเลขเหล่านี้จากน้อยไปหามาก โหนดจะส่งข้อมูลแบบหมุนเวียนตามลำดับคือ จะนำแพ็กเกตจากแถวคอยที่มีหมายเลขต่ำสุดส่งออกไปลำดับแรก ตามด้วยแพ็กเกตจากแถวคอยในลำดับที่สอง เวียนไปจนครบแล้วกลับไปที่แถวแรกอีกดังแสดงในรูปที่ 3.10



รูปที่ 3.10 แสดงการจัดแถวคอยอย่างเป็นธรรม

(a) โหนดที่มีแพ็กเกตรอนำส่งอยู่ใน 5 แถวคอย

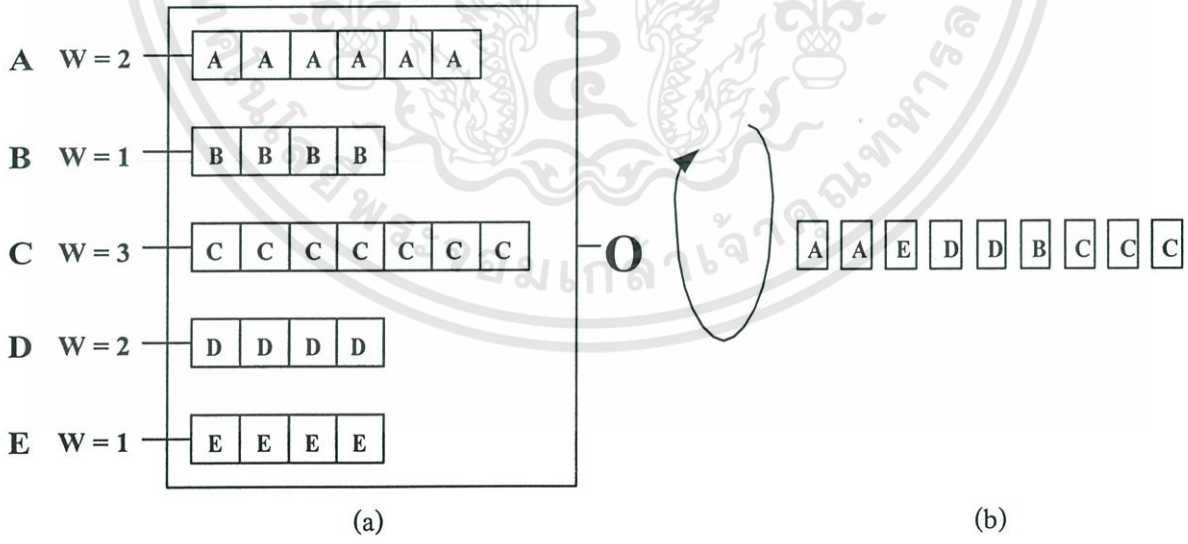
(b) เวลาสิ้นสุดการทำงานของแต่ละแถวคอย

ตามรูปที่มีแพ็กเกตอยู่ในแถวคอย 5 แถวมีความยาวตั้งแต่สองถึงหกหน่วย ในขั้นตอนของการจัดแถวจะให้แพ็กเกตสมมุติมีขนาดเท่ากับหนึ่งหน่วย แพ็กเกตแรกในแถวคอย A (วัดตามขนาดสมมุติ) ถูกกำหนดให้เป็นหมายเลข " 1 " ตามด้วยแพ็กเกตแรกในแถวคอย B, C, D และ E กำหนด

หมายเลขใ+ห้เป็น " 2, 3, 4 และ 5 " ตามลำดับ การหมุนเวียนรอบที่สอง แพ็กเกตที่สองในแถวคอย A, B, C, D และ E จะถูกกำหนดหมายเลขให้เป็น " 6, 7, 8, 9 และ 10 " ตามลำดับท้ายที่สุดแพ็กเกตลำดับสุดท้าย ในแถวคอย A, B, C, D และ E จะมีหมายเลขสูงสุดของแต่ละแถวเป็น " 20, 16, 8, 17 และ 18 " ตามลำดับดังรูป 3.10 (a) ขั้นต่อไปจัดเรียงลำดับแถวคอยใหม่โดยให้เรียงตามหมายเลขสุดท้ายของแต่ละแถวคอยจากน้อยสุดไปหามากสุด ซึ่งจะได้ผลลัพธ์ดังรูป 3.10 (b) ในการส่งข้อมูลจริง โหนดจะนำแพ็กเกต (ตัวจริง) ตัวแรกจากแถวคอย C ส่งออกไปก่อนหนึ่งตัว ตามด้วยแพ็กเกตตัวแรกจากแถวคอย B, D, E, และ A ส่งออกไปตามลำดับ ในรอบที่สองก็จะนำแพ็กเกตตัวแรกในแถวคอย (เดิมเป็นแพ็กเกตตัวที่สอง) จาก C, B, D, E และ A ส่งไปครั้งละหนึ่งแพ็กเกตตามลำดับ และจะดำเนินการหมุนวนไปเรื่อย ๆ จนกว่าจะหมด

การให้ความสำคัญของทุกแถวคอยเท่ากันหมดเป็นปัญหาสำหรับอัลกอริทึมนี้ ในสภาวะแวดล้อมหลาย ๆ แบบมีความนิยมที่จะให้ความสำคัญแก่แพ็กเกตที่ส่งจากผู้ให้บริการ (Servers) มากกว่าแพ็กเกตที่ส่งมาจากผู้ใช้บริการ (Clients) เช่น ส่งข้อมูลที่มาจากผู้ให้บริการเป็นสองเท่าของข้อมูลที่มาจากผู้ใช้บริการ การปรับปรุงอัลกอริทึมในลักษณะนี้เรียกว่า "การจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม" ซึ่งเป็นอัลกอริทึมที่ได้รับความนิยมอย่างกว้างขวาง บางครั้งน้ำหนักที่ให้จะเท่ากับวงจรเสมือนที่ออกมาจากโหนดเพื่อให้แต่ละโพรเซส (Process) มีโอกาสเท่า ๆ กัน

ในวิทยานิพนธ์นี้ จะให้น้ำหนักคือ ค่าระยะทางระหว่างโหนด ถ้าหากให้ [ ] แทนแพ็กเกตที่อยู่ในแถวตามที่มีตัวอักษรระบุไว้ เช่น [A] หมายถึงแพ็กเกตที่อยู่ในแถว A เป็นต้น จากรูปที่ 3.11 จะแสดงหลักการทำงานของการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม



รูปที่ 3.11 การจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม

(a) โหนดที่มีแพ็กเกตรอนำส่งอยู่ใน 5 แถวคอย

(b) แพ็กเกตที่ถูกส่งออกมาสำหรับการหมุนเวียนแต่ละรอบ

ถ้าหากว่าการหมุนเวียนในแต่ละรอบ เป็นไปดังแสดงในรูปที่ 3.11 กล่าวคือแพ็กเกตจากแถว C จะถูกส่งออกมาก่อน แต่เนื่องจากว่าแถว C มีค่าน้ำหนักเท่ากับ 3 ดังนั้นแถว C จึงส่งออกมา 3 แพ็กเกต (โดยที่ทั้ง 3 แพ็กเกต ไม่จำเป็นต้องออกมา ติด ๆ กัน เพียงแต่ว่า ภายในรอบหนึ่ง ๆ ต้องออกมาครบ 3 แพ็กเกต) หลังจากนั้นแพ็กเกตในแถว B จะถูกส่งออกมา แต่เนื่องจากแถว B มีค่าน้ำหนักเท่ากับ 1 ดังนั้นแถว B จึงส่งแพ็กเกตออกมา 1 แพ็กเกต ลำดับต่อไปแถว D จะส่งแพ็กเกตออกมา แต่เนื่องจากว่าแถว D มีค่าน้ำหนักเท่ากับ 2 ดังนั้นแถว D จึงส่งแพ็กเกตออกมา 2 แพ็กเกต ลำดับต่อไปแถว E จะส่งแพ็กเกตออกมา แต่เนื่องจากว่าแถว E มีค่าน้ำหนักเท่ากับ 1 ดังนั้นแถว E จึงส่งแพ็กเกตออกมา 1 แพ็กเกต ลำดับสุดท้ายของรอบแถว A จะส่งแพ็กเกตออกมาแต่เนื่องจากว่าแถว A มีค่าน้ำหนักเท่ากับ 2 ดังนั้นแถว A จะส่งแพ็กเกตออกมา 2 แพ็กเกตดังแสดงในรูปที่ 3.11 (b) การหมุนเวียนรอบที่สองก็จะดำเนินการหมุนวนไปเรื่อยจนกว่าจะหมด

ค่าน้ำหนัก (w) มีค่าดังสมการที่ (3.4)

$$w = \frac{r_q}{R(e)} \tag{3.4}$$

โดยที่ r_q คือจำนวนแบนด์วิดธ์ที่ถูกจองสำหรับการร้องขอของการส่งข้อมูล q (บิต/วินาที)

$R(e)$ คือความจุแบนด์วิดธ์ทั้งหมดของลิงค์ (บิต/วินาที)

การออกแบบอัลกอริทึมการเลือกเส้นทางที่มีการรับประกัน QoS ที่นำเสนอในวิทยานิพนธ์ฉบับนี้ ได้มีการนำเอาระเบียบบริการการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรมมาใช้ ซึ่งจะได้กล่าวรายละเอียดในบทต่อไป

3.5.5 การควบคุมความไม่ต่อเนื่อง

โปรแกรมประยุกต์ประเภทการถ่ายทอดสัญญาณเสียง และภาพเคลื่อนไหวนั้น แม้ว่าจะมีระยะการนำส่งข้อมูล 20 หรือ 30 มิลลิวินาทีต่อแพ็กเกต ก็จะทำให้ไม่เกิดผลเสียหายตราบเท่าระยะเวลานี้คงที่ สมมุติว่าแพ็กเกตบางส่วนถูกส่งมาถึงผู้รับภายใน 20 มิลลิวินาที และแพ็กเกตอีกบางส่วนมาถึงผู้รับภายใน 30 มิลลิวินาที ผลปรากฏคือ ผู้รับจะได้รับเสียงหรือภาพเคลื่อนไหวที่ไม่ต่อเนื่อง ดังนั้นทั้งโหนด และเครือข่ายย่อยจะต้องทำความตกลงในลักษณะการรับประกันระยะเวลาในการนำส่งข้อมูล เช่นจะส่งข้อมูลแต่ละแพ็กเกตจากแหล่งกำเหนิด ไปถึงผู้รับให้ได้ภายในระยะเวลา 24 มิลลิวินาที ถึง 25 มิลลิวินาที กระบวนการนี้เรียกว่า "การควบคุมความไม่ต่อเนื่องของ

ข้อมูล (Jitter Control)" ระยะเวลาที่กำหนดนี้จะต้องเป็นไปได้จริง และสามารถคำนวณได้จากปริมาณความคับคั่งของข้อมูลตลอดเส้นทางเดินที่ต้องการ

การควบคุมความไม่ต่อเนื่องสามารถคำนวณได้จากค่าคาดหวังในการส่งข้อมูลของแต่ละช่วง หรือการส่งข้อมูลจากโหนดหนึ่งไปยังโหนดถัดไป แต่ละโหนดสามารถตรวจสอบจำนวนแพ็กเกตที่อยู่ในแถวคอยว่าแพ็กเกตใดส่งไปเร็ว หรือช้ากว่ากำหนด ข้อมูลนี้จะถูกฝากไว้ในแพ็กเกตและจะได้รับการปรับปรุงทุกครั้งที่ผ่านโหนดแต่ละโหนด ถ้าแพ็กเกตได้รับการนำส่งเร็วกว่าที่กำหนดไว้ ก็จะถูกบังคับให้รอจนกว่าจะถึงเวลาที่ควรนำส่งตามตาราง ถ้าหากช้ากว่าที่กำหนดโหนดจะพยายามนำส่งแพ็กเกตนั้นโดยเร็วที่สุด อันที่จริงแล้วอัลกอริทึมที่ใช้ในการเลือกแพ็กเกตที่ควรนำส่งก่อน จะเลือกนำส่งแพ็กเกตที่อยู่ในแถวคอยที่ช้ากว่ากำหนดเป็นอันดับแรก ในเวลาเดียวกันแพ็กเกตที่เร็วกว่าที่กำหนดก็จะถูกบังคับให้รอคอยโดยอัตโนมัติ ทั้งหมดนี้เป็นผลในการลดความไม่ต่อเนื่องของข้อมูลลงได้

3.5.6 โพรโตคอลสำรองทรัพยากร

การสำรองทรัพยากรในระบบ เป็นกระบวนการหนึ่งที่ทำงานร่วมกับการขั้นตอนการควบคุมปริมาณของข้อมูลในเครือข่าย รายละเอียดในการขอจัดตั้งวงจรเสมือนนั้น ๆ อัตราความเร็วสูงสุดในการส่งข้อมูล ซึ่งระบบจะทำการตรวจสอบและจัดการสำรองทรัพยากรไว้สำหรับวงจรเสมือนนั้น ๆ อัตราความเร็วสูงสุดที่ผู้ใช้ต้องการนี้จะถูกนำไปพิจารณาค้นหาเส้นทางเดินของข้อมูลที่สามารถตอบสนองความต้องการได้

3.6 ความต้องการการรับประกัน QoS

ในเครือข่ายสื่อสารคอมพิวเตอร์ ความต้องการของเครือข่ายและการคำนวณทรัพยากรสำหรับแอพพลิเคชั่นแบบสื่อผสม ถูกแสดงในเทอมของพารามิเตอร์ที่เป็น QoS พารามิเตอร์ที่เป็น QoS ที่มีความสำคัญมากที่สุดที่ระดับเครือข่ายคือ แบนด์วิดธ์คอขวด เอนทูเอนดีเลย์ เอนทูเอนจิตเตอร์ดีเลย์ และ อัตราการสูญเสีย โดยปกติแล้วแอพพลิเคชั่นแบบสื่อผสมที่แตกต่างกันจะต้องการระดับของ QoS ที่แตกต่างกัน ยกตัวอย่างเช่น ในกรณีของ แอพพลิเคชั่นที่เป็นการประชุมทางวิดีโอ ต้องการความเอนทูเอนดีเลย์อย่างเข้มงวด แต่สามารถผ่อนปรนกับอัตราการสูญเสียที่แน่นอนได้ [13] สำหรับแอพพลิเคชั่นที่เป็นวิดิโอออนดีมานด์ ความสำคัญของความต้องการ QoS คือ จิตเตอร์ดีเลย์ แต่แอพพลิเคชั่นการส่งผ่านของมูล บัลค์ (Bulk) เช่น เอฟทีพี (FTP) ต้องการอัตราการสูญเสีย เพื่อที่จะรับประกัน QoS ทรัพยากรในเครือข่ายจะต้องถูกจองสำหรับ เซสชั่น (Session) นั้น ๆ เร็ว ๆ นี้ได้มีการนำเสนอโปรโตคอลการจองทรัพยากร เช่น RSVP [9] ซึ่งพิจารณาจากการจัดตารางแพ็กเกต

หรือระเบียบวิธีการให้บริการ ซึ่งจะสามารถรับประกัน QoS ได้ จำนวนที่ต้องการของแบนด์วิดธ์และบัฟเฟอร์สามารถถูกคำนวณ ดังนั้นจึงทำให้ความต้องการในการประกัน QoS เป็นไปได้

บทที่ 4
การประมวลผลล่วงหน้าเพื่อหาเส้นทาง ที่รับประกันคุณภาพของการให้บริการ

ในบทนี้อธิบายถึงรายละเอียดของอัลกอริทึมการเลือกเส้นทางบนพื้นฐาน QoS และการลดความซับซ้อนของอัลกอริทึมนี้ โดยการใช้ระเบียบการให้บริการมากำหนดเงื่อนไขดังนั้นระเบียบการให้บริการจึงมีความสำคัญต่อการออกแบบอัลกอริทึมการเลือกเส้นทาง ในวิทยานิพนธ์นี้พารามิเตอร์ที่เกี่ยวข้องกับการบังคับ QoS มี 4 พารามิเตอร์คือ แบนด์วิดธ์คอขวด เอนทูเอนดีเลย์ เอนทูเอนจิตเตอร์ดีเลย์ และการไม่มีการสูญเสีย ในวิทยานิพนธ์นี้ได้ใช้ระเบียบบริการ การจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม และเรียกอัลกอริทึมนี้ว่า อัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS ในการให้บริการวิธีของเบลแมนฟอร์ด ซึ่งเป็นการปรับปรุงมากจากอัลกอริทึมการเลือกทางเดินที่สั้นที่สุดของเบลแมนฟอร์ด ซึ่งมีเวลาการคำนวณในกรณีที่เลวที่สุด (Worst Case) จะมีค่าเท่ากับเวลาของเบลแมนฟอร์ด กล่าวคือมีความซับซ้อนเท่ากับ $O(|E| \cdot H_{\max})$ โดยที่ $|E|$ เป็นจำนวนของลิงค์ในเครือข่าย และ $H_{\max}$ เป็นจำนวนอิเทอเรชั่นที่มากที่สุดของอัลกอริทึมการเลือกทางเดินที่สั้นที่สุดเบลแมนฟอร์ด และเป็นค่าขอบเขตบนของจำนวนฮอปในเส้นทางบังคับ QoS อัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS แบบเบลแมนฟอร์ด สามารถหาเส้นทางที่มีจำนวนฮอปที่น้อยที่สุดซึ่งเป็นไปตามความต้องการการบังคับ QoS ถ้าหากว่ามีเส้นทางนั้นอยู่ภายในเครือข่าย และขณะเดียวกันอัลกอริทึมนี้ยังมีประสิทธิภาพในการกระจายการจราจรตลอดเครือข่ายอีกด้วย

ในวิทยานิพนธ์นี้ เราได้ทำการศึกษาอัลกอริทึมการเลือกเส้นทางที่มี QoS สำหรับ เส้นทางที่คำนวณไว้ล่วงหน้า เมื่อเราไม่รู้ล่วงหน้า เกี่ยวกับพารามิเตอร์ บางตัวหรือทุกตัว และจำนวนของแบนด์วิดธ์ที่ถูกจองไว้แล้ว เนื่องจากว่าตารางเส้นทางที่เก็บเส้นทางที่มีการรับประกัน QoS มีขนาดใหญ่มาก ดังนั้นเราได้ทำการทดลองเพื่อลดขนาดของตารางเส้นทาง และผลการซิมมูเลชั่นแสดงให้เห็นว่าเราสามารถลดขนาดของตารางเส้นลงได้ เนื่องจากเส้นทางที่เป็นไปตามบางเงื่อนไขที่เราได้กำหนดไว้ จะถูกละทิ้งได้ เพราะเป็นเส้นทางที่เกินความจำเป็น ในขณะที่เส้นทางที่ดีกว่าได้ถูกเก็บไว้เรียบร้อยแล้ว

4.1 ระเบียบการให้บริการสำหรับอัลกอริทึมการเลือกเส้นทางบนพื้นฐานของคุณภาพของการบริการ (QoS)

สำหรับการหาเส้นทางซึ่งมีการบังคับพารามิเตอร์ของ QoS หลาย ๆ พารามิเตอร์ ได้ถูกพิสูจน์แล้วว่าเป็นปัญหา *NP* สัมบูรณ์ [3] ดังนั้นการออกแบบอัลกอริทึมการเลือกเส้นทางบนพื้นฐานของ QoS ในวิทยานิพนธ์นี้ได้ใช้ระเบียบบริการการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นเพื่อกำหนดขอบเขตของ QoS ดังนั้นประเด็นในการออกแบบที่สำคัญอย่างหนึ่งสำหรับอัลกอริทึมการเลือกเส้นทางบนพื้นฐานของ QoS นั้น เป็นความสามารถที่ได้มาจากระเบียบของการให้บริการ

กลุ่มการบริการการรับประกันของโมเดลบริการร่วมในเครือข่ายคอมพิวเตอร์ ตั้งอยู่บนพื้นฐานการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม [14] ซึ่งสามารถใช้กำหนดขอบเขตบนของเอนทูเอนดีเลย์ที่เข้มงวด, เอนทูเอนจิตเตอร์ดีเลย์ และทำให้แน่ใจว่า ไม่มีแพ็กเกตที่สูญเสียเนื่องจากที่แต่ละสวิตซ์ไม่มีบัฟเฟอร์เพียงพอ สมมุติว่าในองค์ประกอบของเครือข่ายไม่มีความล้มเหลว สมการที่ (4.1) (4.2) และ (4.3) แสดงสมการในการคำนวณ เอนทูเอนดีเลย์ เอนทูเอนจิตเตอร์ดีเลย์ และความต้องการช่องว่างของบัฟเฟอร์ (สำหรับไม่ให้มีการสูญเสียคิวอิ่ง) การใช้ระเบียบบริการการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม ของการส่งข้อมูลใด ๆ ส่งจากโหนดต้นทางใด ๆ ไปถึงโหนดปลายทางใด ๆ (q) ซึ่งมีคุณลักษณะของการจราจรในรูปแบบของ $\left(\sigma_q, \rho_q, S^q_{\max}\right)$ [15] โดยที่

σ_q คือขนาดของโทคเก้น บัคเก็ต (Token Bucket) หรือขนาดของเบิสท์ (Burst) ที่ใหญ่ที่สุด (บิต)

ρ_q คืออัตราการสร้างโทคเก้น บัคเก็ต สำหรับการร้องขอของการส่งข้อมูล q (บิต/วินาที)

$S^q_{\max}$ คือขนาดของแพ็กเกตที่ใหญ่ที่สุดของการร้องขอของการส่งข้อมูล q (บิต)

r_q คืออัตราการรับประกันของการเชื่อมโยง หรือจำนวนของแบนด์วิดธ์ที่ถูกจองสำหรับการร้องขอของการส่งข้อมูล $q\left(r_q \geq \rho_q\right)$ (บิต/วินาที)

$S_{\max}$ คือขนาดของแพ็กเกตที่ใหญ่ที่สุดที่ยอมให้มีได้ในเครือข่าย (บิต)

$R_i(e)$ คือความจุของแบนด์วิดธ์ทั้งหมดของลิงค์ e ซึ่งเป็นลำดับที่ i บนของเส้นทางการส่งข้อมูล q (บิต/วินาที)

$pd_i(e)$ คือพรอพพิเกชั่นดีเลย์ของลิงค์ e ซึ่งเป็นลำดับที่ i บนเส้นทางของการส่งข้อมูล q (วินาที)

H คือจำนวนฮอปของเส้นทาง

ขอบเขตของค่าเอนทูเอนดีเลย์

$$= \frac{\sigma_q + HS^q_{\max}}{r_q} + \sum_{i=1}^{H}\left\{\frac{S_{\max}}{R_i(e)} + pd_i(e)\right\} \quad (4.1)$$

ขอบเขตของค่าเอนทูเอนจิตเตอร์ดีเลย์ $= \frac{\sigma_q + HS^q_{\max}}{r_q}$ (4.2)

ขนาดช่องว่างของบัฟเฟอร์ของสวิตซ์ลำดับที่ n $= \sigma_q + nS^q_{\max}$ (4.3)

4.2อัลกอริทึมการเลือกเส้นทางที่รับประกันคุณภาพการบริการโดยใช้วิธีของเบลแมนฟอร์ด

เราจะอธิบายถึงวิธีการลดความซับซ้อนของปัญหาโดยใช้สมการขอบเขต QoS โดยใช้ระเบียบบริการการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม

4.2.1 การลดความซับซ้อน

ปัญหาการเลือกเส้นทางที่มี QoS ถูกพิสูจน์ว่าเป็นปัญหา *NP* สัมบูรณ์ [3] อย่างไรก็ตาม เมื่อมีการบังคับพารามิเตอร์ของ QoS มากกว่าหนึ่งพารามิเตอร์ ซึ่งก็คือ เพิ่มเติม (Additive) หรือ หลายอย่าง (Multiplicative) และถ้าค่าที่น้อยที่สุด ค่าหนึ่งเท่ากันทุกลิงค์ เราสามารถหาเส้นทางที่บังคับ QoS โดยใช้อัลกอริทึมวิธีของเบลแมนฟอร์ด ในเวลาโพลิโนเมียล เบลแมนฟอร์ดเป็นอัลกอริทึมค้นหาเส้นทางที่มีค่าแบนด์วิดธ์มากที่สุดเป็นอันดับแรก การดำเนินการค้นหาเส้นทางที่สั้นที่สุดโดยการเพิ่มจำนวนฮอป ที่ซึ่งสามารถพิจารณาในฐานะที่มีลักษณะเป็น การบังคับที่เพิ่มเติมอย่างหนึ่ง ที่ซึ่งทุกลิงค์มีค่าเป็น 1 แนวความคิดของการลดความซับซ้อนเกิดจากความจริงที่เป็นไปได้ที่จะแมป (Map) ดังนี้

1.การบังคับเส้นทาง ถึงการบังคับลิงค์

2.บังคับคุณภาพของการบริการในเทอมของจำนวนฮอป เช่นค่าสูงสุดของจำนวนฮอปที่ยอมได้ของเส้นทาง

เราสามารถแมปการบังคับพารามิเตอร์ที่แตกต่างกัน โดยการแมปการบังคับดีเลย์ กับจำนวนฮอป ดังนั้นการลดความซับซ้อนของปัญหานี้บนการเลือกเส้นทางที่สั้นที่สุดโดยการใช้ระเบียบวิธีการบริการการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม เราสามารถลดความซับซ้อน

โดยการใช้สมการ (4.1) (4.2) และ (4.3) เมื่อบังคับพารามิเตอร์ของ QoS ที่เป็น แบนด์วิดธ์ ดีเลย์ จิตเตอร์ดีเลย์ และการไม่มีการสูญเสีย วิธีการการลดความซับซ้อนมีดังต่อไปนี้

4.2.1.1 การบังคับดีเลย์

ดีเลย์ของลิงค์ประกอบด้วย

1.คิวอิ่งดีเลย์ คือเวลาที่แพ็กเกตอยู่ในคิว (วินาที)

2.ทรานสมิตชั่นดีเลย์ คือเวลาที่ใช้ในการส่งข้อมูลตั้งแต่บิตแรก จนถึงบิตสุดท้ายเข้าไปในลิงค์ (วินาที)

3.พรอพพิเกชั่นดีเลย์ คือเวลาในการถ่ายโอนข้อมูลผ่านลิงค์ (วินาที)

คิวอิ่งดีเลย์ของเส้นทางเป็นฟังก์ชั่นของจำนวนฮอป H แต่ทรานสมิตชั่นดีเลย์และพรอพพิเกชั่นดีเลย์ อาจจะแตกต่างกันในลิงค์ที่ต่างกัน ด้วยเหตุนี้เราจึงไม่สามารถลดความซับซ้อนได้จริงในเทอมของจำนวนฮอป อย่างไรก็ตาม ดีเลย์ที่สั้นที่สุด สามารถใช้เป็นจุดประสงค์ที่ดีที่สุดของอัลกอริทึมเบลแมนฟอร์ด จากสมการขอบเขตของเอนทูเอนดีเลย์ และดีเลย์ของลิงค์ $d(e)$ ของลิงค์ $e(u,v)$ ถูกนิยามดังสมการที่ (4.4)

$$d(e) = \frac{\sigma_k}{r_k} + \frac{S^q_{\max}}{r_k} + \frac{S_{\max}}{R\ (e)} + pd(e) \qquad \text{กรณีที่ } u = s \qquad (4.4)$$

$$d(e) = \frac{S^q_{\max}}{r_k} + \frac{S_{\max}}{R\ (e)} + pd(e) \qquad \text{กรณีอื่น ๆ}$$

4.2.1.2 การบังคับแบนด์วิดธ์

การบังคับแบนด์วิดธ์เป็นการบังคับลิงค์ เราสามารถกำจัดออกไปง่าย ๆ ทุกลิงค์ในเครือข่าย ซึ่งไม่เป็นไปตามการบังคับแบนด์วิดธ์ ก่อนการดำเนินการของอัลกอริทึม เราสามารถกำหนดค่าดีเลย์เป็นค่า อินฟินิต (Infinite) ถ้าลิงค์ทั้งหลายเหล่านั้น ดังสมการที่ (4.5)

$$d(e) = \infty \qquad \text{กรณีที่ } b(e) < \Delta_{bandwidth} \qquad (4.5)$$

$$d(e) = d(e) \qquad \text{กรณีอื่น ๆ}$$

ด้วยเหตุนี้การบังคับแบนด์วิดธ์ถูกแมปกับ ดีเลย์ของลิงค์

4.2.1.3 การบังคับจิตเตอร์ดีเลย์

จิตเตอร์ดีเลย์ เป็นฟังก์ชั่นของจำนวนฮอปของเส้นทาง $H_{\max}^{jitter}$ ก่อนการดำเนินการของอัลกอริทึมวิธีของเบลแมนฟอร์ด เส้นทางที่มีจำนวนฮอปมากที่สุด ซึ่งยังคงเป็นไปตาม จิตเตอร์ดีเลย์ ซึ่งสามารถคำนวณได้ดังสมการที่ (4.6) โดยการตั้งค่านี้เป็นจำนวนอิเทอเรชั่นที่มากที่สุดของอัลกอริทึมวิธีเบลแมนฟอร์ด ซึ่งเป็นจำนวนฮอปที่มากที่สุดที่ยอมรับได้ในเส้นทาง เราสามารถแน่ใจได้ว่าเส้นทางจะมีค่าจิตเตอร์ดีเลย์ที่ต่ำกว่า การบังคับจิตเตอร์ดีเลย์ที่ต้องการ

$$H_{\max}^{jitter} = \left\lfloor \frac{\Delta_{jitter} \cdot r_q - \sigma_q}{S_{\max}^q} \right\rfloor \qquad (4.6)$$

ดังนั้น จิตเตอร์ดีเลย์ ถูกแมปกับจำนวนของฮอปที่มากสุดที่ยอมให้มีได้บนเส้นทาง

4.2.1.4 การไม่มีการสูญเสียหรือการบังคับช่องว่างของบัฟเฟอร์

ข้อบังคับนี้เป็นกรณีเฉพาะ เมื่อความน่าจะเป็นในความสูญเสียเท่ากับศูนย์ ซึ่งจะเป็นไปได้เมื่อมีช่องว่างของบัฟเฟอร์ที่ใช้การได้เพียงพอ ที่ทุกโหนดตามเส้นทางที่ติดตั้งสำหรับการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม ขึ้นอยู่กับตำแหน่งของสวิตซ์นั้น (Particular Switch) จากโหนดต้นทาง (ต่อไปข้างหน้าจะใช้จำนวนของฮอป) ต้องการขนาดบัฟเฟอร์เพื่อให้แน่ใจว่าไม่มีการสูญเสียแพ็กเกตซึ่งถูกนิยามดังสมการที่ (4.3) เราจะเห็นว่าการต้องการช่องว่างบัฟเฟอร์เพิ่มขึ้นตามจำนวนฮอปนี้ ถ้าหากว่าปราศจากการสูญเสีย เราแน่ใจว่าแต่ละโหนดมีบัฟเฟอร์ที่สัมพันธ์กับความต้องการที่ตรงกัน ดังนั้นทุกลิงค์ที่เข้ามา เราสามารถกล่าวได้ว่าลิงค์ e มีบัฟเฟอร์เพียงพอ ถ้าจำนวนฮอปของเส้นทางน้อยกว่าหรือเท่ากับ $m(e)_{\max}$ นิยามดังสมการที่ (4.7) โดยที่ $f(e)$ เป็นบัฟเฟอร์ที่ตรงกันกับลิงค์ e ดังนั้นถ้า $m(e)_{\max}$ เป็นค่าสูงสุดของจำนวนฮอปที่อนุญาตของลิงค์ e จะแน่ใจได้ว่าไม่มีการสูญเสียในการส่งผ่าน ในอัลกอลิทึมวิธีของเบลแมนฟอร์ด จำนวนอิเทอเรชั่นเป็นจำนวนเดียวกันกับจำนวนฮอปที่พาติคิวลาร์สเตด (Particular State) ของการค้นหา ดังนั้นจำนวนฮอปของทุกลิงค์เป็นจำนวนเดียวกันในอิเทอเรชั่นพาติคิวลาร์และเท่ากับจำนวนอิเทอเรชั่นนั้น สำหรับอิเทอเรชั่นลำดับที่ i ลิงค์สามารถใช้เมื่อ $i \leq m(e)_{\max}$ สำหรับลิงค์อื่น ๆ ควรจะละเลย เราสมมุติว่าโหนดต้นทางมีช่องว่างบัฟเฟอร์เพียงพอ $\left(\leq \sigma_q + S_{\max}^q \right)$

$$m(e) = \left\lfloor \frac{f(e) - \sigma_q}{S_{\max}^q} \right\rfloor - 1 \qquad \text{โดยที่} \quad e = (u, v) \in E \qquad (4.7)$$

ดังนั้นเราสามารถแมปกับการบังคับช่องว่างของบัฟเฟอร์ เข้ากับ การบังคับลิงค์

4.2.2 รายละเอียดของอัลกอริทึม

ในขั้นตอนแรก ก่อนที่อัลกอริทึมของการเลือกเส้นทางที่รับประกัน QoS ของการบริการโดยวิธีของเบลแมนฟอร์ดจะทำงาน ในวิทยานิพนธ์ฉบับนี้จะสมมติว่าโหนดต้นทางมีข่าวสารเกี่ยวกับโครงสร้างของระบบเครือข่าย และจำนวนของทรัพยากรในเครือข่าย เช่น จำนวนของแบนด์วิดธ์ที่ใช้งานได้ของลิงค์ ช่องว่างของบัฟเฟอร์เป็นต้น ข่าวสารเหล่านี้จะมีอยู่ที่ทุก ๆ โหนดในเครือข่ายเพื่อเตรียมไว้สำหรับโปรโตคอลของการเลือกเส้นทาง อัลกอริทึมนี้เป็นอัลกอริทึมของการเลือกเส้นทางที่รับประกัน QoS ตามความต้องการของโหนดต้นทาง กล่าวคือ เมื่อโหนดต้นทางได้รับการร้องขอการเชื่อมโยง แล้วโหนดต้นทางจะทำการค้นหาเส้นทาง โดยคำนวณหาเส้นทางจากโหนดต้นทางไปยังโหนดปลายทาง

รายละเอียดของอัลกอริทึมการเลือกเส้นทางที่รับประกันคุณภาพ QoS โดยวิธีของเบลแมนฟอร์ด มีลักษณะคล้ายกับอัลกอริทึมการเลือกเส้นทางที่สั้นที่สุดของเบลแมนฟอร์ด กล่าวคือ อิเทอร์เรชั่นแรกของอัลกอริทึมการเลือกเส้นทางที่สั้นที่สุดของเบลแมนฟอร์ด จะหาเส้นทางที่สั้นที่สุด 1 ฮอป จากโหนดต้นทางไปยังโหนดปลายทาง สำหรับอิเทอเรชั่นที่สอง อัลกอริทึมนี้จะหาเส้นทางที่สั้นที่สุด 2 ฮอป และจะดำเนินการเช่นเดียวกันนี้ไปเรื่อย ๆ เมื่อจำนวนอิเทอเรชั่นสูงขึ้นก็จะใช้จำนวนฮอปมากขึ้น อัลกอริทึมนี้จะสิ้นสุดหลังจากที่ถึงอิเทอร์เรชั่นมากที่สุดที่ได้กำหนดไว้ $H_{\max}$ โดยที่ $H_{\max}$ เป็นจำนวนฮอปที่มากที่สุดที่อนุญาตในเครือข่าย หรือเมื่อพบเส้นทางที่เป็นไปตามข้อบังคับที่ตรงตามข้อกำหนดของ QoS ประเด็นที่สำคัญที่สุดของอัลกอริทึมของการเลือกเส้นทางที่รับประกัน QoS โดยวิธีของเบลแมนฟอร์ด คือ การกำจัดลิงค์ที่ไม่เป็นไปตามเงื่อนไขของ QoS และมีการกำหนดจำนวนอิเทอเรชั่นสูงสุดเท่ากับจำนวนฮอปที่มากที่สุดที่กำหนดไว้แล้ว เนื่องจากข้อมูลเหล่านี้จะถูกคำนวณไว้ล่วงหน้าก่อนที่อัลกอริทึมนี้จะทำงาน อัลกอริทึมของการเลือกเส้นทางที่รับประกัน QoS โดยวิธีของเบลแมนฟอร์ด เริ่มคำนวณค่าใช้จ่ายของลิงค์ $C(e)$, ค่าดีเลย์ของลิงค์ $d(e)$, จำนวนออปสูงสุดที่เป็นไปตามเงื่อนไขจิตเตอร์ดีเลย์ $H_{\max}^{jitter}$, จำนวนลิงค์สูงสุดที่สามารถมีได้ภายในเครือข่าย $m(e)_{\max}$ ดังสมการที่ (2.1), (4.4), (4.6) และ (4.7) ตามลำดับ

อัลกอริทึมที่ได้นำเสนอนี้ จะทำการกำจัดลิงค์ในเครือข่ายที่ไม่เป็นไปตามการบังคับแบนด์วิดธ์ ดังสมการที่ (4.5) โดยในที่นี้จะสมมติว่าเราไม่สามารถทราบล่วงหน้าถึงจำนวนแบนด์วิดธ์ที่ต้องการ (r_q) โดยทั่วไปแล้วจะมีค่าเป็นเท่าใดก็ได้แต่ต้องมากกว่าหรือเท่ากับ ρ_q เพื่อว่าดีเลย์ และจิตเตอร์ดีเลย์ก็จะได้เป็นไปตามเงื่อนไขด้วย ในส่วนถัดไปจะกล่าวถึงอัลกอริทึมการเลือกเส้นทางที่คำนวณไว้ล่วงหน้า (Pre-computed Path) ซึ่งเป็นอัลกอริทึมที่ได้มาจากการปรับปรุงและพัฒนามาจากอัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS โดยวิธีของเบลแมนฟอร์ด โดยที่เราไม่ทราบคุณ

สมบัติการไหลของข้อมูลมาก่อนล่วงหน้า โดยสมมติให้โหนดต้นทางแต่ละโหนดสามารถตัดสินใจที่จะไม่ติดตั้งเส้นทาง ในกรณีที่จำนวนฮอปมากเกินกว่าจำนวน ฮอปสูงสุดที่ได้กำหนดไว้, $H_{\max}^{policy}$ โดยจำนวนอิเทอเรชั่นสูงสุดที่สามารถมีได้ภายในเครือข่ายตามอัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS โดยวิธีของเบลแมนฟอร์ด จะถูกกำหนดดังสมการที่ (4.8)

$$H_{\max} = \begin{cases} H_{\max}^{jitter} & \text{ถ้า} \quad H_{\max}^{jitter} < H_{\max}^{policy} \\ H_{\max}^{policy} & \text{กรณีอื่นๆ} \end{cases} \tag{4.8}$$

อัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS โดยวิธีของเบลแมนฟอร์ด จะทำการค้นหาเส้นทางจากโหนดต้นทางไปยังโหนดปลายทาง จนกระทั่งพบเส้นทางที่เป็นไปตามเงื่อนไขของการรับประกัน QoS ตามที่กำหนดไว้ โดยเริ่มจากอิเทอเรชั่นที่ 1, 2, ..., $H_{\max}$ ฮอป ซึ่งในแต่ละอิเทอเรชั่นก่อนที่จะมีการเพิ่มลิงค์ โหนดทมื่อยู่ปลายทางของลิงค์นั้นจะถูกตรวจสอบว่ามีบัฟเฟอร์เพียงพอหรือไม่ ถ้าหากพบว่ามีไม่เพียงพอลิงค์นั้นก็จะถูกตัดออกไป และในแต่ละอิเทอเรชั่นถ้ามีเส้นทางจากโหนดต้นทางไปยังโหนดปลายทางซึ่งสอดคล้องและเป็นไปตามข้อบังคับตามเงื่อนไขของดีเลย์ โดยถ้าเส้นทางใดมีค่าใช้จ่ายต่ำที่สุดก็จะถูกเก็บลงในตารางเส้นทาง ซึ่งการทำงานของอัลกอริทึมนี้จะสิ้นสุด เมื่อพบเส้นทางที่เป็นไปตามเงื่อนไขการรับประกัน QoS หรือเมื่อค่าของอิเทอเรชั่นมีค่าเท่ากับ $H_{\max}$ ฮอป

4.2.3 การวิเคราะห์ความถูกต้องและความซับซ้อน

ทฤษฎี 1

เมื่อระเบียบของการบริการในเครือข่ายเป็นการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม และพารามิเตอร์ของการบังคับ QoS คือ แบนด์วิดธ์ ดีเลย์ จิตเตอร์ดีเลย์ และการไม่มีการสูญเสีย อัลกอลริทึมการเลือกเส้นทางที่รับประกัน QoS วิธีของเบลแมนฟอร์ด จะค้นหาเส้นทางที่เป็นไปตามการบังคับ QoS ตามต้องการได้เสมอ ถ้าหากว่ามีเส้นทางนั้นอยู่ภายในเครือข่าย

พิสูจน์

เนื่องจากลิงค์ใด ๆ ที่ไม่มีแบนด์วิดธ์ที่ใช้การได้อย่างเพียงพอ ลิงค์นั้นจะถูกกำจัดออกก่อนที่จะดำเนินการอัลกอริทึม เส้นทางที่พบโดยอัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS วิธีของเบลแมนฟอร์ด จะประกอบด้วยลิงค์ที่เป็นไปตามเงื่อนไขที่กำหนด ■

ค่าจิตเตอร์ดีเลย์ ของเส้นทางจะขึ้นอยู่กับ จำนวนฮอปของเส้นทางเพียงอย่างเดียวเท่านั้น จำนวนฮอปสูงสุดที่ยอมได้ในเครือข่าย จะถูกกำหนดไว้ก่อนดำเนินการของอัลกอริทึม ดังสมการที่ (4.6) เพื่อเป็นการรับประกันว่าเป็นไปตามการบังคับ ค่าจิตเตอร์ดีเลย์ อัลกอลริทึมการเลือกเส้นทางที่รับประกัน QoS วิธีของเบลแมนฟอร์ด จะค้นหาเส้นทางที่มีดีเลย์น้อยที่สุด โดยเส้นทางที่มีจำนวนฮอปมากที่สุดจะถูกจำกัด โดยจำนวนอิเทอเรชั่น $(H_{\max})$ โดยการกำหนดค่านี้ ให้น้อยกว่าจำนวนที่มากที่สุดของฮอป ถูกกำหนดโดยจิตเตอร์ดีเลย์ เมื่อเส้นทางถูกค้นพบ ดังนั้นจึงแน่ใจได้ว่าเป็นไปตามการบังคับจิตเตอร์ดีเลย์

การบังคับการไม่มีการสูญเสีย ต้องการให้โหนด ใด ๆ ในเส้นทางจำเป็นจะต้องมีช่องว่างของบัฟเฟอร์ที่เพียงพอ เพื่อให้แน่ใจว่าไม่มีการสูญเสียคิวอิ่งของแพ็กเกตที่ลิงค์ที่เข้ามาของมัน จำนวนฮอปที่ยอมได้มากที่สุด $(m(e)_{\max})$ สามารถคำนวณได้สำหรับทุกลิงค์ ขึ้นอยู่กับขนาดของบัฟเฟอร์ที่ใช้การได้ของโหนดนั้น เนื่องจากอัลกอริทึมของเบลแมนฟอร์ด ค้นหาทุกเส้นทางที่สั้นที่สุด i ฮอปที่เป็นไปได้ จากเส้นทางที่สั้นที่สุดที่ $i-1$ ที่แต่ละอิเทอเรชั่น ในเบลแมนฟอร์ด ลิงค์ใด ๆ กับ $m(e)_{\max}$ ต่ำกว่าจำนวนอิเทอเรชั่นจะถูกกำจัด จึงแน่ใจว่าการสื่อสารจะไม่มีการสูญเสีย

ในอัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS วิธีของเบลแมนฟอร์ด ของเราค้นหาเส้นทางที่มีดีเลย์สั้นที่สุด ถ้าเราล้มเหลวในการค้นหาทางเดินที่เป็นไปได้ตามการบังคับดีเลย์ จึงกล่าวได้ว่าไม่มีเส้นทางนั้นภายในเครือข่าย

ด้วยเหตุนี้ อัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS วิธีของเบลแมนฟอร์ด โดยการใช้ระเบียบบริการการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม ทำให้สามารถค้นหาเส้นทางที่เป็นไปตามการบังคับแบนด์วิดธ์ ดีเลย์ จิตเตอร์ดีเลย์ และช่องว่างของบัฟเฟอร์ ถ้าหากว่ามีเส้นทางอยู่อย่างน้อยหนึ่งเส้นทางภายในเครือข่าย

ทฤษฎี 2

เวลาการคำนวณในกรณีที่เลวที่สุดของอัลกอริทึมการเลือกเส้นทางที่รับประกันคุณภาพวิธีของเบลแมนฟอร์ด ดั้งเดิม คือลำดับของ $O(|E| \cdot H_{\max})$ โดยที่ $|E|$ เป็นจำนวนลิงค์ในเครือข่าย และ $H_{\max}$ คือจำนวนอิเทอเรชั่นที่มากที่สุด ของอัลกอริทึมวิธีของเบลแมนฟอร์ด

พิสูจน์

พิจารณา ภาคผนวก ก. แสดงรหัสเทียมของอัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS วิธีของเบลแมนฟอร์ด ลูปของเบลแมนฟอร์ดมีการคำนวณ $(O(|E|))$ ครั้ง ในลูปการลดความซับซ้อนมีการคำนวณ $(O(|E|))$ ครั้ง ในลูปการเริ่มต้นหาทางเดินจากโหนดต้นทางมีการคำนวณ $(O(|E|))$ ครั้งในส่วนการลดความซับซ้อนมีการคำนวณให้สำเร็จในครั้งเดียว ส่วนการเริ่มต้นหา

ทางเดินจากโหนดต้นทางมีการคำนวณ $H_{\max}+1$ และในส่วนของรีแล็ก (Relax) $H_{\max}$ ครั้ง ดังนั้น เวลารวมทั้งหมดมีค่าเท่ากับ $O(|E|)+O(|E|\cdot(H_{\max}+1))+O(|E|\cdot H_{\max})=O(|E|)\cdot H_{\max}$ ดังนั้นกรณีที่เลวที่สุดของการคำนวณ ของอัลกอริทึมนี้มีค่าเท่ากับ $O(|E|\cdot H_{\max})$ ■

4.3 อัลกอริทึมการเลือกเส้นทางที่รับประกันคุณภาพการบริการสำหรับทางเดินที่คำนวณไว้ล่วงหน้า ($k-WSP_{EP}$)

ในส่วนก่อนหน้านี้เราได้อธิบายถึงอัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS วิธีของเบลแมนฟอร์ด เป็นอัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS ที่มีพื้นฐานความต้องการของโหนดต้นทาง กล่าวคือเราสมมุติว่าเราทราบล่วงหน้าเกี่ยวกับลักษณะการไหล และอัตราการรับประกัน ในส่วนนี้เรานำเสนอของอัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS ที่เรียกว่า $k-WSP_{EP}$ ในที่นี้ WSP เป็นอัลกอริทึมการเลือกเส้นทางที่มีจำนวนฮอปน้อยที่สุด ถ้าหากว่ามีเส้นทางมากกว่าหนึ่งเส้นทางแล้ว เส้นทางที่มีแบนด์วิดธ์มากที่สุดจะถูกเลือก และ k เป็นจำนวนแบนด์วิดธ์ ในแต่ละอิเทอเรชั่นของเบลแมนฟอร์ดจะมีการค้นหาเส้นทางที่มีแบนด์วิดธ์มากที่สุดอันดับที่หนึ่ง เส้นทางที่มีแบนด์วิดธ์มากอันดับที่สอง เส้นทางที่มีแบนด์วิดธ์มากอันดับที่สาม และดำเนินอัลกอริทึมตามขั้นตอนดังกล่าวนี้ไปเรื่อย ๆ จนกระทั่งถึงเส้นทางที่มีแบนด์วิดธ์มากอันดับที่ k สำหรับ $k-WSP_{EP}$ สามารถสร้างตารางการเลือกเส้นทาง ที่รองรับการร้องขอการจราจรที่มีข้อกำหนดคุณลักษณะของการไหลแต่ละพารามิเตอร์ ซึ่งอาจเป็นค่าใด ๆ ที่ไม่รู้ล่วงหน้ามาก่อน

ค่าที่เป็นไปได้มากที่สุดของค่า k เท่ากับจำนวนที่มากที่สุดของแบนด์วิดธ์ที่เหลือที่แตกต่าง K โดยที่ $K\le|E|$ โดยที่ $|E|$ เป็นจำนวนของลิงค์ในเครือข่าย ดังนั้นในกรณีที่เลวที่สุดขนาดของตารางเท่ากับ K คูณ กับ WSP อย่างไรก็ตามเราแสดงว่าในอิเทอเรชั่น วิธีของเบลแมนฟอร์ดไม่ใช่ทุก ๆ ค่า K ที่เป็นไปได้ เส้นทางที่มีจำนวนแบนด์วิดธ์มากที่สุดโดยที่มีค่าดีเลย์น้อยที่สุด เส้นทางนั้นเป็นเส้นทางที่จำเป็นที่จะถูกเก็บไว้ในตารางเส้นทาง เส้นทางที่เป็นไปตามเงื่อนไข ที่ได้กำหนดไว้จะเป็นเส้นทางที่เกินความจำเป็น เส้นทางนั้นจะถูกละทิ้ง เพราะว่าเส้นทางที่ดีกว่าได้ถูกเก็บไว้ในตารางเรียบร้อยแล้ว ทั้งนี้ขึ้นอยู่กับชนิดของเครือข่าย รูปร่างของเครือข่าย และสถานการณ์การจราจรภายในเครือข่าย ดังนั้น ตารางเส้นทางจึงสามารถถูกลดขนาดลงได้

เมื่อ $k = K$, $k-WSP_{EP}$ จะสามารถหาเส้นทางที่มีจำนวนฮอปที่ต่ำที่สุดเสมอ ซึ่งเป็นไปตามเงื่อนไข การบังคับแบนด์วิดธ์ ดีเลย์ และจิตเตอร์ดีเลย์ในเวลาโพลิโนเมียล ถ้าหากว่ามีเส้นทางนั้นอยู่ภายในเครือข่าย อีกทั้งเป็นไปตามเงื่อนไขการบังคับขนาดของบัฟเฟอร์อีกด้วย กรณีที่เลวที่สุดของความซับซ้อนของอัลกอริทึมนี้คือ $O(KH_{\max}|E|)$ โดยที่ $H_{\max}$ เป็นขอบเขตบนของจำนวนฮอปของเส้นทางนั้น

4.3.1 รายละเอียดของอัลกอริทึม

อัลกอริทึม $k-WSP_{EP}$ เป็นอัลกอริทึมที่ปรับปรุงมาจาก WSP ซึ่งปรับปรุงมาจากอัลกอริทึมวิธีของเบลแมนฟอร์ด ซึ่งอัลกอริทึมวิธีของเบลแมนฟอร์ดถูกนำเสนอโดย เกอรินและคณะ [2] และมาและคณะ [16] ในอิเทอเรชั่นแรก อัลกอริทึมนี้จะหาจำนวนฮอป $k-WSP$ หนึ่งออป (WSP ลำดับที่หนึ่ง WSP ลำดับที่สอง WSP ลำดับที่สาม ไปเรื่อย ๆ จนกระทั่งถึง WSP ลำดับที่ k) จากโหนดต้นทางจนถึงโหนดปลายทาง ในเครือข่าย ในอิเทอเรชั่นที่สอง อัลกอริทึมนี้จะหาจำนวนฮอป $k-WSP$ สองฮอป และดำเนินการอย่างนี้ในอิเทเรชั่นที่สูงขึ้นไปกับจำนวนฮอปที่มากขึ้น และจะลิ้นสุดเมื่อถึงหลังอิเทอเรชั่นที่ $H_{\max}$ โดยที่ $H_{\max}$ จำนวนฮอปมากที่สุดที่กำหนดไว้ล่วงหน้า ถ้าหากว่ามีจำนวนเส้นทางมากกว่าหนึ่งเส้นทางที่มีแบนด์วิดธ์คอขวดเท่ากัน ให้เลือกเส้นทางที่มีดีเลย์สถิตที่ต่ำที่สุดเก็บไว้ และตัวอย่างการคำนวณเส้นทางโดยใช้อัลกอริทึม $k-WSP$ และ $k-WSP_{EP}$ แสดงในภาคผนวก ข.

สมมุติว่าเราได้กำหนดค่า k ล่วงหน้า ถ้า $k=K$ เราจะสามารถหาเส้นทางที่มี QoS ที่เป็นไปตามการบังคับแบนด์วิดธ์ ดีเลย์ จิตเตอร์ดีเลย์ในเวลาโพลิโนเมียล ได้เสมอ ถ้าหากว่ามีเส้นทางนั้นอยู่ภายในเครือข่าย อัลกอริทึมนี้ขึ้นอยู่กับลักษณะของเครือข่าย เมื่อ $k<<K$ เราจะสามารถประสพความสำเร็จในการค้นหาเส้นทางด้วยความน่าจะเป็นสูง กล่าวคือ ในทางปฏิบัติแล้วเส้นทาง $K-WSP$ ทุกเส้นทางไม่จำเป็นที่จะต้องถูกเก็บไว้ในตารางเส้นทาง เนื่องจากว่าบางเส้นทางนั้นเกินความจำเป็นและสามารถที่จะถูกตัดทิ้งได้ ดังนั้นขนาดของตารางเส้นทางจึงมีขนาดเล็กลงได้

ดีเลย์ของลิงค์ประกอบด้วย คิวอิ่งดีเลย์ ทรานสมิตชั่นดีเลย์ และ พรอพพิเกชั่นดีเลย์ ซึ่งค่าคิวอิ่งดีเลย์เป็นค่าเดียวกันกับค่าจิตเตอร์ดีเลย์ ทั้งสองค่าเป็นค่าพลวัต กล่าวคือค่าทั้งสองจะเปลี่ยนแปลงขึ้นอยู่กับค่าข้อกำหนดการไหลที่ถูกร้องขอ $\left(\sigma_q,\rho_q,S^q_{\max}\right)$ [17] และจำนวนแบนด์วิดธ์ ที่ถูกจอง $\left(r_q\right)$ สำหรับการไหล q สำหรับ $k-WSP_{EP}$ พารามิเตอร์ซึ่งเป็นข้อกำหนดการไหลและ r_q ไม่ทราบล่วงหน้ามาก่อน ค่ามากที่สุดของ r_q ของเส้นทางสามารถถูกจองเป็นค่าแบนด์วิดธ์คอขวดของเส้นทาง ดังนั้นค่าคิวอิ่งดีเลย์ต่ำสุดจะขึ้นอยู่กับค่าแบนด์วิดธ์คอขวดของเส้นทางด้วย สำหรับค่าทรานสมิตชั่นดีเลย์ และพรอพพิเกชั่นดีเลย์ เป็นค่าสถิต ทั้งสองค่านี้ขึ้นอยู่กับพารามิเตอร์ที่มีค่าคงที่ ยกตัวอย่างเช่น ความจุแบนด์วิดธ์ของลิงค์ ความยาวของลิงค์ ตัวกลางที่ส่งผ่านข้อมูล เป็นต้น ดังนั้นค่าดีเลย์ลิงค์สถิตถูกนิยาม ดังสมการที่ (4.9)

$$d_{static}(e) = \frac{S_{\max}}{R(e)} + pd(e) \qquad (4.9)$$

ค่าเอนทูเอนดีเลย์ของเส้นทาง P, $D(P)$ ถูกนิยามดังสมการที่ (4.10)

$$D(P) = D_{static}(P) + D_{dynamic}(P) \quad (4.10)$$

โดยที่

$$D_{static}(P) = \sum_{e \in P} d_{static}(e) \quad (4.11)$$

$$D_{dynamic}(P) = \frac{\sigma_q + H(P)S^q_{\max}}{r_q} \quad (4.12)$$

โดยที่ r_q ที่ถูกเลือกจะเท่ากับค่าแบนด์วิดธ์คอขวด $B(P)$ ของเส้นทาง P เราได้ค่าเอนทูเอนดีเลย์ต่ำที่สุดของเส้นทาง $P, D^{\min}(P)$ ซึ่งสามารถนิยามได้ดังสมการที่ (4.13) และ (4.14)

$$D^{\min}(P) = D_{static}(P) + D^{\min}_{dynamic}(P) \quad (4.13)$$

โดยที่

$$D^{\min}_{dynamic}(P) = \frac{\sigma_q + H(P)S^q_{\max}}{B(P)} \quad (4.14)$$

ค่าจิตเตอร์ดีเลย์ และค่าจิตเตอร์ดีเลย์ที่ต่ำที่สุดของเส้นทาง P สามารถนิยามโดยใช้สมการที่ (4.15) และ (4.16)

$$J(P) = D_{dynamic}(P) \quad (4.15)$$

$$J^{\min}(P) = D^{\min}_{dynamic}(P) \quad (4.16)$$

อิทเทอเรชั่นที่ i ของ $k-WSP_{EP}$ ซึ่งมี i ฮอปถึงทุกโหนดที่สามารถไปถึงได้จะถูกค้นพบ ถ้าหากว่า มีเส้นทางที่มีแบนด์วิดธ์คอขวดเท่ากัน มากกว่าหนึ่งเส้นทาง จะเลือกเก็บเส้นทางที่มี $D_{static}(P)$ ที่ต่ำที่สุดของอิเทอเรชั่นนั้น หลังจากเส้นทาง P ถูกค้นพบเราจะได้ค่า $H(P), B(P)$ และ $D_{static}(P)$ สำหรับค่า $D^{\min}(P)$ และ $J^{\min}(P)$ สามารถคำนวณได้ ให้เส้นทางนี้คือ P_y

สมมุติให้ มี $p-WSPs$ ถูกค้นพบ และเก็บไว้ในตารางเส้นทาง สำหรับมีคู่โหนดต้นทาง และโหนดปลายทางเดียวกัน และสมมุติให้เป็น P_x ซึ่งมีจำนวนฮอปน้อยกว่าหรือเท่ากับ P_y

ถ้า P_x มีแบนด์วิดธ์คอขวดมากกว่า หรือเท่ากับ และมีดีเลย์สถิตต่ำกว่าหรือเท่ากับ P_y แล้วเป็นไปตามเงื่อนไข ดังสมการที่ (4.17), (4.18) และ (4.19) ดังนั้นในกรณีนี้ เราไม่จำเป็นที่จะต้องเก็บ P_y ลงในตารางเส้นทาง เนื่องจากว่าเส้นทาง P_x เป็นเส้นทางที่ดีกว่าถูกเก็บไว้แล้ว

$$B(P_y) \leq B(P_x) \tag{4.17}$$

$$H(P_y) \geq H(P_x) \tag{4.18}$$

$$D_{static}(P_y) \geq D_{static}(P_x) \tag{4.19}$$

เมื่อเป็นไปตามเงื่อนไขดังสมการที่ (4.17) ~ (4.19) แล้วสำหรับการไหลที่มีข้อกำหนดเดียวกัน เส้นทางที่มีแบนด์วิดธ์คอขวดมากกว่า หรือเท่ากับ P_x และมีดีเลย์พลวัตต่ำกว่า หรือเท่ากับ ดีเลย์พลวัตที่ต่ำที่สุด เมื่อเปรียบเทียบกับเส้นทาง P_y โดยพิจารณาจากสมการที่ (4.14) ดังนั้นถ้า P_x มีดีเลย์สถิตต่ำกว่า หรือเท่ากับ และจำนวนฮอปต่ำกว่าหรือเท่ากับ P_y และ P_y จะมีค่าดีเลย์ที่ต่ำที่สุด สูงกว่า หรือเท่ากับ P_x คิวอิ่งดีเลย์เท่ากับจิตเตอร์ดีเลย์ของเส้นทาง ดังนั้น เส้นทาง P_y จะไม่มีจำนวนฮอป จิตเตอร์ดีเลย์ต่ำที่สุด สูงกว่าหรือเท่ากับ เส้นทาง P_x ดังนั้นเส้นทาง P_y จึงเป็นเส้นทางที่เกินความจำเป็นและไม่จำเป็นที่จะต้องเก็บเส้นทาง P_y ลงในตารางเส้นทาง

ปัจจัยที่ทำให้เส้นทาง P_y มีค่าดีเลย์ทั้งหมด ต่ำกว่าเส้นทาง P_x คือ ทรานสมิตชั่นดีเลย์ และพรอพพิเกชั่นดีเลย์ของเส้นทาง P_y กล่าวคือเส้นทาง P_y จะมีคิวอิ่งดีเลย์ที่ต่ำที่สุด สูงกว่าหรือเท่ากับของเส้นทาง P_x เสมอ เส้นทางที่ใช้ลิงค์ที่มีความเร็วสูง จะมีทรานสมิตชั่นดีเลย์ต่ำ เช่นเดียวกันกับเส้นทางที่มีแบนด์วิดธ์คอขวดสูงจะมีมีทรานสมิตชั่นดีเลย์ต่ำ สำหรับ พรอพพิเกชั่นดีเลย์จะขึ้นอยู่กับสื่อกลางที่ใช้ในการส่งผ่านข้อมูล และความยาวของลิงค์ ลิงค์ที่มีความเร็วสูงโดยทั่วไปแล้วจะเป็นเส้นใยแก้วนำแสง และมีพรอพพิเกชั่นดีเลย์ต่ำกว่าสื่อที่ใช้ส่งผ่านข้อมูลอื่น ๆ เส้นทางที่มีจำนวนฮอปมากกว่ามักจะมีแนวโน้มที่จะมีดีเลย์สถิตที่สูงกว่า ในหลาย ๆ โอกาสสำหรับเส้นทางที่ใช้ลิงค์ที่มีความเร็วต่ำแล้วมักจะเป็นไปตามเงื่อนไขดังสมการที่ (4.17), (4.18) และ (4.19) อย่างพร้อมเพรียงกัน ดังนั้นจำนวนเส้นทางหลายเส้นทางจำเป็นจะถูกเก็บไว้ในตารางเส้นทาง จะมีจำนวนน้อยกว่าจำนวนที่น่าจะเป็นไปได้มากที่สุด เช่น โดยทั่วไป $k << K$

ขณะที่มีการคัดเลือกเส้นทางจากทุกเส้นทางที่มีในตารางเส้นทาง สำหรับจำนวนฮอปเท่ากับ i เส้นทางที่มีจำนวนฮอปที่ต่ำที่สุดจะถูกตรวจสอบการบังคับแบนด์วิดธ์อันดับแรก ดีเลย์ และ

จิตเตอร์ดีเลย์ โดยใช้สมการที่ (4.20), (4.21) และ (4.22) เส้นทางที่มีแบนด์วิดธ์มากที่สุดจะถูกตรวจสอบก่อน ถ้าไม่เป็นไปตามการบังคับดีเลย์แล้ว เส้นทางที่มีแบนด์วิดธ์ถัดไปจะถูกตรวจสอบ จะดำเนินเช่นนี้ต่อไปเรื่อย ๆ จนกระทั่งเป็นไปตามเงื่อนไขใดเงื่อนไขหนึ่งดังนี้

(1) ไม่เป็นไปตามการบังคับดีเลย์หรือแบนด์วิดธ์

หรือ (2) ไม่มีเส้นทางในตารางเส้นทางสำหรับจำนวนฮอปนั้น

หลังจากนั้นจึงมีการพิจารณา เส้นทางที่มีจำนวนฮอปมากกว่าหนึ่งฮอป $(=i+1)$ เช่นเดียวกันกับที่กล่าวมาข้างต้น คือจะหาเส้นทางที่มีแบนด์วิดธ์มากที่สุดเป็นอันดับแรก การค้นหาเส้นทางดำเนินต่อไปเรื่อย ๆ จนกระทั่งทุกเส้นทางมีจำนวนฮอปมากที่สุดเป็น $H_{\max}$ แล้วจึงจะเสร็จสิ้น

$$B(P) \geq \Delta_{bandwidth} \tag{4.20}$$

โดยที่ $\Delta_{bandwidth} = \rho_q$

$$D^{\min}(P) \leq \Delta_{delay} \tag{4.21}$$

$$J^{\min}(P) \leq \Delta_{jitter} \tag{4.22}$$

จำนวนแบนด์วิดธ์ที่ต่ำที่สุด r_q จะถูกจองสำหรับเส้นทางที่ถูกเลือก ซึ่งสามารถคำนวณได้จากสมการที่ (4.23), (4.24) และ (4.25) โดยที่ r_q^{delay} และ r_q^{jitter} เป็นแบนด์วิดธ์ที่ต่ำที่สุดที่ต้องการเพื่อให้เป็นไปตามการบังคับดีเลย์ และจิตเตอร์ดีเลย์ตามลำดับ

$$r_q^{delay} = \frac{\sigma_q + H(P)S^q_{\max}}{\Delta_{delay} - D_{static}(P)} \tag{4.23}$$

$$r_q^{jitter} = \frac{\sigma_q + H(P)S^q_{\max}}{\Delta_{jitter}} \tag{4.24}$$

$$r_q = MAX\{\sigma_q, r_q^{delay}, r_q^{jitter}\} \tag{4.25}$$

4.3.1.1 ตารางเส้นทางที่มีคุณภาพของการบริการ

ตารางเส้นทางที่มี QoS จะมีลักษณะเป็นเมตริก 3 มิติ คือ $\left[|V| \bullet H_{\max} \bullet k\right]$

โดยที่

$|V|$ เป็นจำนวนของโหนดปลายทางในเครือข่าย

$H_{\max}$ เป็นจำนวนฮอปที่มากที่สุดสำหรับเส้นทางที่ยอมให้มีได้

k เป็นจำนวน *WSPs* ถึงปลายทาง

ตารางเส้นทางที่เข้ามา $[u,v,w]$ ที่เข้ามาที่โหนด x จะมีข่าวสารเส้นทางของ *WSP* ลำดับที่ w ของเส้นทาง $P(x,u)$ จากโหนดต้นทาง x ถึงโหนดปลายทาง u ซึ่งมีจำนวนฮอปเท่ากับ v ฮอป ดังนั้นข่าวสารที่เข้ามาแต่ละครั้งจึงมี 3 ฟิลด์ (Field) ดังต่อไปนี้

(1) แบนด์วิดธ์คอขวดของเส้นทาง : $B(P(x,u))$

(2) โหนดข้างเคียง : ที่อยู่ของเส้นทางที่อยู่ข้างเคียงถัดไป $P(x,u)$

(3) ดีเลย์สถิตของเส้นทาง : $D_{static}(P(x,u))$

4.3.1.2 การวิเคราะห์ความถูกต้องและความซับซ้อน

การวิเคราะห์ความถูกต้องและความซับซ้อน ของอัลกอริทึม $K-WSP_{EP}$

ทฤษฎีที่ 3

เมื่อระเบียบการให้บริการที่ใช้ในเครือข่ายคือ การจัดแถวคอยให้น้ำหนักอย่างเป็นธรรม และพารามิเตอร์ของการบังคับ QoS เช่น แบนด์วิดธ์คอขวด ดีเลย์ และจิตเตอร์ดีเลย์ อัลกอริทึม $K-WSP_{EP}$ สามารถหาเส้นทางที่คำนวณไว้ล่วงหน้า ถ้าหากว่ามีเส้นทางที่เป็นไปตามการบังคับได้เสมอ ถ้าหากว่ามีเส้นทางนั้นอยู่ภายในเครือข่าย

พิสูจน์

มีแบนด์วิดธ์ของลิงค์ในเครือข่ายที่มีอยู่ที่เป็นไปได้ K ค่า ถ้า $k=K$ ยกตัวอย่างเช่น ในแต่ละอิเทอเรชั่นของอัลกอริทึม $K-WSP_{EP}$ เราจะได้เส้นทางมีดีเลย์ที่สั้นต่ำที่สุดและ ค่าแบนด์วิดธ์ที่มากที่สุดที่เป็นไปได้ อย่างมากที่สุด K ค่า ที่มีแบนด์วิดธ์คอขวดของเส้นทางจะถูกเรียงลำดับจากค่าสูงที่สุด $(1^{st}WSP)$ ถึงต่ำที่สุด $(K^{th}WSP)$ ดังนั้น $K-WSP_{EP}$ สามารถหาเส้นทางที่

เป็นไปตามการบังคับแบนด์วิดธ์ได้เสมอ ถ้าหากว่ามีเส้นทางนั้นอยู่ภายในเครือข่ายภายในเครือข่าย ■

คิวอิ่งดีเลย์ขึ้นอยู่กับข้อกำหนดของการไหล เช่น ขนาดของเบิสท์ที่ใหญ่ที่สุด (σ_q) และขนาดของแพ็กเกตที่ใหญ่ที่สุด $(S^q_{\max})$ และอัตราการรับประกันของข้อกำหนดของการไหล (r_q) เราสมมุติว่าได้กำหนดขนาดของเบิสท์ที่ใหญ่ที่สุดและขนาดของแพ็กเกตที่ใหญ่ที่สุดแล้ว คิวอิ่งดีเลย์ของเส้นทาง จะมีค่าต่ำที่สุดเมื่ออัตราการรับประกันนั้นสูงที่สุด สำหรับเส้นทางใด ๆ ที่มีจำนวนฮอปเท่ากันแล้ว เส้นทางที่มีอัตราการรับประกันที่สูงที่สุดจะเป็นแบนด์วิดธ์คอขวดของเส้นทางนั้น เส้นทางที่มีแบนด์วิดธ์คอขวดสูงกว่า จะมีคิวอิ่งดีเลย์ที่ต่ำกว่าเส้นทางที่มีแบนด์วิดธ์คอขวดต่ำกว่า สำหรับการจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรมแล้ว จิตเตอร์ดีเลย์มีค่าเท่ากับคิวอิ่งดีเลย์ของเส้นทาง ที่อิเทอเรชั่นใด ๆ ก็ตาม (จำนวนฮอป) ค่า $K-WSP_{EP}$ ที่มากที่สุดจะถูกค้นพบค่าคิวอิ่งดีเลย์จะเรียงลำดับจากต่ำที่สุด $(1^{st}WSP)$ ถึงสูงที่สุด $(K^{th}WSP)$ ดังนั้น $K-WSP_{EP}$ จะได้เส้นทางที่เป็นไปตามการบังคับจิตเตอร์ดีเลย์เสมอ ถ้าหากว่ามีเส้นทางนั้นอยู่ภายในเครือข่าย

ค่าเอนทูเอนดีเลย์ทั้งหมดคือผลรวมของคิวอิ่งดีเลย์ ทรานสมิตชั่นดีเลย์ และพรอพพิเกชั่นดีเลย์ ค่า $K-WSPs$ ที่แตกต่างกัน (กล่าวคือค่าแบนด์วิดธ์คอขวดที่แตกต่างกัน) จะมีค่า ทรานสมิตชั่นดีเลย์ และพรอพพิเกชั่นดีเลย์ (ดีเลย์สถิต) ที่แตกต่างกัน ดีเลย์สถิตจะไม่เปลี่ยนไปตามโหลดหรือแบนด์วิดธ์คอขวดของเส้นทาง ดังได้อธิบายมาแล้ว ถ้าหากว่ามีเส้นทางที่มีแบนด์วิดธ์คอขวดเท่ากันมากกว่าหนึ่งเส้นทางแล้ว เส้นทางที่มีค่าดีเลย์สถิต (ทรานสมิตชั่นดีเลย์ และพรอพพิเกชั่นดีเลย์) ต่ำที่สุดจะถูกเลือก สำหรับแต่ละอิเทอเรชั่น อัลกอลิทึม $K-WSP_{EP}$ จะไม่บันทึกข่าวสารการจัดเส้นทางของเส้นทาง P ถ้าเส้นทาง P มีแบนด์วิดธ์คอขวดต่ำกว่าหรือเท่ากับ และดีเลย์สถิตต่ำกว่าหรือเท่ากับ เส้นทางที่ถูกค้นพบแล้ว เส้นทางที่มีอยู่นี้ดีกว่า ในทำนองเดียวกัน บางเส้นทางที่ถูกค้นพบแล้วจะถูกละทิ้งด้วย ถ้าหากว่ามีเส้นทางใหม่ที่ดีกว่า การละทิ้งเส้นทางจะไม่มีผลกระทบกับประสิทธิภาพของอัลกอริทึม $K-WSP_{EP}$ ในการหาเส้นทางที่มี QoS

ในแต่ละอิเทอเรชั่น ดีเลย์สถิตจะถูกเรียงจากมากที่สุด $(1^{st}WSP)$ ถึงต่ำที่สุด $(K^{th}WSP)$ เราจะได้เส้นทางทุกค่า K ที่เป็นไปได้ กับคิวอิ่งดีเลย์ และดีเลย์สถิตที่แตกต่างกัน ซึ่งจะประกอบด้วยคิวอิ่งดีเลย์ที่ต่ำที่สุด และดีเลย์สถิตที่ต่ำที่สุด สำหรับแต่ละค่า K ที่เป็นไปได้ที่แตกต่างกันของแบนด์วิดธ์ของลิงค์ที่เหลืออยู่ ดังนั้นเราจะได้ $K-WSPs$ กับเอนทูเอนดีเลย์ทั้งหมดที่แตกต่างกันที่อิเทอเรชั่น (จำนวนฮอป) ถ้ามีเส้นทางที่เป็นไปตามเงื่อนไขการบังคับแบนด์วิดธ์ ดีเลย์ และจิตเตอร์ดีเลย์แล้ว เส้นทางนั้นจะเป็นเส้นทางหนึ่งใน $K-WSPs$ [2] โดยการดำเนินการตามอัลกอริทึม $K-WSP_{EP}$ ที่ อิเทอเรชั่นที่ $H_{\max}$ ถ้าหากว่ามีเส้นทางที่เป็นไปตามเงื่อนไขการบังคับแบนด์วิดธ์ ดีเลย์ และจิตเตอร์ดีเลย์ แล้วอัลกอริทึม $K-WSP_{EP}$ จะหาเส้นทางนั้นได้เสมอ

ทฤษฎีที่ 4

ความซับซ้อนของอัลกอริทึม $K-WSP_{EP}$ คือ $O\left(KH_{\max}^{policy}|E|\right)$

โดยที่

K เป็นค่าที่แตกต่างกันทั้งหมดของค่าแบนด์วิดธ์ที่เหลืออยู่ของลิงค์ที่เป็นไปได้ทั้งหมด

$H_{\max}^{policy}$ เป็นจำนวนของอิเทอเรชั่นของอัลกอริทึมวิธีของเบลแมนฟอร์ด

$|E|$ เป็นจำนวนของลิงค์ในเครือข่าย

พิสูจน์

อัลกอริทึม $K-WSP_{EP}$ จะคล้ายกับอัลกอริทึม $QoSR_{BF}$ กล่าวคืออัลกอริทึมนี้จะเริ่มจากการจัดเรียงแบนด์วิดธ์ที่เหลืออยู่ ของลิงค์ที่เป็นไปได้ทั้งหมด (K) โดยเรียงจากมากไปหาน้อย ซึ่งมีการทำงานเช่นเดียวกับอัลกอริทึมการเลือกเส้นทางที่มีการรับประกัน QoS โดยใช้วิธีของเบลแมนฟอร์ด ดังแสดงในบล็อกของ รีแล็ก ดังแสดงในภาคผนวก ก. ดังนั้นความซับซ้อนของอัลกอริทึม $K-WSP_{EP}$ คือ $O\left(KH_{\max}^{policy}|E|\right)$ โดยที่ K เป็นค่าที่แตกต่างกันทั้งหมดของค่าแบนด์วิดธ์ที่เหลืออยู่ของลิงค์ที่เป็นไปได้ทั้งหมด $H_{\max}^{policy}$ เป็นจำนวนของอิเทอเรชั่นของอัลกอริทึมวิธีของเบลแมนฟอร์ด และ $|E|$ เป็นจำนวนของลิงค์ในเครือข่าย ■

บทที่ 5
การซิมมูเลชั่น และผลการซิมมูเลชั่น

5.1 การซิมมูเลชั่น

5.1.1 การสร้างเครือข่ายแบบสุ่ม

แบบจำลองของ แวกซ์แมน (Waxman Model) [18] เป็นแบบจำลองของเครือข่ายที่ใช้ในการซิมมูเลชั่นนี้เป็นเครือข่ายแบบสุ่ม โดยเครือข่ายที่ได้จะมีลักษณะใกล้เคียงกับเครือข่ายที่มีใช้งานอยู่จริงในปัจจุบัน แบบจำลองของเครือข่ายจะมีลักษณะเป็นกราฟโดยมีโหนด n โหนดกระจายแบบนอร์มอล (Normal Distribution) อยู่ในตำแหน่งต่าง ๆ ในพื้นที่สี่เหลี่ยมผืนผ้า (Cartesian Coordinate Grid) โดยเส้นเชื่อมโยง (Edges) ระหว่างแต่ละคู่ของโหนด (u, v) ภายในเครือข่าย จะขึ้นอยู่กับฟังก์ชั่นความน่าจะเป็นของการเชื่อมต่อแต่ละคู่ของโหนด $P_e(u,v)$ ดังสมการที่ (5.1)

$$P_e(u,v) = \beta \quad X \quad \exp\left(\frac{-d}{\alpha L}\right) \qquad (5.1)$$

โดยที่

β จะมีค่าเพิ่มขึ้นตามจำนวนของเส้นเชื่อมโยงระหว่างโหนดต่าง ๆ มีค่า (0,1]

d เป็นระยะทางระหว่างโหนด u ละโหนด v

α จะมีค่าเพิ่มขึ้นตามระยะทางระหว่างโหนด u และโหนด v มีค่า (0,1]

L เป็นระยะทางที่มากที่สุดที่เป็นได้ ระหว่างโหนด u และโหนด v

ในวิทยานิพนธ์นี้ ค่า $\beta = 0.25$ และ $\alpha = 0.20$ ซึ่งจะทำให้ได้เครือข่ายที่มีลักษณะใกล้เคียงกับเครือข่ายที่มีใช้งานอยู่จริงในปัจจุบัน

สำหรับค่าน้ำหนัก (Weight) ของเส้นที่เชื่อมโยงระหว่างแต่ละคู่ของโหนดนั้น จะแทนด้วย $w(u,v)$ ซึ่งภายในเครือข่ายที่ได้จำลองขึ้นนี้ค่า $w(u,v)$ จะถูกกำหนดให้เป็นค่าระยะทางระหว่างโหนด u และโหนด v ใด ๆ

แบบจำลองของเครือข่ายที่สร้างขึ้นในการทดลองนี้ ได้ผ่านการทดสอบแล้วว่าทุก ๆ โหนดภายในเครือข่าย ได้มีการเชื่อมต่อกัน

5.1.2 วิธีการซิมมูเลชั่น

แบบจำลองเครือข่ายในวิทยานิพนธ์นี้ มีสื่อกลาง (Media) เป็นสายใยแก้วนำแสง (Fiber Optic Cable) ทิศทางการสื่อสารข้อมูลเป็นแบบ สองทิศทางพร้อมกัน (Full Duplex) ลิงค์มีลักษณะเหมือนกัน (Homogeneous) คือมีความจุของแบนด์วิดธ์ขนาด 155.52 เมกกะบิตต่อวินาที (OC-3) เครือข่ายแบบสุ่ม ถูกสร้างขึ้นตามโมเดลของแวกซ์แมน โดยที่ตำแหน่งของโหนดถูกกำหนดให้อยู่ภายในสี่เหลี่ยมผืนผ้า ที่มีขนาดเท่ากับ 4000 x 2400 ตารางกิโลเมตร ซึ่งเป็นขนาดของประเทศสหรัฐอเมริกาอย่างคร่าว ๆ โหนดทั้งหมดของเครือข่ายเชื่อมต่อกันด้วยดีกรีอย่างน้อยที่สุดเท่ากับ 2 และมีดีกรีเฉลี่ยเท่ากับ 3 รูปที่ 5.1, 5.2, 5.3 และ 5.4 แสดงตัวอย่างเครือข่ายแบบสุ่มที่ใช้ในการทดลอง มีจำนวนโหนดภายในเครือข่ายเท่ากับ 15, 30, 45 และ 60 โหนด ในวิทยานิพนธ์นี้เริ่มต้น โดยแบ่งการจราจรพื้นฐานให้มีค่าแบนด์วิดธ์ที่เหลืออยู่ มีลักษณะเป็นค่าสุ่มที่ไม่ต่อเนื่อง ในแต่ละลิงค์เป็น 3 กรณี ดังต่อไปนี้

กรณีที่ 1 :กำหนดแบนด์วิดธ์ที่เหลืออยู่ของลิงค์ในเครือข่ายมีค่าแตกต่างกันทั้งหมด 31 ค่า (K=31) มีค่าดังต่อไปนี้ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115, 120, 125, 130, 135, 140, 145, 150 และ 155 เมกกะบิตต่อวินาที

กรณีที่ 2 : กำหนดแบนด์วิดธ์ที่เหลืออยู่ของลิงค์ในเครือข่ายมีค่าแตกต่างกันทั้งหมด 16 ค่า (K=16) มีค่าดังต่อไปนี้ 5, 15, 25, 35, 45, 55, 65, 75, 85, 95, 105, 115, 125, 135, 145 และ 155 เมกกะบิตต่อวินาที

กรณีที่ 3 : กำหนดแบนด์วิดธ์ที่เหลืออยู่ของลิงค์ในเครือข่ายมีค่าแตกต่างกันทั้งหมด 7 ค่า (K=7) มีค่าดังต่อไปนี้ 5, 30, 55, 80, 105, 130,และ 155 เมกกะบิตต่อวินาที

ในวิทยานิพนธ์นี้ ได้ทำการสร้างเครือข่ายแบบสุ่มอีกกรณีละ 5 เครือข่าย และในแต่ละเครือข่ายได้ทำการเลือก 10 คู่ของโหนดต้นทางและโหนดปลายทางอย่างสุ่ม เพื่อคำนวณหาทางเดินที่มีการรับประกัน QoS โดยใช้อัลกอลิทึม $k^{th} - WSP$ และ $k - WSP_{EP}$ แต่ละจุดที่พล็อตกราฟคำนวณได้จาก จำนวนเส้นทางเฉลี่ยต่อโหนดต้นทางและปลายทาง 1 คู่ ดังแสดงในสมการที่ (5.2)

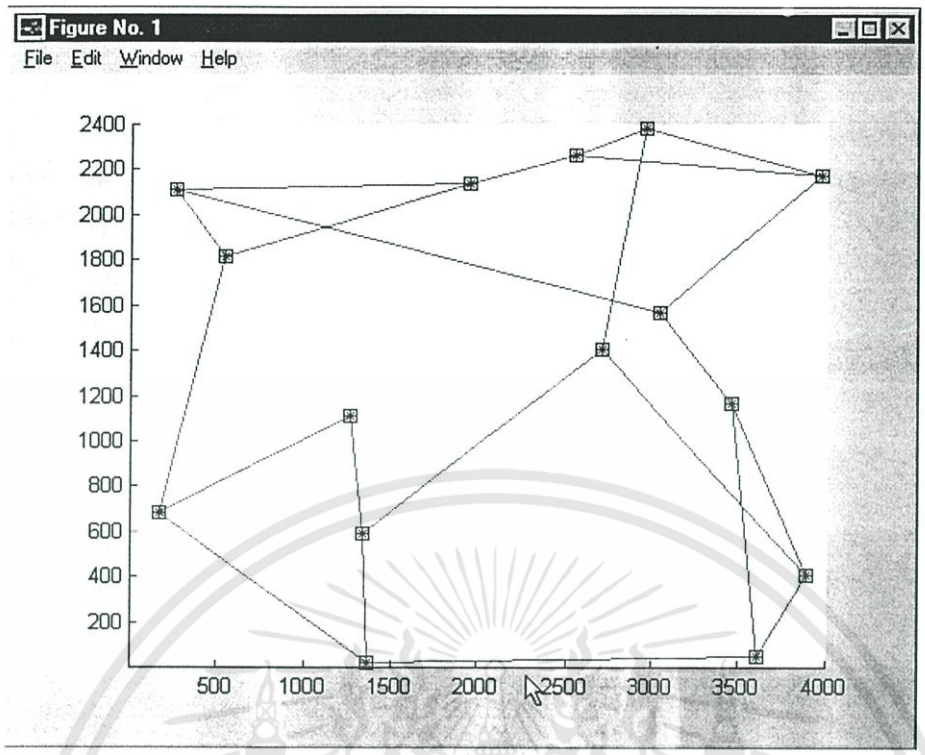
$$\text{แต่ละจุดที่พล็อตกราฟ} = \sum_{i=1}^{N} \frac{\textit{จำนวนเส้นทางทั้งหมดคู่ที่ } i}{N} \qquad (5.2)$$

โดยที่ N คือจำนวนคู่ทั้งหมด

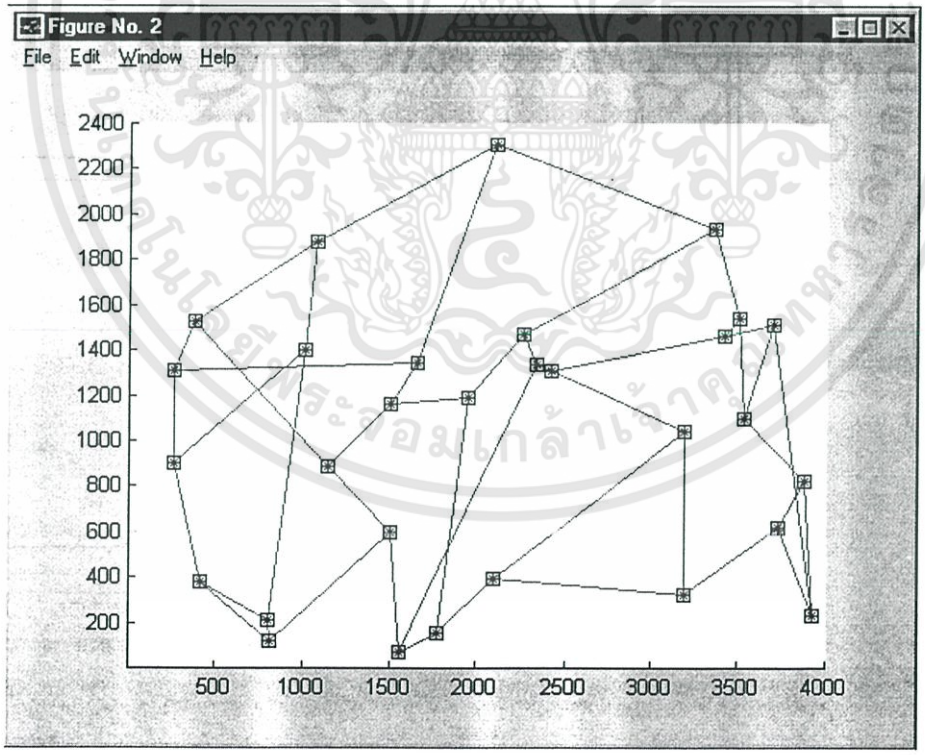
ดังนั้นแต่ละจุดที่พล็อตกราฟใช้เส้นทางเท่ากับ 1500, 5700, 10400, และ 16550 สำหรับเครือข่าย 15, 30, 45 และ 60 โหนด ตามลำดับ

ตารางที่ 5.1 แสดงรายละเอียดของพารามิเตอร์ที่ใช้ในการซิมมูเลชั่น

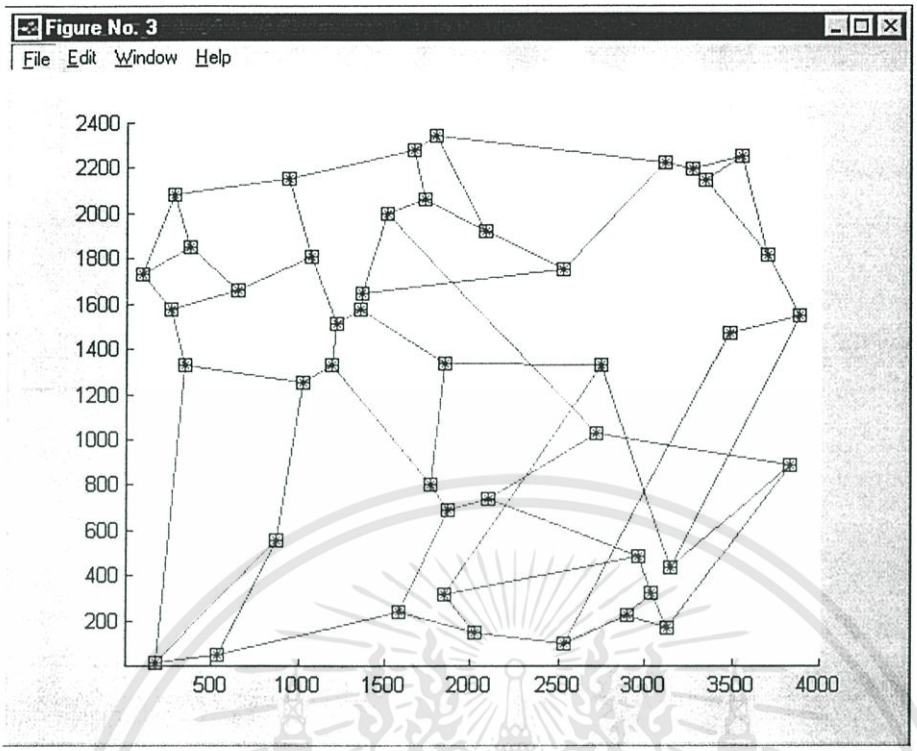
พารามิเตอร์	ค่าที่ใช้ในการทดลอง (หน่วย)	รายละเอียด
$R(e)$	155.52 x 10^6 บิต/วินาที	แบนด์วิดธ์ของลิงค์ e (OC-3)
$H_{\max}$	$H_{\max} = \|V\| - 1$	จำนวนฮอปที่มากที่สุดที่อนุญาตในเครือข่าย
V	15, 30, 45 และ 60 โหนด	จำนวนโหนดในเครือข่าย
$v(e)$	2 x 10^8 เมตร/วินาที	ความเร็วในการนำข้อมูลของลิงค์ e
$L(e)$	ความยาวของลิงค์ เมตร	ความยาวระหว่างโหนด u และโหนด v
$pd(e)$	$\frac{L(e)}{v(e)}$ วินาที	พรอพพิเกชั่นดีเลย์ของลิงค์e
$(\sigma_{video}, \rho_{video}, S_{\max}^{video})$	(4240บิต,1.5x10^6บิต/วินาที,424บิต)	ข้อกำหนดการไหลของข้อมูล q วิดีโอ $(10S_{\max}, 1.5x10^6, S_{\max})$
$S_{\max}$	53 ไบต์ = 424 บิต	ขนาดเซลล์ของ ATM
$\Delta_{bandwidth} = \rho_{video}$	1.5 x 10^6 บิต/วินาที	ค่าบังคับแบนด์วิดธ์ เป็น อัตราการสร้างโทคเก้นที่ร้องขอสำหรับการส่งข้อมูล วิดีโอ
Δ_{delay}	400 x 10^{-3} วินาที	ค่าบังคับเอนทูเอนดีเลย์ของการส่งข้อมูล วิดีโอ
Δ_{jitter}	100 x 10^{-3} วินาที	ค่าบังคับเอนทูเอนจิตเตอร์ดีเลย์ของการส่งข้อมูล วิดีโอ
Δ_{buffer}	$\Delta_{buffer}^{e} = \sigma_{video} + nS_{\max}^{video}$ บิต	ค่าบังคับบัฟเฟอร์ ลำดับที่ n
$f(e)$	ค่าสุ่มระหว่าง 4.6 x 10^3 และ 128 x 10^3 บิต	พื้นที่ว่างของบัฟเฟอร์ที่โหนดใด ๆ



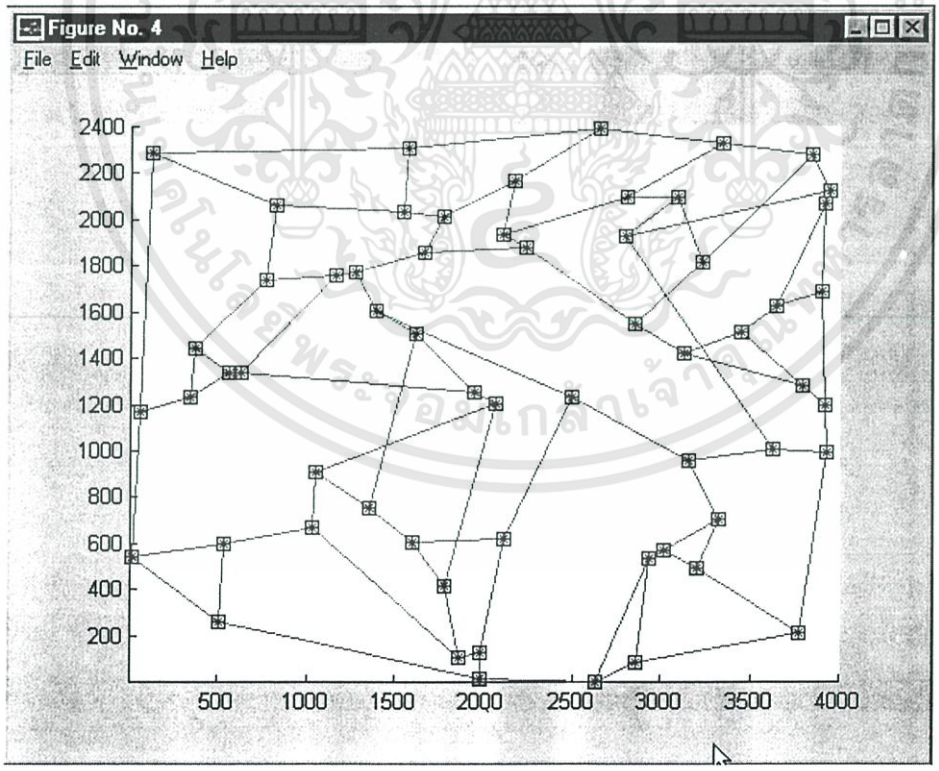
รูปที่ 5.1 ตัวอย่างของเครือข่ายแบบสุ่มที่มี 15 โหนด



รูปที่ 5.2 ตัวอย่างของเครือข่ายแบบสุ่มที่มี 30 โหนด



รูปที่ 5.3 ตัวอย่างของเครือข่ายแบบสุ่มที่มี 45 โหนด



รูปที่ 5.4 ตัวอย่างของเครือข่ายแบบสุ่มที่มี 60 โหนด

5.2 ผลการทดลอง

ผลการทดลองแสดงดังตารางที่ 5.1, 5.2 และ 5.3 และแสดงดังรูปที่ 5.5 แสดงจำนวนเฉลี่ยของเส้นทาง (สำหรับกรณีที่ 1) ที่ได้จากอัลกอริทึม $k-WSP_{EP}$ จะมีจำนวนน้อยกว่าจำนวนเฉลี่ยของเส้นทางที่ได้จากอัลกอริทึม $k^{th}-WSP$ ผลการทดลองนี้ [19] (รายละเอียดแสดงในภาคผนวก ค) ได้ผลเช่นเดียวกันกับ กรณีที่ 2 และ กรณีที่ 3 ด้วย ซึ่งสามารถเปรียบเทียบได้ระหว่างรูปที่ 5.6 และ 5.7 เนื่องจากในกรณีที่ 1 มีขนาดค่า K มากกว่ากรณีอื่น ๆ จึงมีเส้นทางที่มี QoS มากกว่ากรณีที่ 2 และ 3 ตามลำดับ

ตารางที่ 5.2 แสดงการเปรียบเทียบระหว่างจำนวนเฉลี่ยของเส้นทางที่ได้จากอัลกอริทึม $k^{th}-WSP$ และ $k-WSP_{EP}$ สำหรับกรณีที่ 1

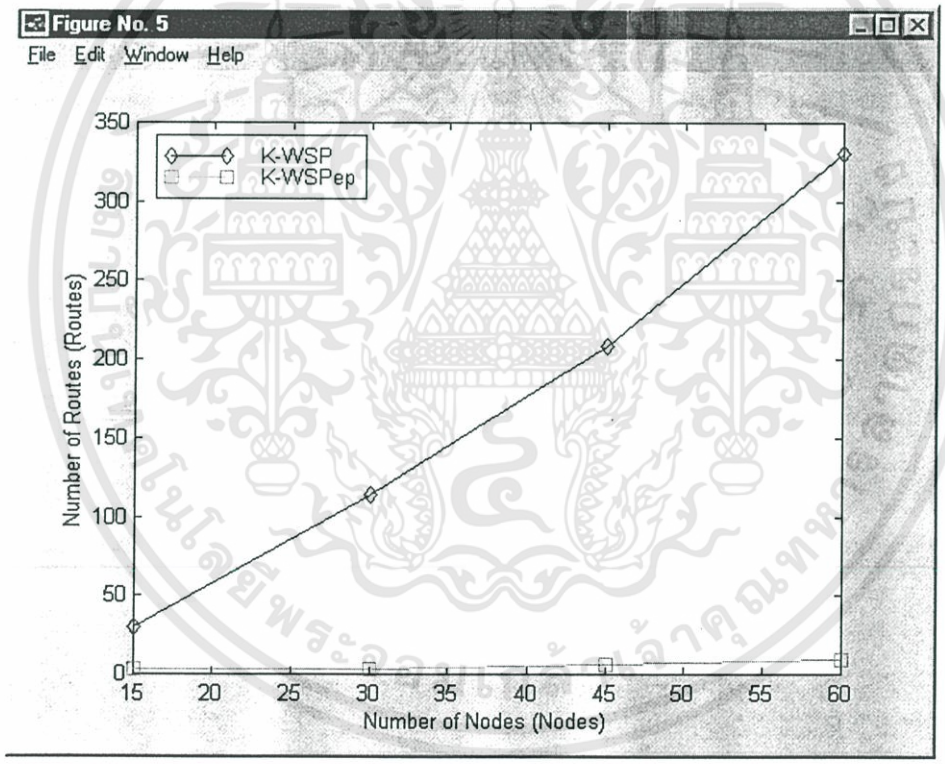
จำนวนโหนด (โหนด)	จำนวนเส้นทางเฉลี่ย (เส้นทาง)	
	$k^{th}-WSP$	$k-WSP_{EP}$
15	29.825	2.632
30	113.451	3.509
45	207.895	6.141
60	330.409	8.772

ตารางที่ 5.3 แสดงจำนวนเฉลี่ยของเส้นทางที่ได้จากอัลกอริทึม $k^{th}-WSP$ สำหรับกรณีที่ 1, กรณีที่ 2 และกรณีที่ 3 จากเครือข่ายแบบสุ่ม (15, 30, 45 และ 60 โหนด)

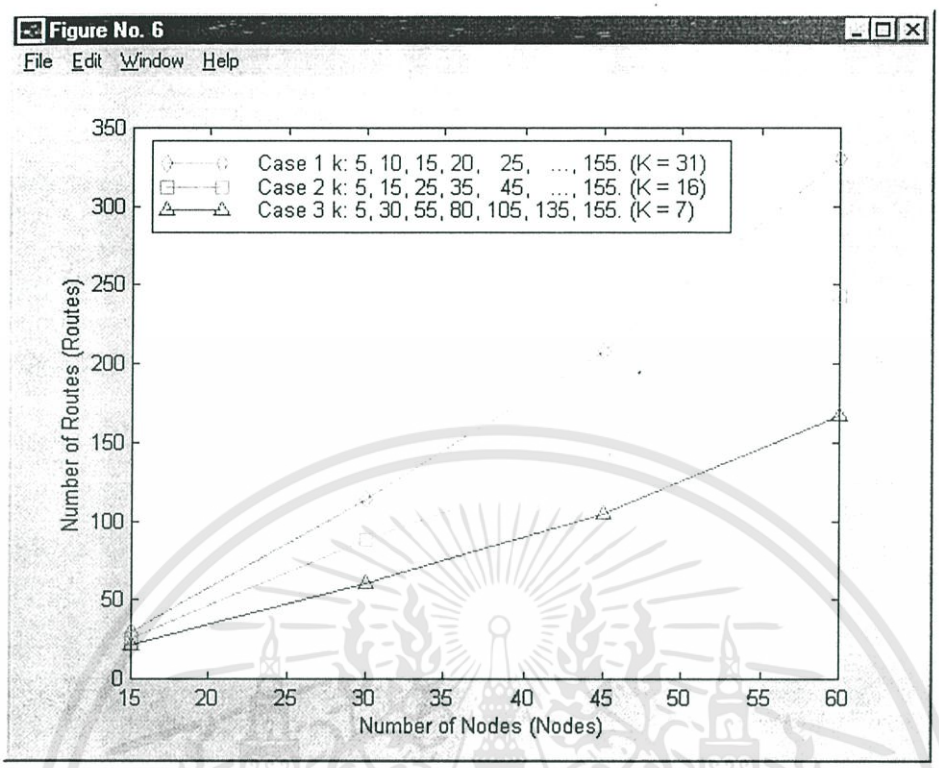
จำนวนโหนด (โหนด)	จำนวนเส้นทางเฉลี่ย (เส้นทาง)		
	กรณีที่ 1	กรณีที่ 2	กรณี 3
15	29.825	25.758	21.515
30	113.451	88.182	60.606
45	207.895	140.909	104.545
60	330.409	243.031	166.971

ตารางที่ 5.4 แสดงจำนวนเฉลี่ยของเส้นทางที่ได้จากอัลกอริทึม $k-WSP_{EP}$ สำหรับกรณีที่ 1, กรณีที่ 2 และกรณีที่ 3 จากเครือข่ายแบบสุ่ม (15, 30, 45 และ 60 โหนด)

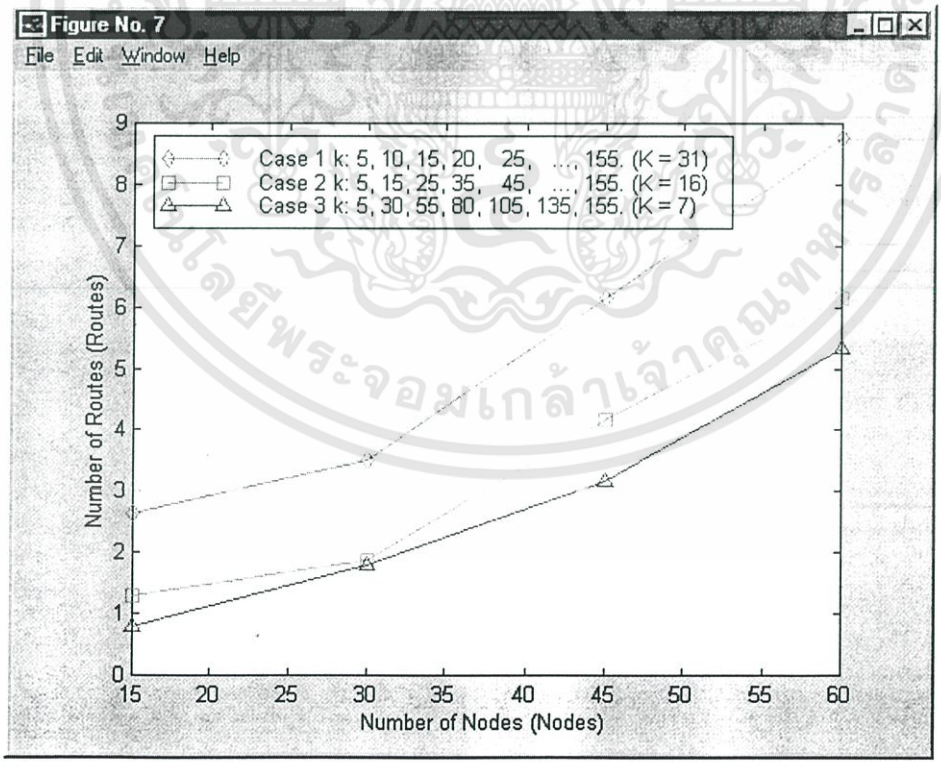
จำนวนโหนด (โหนด)	จำนวนเส้นทางเฉลี่ย (เส้นทาง)		
	กรณีที่ 1	กรณีที่ 2	กรณี 3
15	2.632	1.291	0.793
30	3.509	1.869	1.807
45	6.141	4.165	3.179
60	8.772	6.138	5.324



รูปที่ 5.5 แสดงจำนวนเฉลี่ยของเส้นทางที่ได้จากอัลกอริทึม $k^{th}-WSP$ และ $k-WSP_{EP}$ สำหรับกรณีที่ 1



รูปที่ 5.6 แสดงจำนวนเฉลี่ยของเส้นทางที่ได้จากอัลกอริทึม $k^{th} - WSP$



รูปที่ 5.7 แสดงจำนวนเฉลี่ยของเส้นทางที่ได้จากอัลกอริทึม $k - WSP_{EP}$

ตารางที่ 5.5 แสดงการเปรียบเทียบระหว่างเวลาเฉลี่ยในการหาเส้นทางที่ได้จากอัลกอริทึม $k^{th} - WSP$ และ $k - WSP_{EP}$ สำหรับกรณีที่ 1

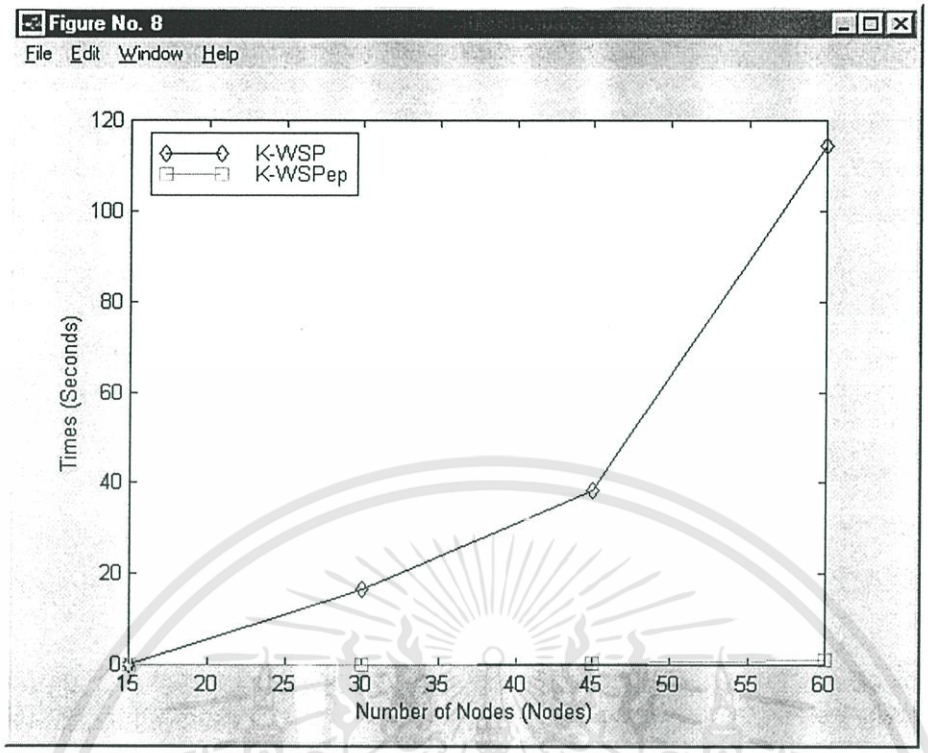
จำนวนโหนด (โหนด)	เวลาเฉลี่ยในการหาเส้นทาง (วินาที)	
	$k^{th} - WSP$	$k - WSP_{EP}$
15	0.056	0.007
30	16.413	0.079
45	38.380	0.267
60	114.490	0.945

ตารางที่ 5.6 แสดงเวลาเฉลี่ยในการหาเส้นทางที่ได้จากอัลกอริทึม $k^{th} - WSP$ สำหรับกรณีที่ 1, กรณีที่ 2 และกรณีที่ 3 จากเครือข่ายแบบสุ่ม (15, 30, 45 และ 60 โหนด)

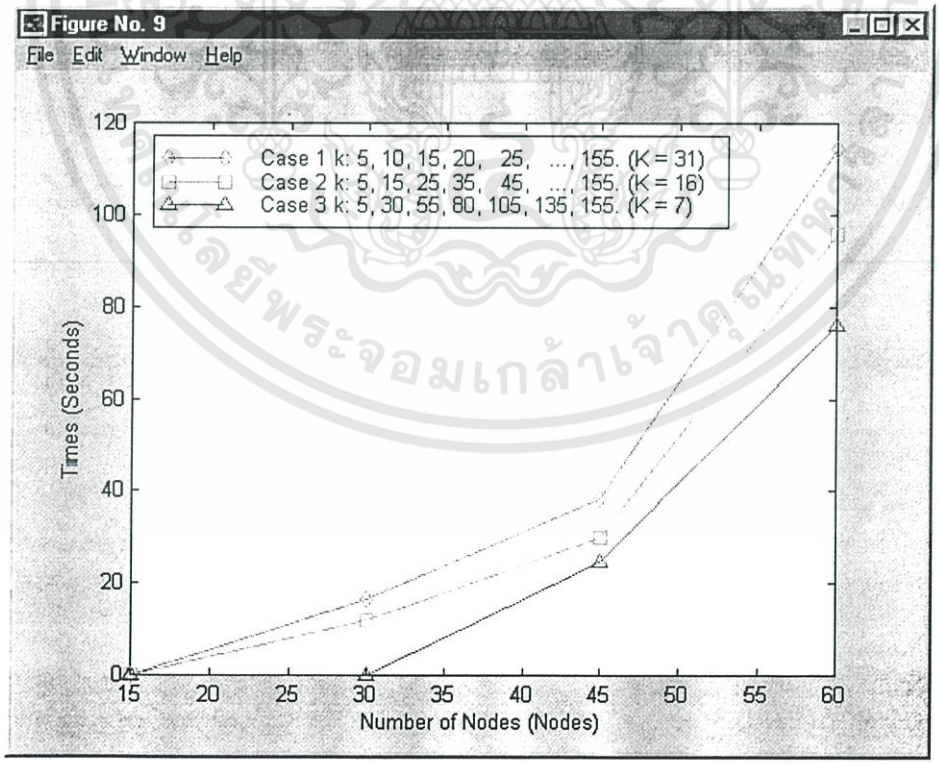
จำนวนโหนด (โหนด)	เวลาเฉลี่ยในการหาเส้นทาง (วินาที)		
	กรณีที่ 1	กรณีที่ 2	กรณี 3
15	0.056	0.042	0.371
30	16.413	12.005	9.072
45	38.380	29.966	24.574
60	114.490	95.820	75.968

ตารางที่ 5.7 แสดงเวลาเฉลี่ยในการหาเส้นทางที่ได้จากอัลกอริทึม $k - WSP_{EP}$ สำหรับกรณีที่ 1, กรณีที่ 2 และกรณีที่ 3 จากเครือข่ายแบบสุ่ม (15, 30, 45 และ 60 โหนด)

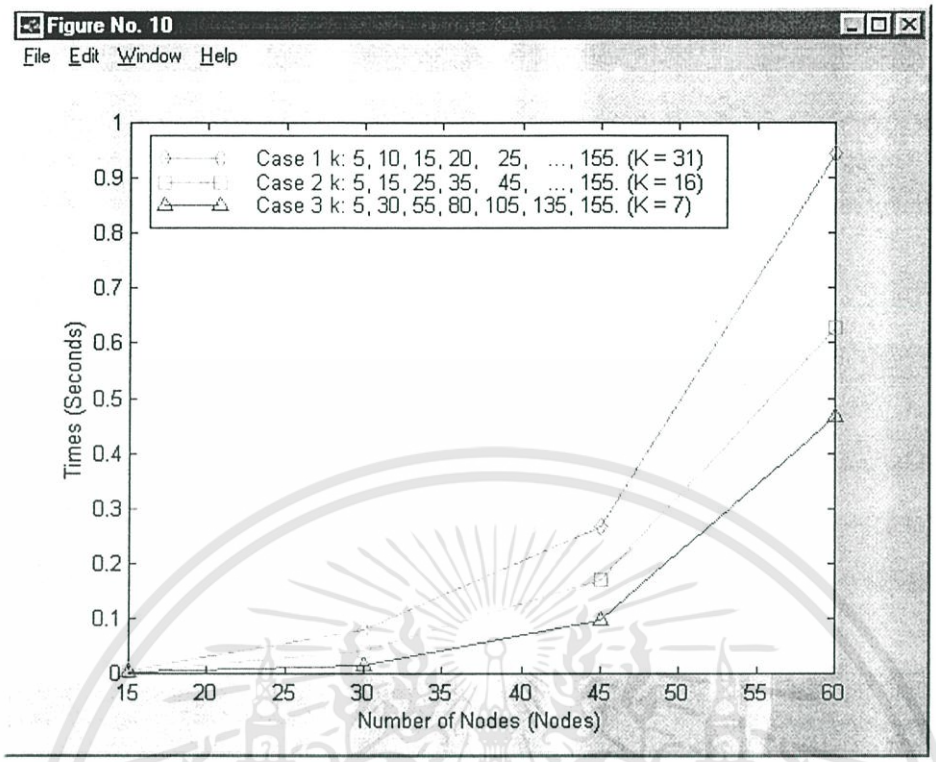
จำนวนโหนด (โหนด)	เวลาเฉลี่ยในการหาเส้นทาง (วินาที)		
	กรณีที่ 1	กรณีที่ 2	กรณี 3
15	0.007	0.004	0.002
30	0.079	0.041	0.016
45	0.267	0.171	0.098
60	0.945	0.628	0.465



รูปที่ 5.8 แสดงเวลาเฉลี่ยในการหาเส้นทางที่ได้จากอัลกอริทึม $k^{th} - WSP$ และ $k - WSP_{EP}$ สำหรับกรณีที่ 1



รูปที่ 5.9 แสดงเวลาเฉลี่ยในการหาเส้นทางที่ได้จากอัลกอริทึม $k^{th} - WSP$



รูปที่ 5.10 แสดงเวลาเฉลี่ยในการหาเส้นทางที่ได้จากอัลกอริทึม $k-WSP_{EP}$

บทที่ 6
บทสรุปและแนวทางในการพัฒนา

6.1 สรุปผลการซิมมูเลชั่น

1. อัลกอริทึมการเลือกเส้นทางที่มีการรับประกัน QoS ในเครือข่ายคอมพิวเตอร์สามารถรองรับการส่งผ่านข้อมูลที่มีแอพพลิเคชั่นแบบสื่อผสม ดังนั้นถ้าหากไม่มีอัลกอริทึมการเลือกเส้นทางที่มีการรับประกัน QoS แล้ว เครือข่ายอาจจะไม่สามารถค้นหาเส้นทางได้ และปฏิเสธการร้องขอการเชื่อมต่อ แม้ว่าจะมีทรัพยากรในเครือข่ายเพียงพอสำหรับการติดตั้งของการร้องขอนั้นก็ตาม แต่ปัญหาการเลือกเส้นทางที่มีการรับประกัน QoS หลาย ๆ พารามิเตอร์ นั้นถูกพิสูจน์แล้วว่าเป็นปัญหา *NP* สัมบูรณ์ [3] ซึ่งในการพิสูจน์นี้ ไม่ได้มีการสมมุติของระเบียบบริการที่ใช้ในเครือข่าย ค่าขอบเขตของคุณภาพของการบริการ เช่น ขอบเขตของดีเลย์ ขอบเขตของจิตเตอร์ดีเลย์ ซึ่งขึ้นอยู่กับระเบียบของการให้บริการ ในวิทยานิพนธ์นี้จึงได้นำเสนออัลกอริทึมการเลือกเส้นทางที่มี QoS เพื่อสนันสนุนการจองทรัพยากรเพื่อรับประกันความต้องการ QoS สำหรับแอพพลิเคชั่นแบบสื่อผสม

ในการออกแบบอัลกอริทึมการเลือกเส้นทางที่มีการรับประกัน QoS สำหรับแอพพลิเคชั่นแบบสื่อผสมนั้น ต้องใช้ระเบียบบริการ เนื่องจากค่าขอบเขตของ QoS ขึ้นอยู่กับระเบียบของการให้บริการ

อัลกอริทึมการเลือกเส้นทางที่มี QoS ที่มีประสิทธิภาพในวิทยานิพนธ์นี้ ได้มีการนำเอาระเบียบบริการในเครือข่ายมาใช้สำหรับการออกแบบอัลกอริทึม กล่าวคือได้พิจารณาระเบียบบริการ "การจัดแถวคอยที่ให้น้ำหนักอย่างเป็นธรรม" ซึ่งอัลกอริทึมนี้เรียกว่า "การเลือกเส้นทางที่มี QoS วิธีของเบลแมนฟอร์ด" เป็นอัลกอริทึมที่สามารถค้นหาเส้นทางที่เป็นไปตามเงื่อนไขที่มีการบังคับ แบนด์วิดธ์คอขวด เอนทูเอนดีเลย์ เอนทูเอนจิตเตอร์ดีเลย์ และช่องว่างของบัฟเฟอร์ (ไม่มีการสูญเสีย) ได้เสมอถ้าหากว่ามีเส้นทางนั้นอยู่ในเครือข่าย

เวลาที่ใช้ในการคำนวณของอัลกอริทึม "การเลือกเส้นทางที่มี QoS วิธีของเบลแมนฟอร์ด" ในกรณีที่เลวที่สุด จะเร็วเท่ากับ อัลกอริทึมการหาเส้นทางที่สั้นที่สุดวิธีของเบลแมนฟอร์ด กล่าวคืออยู่ในลำดับของ $O(|E \bullet H_{\max}|)$ โดยที่ $|E|$ คือจำนวนของลิงค์ในเครือข่าย และ $H_{\max}$ เป็นจำนวนอิเทอเรชั่นที่มากที่สุด ของอัลกอริทึมวิธีของเบลแมนฟอร์ด ซึ่งเป็นขอบเขตบนของจำนวนฮอปของเส้นทางนั้น

2. อัลกอริทึมการคำนวณไว้ล่วงหน้าของทางเดินที่รับประกัน QoS (k-WSP_{EP}) ซึ่งเป็นอัล

กอริทึมที่ปรับปรุงมาจาก "อัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS วิธีของเบลแมนฟอร์ด" ผลของการประเมินของอัลกอริทึมนี้คือ เส้นทางที่เป็นไปตามเงื่อนไขที่กำหนดไว้เป็นเส้นทางที่เกินความจำเป็น จึงสามารถถูกละทิ้งได้ ดังนั้นขนาดของตารางเส้นทางจึงมีขนาดลดลง โดยยังคงมีประสิทธิภาพ เนื่องจากมีเส้นทางที่ดีถูกเก็บไว้เรียบร้อยแล้ว

กล่าวคือ จากการเปรียบเทียบจำนวนเส้นทางเฉลี่ยที่ได้จากอัลกอริทึม k^{th}-WSP และ k-WSP_{EP} ปรากฏว่า จำนวนเส้นทางเฉลี่ยที่ได้จากอัลกอริทึม k-WSP_{EP} ลดลงกว่า 95.73 % โดยที่ใช้เวลาเฉลี่ยในการดำเนินอัลกอริทึมน้อยกว่า 99.37 % โดยที่เส้นทางที่ได้จากอัลกอริทึม k-WSP_{EP} ยังคงมีการรับประกัน QoS ทั้ง 4 พารามิเตอร์ คือ แบนด์วิดธ์คอขวด เอนทูเอนดีเลย์ เอนทูเอนจิตเตอร์ดีเลย์ และอัตราการสูญเสีย นอกจากนั้นเส้นทางที่ได้ ยังมีแบนด์วิดธ์คอขวดมากกว่า เอนทูเอนดีเลย์น้อยกว่า และเอนทูเอนจิตเตอร์ดีเลย์น้อยกว่าอีกด้วย

6.2 แนวทางในการพัฒนา

ในวิทยานิพนธ์ได้เสนอ อัลกอริทึมที่มีการเลือกที่มีการรับประกัน QoS เป็นการเลือกเส้นทางแบบ ยูนิคาสท์ โดยมีการใช้ระเบียบบริการแบบ "การจัดแถวคอยแบบให้น้ำหนักอย่างเป็นธรรม" ดังนั้นแนวทางในการพัฒนาต่อจากวิทยานิพนธ์นี้ มีดังต่อไปนี้

6.2.1 พัฒนาอัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS แบบจากจุดหนึ่ง ถึงกลุ่มผู้รับจำนวนหนึ่งที่ระบุไว้

6.2.2 พัฒนาอัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS แบบจากหลายจุด ถึงหลายจุดพร้อม ๆ กัน

6.2.3 พัฒนาอัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS โดยการใช้ระเบียบการให้บริการแบบอื่น ๆ

6.2.4 พัฒนาอัลกอริทึมการเลือกเส้นทางที่รับประกัน QoS ให้ใช้ได้ในเครือข่ายที่แตกต่างกัน

เอกสารอ้างอิง

(1.) Christian, H. **Routing in the Internet.** New Jersey : Prentice Hall, Inc. 1995.

(2). Guerin, R., Orda, A., and Williams, D. "Qos routing mechanisms and ospf extensions". **Internet Draft.** January 1998.

(3). Wang, Z. and Crowcroft, J. "Quality of service routing for supporting multimedia applications". **IEEE Journal Selected Areas Communication.** 14 (7) : 1228-1234, 1996.

(4). ชิดชนก เหลือสินทรัพย์. **Analysis & Design of Algorithms.** กรุงเทพฯ : SUM System Company. 2543.

(5). Salama, F., Reeves, S. H. and Viniotis, Y. D. "A distributed algorithm for delay constrained unicast routing". **In Proceeding of IEEE INFOCOM.** Kobe, Japan, April 1997.

(6). Rao, S.V. N. and Batsel, S.G. "Algorithm for minimum end-to-end delay paths", **IEEE Communication Letter.** 1(5) , September 1997.

(7). Parris, C., Zhang, V. and Ferrari, D. "Dynamic management of quaranteed performance multimedia connections". **Multimedia Systems Journal.** 1:267-283, 1994.

(8). Guerin, R. and Orda, A. "Qos-based routing in networks with inaccurate information : Theory and algorithms". **In proceeding of IEEE INFOCOM**. Kobe, Japan. April 1997.

(9). สุรศักดิ์ สงวนพงษ์. **สถาปัตยกรรม และ โปรโตคอลทีซีพี/ไอพี.** กรุงเทพฯ. : โครงการตำราวิชาการ ภาค วิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์. 2543.

(10). Stallings, W. **Data and Computer Communications.** 5th ed. New Jersey: : Prentice Hall, Inc. 1997.

(11). Andrew S, Tanenbaum. **Computer Networks.** New Jersey: : Prentice Hall, Inc. 1996.

(12). Pornavalai, C., Chakraborty, G. and Shiratori. N. "QoS based routing algorithm in intergrated service packet networks". **Journal of High Speed Networks.** 1998.

(13). Pornavalai, C., Chakraborty, G. and Shiratori, N. "QoS routing algorithm for precomputed paths". **In 6th International Conference on computer Communications and Networks (IC3N' 97).** September, 1997.

(14). Parekh, K. A., and Gallager, G. R. "A generalized processor sharing approach to flow control in intergrated services networks: The multiple node case". **IEEE/ACM Trans. Networking.** 2(2) : 137 - 150, April 1994.

(15). Zhang, H. "Service disciplines for guaranteed performance service in packet-switching networks". **Proceeding of the IEEE.** 83(10), October 1995.

(16). Ma, Q. and Steenkiste, P. "Quality-of-service routing for traffic with performance guarantees". **In IFIP Fifth Internation Workshop on Quality of Service (IWQOS'97).** May 1997.

(17). Pornavalai, C. **"Quality of Service Guaranteed Routing In Integrated Service Networks".** PH.D. dissertation, Tohoku University. Japan. 1997.

(18). Waxman, M. B. "Routing of multipoint connections". **IEEE Journal Selected Areas Communications.** 6(9) : 1617 – 1622. 1988. pp. 1617 – 1622.

(19). Kaewyongphang P., Pornavalai C., Varakulsiripunth R., Chakraborty G. and Shiratori. N. "On Precomputing of QoS Paths." **In IEEE International Workshop on Intelligent Signal Processing and Communication Systems. (ISPACS' 99).** Phuket, Thailand, December 1999. pp. 17 - 20.

(20). Casetti, C., Lo Cigno R., Mellia M. and Munafo M. 2002. **A New Class of QoS Routing Strategies Based on Network Graph Reduction.** [Online] Available: http://www.ieeeinfocom.org/2002/paper/551.pdf

ภาคผนวก ก.

รหัสเทียมของอัลกอริทึมการเลือกเส้นทางที่รับประกันคุณภาพโดยใช้วิธีของเบลแมน – ฟอร์ด ($QosR - BF$)

อินพุต : กราฟ $G = (V, E)$ โดยที่ $v \in V$ แทนโหนดในเครือข่าย และ $e = (u, v) \in E$ แทนลิงค์ e ซึ่งเป็นลิงค์ที่เชื่อมโดยตรงจากโหนด u ไปยังโหนด v

s เป็นโหนดต้นทาง และ z เป็นโหนดปลายทาง

$b(e)$ คือ จำนวนแบนด์วิดธ์ของลิงค์ e ที่ยังไม่ถูกจอง

$R(e)$ คือ ความจุของลิงค์ e

$f(e)$ คือ ขนาดของบัฟเฟอร์ของโหนด v ที่เชื่อมกับลิงค์ e

$\tau = (\sigma_q, \rho_q, S^q_{\max})$ คือ ข้อกำหนดการไหล (ขนาดของเบิสท์ที่ใหญ่ที่สุด, อัตราโทคเก้น และขนาดของแพ็กเกตที่ใหญ่ที่สุด)

$pd(e)$ คือ พรอพพิเกชั่นดีเลย์ของลิงค์ e

Δ_s คือ การบังคับคุณภาพของการให้บริการ (โดยพารามิเตอร์ที่บังคับจะขึ้นอยู่กับตัวอักษรที่ต่อท้าย)

$H^{policy}_{\max}$ คือ จำนวนฮอปมากที่สุดที่โหนดต้นทางเป็นผู้กำหนด

เอาท์พุท : เส้นทางตามข้อกำหนดของการรับประกันคุณภาพของการให้บริการ

$\pi_i[v]$ คือ โหนดที่อยู่ก่อนหน้าโหนด v ตามเส้นทาง P ซึ่งจะมีจำนวนฮอปเท่ากับ i ฮอป

$P(s, z)$ คือ เส้นทางที่มีการรับประกันคุณภาพของการให้บริการจากโหนด s ถึงโหนด z

$C(P), D(P), B(P), H(P), J(P)$ คือ ค่าใช้จ่าย, ค่าดีเลย์, ค่าแบนด์วิดธ์คอขวด, จำนวนฮอป และค่าจิตเตอร์ดีเลย์ของ $P(s, z)$ ตามลำดับ

$c_i[v], d_i[v], b_i[v], h_i[v], j_i[v]$ คือ ค่าใช้จ่าย, ค่าดีเลย์, ค่าแบนด์วิดธ์คอขวด, จำนวนฮอป และค่าจิตเตอร์ดีเลย์ของเส้นทาง $, (i = h_i[v])$, ถึงโหนด v ตามลำดับ

$QosR - BF(G, s, z, b, f, \tau, pd, \Delta, H^{policy}_{\max})$

1. $COMPLEXITY\_REDUCTION(G, s, b, f, \tau, pd, \Delta, H^{policy}_{\max})$
2. $INITIALIZE - SINGLE - SOURCE(G, 0, s)$
3. $for \quad i \leftarrow 1 \quad to \quad H^{policy}_{\max}$
4. $\quad INITIALIZE \quad SINGLE \quad SOURCE(G, i, s)$
5. $\quad do\ for \quad each \quad edge \quad e = (u, v) \in E[G]$
6. $\quad\quad if\ (((i \le m(e)_{\max})\ and\ (h_{i-1}[u] == i - 1))\ and\ (d_{i-1}[u] \le \Delta_{delay}))$
7. $\quad\quad then \quad RELAX(u, v, i, z, \Delta_{delay}$
8. $\quad if\ ((d_i[z] \le \Delta_{delay})$
9. $\quad\quad C(P) \leftarrow c_i[z]$
10. $\quad\quad D(P) \leftarrow d_i[z]$
11. $\quad\quad B(P) \leftarrow b_i[z]$

12. $H(P) \leftarrow h_i[z]$
13. $J(P) \leftarrow$ *jitter bound* ดังสมการที่ 4.2
14. *return* *TRUE*
15. *return* *FALSE*

COMPLEXITY_REDUCTION$(G, s, b, f, \tau, pd, \Delta, H_{\max}^{policy})$

16. $r_q \leftarrow \Delta_{bandwidth}$
17. *do for each edge* $e = (u, v) \in E[G]$
18. $c(e) \leftarrow$ สมการที่ 2.1
19. $d(e) \leftarrow$ สมการที่ 4.4, และ สมการที่ 4.5
20. $m(e)_{\max} \leftarrow$ สมการที่ 4.7
21. $H_{\max}^{jitter} \leftarrow$ สมการที่ 4.6
22. $H_{\max} \leftarrow$ สมการที่ 4.8
23. $C(P) \leftarrow \infty$
24. $D(P) \leftarrow \infty$
25. $J(P) \leftarrow \infty$
26. $B(P) \leftarrow 0$
27. $H(P) \leftarrow \infty$

INITIALIZE $-$ *SINGLE* $-$ *SOURCE*(G, i, s)

28. *do for each vertex* $v \in V[G]$
29. $c_i[v] \leftarrow \infty$
30. $b_i[v] \leftarrow 0$
31. $d_i[v] \leftarrow \infty$
32. $h_i[v] \leftarrow \infty$
33. $\pi_i[v] \leftarrow NIL$
34. $c_i[s] \leftarrow 0$
35. $b_i[s] \leftarrow \infty$
36. $h_i[s] \leftarrow 0$
37. $d_i[s] \leftarrow 0$

RELAX$(u, v, i, z, \Delta_{delay})$

38. *if* $v \; != z$

39. *if* $d_i[v] > d_{i-1}[u] + d(u,v)$

40. *then* $d_i[v] \leftarrow d_{i-1}[u] + d(u,v)$

41. $c_i[v] \leftarrow c_{i-1}[u] + c(u,v)$

42. $h_i[v] \leftarrow h_{i-1}[u] + 1$

43. $b_i[v] \leftarrow MIN(b_{i-1}[u], b(u,v))$

44. $\pi_i[v] \leftarrow u$

45. *else if* $((d_{i-1}[u] + d(u,v) \leq \Delta_{delay})$ *and* $(c_i[v] > c_{i-1}[u] + c(u,v)))$

46. *then* $d_i[v] \leftarrow d_{i-1}[u] + d(u,v)$

47. $c_i[v] \leftarrow c_{i-1}[u] + c(u,v)$

48. $h_i[v] \leftarrow h_{i-1}[u] + 1$

49. $b_i[v] \leftarrow MIN(b_{i-1}[u], b(u,v))$

50. $\pi_i[v] \leftarrow u$

$MIN(x,y)$

51. *if* $x < y$

52. *return* x

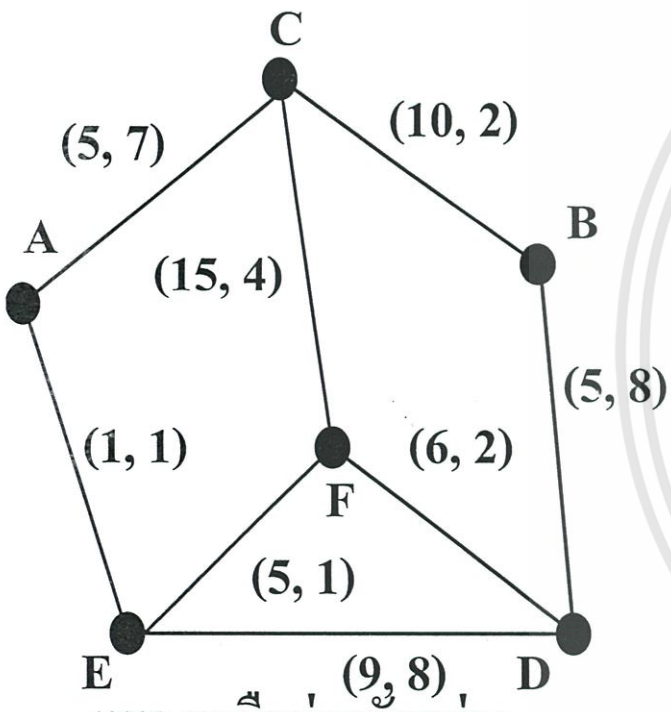
53. *else*

54. *return* y

ภาคผนวก ข

แสดงตัวอย่างการคำนวณเส้นทางโดยใช้อัลกอริทึม $k^{th} - WSP$ และ $k - WSP_{EP}$

สมมุติว่ามีเครือข่ายดังแสดงในรูปข้างล่าง ดังนั้นผลการคำนวณเส้นทางโดยใช้อัลกอริทึม $K-WSP$ และ $K-WSP_{EP}$ แสดงดังตารางดังต่อไปนี้



แสดงเครือข่ายตัวอย่าง

A คือ โหนดต้นทาง

B คือ โหนดปลายทาง

th อิเทอเรชั่น	แบนด์วิดธ์	ดีเลย์	เส้นทาง	K^{th}-WSP	K-WSP_{EP}
1	-	-	-	-	-
2	5	9	A-C-B	1st	1
3	1	17	A-E-D-B	1st	-
4	5	21	A-C-F-D-B	1st	-
4	1	8	A-E-F-C-B	2nd	1
4	1	12	A-E-F-D-B	-	-
5	5	28	A-C-F-E-D-B	1st	-
5	1	17	A-E-D-F-C-B	2nd	-
จำนวนของเส้นทางที่จะบันทึกในตาราง				6	2

ภาคผนวก ค

ผลงานวิจัย

1. Kaewyongphang P., Pornavalai C., Varakulsiripunth R., Chakraborty G. and Shiratori. "On Precomputing of QoS Paths." **In IEEE International Workshop on Intelligent Signal Processing and Communication Systems. (ISPACS' 99).** Phuket, Thailand, December 1999. pp. 17 - 20.

PROCEEDINGS

1999 IEEE International Symposium on Intelligent Signal Processing and Communication Systems

ISPACS'99

Signal Processing and Communications Beyond 2000

December 8-10, 1999

Phuket Arcadia Hotel & Resort, Phuket, Thailand

On Precomputing of QoS Paths

Phongkaew Kaewyongphang† Chotipat Pornavalai‡ Ruttikorn Varakulsiripunth†

†Dept. of Electronics, Faculty of Engineering ‡Faculty of Information Technology
and Research Center for Communications and Information Technology (ReCCIT)
King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok 10250 Thailand
†{s0061066, kvruttik}@kmitl.ac.th ‡chotipat@it.kmitl.ac.th

Goutam Chakraborty††

††Dept. of Software and Information Science
Iwate Prefectural University
Iwate 020-0193, Japan
goutam@soft.iwate-pu.ac.jp

Norio Shiratori‡‡

‡‡RIEC
Tohoku University
Sendai 980-77, Japan
norio@shiratori.riec.tohoku.ac.jp

Abstract

Though routing with multiple QoS constraints is known to be a NP-complete problem, we have shown in our previous works [1] [2] that, for a number of specific QoS constraints, it can be solved in polynomial time, when Weighted Fair Queueing (WFQ) service discipline is employed. For precomputation of QoS paths, the size of the routing table must be very large in order to store all QoS routes. However, in [2], we have proposed that many paths are likely to be redundant as paths satisfying defined conditions could be neglected. In this paper, results from the simulation showed, as expected, that the size of routing table can be greatly reduced. Many redundant paths are deleted, while still achieving the optimal performance.

1 Introduction

Searching a route with multiple QoS constraints for supporting performance guaranteed service, is known to be a NP-complete problem [3]. However, we have shown in [1] that considerations from the service discipline could greatly reduce the complexity of the problem. For a number of specific QoS constraints (bandwidth, delay, delay jitter, buffer space constraints), it can be solved in polynomial time when Weighted Fair Queueing (WFQ) service discipline is employed [1].

In this paper, we propose a QoS routing algorithms, called "k-WSP_{EP}" for exact pre-computed paths. It is the modified version of Bellman-Ford (BF) algorithm. Here WSP stands for Widest Shortest Path, and k is the maximum number of WSPs for each BF iteration, i.e. widest, 2^{nd}widest, 3^{rd}widest, ..., k^{th}widest. k-WSP_{EP} could create routing table supporting traffic request for which flow specification parameters may take any values and are not known a priori.

The possible maximum value of k is equal to the maximum number of different residual bandwidth[1], K, where $K \leq |E|$, and $|E|$ is the number of links in the network. Thus in the worst case, the size of the routing table would be K times that of WSP. However, using simulation, we show that, in each iteration of BF, not all of the K possible WSPs are needed to be stored in the routing table. Many paths could be ignored, when the defined conditions are satisfied, because better path is already available.

This paper is organized as follows. In Sect. 2, we briefly describe the definition of QoS routing problem for pre-computed paths. In Sect. 3, the QoS bound expressions of WFQ service discipline are given. The proposed k-WSP_{EP} algorithm and simulation results are presented in Sect. 4 and 5 respectively. Related works are mentioned in Sect. 6. Conclusion is in Sect. 7.

2 QoS Routing Problem for Pre-Computed Paths

We simply assume that a communication network can be modeled as a direct connected graph $G = (V, E)$. Here V represents the set of nodes, which could be routers, servers or switches, and E represents the set of edges or links of the network. $h(e) = h(u,v) = 1$, $d(e) = d(u,v)$, $b(e) = b(u,v)$, $j(e) = j(u,v)$, $l(e) = l(u,v)$ are the hop, delay, available bandwidth, jitter and loss functions of link e, connecting directly the node u to node v. A source node s and a destination node z, $(s, z \in V)$, are given. The problem is to find a route or path from source node s to destination node z, $P(s,z) = (s, j, k, ..., l, z)$, such that it (i) would consume minimum resources by using minimum hop route, (ii) would balance the load in the network by choosing widest bandwidth route, (iii) would satisfy all of the multiple QoS constraints.

Let $H(P)$, $D(P)$, $J(P)$, $L(P)$ and B(P) are the number of hops, delay, jitter, loss and bottleneck bandwidth on the path $P(s,z)$ respectively. The QoS constraints are symbolized by Δ with respective suffixes. In this paper, we consider the deterministic guaranteed service. The loss probability is considered to be zero (buffer space constraint). In other words, every node or switch along the selected route must have enough buffer space, so that queueing loss will not occur for any packet at that node.

[1]In practice, link bandwidth will take only discrete values.

end-to-end delay bound	$\frac{\sigma_q + HS^q_{max}}{r_q} + \sum_{i=1}^{H}\{\frac{S_{max}}{R_i(e)} + pd_i(e)\}$
end-to-end jitter bound	$\frac{\sigma_q + HS^q_{max}}{r_q}$
buffer space at n^{th} switch	$\sigma_q + nS^q_{max}$

Table 1: End-to-end delay bound, delay-jitter bound, and buffer space requirement of Weighted Fair Queueing (WFQ) service discipline.

3 Weighted Fair Queueing (WFQ) Service Discipline

The guaranteed service class of the Integrated Service model [4] in the Internet is based on the Weighted Fair Queueing (WFQ) [5]. It can be used to provide a tight upper bound on the end-to-end delay, end-to-end jitter, and ensure that no packet is lost due to non availability of buffer at each switch, assuming no failure of network components. Table 1 shows the expressions to calculate such bounds and buffer space requirement (for no queueing loss) of a flow q, for which the traffic characteristic is in the form $(\sigma_q, \rho_q, S^q_{max})$ [6]. Here σ_q is the token bucket size or maximum burst size, ρ_q is the token generation rate of the leaky bucket, S^q_{max} is the maximum packet size of the requested flow q respectively. r_q is the guaranteed rate for the connection or the amount of bandwidth to be reserved for the requested flow q ($r_q \geq \rho_q$). S_{max} is the maximum packet size allowed in the network. $R_i(e)$ and $pd_i(e)$ are the total bandwidth (link capacity) and propagation delay of link e which is the i^{th} hop on the route traversed by the connection. H is the number of hops of the route.

4 QoS Routing Algorithm for Exact Pre-Computed Paths (k-WSP_{EP})

4.1 Algorithm Description

The proposed algorithm k-WSP_{EP} is modified from Widest-Shortest Path (WSP) algorithm (using BF algorithm) proposed by Guerin et. al. [7], and Ma and Steenkiste [8]. In the first iteration, it find one-hop k WSPs (1^{st}WSP, 2^{nd}WSP, ..., k^{th}WSP) from a source to all destination nodes in the network. In the second iteration it finds two-hops k WSPs, and continue to higher iterations with longer hops. The algorithm terminates after H_{max} iterations, where H_{max} is the predefined maximum hop length. If there are more than one paths having the same bottleneck bandwidth and same number of hops, the one having minimum path static delay (defined later) is kept.

Let us suppose that k is a predefined number. If $k = K$, we can always find QoS (bandwidth, delay, and delay jitter) satisfying route in polynomial time, if there exists one [1] [8]. However, later in this section, we will explain that not all of the K WSPs are needed to be stored in practical situations. Some of them are actually redundant and could be ignored. Thus the required routing table size could be reduced.

Link delay is composed of queueing, transmission, and propagation delay. Queueing delay is the same as the delay jitter (see Table 1). They are dynamic as they change with the requested flow specification $(\sigma_q, \rho_q, S^q_{max})$ and the amount of bandwidth to be reserved (r_q) for flow q. For k-WSP_{EP}, neither any of the flow specification parameters nor r_q are known a priori. The maximum value of r_q of a path that could be reserved is the bottleneck bandwidth of that path. Thus minimum queueing delay also depends on the bottleneck bandwidth of that path. Transmission and propagation delay are static. They depend on the static or constant parameters such as the link bandwidth capacity, length of the links, transmission media, etc. Thus the static link delay is defined as

$$d_{static}(e) = \frac{S_{max}}{R(e)} + pd(e) \tag{1}$$

The end-to-end delay of a path P, $D(P)$ is defined as:

$$D(P) = D_{static}(P) + D_{dynamic}(P) \tag{2}$$

where

$$D_{static}(P) = \sum_{e \in P} d_{static}(e) \tag{3}$$

$$D_{dynamic}(P) = \frac{\sigma_q + H(P)S^q_{max}}{r_q} \tag{4}$$

When r_q is selected equal to bottleneck bandwidth $B(P)$ of path P, we get the minimum end-to-end delay of path P, $D^{min}(P)$, which can be defined as:

$$D^{min}(P) = D_{static}(P) + D^{min}_{dynamic}(P) \tag{5}$$

where

$$D^{min}_{dynamic}(P) = \frac{\sigma_q + H(P)S^q_{max}}{B(P)} \tag{6}$$

The delay jitter and minimum delay jitter of the path P can be defined using Eq. (7) and Eq. (8) respectively.

$$J(P) = D_{dynamic}(P) \tag{7}$$

$$J^{min}(P) = D^{min}_{dynamic}(P) \tag{8}$$

At iteration i of k-WSP_{EP}, k WSPs with i hops to all the reachable nodes are found. If there are more than one paths having the same bottleneck bandwidth, the one having minimum $D_{static}(P)$ is saved in that iteration. After a route P is found, $H(P)$, $B(P)$ and $D_{static}(P)$ are then known. $D^{min}(P)$ and $J^{min}(P)$ can then be computed. Let this path be P_y. Assume that there are p WSPs already found and stored in the routing table for the same source-destination pair. Also suppose that among them, there exists a path P_x with number of hops less than or equal to P_y.

If P_x has greater (or equal) bottleneck bandwidth and lower (or equal) static delay than P_y, then all the conditions stated in Eq. (9), (10) and (11) are valid. We claim that in that case we do not need to save P_y in the routing table, as P_x is a better route.

$$B(P_y) \leq B(P_x) \tag{9}$$

$$H(P_y) \geq H(P_x) \tag{10}$$

$$D_{static}(P_y) \geq D_{static}(P_z) \quad (11)$$

Contrary to the above, if there already exists a path P_z (for the same s-d pair) such that when P_y is compared with P_z , conditions stated in Eq. (12), (13) and (14) are satisfied, then P_z could be deleted from the routing table. Of the already stored paths, those satisfying Eq. (12), (13) and (14) will be deleted and P_y will be included. The set conditions Eq. (9)~(11) and Eq. (12)~(14) are checked in turn for every new entry. Either none or only one of the set of conditions would be satisfied.

$$B(P_y) \geq B(P_z) \quad (12)$$

$$H(P_y) \leq H(P_z) \quad (13)$$

$$D_{static}(P_y) \leq D_{static}(P_z) \quad (14)$$

When Eq. (9)~(11) are satisfied, for the same flow specification, the larger (or equal) bottleneck bandwidth route P_x, has lower (or equal to) the minimum dynamic delay (see Eq. (6)) compared to path P_y. Thus if P_x has lower (or equal) static delay path and number of hops than P_y, P_y will have higher (or equal) minimum total delay than P_x. Queueing delay is equal to delay jitter of the route. Therefore path P_y also has higher (or equal) minimum delay jitter than P_x. Thus P_x is better than (or equal to) P_y considering all the QoS criteria (bandwidth, delay, and delay jitter). P_y also does not have lower number of hops than P_x. Path P_y is redundant and not needed to be stored in the routing table. Following similar arguments, we can say that P_z is to be deleted, if conditions stated in Eq. (12), (13) and (14) are valid.

The factors that may make P_y having lower total delay than path P_x are its transmission and propagation delay, as it would always have higher or equal minimum queueing delay than P_x. The route using high speed links has low transmission delay. It is likely that a route with high bottleneck bandwidth will have low transmission delay. The propagation delay depends on the transmission media and link length. The high-speed links usually are optical-fiber and has lower propagation delay than other transmission medias. Paths with longer hops also tend to have higher static delay. The conditions in Eq. (9), (10), and (11) are likely to be satisfied simultaneously in many occasions for paths using low-speed links. Therefore the number of paths needed to be stored in the routing table could be much less than maximum possible. Similarly, conditions in Eq. (12), (13), and (14) would likely be satisfied simultaneously.

While choosing a path from all of the routes available in the routing table, the minimum hop (say hop-count = i) routes are first checked for bandwidth, delay and delay jitter constraints, using Eq. (15), (16) and (17). The widest path is first checked. If it can not satisfy delay constraint, the next widest path is checked. This continues until either (1) the bandwidth or delay jitter constraint is violated, or (2) no path is available in the routing table for that hop-count. Routes with one more hop-count ($= i + 1$) are tried next. As before, the widest path is tried first. The search goes on until all routes up to maximum hop-count H_{max} is exhausted.

$$B(P) \geq \Delta_{bandwidth} \quad (15)$$

where $\Delta_{bandwidth} = \rho_q$.

$$D^{min}(P) \leq \Delta_{delay} \quad (16)$$

$$J^{min}(P) \leq \Delta_{jitter} \quad (17)$$

The minimum amount of bandwidth r_q that has to be reserved for the selected path, can be computed by Eq. (18), (19) and (20), where r_q^{delay} and r_q^{jitter} are the minimum bandwidth required to satisfy delay and delay jitter constraints respectively.

$$r_q^{delay} = \frac{\sigma_q + H(P)S_{max}^q}{\Delta_{delay} - D_{static}(P)} \quad (18)$$

$$r_q^{jitter} = \frac{\sigma_q + H(P)S_{max}^q}{\Delta_{jitter}} \quad (19)$$

$$r_q = MAX\{\rho_q, r_q^{delay}, r_q^{jitter}\} \quad (20)$$

4.2 QoS Routing Table

QoS routing table in our scheme is a 3 dimension matrix $[|V| \cdot H_{max} \cdot k]$, where $|V|$ is the number of nodes (destinations) in the network, H_{max} is the maximum number of hops for a path allowed, and k is the maximum number of WSPs to a destination. Routing table entry $[u, v, w]$ at node x contains routing information of the w^{th}WSP, $P(x, u)$, from node x to destination u, with v hops. There are 3 fields of informations in each entry.

- Path's bottleneck bandwidth: $B(P(x, u))$.
- neighbor: the next neighbor's router address of the path $P(x, u)$.
- path's static delay: $D_{static}(P(x, u))$.

5 Simulation Results

The communication networks used in our experiments are optical fiber, full duplex and directed, with homogeneous link capacity bandwidth of 155.52 Mbps (OC3). Random networks (15, 30, 45, and 60 nodes) are generated by modifying the routine described in [9]. The positions of the nodes were fixed in a rectangle of size 4000×2400 km^2, roughly the size of the USA. Nodes of all these networks were so connected that the degree of each node is at least 2, and the average node degree is 3. We initialize and divide the background traffic in each link in to 3 cases, with the following discrete random values.

1. Case 1: 5, 10, 15, ... 155 Mbps ($K = 31$)
2. Case 2: 5, 15, 25, ... 155 Mbps ($K = 15$)
3. Case 3: 5, 30, 55, ... 155 Mbps ($K = 7$)

In the algorithm, we set k to be K of each case and H_{max} to be the $|V| - 1$. Results in Figure 1 show that the average number of routes (for Case 1) after deleting redundant routes is very much smaller than before deleting. This is also true for Case 2 and Case 3, as can be compared Figure 2 with Figure 3. Because Case 1 has the largest value of K than others, there are QoS routes more than Case 2 and 3 respectively.

6 Related Works

Guerin et. al. [7] proposed WSP algorithm for exact pre-computed paths. Unlike k-WSP_{EP}, propagation and transmission delay are neglected in their paper. WSP can find a route that can satisfy bandwidth and delay jitter only. Guerin et. al. [10], and Ma and Steenkiste [8] proposed to run shortest path algorithm

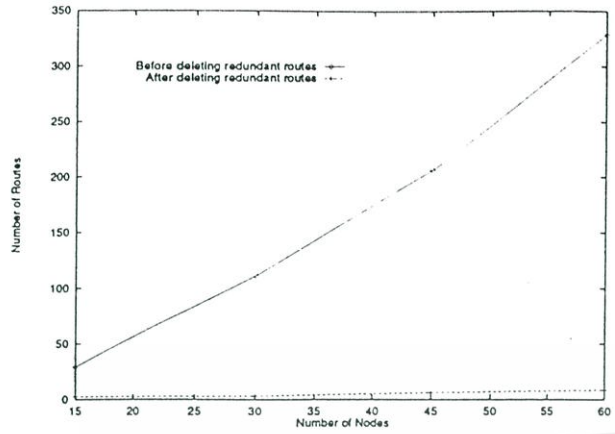


Figure 1: Average number of routes for Case 1

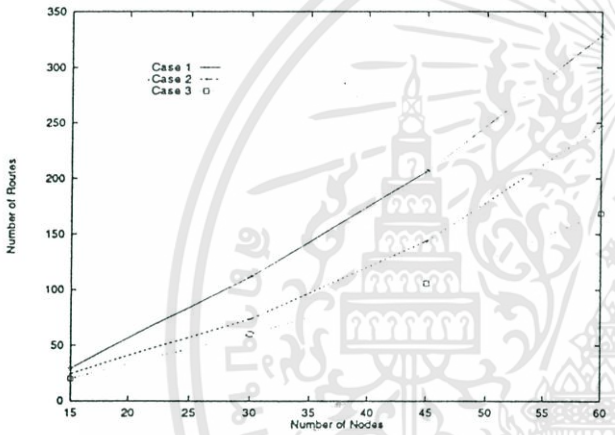


Figure 2: Average number of routes before deleting redundant routes

for each possible residual bandwidth when r_q is not known a priori. However, their algorithms were presented as on-demand algorithms. σ_q and S^q_{max} are assumed to be known a priori. Definition of link delay is also different from ours. Moreover, we show that the paths satisfying the defined conditions, are not needed to be stored in the routing table, as better path is already stored.

7 Conclusion

In this paper we present a precomputed QoS routing algorithm that could create routing table supporting traffic request for which flow specification parameters could take any values and may not be known a priori. The size of the routing table would be $|E|$ times that of WSP, in the worst case. However, paths satisfying the defined conditions could be ignored. Simulation results showed that many paths are redundant. Therefore the required routing table is very much smaller.

Acknowledgments

The authors would like to thank to JICA for their supports.

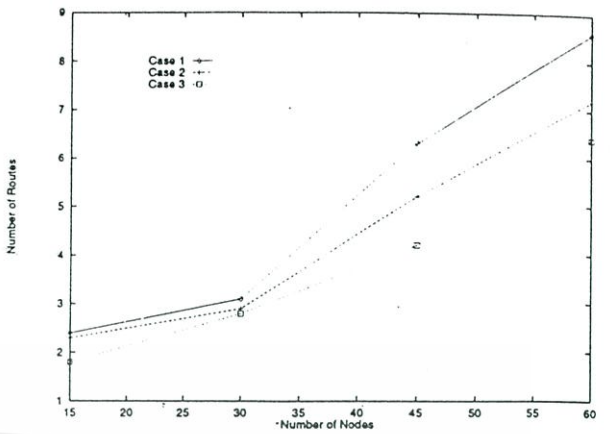


Figure 3: Average number of routes after deleting redundant routes

References

[1] C. Pornavalai, G. Chakraborty, and N. Shiratori. Qos based routing algorithm in integrated services packet networks. *Journal of High Speed Network*, 1998.

[2] C. Pornavalai, G. Chakraborty, and N. Shiratori. Qos routing algorithms for pre-computed paths. In *6th International Conference on Computer Communications and Networks (IC3N'97)*, September 1997.

[3] Z. Wang and J. Crowcroft. Quality of service routing for supporting multimedia applications. *IEEE J. Select. Areas Commun.*, 14(7):1228–1234, 1996.

[4] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. *Internet Draft*, August 1996.

[5] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM Trans. Networking*, 2(2):137–150, April 1994.

[6] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proc. of the IEEE*, 83(10), October 1995.

[7] R. Guerin, A. Orda, and D. Williams. Qos routing mechanisms and ospf extensions. *Internet Draft*, March 1997.

[8] Q. Ma and P. Steenkiste. Quality-of-service routing for traffic with performance guarantees. In *IFIP Fifth International Workshop on Quality of Service (IWQOS'97)*, May 1997.

[9] B. M. Waxman. Routing of multipoint connections. *IEEE J. Select. Areas Commun.*, 6(9):1617–1622, 1988.

[10] R. Guerin and A. Orda. Qos-based routing in networks with inaccurate information: Theory and algorithms. In *Proc. of IEEE INFOCOM*, April 1997.

ประวัติผู้เขียน

ชื่อผู้เขียน	นางสาวฟองแก้ว แก้วยองผาง
วันเดือนปีเกิด	17 พฤศจิกายน 2515.
สถานที่เกิด	จังหวัดลำพูน
วุฒิการศึกษาระดับปริญญาตรี	สำเร็จการศึกษาวิทยาศาสตรบัณฑิต สาขาวัสดุศาสตร์
สถานที่สำเร็จการศึกษา	มหาวิทยาลัยเชียงใหม่
ปีที่สำเร็จการศึกษา	ปีการศึกษา 2537.
ผลงานวิจัยที่ได้รับการตีพิมพ์	

1. ฟองแก้ว แก้วยองผาง, ศักดิ์ชัย ทิพย์จักษุรัตน์ และรัตติกร วรากรศิริพันธุ์. **"การออกแบบ และการวิเคราะห์ทางสถาปัตยกรรมของเครือข่ายสื่อสารคอมพิวเตอร์".** วิศวสาร ลาดกระบัง คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. ปีที่ 13, ฉบับที่ 2, เมษายน 2540. หน้า 59 - 66.

2. Kaewyongphang P., Pornavalai C., Varakulsiripunth R., Chakraborty G. and Shiratori. N. "On Precomputing of QoS Paths." **In IEEE International Workshop on Intelligent Signal Processing and Communication Systems. (ISPACS' 99).** Phuket, Thailand, December 1999. pp. 17 - 20.