การพัฒนาโปรแกรมจำลองการทำงานสำหรับเครือข่ายคอมพิวเตอร์

DEVELOPMENT OF SIMULATION PROGRAM FOR COMPUTER NETWORKS

กำพล ฉัตรมณีฤกษ์

นันทวัฒน์ เดชเจริญ

นิติรัฐ แสงชัยอรุณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

ปริญญานิพนธ์ปีการศึกษา 2553

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาโปรแกรมจำลองการทำงานสำหรับเครือข่ายคอมพิวเตอร์

DEVELOPMENT OF SIMULATION PROGRAM FOR COMPUTER NETWORKS

ผู้จัดทำ

1. นายกำพล ฉัตรมณีฤกษ์ รหัสนักศึกษา 50010092
2. นายนันทวัฒน์ เดชเจริญ รหัสนักศึกษา 50010802
3. นายนิติรัฐ แสงชัยอรุณ รหัสนักศึกษา 50010820

_____ อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ ดร. ศักดิ์ชัย ทิพย์จักษุรัตน์)

การพัฒนาโปรแกรมจำลองการทำงานสำหรับเครือข่ายคอมพิวเตอร์

นายกำพล ฉัตรมณีฤกษ์ 50010092
นายนันทวัฒน์ เดชเจริญ 50010802
นายนิติรัฐ แสงชัยอรุณ 50010820
ผศ.ดร. ศักดิ์ชัย ทิพย์จักษุรัตน์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2553

บทคัดย่อ

การจำลองการทำงานด้วยระบบคอมพิวเตอร์เป็นเทคนิคที่ทำให้เราสามารถศึกษาการทำงานของระบบต่างๆ ได้โดยไม่จำเป็นต้องติดตั้งระบบจริงขึ้น ซึ่งทำให้เราประหยัดค่าใช้จ่ายในการศึกษาการทำงานของระบบ ปริญญานิพนธ์ฉบับนี้เราได้พัฒนาระบบการจำลองการทำงานสำหรับเครือข่ายคอมพิวเตอร์ โดยใช้ทฤษฎีการจำลองการทำงานด้วยระบบคอมพิวเตอร์มาประยุกต์ใช้สำหรับการสื่อสารในเครือข่ายคอมพิวเตอร์โดยอ้างอิงกับการส่งข้อมูลของโพรโทคอลทีซีพี/ไอพีซึ่งเป็นโพรโตคอลที่ใช้สำหรับเครือข่ายอินเทอร์เน็ตในปัจจุบัน และเพื่อให้การแสดงผลการจำลองการทำงานของระบบมีความน่าสนใจมากยิ่งขึ้น เราได้แสดงผลในรูปแบบของภาพเคลื่อนไหว และแสดงประสิทธิภาพของระบบในรูปแบบของกราฟดังนี้คือ จำนวนแพกเกตที่ส่งสำเร็จ, จำนวนแพกเกตที่สูญหาย, ดีเลย์เฉลี่ยของการรอในคิวและทรูพุท นอกจากนี้ประโยชน์ที่ได้จากระบบที่เราพัฒนาขึ้น สามารถนำไปช่วยในการสอนวิชาเครือข่ายคอมพิวเตอร์ได้

DEVELOPMENT OF SIMULATION PROGRAM FOR COMPUTER NETWORKS

Mr.Kumpon Chatmaneelerk 50010092

Mr.Nantawat Dejcharoen 50010802

Mr.Nitirat Saengchaiarun 50010820

Asst. Prof. Dr. Sakchai Thipchaksurat Advisor

Academic Year 2010

ABSTRACT

Simulation of computer network is technique that helpful of us to study the behavior of any system without setting up the real system. Therefore, the cost of this study will be reduced. In this thesis, we have developed the simulation for computer networks by using the theory of computer simulation applied to the communication in computer networks. We focus on the TCP/IP Protocols which are widely used in the Internet. To make our system more interesting, the simulation processes are shown as the animation including the graphs of system performance. In our system we have shown four performance metrics such as the number of successful packet, the number of dropped packet, the average queuing delayed and throughput. Moreover, the benefits from this system that we have developed can be used for Computer-Aided Instruction in the computer networks course.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดีด้วยคำแนะนำและคำปรึกษาจาก ผศ.ดร.ศักดิ์ชัย ทิพย์จักษุรัตน์ ซึ่งเป็นอาจารย์ผู้ควบคุมปริญญานิพนธ์ พวกเรารู้สึกซาบซึ้งในความอนุเคราะห์จากท่านอาจารย์ และขอขอบพระคุณเป็นอย่างสูง

ขอกราบขอบพระคุณคณาจารย์สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกๆ ท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับพวกเรา

ขอขอบคุณห้อง Multimedia Laboratory ณ ECC 801 ที่ได้เอื้อเฟื้อสถานที่ทำงานให้แก่คณะผู้จัดทำ และขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ในสาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกคนที่ให้คำแนะนำต่างๆ และคอยให้กำลังใจเสมอมา

สุดท้ายนี้พวกเราขอกราบขอบพระคุณบิดา มารดา และครอบครัวของพวกเราที่เป็นกำลังใจและให้การสนับสนุนในทุกๆ เรื่อง ทำให้พวกเราสามารถทำปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี โดยที่คุณค่าและประโยชน์อันพึงมาจากปริญญานิพนธ์ฉบับนี้ พวกเราขอมอบแด่ผู้มีพระคุณทุกท่าน

กำพล ฉัตรมณีฤกษ์
นันทวัฒน์ เดชเจริญ
นิติรัฐ แสงชัยอรุณ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

รูป หน้า

สารบัญรูป (ต่อ)

รูป หน้า

สารบัญรูป (ต่อ)

รูป หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญและความเป็นมา

เนื่องจากในปัจจุบันได้มีการพัฒนาระบบเครือข่ายมาอย่างต่อเนื่องตั้งแต่อดีตจนถึงปัจจุบัน ซึ่งการที่พัฒนาระบบเครือข่ายนั้นจะต้องมีการวางโครงสร้างและระบบผ่านอุปกรณ์ต่างๆ ที่ใช้ในการเชื่อมต่อเครือข่ายมากมาย โดยการที่จะติดตั้งระบบเครือข่ายสำหรับการใช้งานนั้นจะต้องมีการทดสอบการทำงานของระบบเครือข่ายจึงเป็นการยากที่จะทดสอบการทำงานของระบบเครือข่ายโดยการทดสอบผ่านอุปกรณ์ต่างๆ ที่ได้ทำการวางระบบไว้แล้ว เนื่องจากการวางระบบเครือข่ายนั้นเป็นสิ่งที่ใช้งบประมาณและเวลามากจึงไม่คุ้มที่จะทดสอบการทำงานของระบบเครือข่ายโดยการทดสอบผ่านอุปกรณ์ที่ใช้ในความเป็นจริง ทำให้เกิดปัญหาสำหรับผู้ที่จะศึกษาค้นคว้าเพื่อทำความเข้าใจการทำงานของระบบเครือข่าย จึงเกิดความจำเป็นที่จะต้องมีการจำลองการทำงานของระบบเครือข่ายเพื่อที่จะได้ทำการศึกษาการทำงานของระบบเครือข่ายในด้านต่างๆ ซึ่งในปัจจุบันได้มีโปรแกรมที่ช่วยในการจำลองการทำงานของระบบเครือข่ายมากมาย เช่น NS2 และ Packet Tracer เป็นต้น เมื่อมีโปรแกรมที่ช่วยในการจำลองการทำงานของระบบเครือข่ายจึงเป็นประโยชน์ต่อการศึกษาค้นคว้า

โครงงานนี้เป็นการนำเสนอโปรแกรมการจำลองการทำงานของระบบเครือข่าย (Network Simulation) ไม่ว่าจะเป็นการจำลองส่งแพ็คเกต การให้บริการแบบ M/M/1 จำลองการทำงานของเราท์เตอร์ ประกอบกับมีการแสดงประสิทธิภาพของระบบผ่านในรูปแบบกราฟแสดงผล

โครงงานนี้เป็นโครงงานที่พัฒนาจากโครงงานระบบจำลองการทำงานสำหรับเครือข่ายคอมพิวเตอร์โดยได้ปรับปรุงส่วนการเลือกอุปกรณ์ให้มีความอิสระมากขึ้นเพื่อตอบสนองความต้องการของผู้ที่มาศึกษาได้อย่างเต็มที่ อีกทั้งยังมีการนำโพรโทคอลทีซีพีมาใช้ในการรับส่งข้อมูลเพื่อให้โปรแกรมการจำลองการทำงานของระบบเครือข่ายมีความใกล้เคียงกับความเป็นจริงมากขึ้น และเพื่อให้การใช้งานของโปรแกรมสะดวกต่อการใช้งานเราจึงนำโปรแกรมจำลองการทำงานของระบบเครือข่ายมาไว้บนเว็บไซต์

1.2 วัตถุประสงค์ของโครงงาน

1) ใช้เป็นสื่อการเรียนการสอนวิชาคอมพิวเตอร์เน็ตเวิร์ค

2) เพื่อศึกษาและพัฒนาระบบจำลองการทำงานของเครือข่ายคอมพิวเตอร์

1.3 ขอบเขตของโครงงาน

เป็นโปรแกรมเพื่อศึกษาและพัฒนาระบบจำลองการทำงานของเครือข่ายคอมพิวเตอร์และใช้ประกอบในการเรียนการสอนวิชาคอมพิวเตอร์เน็ตเวิร์ค ซึ่งโปรแกรมนี้จำลองการส่งแพ็คเกตโดยเน้นความรู้การส่งข้อมูลโดยใช้โพรโทคอลทีซีพีไอพีเป็นหลัก โดยใช้เทคนิคการพัฒนาระบบการจำลองการทำงานแบบ Event-Driven และแสดงประสิทธิภาพการทำงานของระบบออกมาในรูปแบบของกราฟ โดยให้แสดงผลการจำลองการทำงานออกมาในรูปแบบของแอนิเมชัน (Animation) เพื่อให้ผู้ศึกษามีความเข้าใจมากยิ่งขึ้น

1.4 วิธีการดำเนินงาน

1) วางแผนการทำงานโครงงานทั้งหมด
2) ศึกษาถึงการทำงานของการจำลองการทำงานของระบบเครือข่าย (Network Simulation)
3) วิเคราะห์และวางแผนรูปแบบของโปรแกรมที่ต้องการจะพัฒนาและออกแบบโปรแกรม
4) ศึกษาการใช้งานแอคชั่นสคริปต์ (Action Script)
5) พัฒนาโปรแกรมตามที่ได้ออกแบบไว้
6) ตรวจสอบการทำงานของโปรแกรม
7) สรุปผลการทำงานและจัดทำเอกสาร

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้ความรู้เกี่ยวกับการออกแบบและพัฒนาโปรแกรมประเภทการจำลองการทำงานของระบบเครือข่ายคอมพิวเตอร์ (Computer Network Simulation)
2) ได้ความรู้เกี่ยวกับการเขียนโปรแกรมด้วยภาษาแอคชั่นสคริปต์ (Action Script)
3) ได้ความรู้เกี่ยวกับการทำแอนิเมชัน (Animation)

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ทฤษฎีแถวคอย

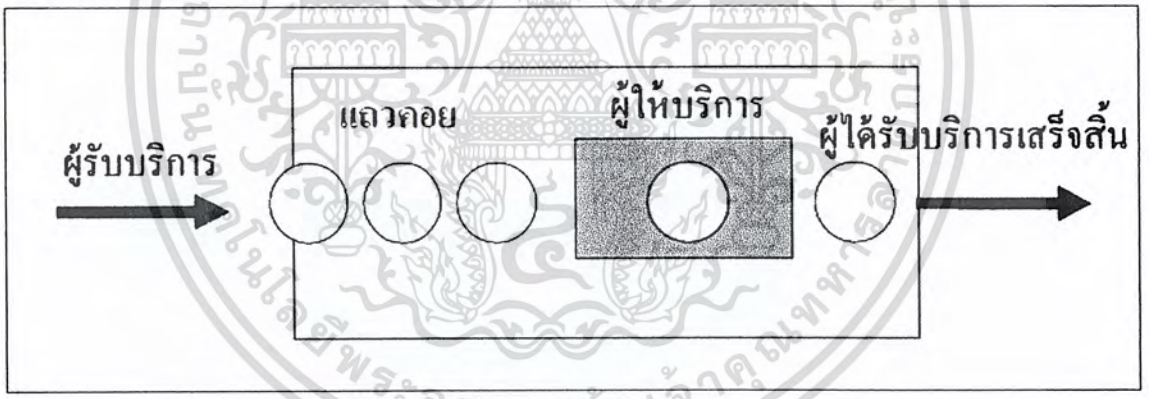
ทฤษฎีแถวคอย (Queuing Theory) จะมีแบบจำลองเชิงปริมาณที่มีลักษณะแตกต่างกันหลายแบบขึ้นอยู่กับรูปแบบและลักษณะของผู้เข้ามารับบริการลักษณะของผู้ให้บริการและลักษณะของแถวคอยนอกจากนี้ยังเกี่ยวข้องกับปัจจัยอื่นๆเช่นพฤติกรรมของผู้ที่เข้ามารับบริการที่อยู่ในระบบแถวคอย เป็นต้น ซึ่งในการศึกษาแถวคอย ผู้ศึกษาจะต้องแยกส่วนประกอบต่าง ๆ ของโครงสร้างระบบแถวคอยให้มีความชัดเจน เพื่อที่จะสามารถทำความเข้าใจแถวคอยนั้น ๆ ได้อย่างถูกต้อง

2.1.1 องค์ประกอบของระบบแถวคอย

1) ผู้รับบริการ (Customer)

2) ผู้ให้บริการ (Server)

3) แถวคอย (Queue)



รูป 2.1 องค์ประกอบระบบของแถวคอย

2.1.2 ลักษณะของระบบแถวคอย

เราสามารถพิจารณาลักษณะของแถวคอยได้ 2 แบบ คือ ระเบียบการให้บริการ (Queue Discipline) และผังการให้บริการ (Physical Layouts)

1) ระเบียบการให้บริการ กฎเกณฑ์ที่ใช้ในการให้บริการว่าจะให้บริการแก่ผู้มารับบริการรายใดก่อน เช่น

1.1) มาก่อนรับบริการก่อน (First Come First Serve, FCFS)

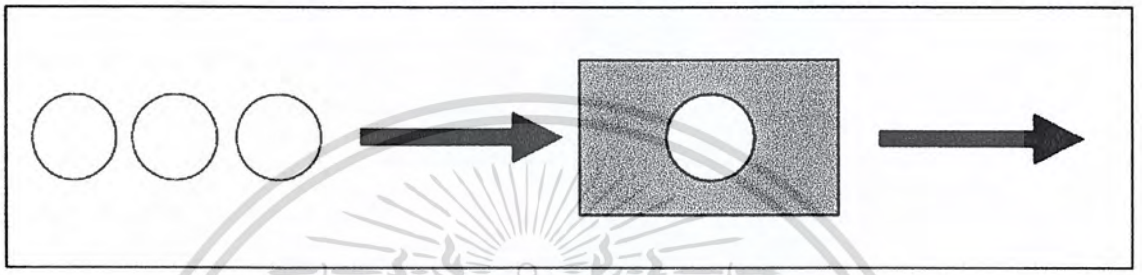
1.2) มาทีหลังรับบริการก่อน (Last Come First Serve, LCFS)

1.3) ผู้มารับบริการที่มีความจำเป็นได้รับบริการก่อน

2) ผังการให้บริการ จำนวนผู้ให้บริการมีจำนวนเท่าไร และขั้นตอนการให้บริการมีกี่ขั้นตอน ซึ่งเราสามารถที่จะสรุปผังการให้บริการ

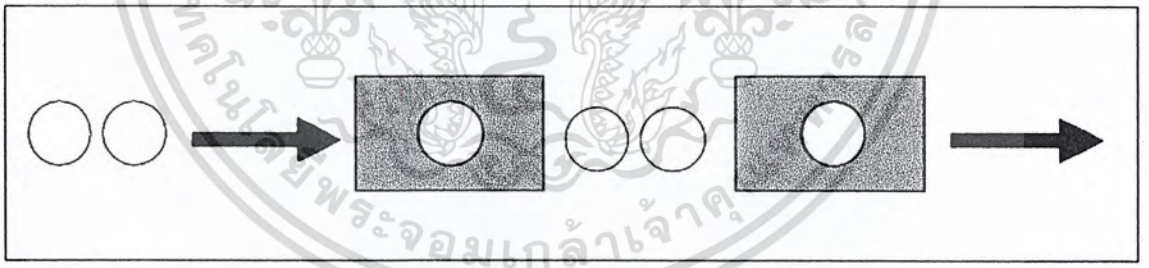
2.1.3 รูปแบบของระบบแถวคอย

1) ระบบแถวคอยแบบช่องทางเดียว – ขั้นตอนเดียว (single-channel-single-phase system) คือระบบแถวคอยที่มีผู้ให้บริการคนเดียวและมีขั้นตอนเดียว เมื่อผู้มารับบริการรับบริการเสร็จแล้วก็จะออกจากระบบไป



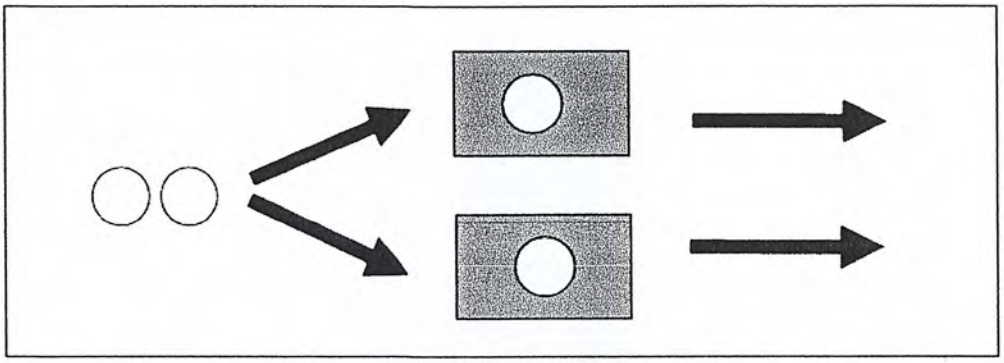
รูป 2.2 ระบบแถวคอยแบบช่องทางเดียว – ขั้นตอนเดียว

2) ระบบแถวคอยแบบช่องทางเดียว – หลายขั้นตอน (single channel multiple phase system) คือ ระบบแถวคอยที่ขั้นตอนการบริการหลายขั้นตอน (มากกว่า1ขั้นตอน) และแต่ละขั้นตอนมีผู้ให้บริการเดียว



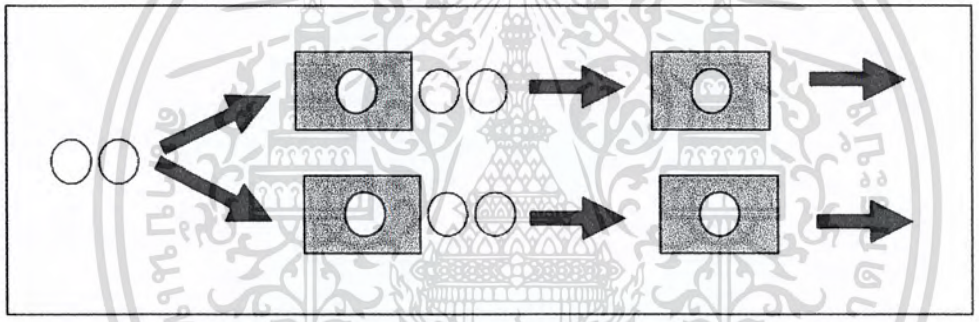
รูป 2.3 ระบบแถวคอยแบบช่องทางเดียว - หลายขั้นตอน

3) ระบบแถวคอยแบบหลายช่องทาง-ขั้นตอนเดียว (multiple channel single phase system) คือ ระบบแถวคอยที่มีขั้นตอนการบริการขั้นตอนเดียว แต่มีผู้ให้บริการมากกว่า 1 คน



รูป 2.4 ระบบแถวคอยแบบหลายช่องทาง-ขั้นตอนเดียว

4) ระบบแถวคอยแบบหลายช่องทาง - หลายขั้นตอน (multiple channel multiple phase system) คือ ระบบแถวคอยที่มีขั้นตอนการบริการหลายขั้นตอน และแต่ขั้นตอนมีผู้ให้บริการ มากกว่า 1 คน



รูป 2.5 ระบบแถวคอยแบบหลายช่องทาง - หลายขั้นตอน

2.1.4 ลักษณะของผู้มารับบริการ

2.1.4.1 การเข้ามารับบริการแบบคงที่

หมายถึง การเข้ามารับบริการในอัตราที่สม่ำเสมอ เช่น ในระบบสายการผลิตของโรงงานอุตสาหกรรมต่างๆ เป็นต้น

2.1.4.2 การเข้ามารับบริการแบบสุ่ม

หมายถึง การเข้ามารับบริการมีลักษณะที่ไม่แน่นอนไม่สม่ำเสมอ ไม่สามารถทราบล่วงหน้าและการเข้ามารับบริการในแต่ละรายจะมีความเป็นอิสระต่อกัน โดยปกติแล้วลักษณะการเข้ามารับบริการส่วนใหญ่จะเป็นแบบสุ่ม โดยที่อัตราการเข้ามารับบริการจะมีการแจกแจงแบบปัวส์ซอง

ในการเก็บข้อมูลการเข้ามารับบริการของผู้มารับบริการทำได้ 2 ลักษณะคือ

1) อัตราการเข้ารับบริการ (arrival rate) หมายถึง ผู้มารับบริการเข้ามารับบริการโดยเฉลี่ยกี่คนในหนึ่งหน่วยเวลา

2) เวลาระหว่างการเข้ารับบริการ (arrival time interval) หมายถึง เวลาห่างโดยเฉลี่ยระหว่างผู้มารับบริการแต่ละคน

ระบบแถวคอยส่วนใหญ่จะมีลักษณะการมารับบริการแบบสุ่ม โดยที่อัตราการเข้ามารับบริการมีการแจกแจงแบบปัวส์ซอง ซึ่งสามารถคำนวณความน่าจะเป็นที่จะมีผู้มารับบริการเข้ามา X รายได้ดังนี้

$$P(x) = \frac{e^{-\lambda t}(\lambda t)^n}{n!} \qquad x = 1,2,3,\ldots n \qquad (2.1)$$

x = จำนวนผู้มารับบริการต่อหน่วยเวลา

λ = อัตราการเข้ามารับบริการ

e = 2.7183

2.1.5 ลักษณะหน่วยบริการ

มีประเด็นสำคัญที่ต้องพิจารณาจะเกี่ยวข้องกับอัตราการให้บริการแก่ผู้มารับบริการ (Service Rate) ซึ่งการให้บริการของหน่วยบริการจะมีการให้บริการ 2 ลักษณะ

2.1.5.1 อัตราการให้บริการแบบคงที่

อัตราการให้บริการแบบคงที่ หมายถึง การให้บริการในแต่ละรายจะใช้เวลาที่เท่าๆ กัน เช่น การบรรจุน้ำดื่มลงขวดด้วยเครื่องจักร ซึ่งแต่ละขวดจะใช้เวลา 5 วินาที ดังนั้น อัตราการให้บริการจะเท่ากับ 12 ขวดต่อนาที

2.1.5.2 อัตราการให้บริการแบบสุ่ม

อัตราการให้บริการแบบสุ่ม หมายถึง การให้บริการในแต่ละรายใช้เวลาที่ไม่เท่ากัน ซึ่งจะขึ้นอยู่กับความต้องการของผู้มารับบริการแต่ละราย การรวบรวมข้อมูลของการให้บริการมักจะอยู่ในรูปของเวลาที่ใช้ในการบริการ (Service Time) ของแต่ละรายแล้วนำมาหาค่าเฉลี่ย โดยส่วนใหญ่ลักษณะหน่วยบริการจะเป็นเวลาที่ใช้ในการบริการแบบสุ่มและมีการแจกแจงแบบเอ็กซ์โปแนนเชียล

$$P(\text{service time} > x) = e^{-\mu_t}, \quad x \geq 0 \qquad (2.2)$$

x = จำนวนผู้มารับบริการต่อหน่วยเวลา

μ = อัตราการให้บริการ

e = 2.7183

2.1.6 รูปแบบต่างๆของปัญหาแถวคอย

ในปัญหาของแถวคอยจะมีแบบจำลองที่มีความหลากหลายขึ้นอยู่กับลักษณะของข้อมูลพื้นฐานของระบบ ซึ่งจะเป็นการพิจารณาถึงลักษณะการแจกแจงความน่าจะเป็นของการเข้าสู่ระบบ และเวลาที่ใช้ในการให้บริการ ดังนั้นการแสดงลักษณะของแบบจำลองจึงมีความสำคัญที่จะทำให้เกิดความเข้าใจที่ง่ายและตรงกัน โดยใช้สัญลักษณ์เคนดอล (Kendoll Notation) ดังนี้

$$A / B / s \qquad (2.3)$$

ซึ่งสามารถอธิบายสัญลักษณ์ได้ดังนี้

1) A หมายถึง การแจกแจงความน่าจะเป็นของการเข้าสู่ระบบแถวคอย

2) B หมายถึง การแจกแจงความน่าจะเป็นของเวลาการให้บริการ

3) s หมายถึง จำนวนหน่วยให้บริการ (s = 1, 2, ...)

การแจกแจงของการเข้าสู่ระบบแถวคอยและการแจกแจงของเวลาให้บริการ จะมีการแจกแจงในรูปแบบต่างๆ ซึ่งจะมีการกำหนดสัญลักษณ์ที่จะบอกถึงลักษณะของการแจกแจงดังนี้

1) M หมายถึง การแจกแจงแบบปัวส์ซอง สำหรับการเข้ามารับบริการและการแจกแจงแบบเอ็กซ์โปแนนเชียลสำหรับเวลาที่ให้บริการ

2) D หมายถึง การแจกแจงแบบคงที่

3) G หมายถึง เวลาที่ให้บริการมีการแจกแจงแบบทั่วไป

เช่น M/M/1 จะแสดงถึงการเข้ามารับบริการแจกแจงแบบปัวส์ซอง ส่วนเวลาในการให้บริการจะมีการแจกแจงแบบเอ็กซ์โปแนนเชียลและมีหน่วยบริการ 1 หน่วย เป็นต้น

2.1.7 รูปแบบพื้นฐาน M/M/1

รูปแบบพื้นฐาน M/M/1 จะมีลักษณะที่สำคัญของแบบจำลอง ดังนี้

1) อัตราการเข้ามารับบริการมีการแจกแจงแบบปัวส์ซอง

2) เวลาที่ให้บริการที่การแจกแจงแบบเอ็กซ์โปแนนเชียล

3) ระเบียบบริการเป็นแบบมาก่อนได้รับบริการก่อน

4) ความยาวแถวคอยไม่จำกัด

5) จำนวนประชากรไม่จำกัด

6) มีผู้ให้บริการ 1 หน่วย

7) อัตราการเข้ารับบริการน้อยกว่าอัตราการให้บริการ

2.2 หลักการทำงานพื้นฐานของระบบการจำลอง (Simulation)

Simulation คือ การนำเสนอหรือการจำลองลักษณะของระบบอื่นๆ ตลอดช่วงเวลาที่สนใจ ซึ่งในกรณีที่กล่าวถึง Computer simulation จะหมายถึงโปรแกรมคอมพิวเตอร์ที่จำลองการทำงานของระบบที่สนใจ

ระบบ Simulation จะแบ่งเป็นสี่ส่วนหลักๆ คือ

1) Computer Simulation คือ โปรแกรมคอมพิวเตอร์ที่จำลองลักษณะการทำงานของระบบที่สนใจตลอดเวลาหรือเฉพาะช่วงเวลาที่ให้ความสนใจ

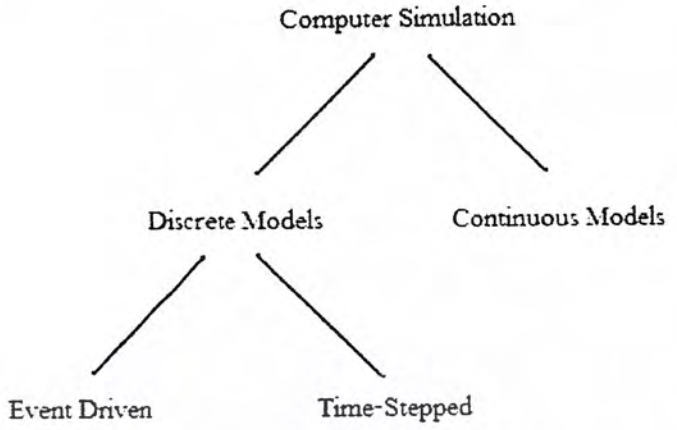
2) Program variables หรือ State variable เป็นตัวแปรซึ่งใช้เป็นสื่อกลางในการแสดงถึงสถานะปัจจุบันของระบบที่จำลอง

3) Simulation Program จะปรับเปลี่ยนค่าของ State variable เพื่อที่จะนำไปพัฒนาแบบจำลองไปตามช่วงเวลาที่เปลี่ยนแปลงไปเรื่อยๆ

4) รูปแบบของเวลาที่มีการใช้งานในระบบ Simulation

2.3 SYSTEM, MODEL, AND SIMULATION

ระบบ Computer Simulation จะแบ่งประเภทรูปแบบการทำงานออกเป็นสองประเภท คือ Continuous time simulation และ discrete time simulation โดยที่แต่ละแบบนั้นจะมีลักษณะการทำงานที่แตกต่างกันการเลือกใช้จึงควรเลือกให้เหมาะกับลักษณะของงาน



รูป 2.6 Model แสดงประเภทของการทำงานใน Computer Simulation

2.3.1 Continuous time simulation

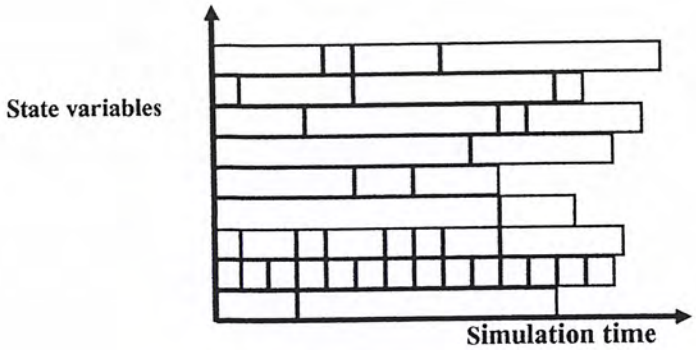
Continuous time simulation เป็นระบบจำลองที่ให้ความสนใจการเปลี่ยนแปลงในแบบจำลองตลอดเวลาตั้งแต่เริ่มต้นทำงานจนสิ้นสุดการทำงาน ไม่ว่าระหว่างการทำงานของแบบจำลองจะมีการเปลี่ยนแปลงหรือไม่มีการเปลี่ยนแปลงใดๆเกิดขึ้นก็ตาม ซึ่งลักษณะของแบบจำลองที่ได้มักจะขึ้นกับสมการที่จะนำมาใช้ในการอธิบายแบบจำลอง แบบจำลองที่เหมาะสมกับการทำงานแบบนี้มักจะเป็นแบบจำลองที่ต้องการความต่อเนื่องในการทำงาน

2.3.2 Discrete time simulation

Discrete time simulation เป็นระบบจำลองที่ให้ความสนใจการเปลี่ยนแปลงในลักษณะเป็นช่วงของเวลาที่ไม่ต่อเนื่อง คือ ไม่ต้องเก็บข้อมูลตลอดเวลาที่จำลองการทำงาน ซึ่งสามารถแบ่งเป็นรูปแบบย่อยๆ ได้ 2 รูปแบบ คือ

2.3.2.1 Event stepped

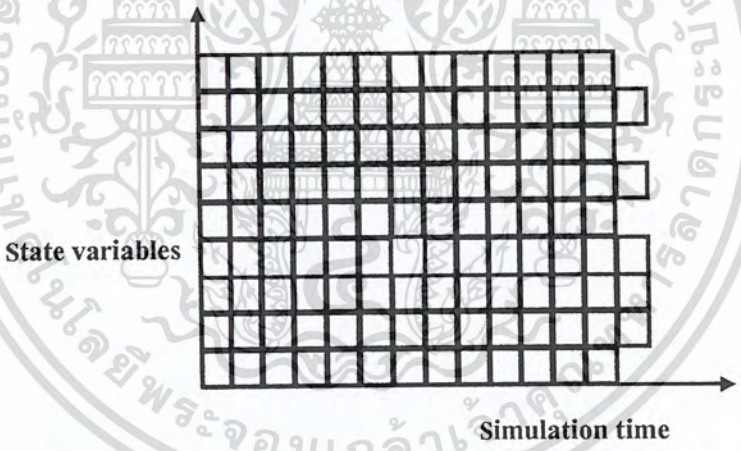
เป็นแบบจำลองที่จะสนใจต่อเหตุการณ์ที่มีการเปลี่ยนแปลงไป ดังนั้นการเลื่อนไปของเวลาที่ใช้ในแบบจำลองจะขึ้นอยู่กับการเกิดเหตุการณ์ต่างๆ หากมีเหตุการณ์ใดเหตุการณ์หนึ่งเกิดขึ้นจึงจะมีการเลื่อนไปของเวลา แบบจำลองที่มีความเหมาะสม เช่น แบบจำลองของสนามบินอย่างง่ายซึ่งจะสนใจการขึ้นและลงของเครื่องบิน นั้นคือจะมีการเปลี่ยนแปลงเมื่อเครื่องบันมีการบินขึ้นและบินลงในสนามบิน



รูป 2.7 state variable ของ DISCRETE-EVENT SIMULATION

2.3.2.2 Time stepped

เป็นแบบจำลองที่จะเก็บข้อมูลเป็นช่วงๆ ของเวลา ดังนั้นเวลาที่จะใช้ในการทำงานจะมีการเลื่อนไปด้วยอัตราคงที่ ซึ่งจะขึ้นอยู่กับการกำหนดของแบบจำลอง เช่น แบบจำลองการเปลี่ยนแปลงระดับน้ำของทะเล ซึ่งจะเก็บการเปลี่ยนแปลงทุกๆ ครึ่งชั่วโมง



รูป 2.8 state variable ของ DISCRETE-TIME SIMULATION

2.4 DISCRETE-EVENT SIMULATION (DES)

Discrete event simulation เป็นโมเดลทางคอมพิวเตอร์ สำหรับระบบใดๆ การที่เปลี่ยนแปลงทางสถานะของระบบเกิดขึ้น ณ จุดที่ไม่ต่อเนื่องกันและไม่มีความสัมพันธ์กัน บนแกนเวลาของการจำลอง แนวคิดพื้นฐานคือ ระบบประกอบไปด้วย

1) สถานะของระบบ (System State) ในระบบจำลองแทนด้วยตัวแปรสถานะ (State Variable)

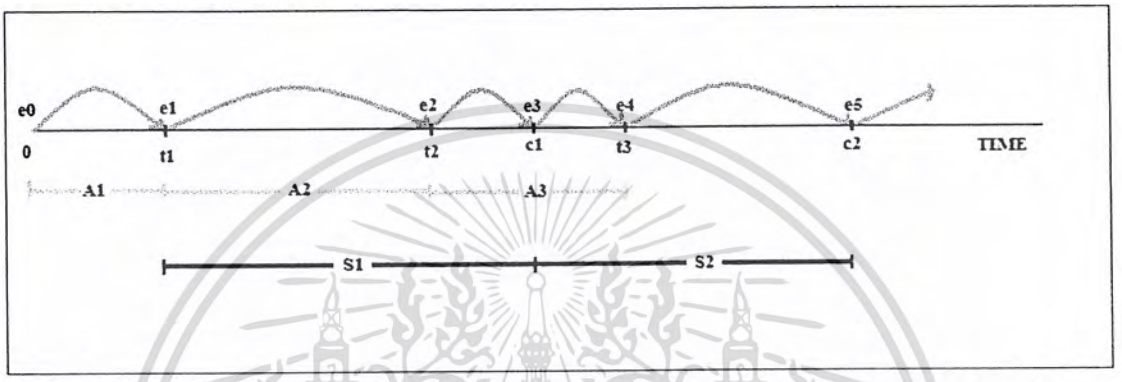
2) การเปลี่ยนแปลงสถานะ (State transitions) ในระบบจำลองแทนด้วยเหตุการณ์ (event)

สรุปก็คือเหตุการณ์ที่เกิดขึ้นส่งผลให้สถานะของระบบเปลี่ยนแปลง หากไม่มีเหตุการณ์ใดๆ เกิดขึ้น สถานะของระบบใน Discrete event simulation ก็จะไม่เปลี่ยนแปลง การประมวลผลหรือ

การคำนวณ ใน DES สามารถเปรียบได้กับลำดับของการประมวลผลเหตุการณ์ต่างๆ หรือลำดับของการเกิดเหตุการณ์ต่างๆ โดยที่แต่ละเหตุการณ์จะถูกกำหนดให้เกิดขึ้น ณ เวลาใดเวลาหนึ่ง ในเวลาจำลองหรือ time stamp

2.4.1 TIME-ADVANCE MECHANISMS

TIME-ADVANCE MECHANISMS หรือ ตัวจัดการการขยับเวลาในระบบจำลอง ยกตัวอย่างจากระบบจำลอง แบบ Discrete-event



รูป 2.9 TIME-ADVANCE MECHANISMS ของ DISCRETE-EVENT SIMULATION

t_i = เวลาเริ่มต้นเมื่อมีงานเข้ามาในระบบ

A_i = $t_i - t_{i-1}$ หรือระยะเวลาที่งานเข้ามาในระบบลบกับเวลาที่งานต่อไปเข้ามา

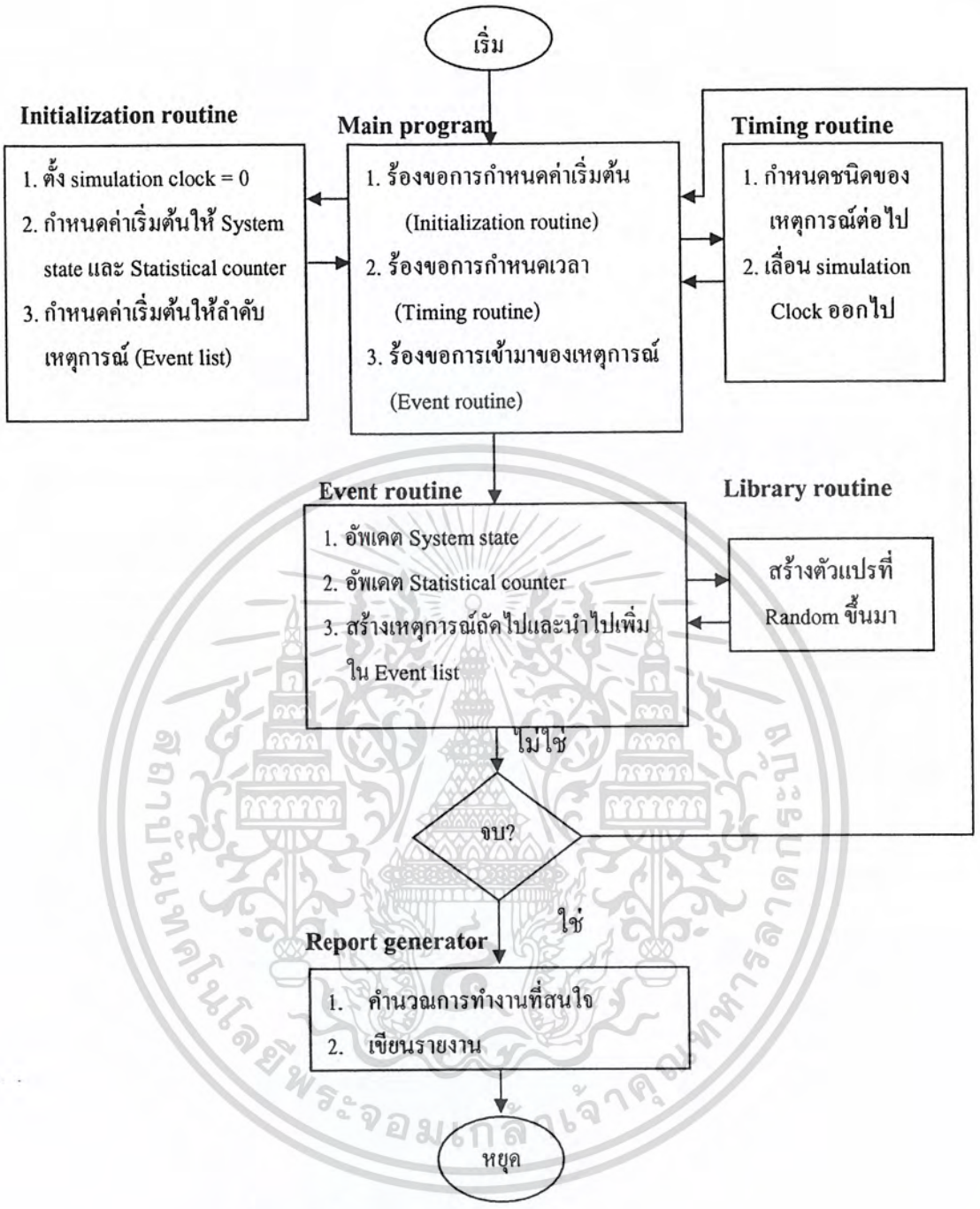
S_i = เวลารวมที่งานนั้นใช้ทรัพยากรในระบบจนงานเสร็จสิ้น

D_i = ค่าเวลารวมที่งานรออยู่ในคิว

c_i = $t_i + D_i + S_i$ หรือ เวลาเริ่มต้นที่งานเข้ามาในระบบ รวมกับค่าดีเลย์และเวลาที่งานใช้ทรัพยากรในระบบ

e_i = เวลาใดๆก็ตามที่มีเหตุการณ์เข้ามาในระบบ

อธิบายการทำงานแบบคร่าวๆ ได้ว่าเมื่อมี งาน$_1$ เข้ามา ที่เวลา t_1ขณะนั้นทรัพยากรในระบบไม่ได้ใช้งาน งาน$_1$ จึงสามารถใช้งานทรัพยากรได้เลย จากรูปที่ 2.9 ระยะเวลา S_1 คือระยะเวลาที่งาน$_1$ ใช้ทรัพยากรในระบบ และทำงานเสร็จเมื่อเวลา c_1 ระหว่างที่ งาน$_1$ ใช้งานระบบอยู่นั้น งาน$_2$ ได้เข้ามาขอใช้ระบบที่เวลา t_2 จะเห็นว่าระบบจะไม่ว่างเนื่องจาก งาน$_1$ กำลังใช้งานอยู่ งาน$_2$ ก็จะไปรอในคิว และจะรอจนกว่า งาน$_1$ทำงานเสร็จเรียบร้อยจึงจะเริ่มการใช้ทรัพยากรในระบบต่อไป



รูป 2.10 ผังการทำงานของ Discrete-Event Simulation

2.4.2 Components and Organization of Discrete-Event Simulation Model

1) System state สถานะของตัวแปรทั้งหลายที่จำเป็นต่อระบบในเวลาต่างๆ

2) Simulation clock ตัวแปรที่เป็นเก็บเวลาของการ Simulation

3) Event list รายการที่เก็บเวลาถัดไปเมื่อมี event เกิดขึ้น

4) Statistical counters ตัวแปรที่ใช้เก็บรายละเอียดสถิติเกี่ยวกับคุณสมบัติของระบบ

5) Initialization routine โปรแกรมย่อย ที่กำหนดค่าเริ่มต้นให้ simulation ที่เวลา

เท่ากับ 0

6) Timing routing โปรแกรมย่อยที่กำหนดเหตุการณ์ที่จะเกิดต่อไป โดยอ้างจาก Event List จากนั้นเมื่อถึงเวลาที่กำหนดไว้เหตุการณ์นั้นก็จะเริ่มทำงาน

7) Event routine โปรแกรมย่อยที่อัพเดต system state เมื่อมี event ชนิดต่างๆเกิดขึ้น

8) Library routine Library ที่ใช้กำหนดค่าให้โปรแกรมย่อย ใช้โดยการสุ่มจากกฎของความน่าจะเป็น เพื่อใช้กำหนดเส้นทางของ ระบบ Simulation

9) Report generator โปรแกรมย่อยที่คำนวณและแสดงผลค่าต่างๆที่ต้องการเมื่อการ Simulation จบสิ้นลง

10) Main program โปรแกรมย่อยที่ต้องการใช้ Timing routine ในการกำหนดเหตุการณ์ ต่อไปและต่อจากนั้นก็ส่งการควบคุมไปยัง event routine เพื่ออัพเดตข้อมูลที่เกี่ยวข้อง ของ system state Main program จะเป็นตัวตรวจสอบการสิ้นสุดของการ Simulation และร้องขอไปยัง Report generator ให้แสดง Report เมื่อสิ้นสุดการทำงาน

2.4.3 การอธิบายโดยการสมมติค่า (INTUITIVE EXPLANATION)

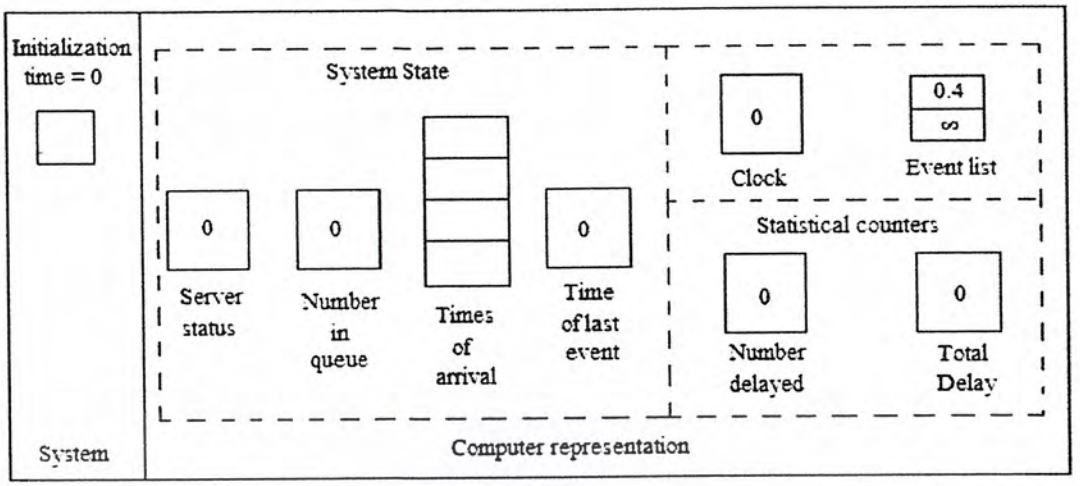
ก่อนอื่นจะเริ่มต้นการอธิบายเกี่ยวกับระบบจำลอง Single-server queuing โดยจะเป็นการจำลองภายในเครื่องคอมพิวเตอร์ที่เวลา $e_0 = 0$ และที่เวลา $e_1, e_1, \ldots, e_{13}$ ซึ่งลำดับเหตุการณ์ทั้ง 13 เหตุการณ์ที่ปรากฏขึ้นนั้นจำเป็นที่จะต้องอยู่ภายใต้การทำงานของจำนวนดีเลย์ในคิว (n = 6) โดยการอธิบายที่เหมาะสมนั้นเราจะสมมติ Interarrival และเวลาของการให้บริการดังแสดงในตัวอย่าง 2.1

ตัวอย่าง 2.1 ค่าสมมติ Interarrival และเวลาของการให้บริการ

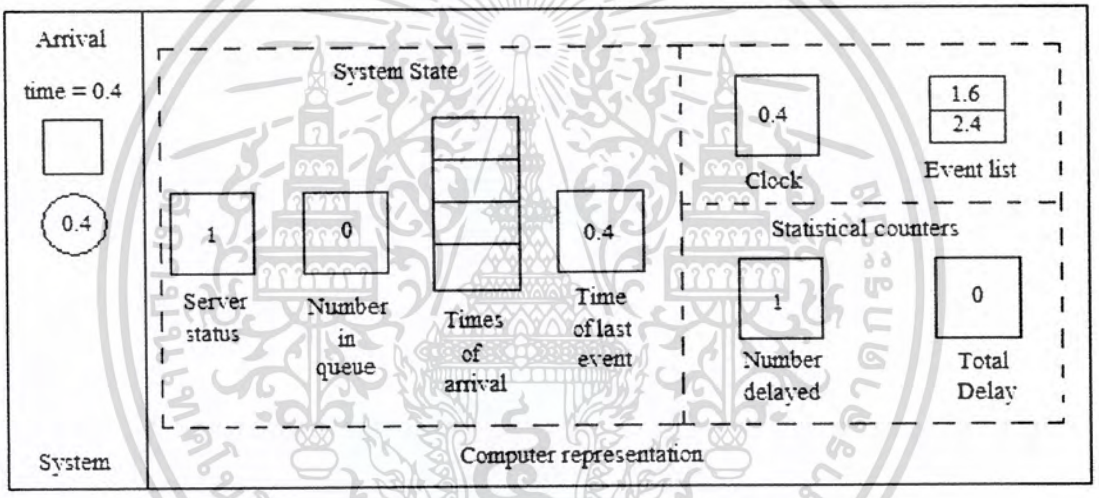
$A_1 = 0.4, A_2 = 1.2, A_3 = 0.5, A_4 = 1.7, A_5 = 0.2, A_6 = 1.6, A_7 = 0.2, A_8 = 1.4, A_9 = 1.9, \ldots$

$S_1 = 2.0, S_2 = 0.7, S_3 = 0.2, S_4 = 1.1, S_5 = 3.7, S_6 = 0.6, \ldots$

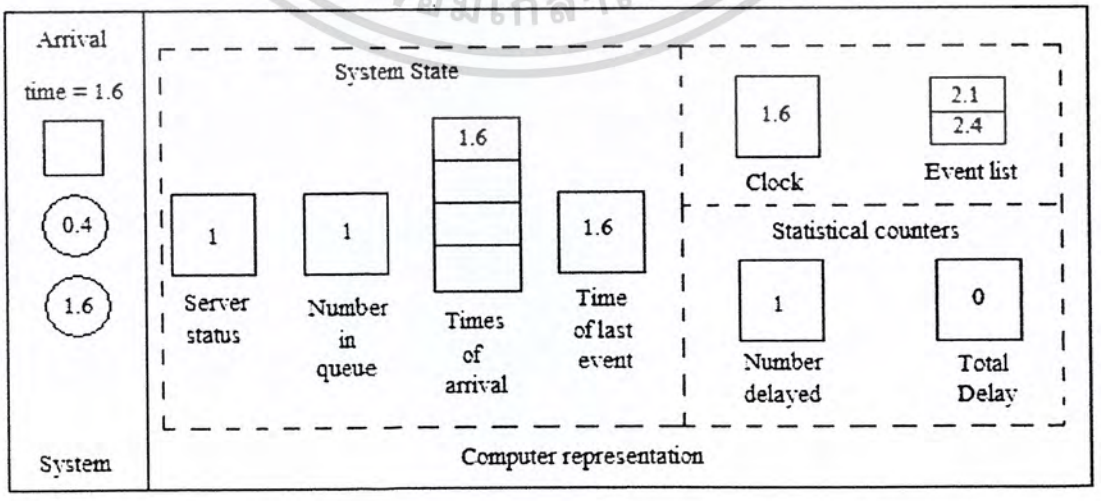
ดังนั้นระหว่างเวลา 0 และเวลาของงานที่เข้ามาในระบบชิ้นแรก คือ 0.4 และเวลาของงานที่เข้ามาในระบบชิ้นแรกและชิ้นที่สอง คือ 1.2 เป็นต้น โดยที่รูปที่ 2.11จะแสดงถึงระบบการจำลองในแต่ละช่วงเวลา $e_0 = 0$, $e_1 = 0.4 \ldots$ $e_{13} = 8.6$ ในช่อง System รูปสี่เหลี่ยมจะหมายถึงเซิร์ฟเวอร์และวงกลมจะหมายถึงผู้ใช้บริการ นอกจากนี้ตัวเลขภายในวงกลมจะแสดงถึงเวลาที่ผู้ใช้บริการรายนั้นๆ เข้ามาในระบบ



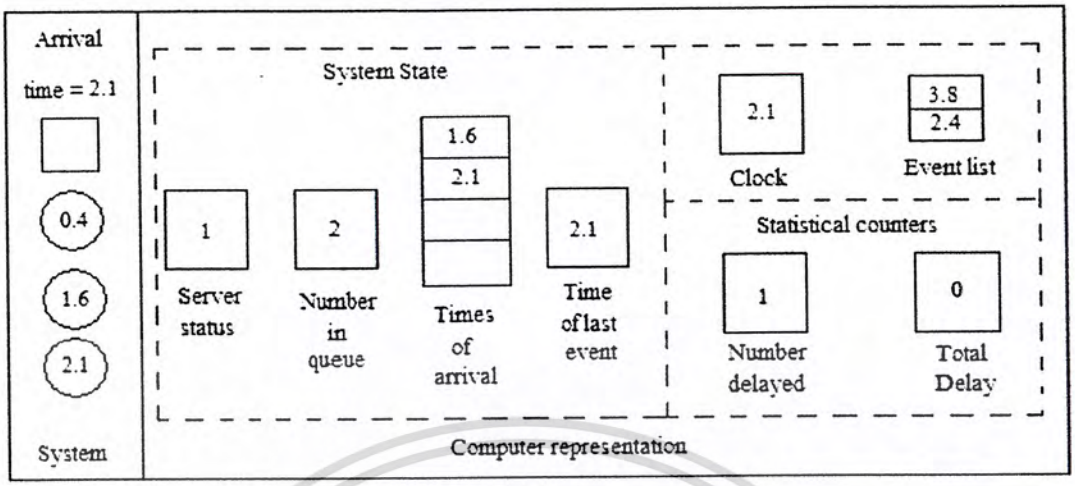
รูป 2.11 ระบบการจำลองที่เวลา 0



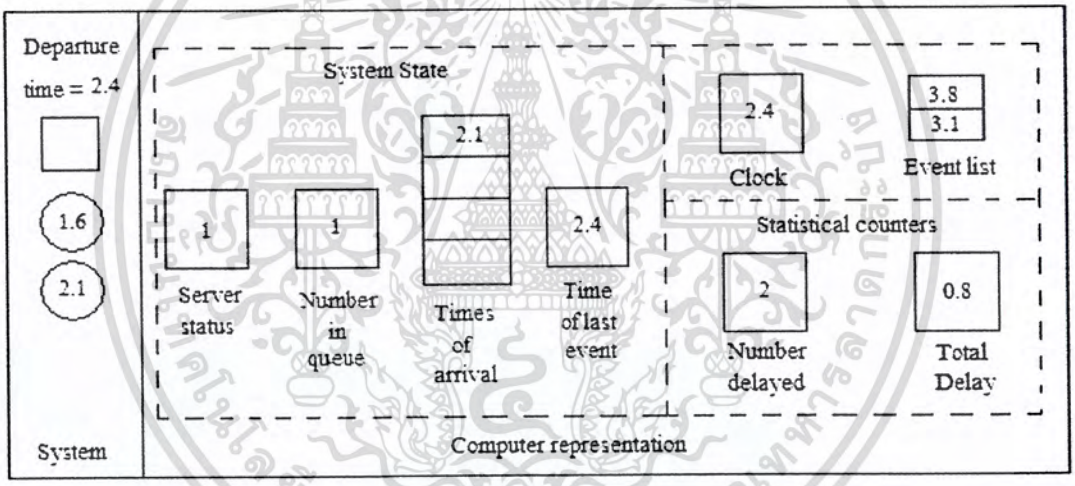
รูป 2.12 ระบบการจำลองที่เวลา 0.4



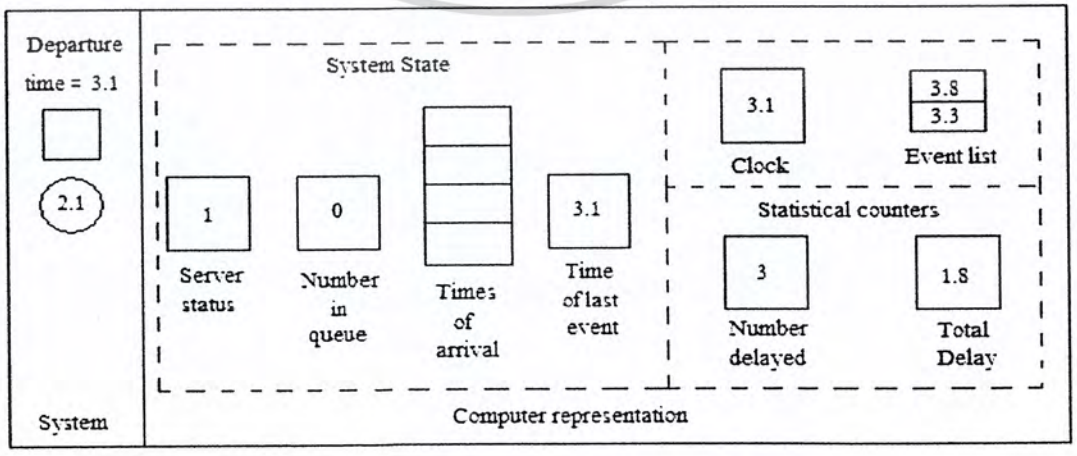
รูป 2.13 ระบบการจำลองที่เวลา 1.6



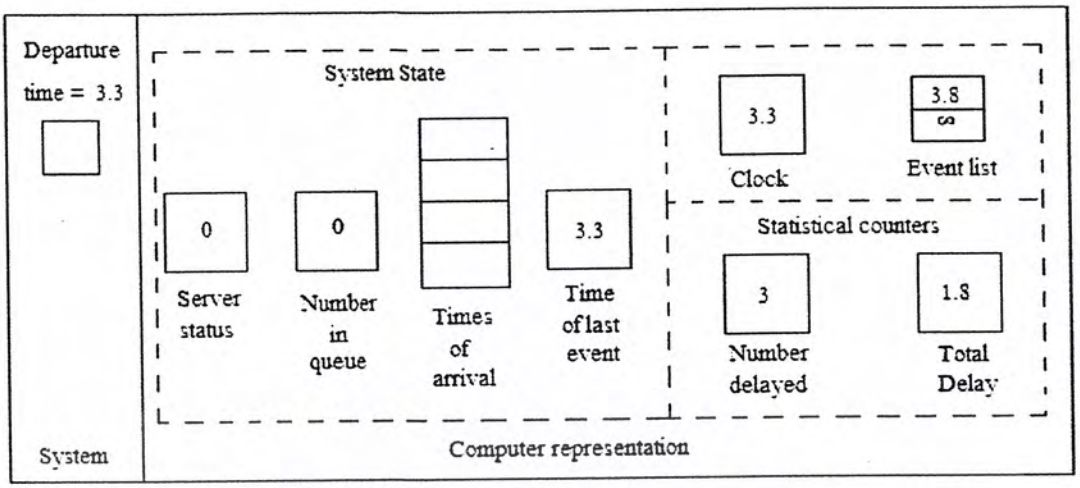
รูป 2.14 ระบบการจำลองที่เวลา 2.1



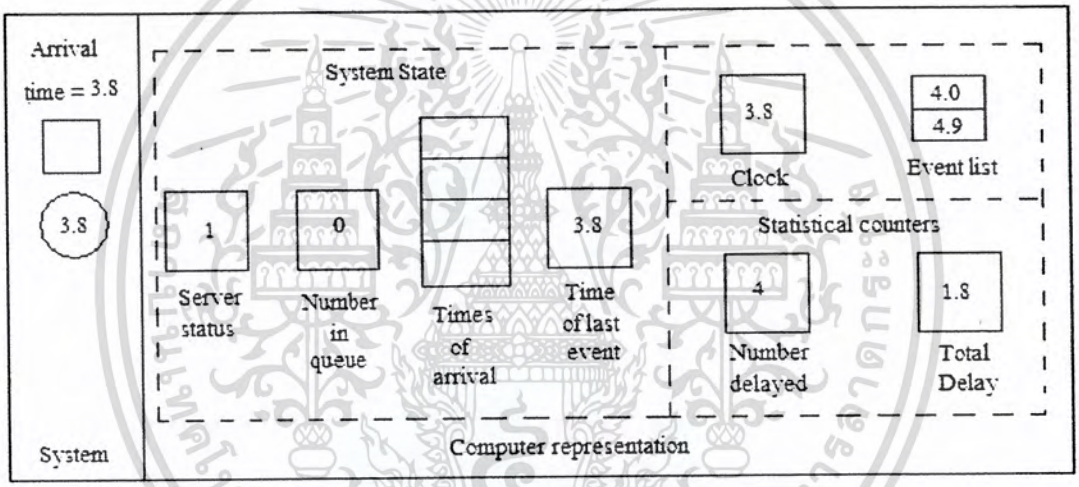
รูป 2.15 ระบบการจำลองที่เวลา 2.4



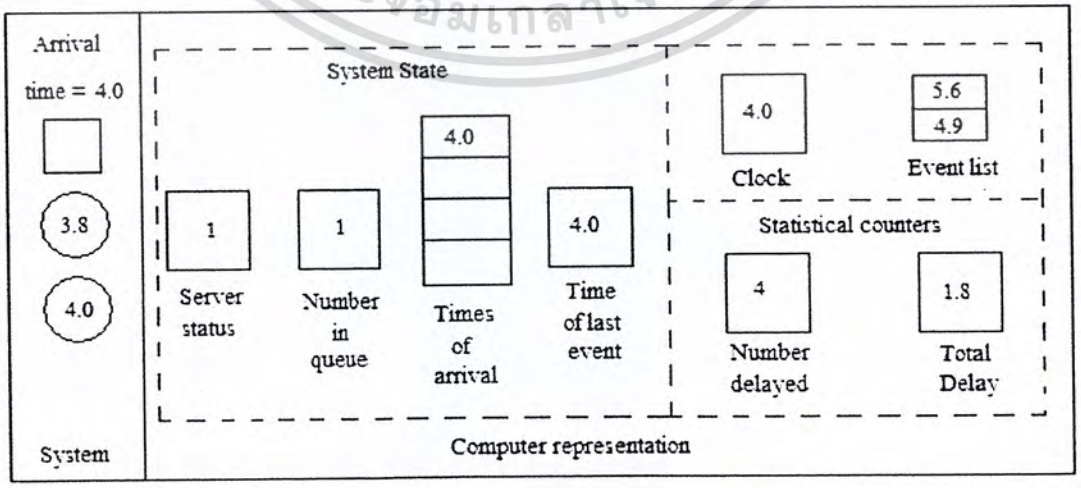
รูป 2.16 ระบบการจำลองที่เวลา 3.1



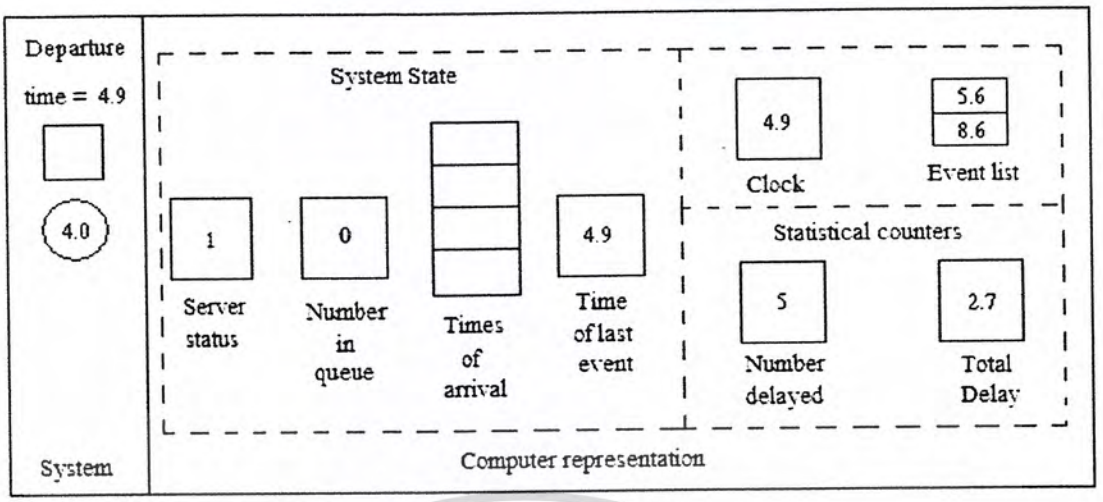
รูป 2.17 ระบบการจำลองที่เวลา 3.3



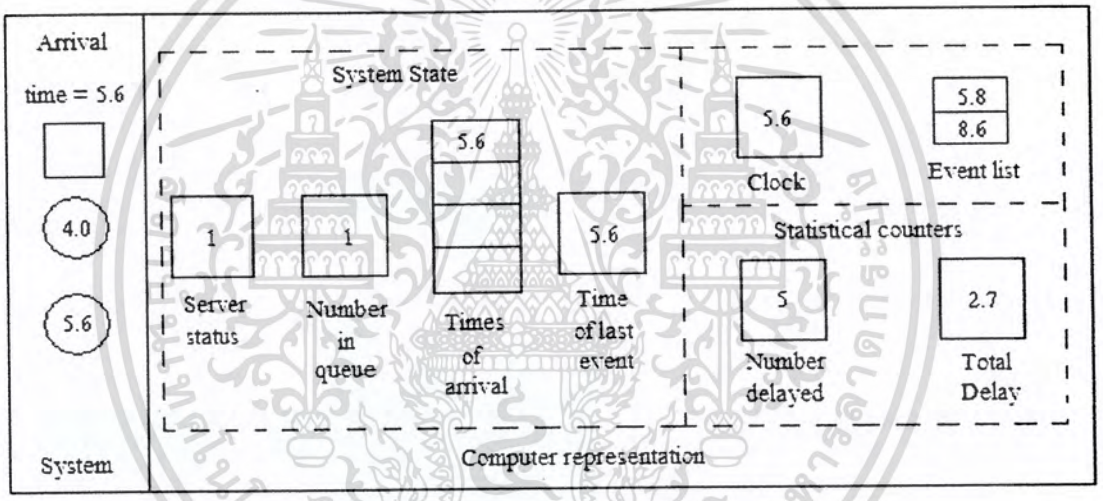
รูป 2.18 ระบบการจำลองที่เวลา 3.8



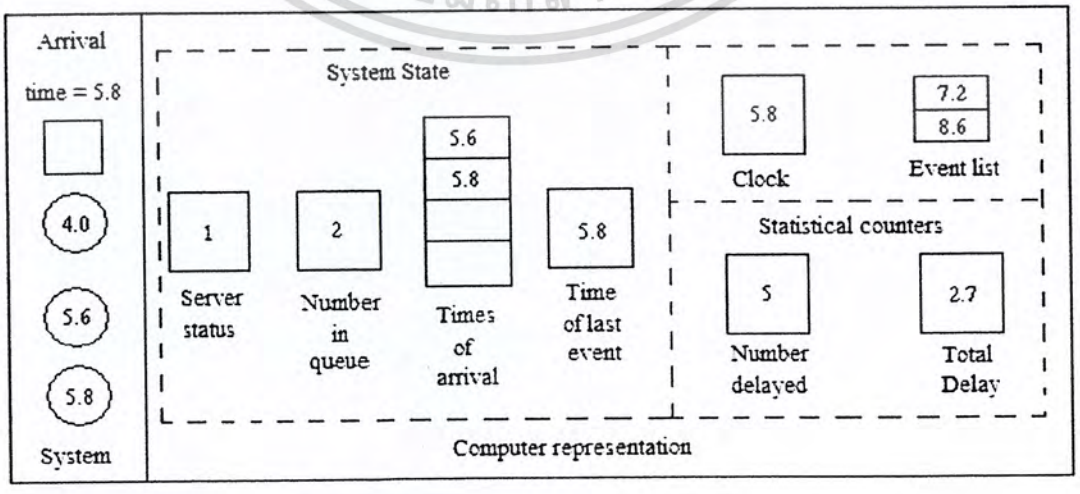
รูป 2.19 ระบบการจำลองที่เวลา 4.0



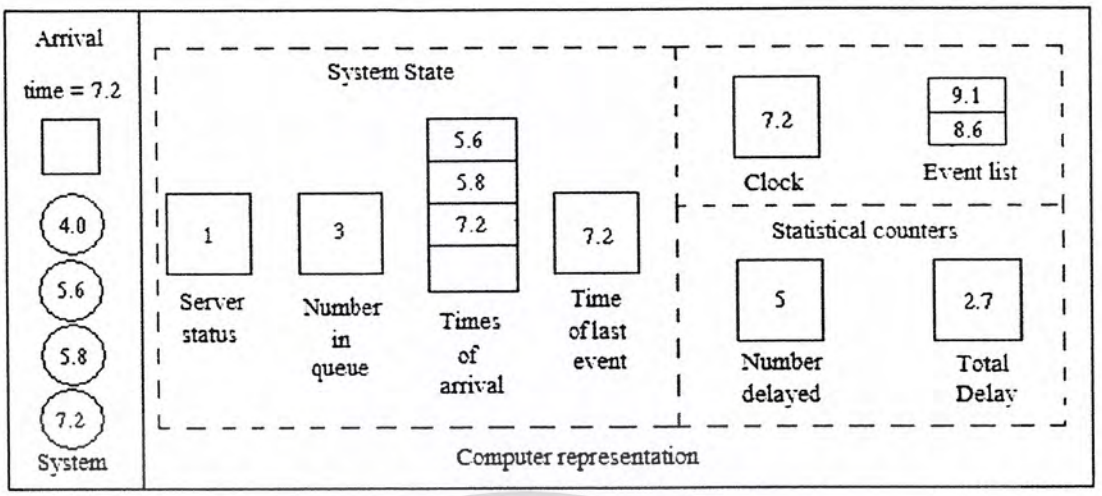
รูป 2.20 ระบบการจำลองที่เวลา 4.9



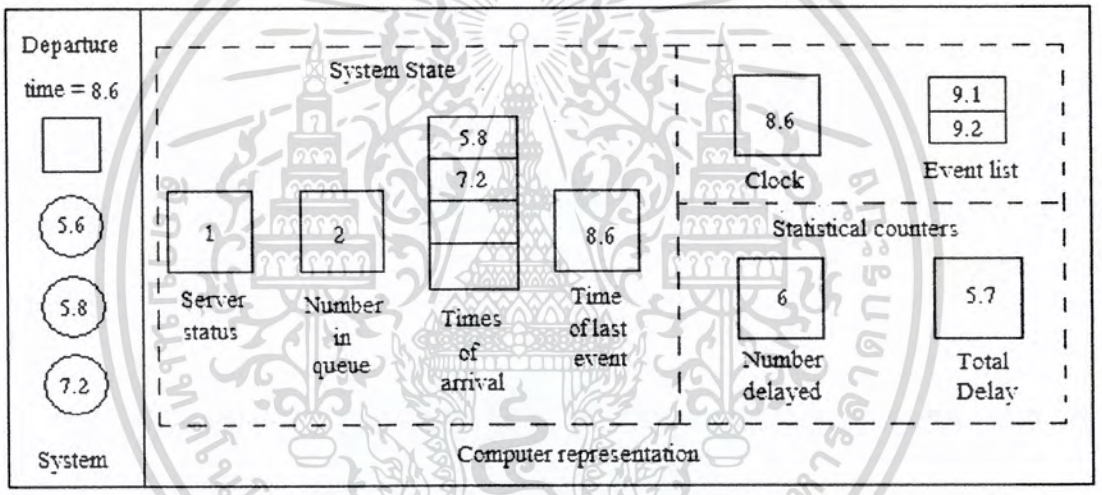
รูป 2.21 ระบบการจำลองที่เวลา 5.6



รูป 2.22 ระบบการจำลองที่เวลา 5.8



รูป 2.23 ระบบการจำลองที่เวลา 7.2



รูป 2.24 ระบบการจำลองที่เวลา 8.6

จากรูปข้างต้นสามารถอธิบายตัวแปรต่างๆ ได้ดังนี้

1) Server Status สถานะของเซิร์ฟเวอร์ (0=ว่าง, 1=ไม่ว่าง)

2) Number in queue จำนวนผู้ใช้บริการที่รออยู่ในคิว

3) Times of arrival อาร์เรย์ที่ใช้เก็บเวลาของการมาถึงของผู้ใช้บริการ

4) Time of last event เวลาของเหตุการณ์สุดท้ายที่เกิดขึ้น

5) Clock ตัวเก็บเวลาของการ Simulation

6) Event list รายการที่เก็บเวลาถัดไปเมื่อมีเหตุการณ์เกิดขึ้นเกิดขึ้น

7) Number Delay จำนวนดีเลย์ที่เกิดขึ้น

8) Total Delay เวลาดีเลย์รวม

ที่เวลา t = 0: Initialization การจำลองจะเริ่มต้นที่ Main Program โดยที่ Main program จะนำมาซึ่งชุดคำสั่งของการ Initialization ในตอนแรกระบบจะยังไม่มีผู้ใช้บริการและยังไม่เกิดเหตุการณ์ใดๆ ขึ้นกับเซิร์ฟเวอร์ซึ่งจะแสดงให้เห็นในรูป 2.11 โดยจะมีค่าต่างๆ ดังนี้

สถานะของเซิร์ฟเวอร์เท่ากับ 0 ในที่นี้ 0 จะเปรียบได้กับสถานะที่เซิร์ฟเวอร์ยังไม่ได้ทำงานอะไร ผู้ใช้จะสามารถใช้งานได้ แต่ถ้าเป็น 1 จะเปรียบได้กับการที่เซิร์ฟเวอร์ไม่ว่าง], จำนวนผู้ใช้ในคิวจะยังเท่ากับ 0 โดยที่คิวในที่นี้จะเก็บอยู่ในรูปแบบของอาร์เรย์แบบแถวเดียวซึ่งจะเก็บเวลาที่ผู้ใช้บริการเข้ามารับการบริการ, เวลาของเหตุการณ์สุดท้ายก็ยังคงเป็น 0, Simulation clock ก็จะถูกตั้งให้เป็น 0 ส่วน Even list คือ การแสดงเวลาของเหตุการณ์ถัดไปที่จะเกิดขึ้นของแต่ละเหตุการณ์ต่างๆ โดยการกำหนดค่าจะเป็นดังนี้ เวลาของผู้ที่มาถึงคนแรกคือ 0 รวมกับ A_1 เท่ากับ 0.4 (ซึ่งจะหมายถึง A) เนื่องจากตอนนี้ยังไม่มีผู้บริการเข้ามาขอรับบริการ ดังนั้นจึงไม่ยังไม่จำเป็นต้องกล่าวถึงเวลาขาออกของตัวถัดไป (ซึ่งจะหมายถึง D) ซึ่งในที่นี้เราทราบว่าผู้ใช้บริการคนแรกจะมาถึงที่เวลา 0.4 แต่อย่างไรก็ตามการโดยปกติแล้วการจำลองจะดูที่ Event list และจะเลือกตัวที่มีค่าน้อยที่สุดเพื่อใช้ในการตัดสินใจว่าเหตุการณ์ไหนจะเป็นเหตุการณ์ถัดไป โดยในที่นี้มีการกำหนดเวลาขาออกเป็น ∞ (หรือจะเลือกค่าที่มากๆ ในโปรแกรมคอมพิวเตอร์) สุดท้าย Statistical counter ทั้งสองตัวจะถูกกำหนดให้เท่ากับ 0 และเมื่อทำการกำหนดค่าให้ทุกตัวเสร็จเรียบร้อยแล้ว ตัวควบคุมจะส่งกลับไปที่ Main Program

ที่เวลา t = 0.4: Arrival of customer 1 ที่เวลา 0.4 Main Program จะส่งไปให้ชุดคำสั่งของ Arrival เพื่อดำเนินการกับการมาถึงของผู้ใช้บริการคนแรก ดังแสดงในรูป 2.12 เมื่อผู้ใช้บริการเข้ามาถึงจะพบว่าเซิร์ฟเวอร์พร้อมที่จะให้บริการเนื่องจากเซิร์ฟเวอร์ยังว่างอยู่ (สถานะเท่ากับ 0) ดังนั้นระบบก็จะเริ่มให้บริการทันทีดีเลย์ในคิวของ D_1 จะเท่ากับ 0 ส่วนสถานะเซิร์ฟเวอร์จะถูกกำหนดให้เป็น 1 เพื่อแสดงให้เห็นว่าในขณะนี้เซิร์ฟเวอร์ไม่ว่างที่จะให้บริการผู้ใช้คนอื่นในขณะนี้ แต่ในทางกลับกันคิวจะยังคงว่างอยู่ ส่วน Clock นั้นจะถูกเปลี่ยนเป็นเวลาปัจจุบันนั่นก็คือ 0.4 และ Even list จะถูกเปลี่ยนแปลงไปตามเหตุการณ์ที่มีผู้ใช้บริการเข้ามารับบริการ ซึ่งในที่นี้เหตุการณ์ถัดไป คือ A_2 เท่ากับ 1.2 ดังนั้นเวลาที่จะ ได้คือ 0.4 รวมกับ 1.2 เท่ากับ 1.6 และ เวลาขาออกถัดไป คือ S_1 เท่ากับ 2.0 หน่วยเวลาจากปัจจุบัน ดังนั้นเวลาที่ได้คือ 0.4 รวมกับ 2.0 เท่ากับ 2.4 ส่วนจำนวนดีเลย์จะถูกเพิ่มเป็น 1 (ถ้าเพิ่มไปจนถึง 6 การจำลองจะสิ้นสุดลง) D_1 เท่ากับ 0 จะถูกเพิ่มเข้าไปรวมกับดีเลย์ทั้งหมด (ซึ่งในขณะนี้ยังคงเป็น 0) สุดท้ายนี้เวลาของเหตุการณ์สุดท้ายจะนำมาซึ่งเวลาปัจจุบันคือ 0.4 จากนั้นตัวควบคุมจะส่งกลับไปที่ Main Program

ที่เวลา t = 1.6: Arrival of customer 2 จะแสดงให้เห็นดังรูป 2.13 เมื่อผู้ใช้บริการเข้ามาถึงจะพบว่าเซิร์ฟเวอร์ไม่ว่างให้บริการ ดังนั้นผู้ใช้บริการรายนี้จะต้องเข้าไปรอในคิว โดยที่เวลาที่เข้ามาถึงจะถูกเก็บไว้ในตำแหน่งแรกของอาร์เรย์จึงทำให้จำนวนงานที่รอในคิวเพิ่มเป็น 1 ส่วนเวลาของเหตุการณ์ถัดไปใน Even list จะเปลี่ยนเป็น A_3 เท่ากับ 0.5 หน่วยเวลาจากปัจจุบัน ดังนั้นเวลาที่

จะได้คือ 1.6 รวมกับ 0.5 เท่ากับ 2.1 ส่วนเวลาถัดไปของขาออกจะยังไม่เปลี่ยนแปลง โดยที่ยังคงเป็นเวลา 2.4 ของผู้ใช้บริการรายที่ 1 ซึ่งเป็นผู้ที่ยังคงรับบริการอยู่ในเวลานี้ ส่วนจำนวนดีเลย์และดีเลย์รวมก็ยังไม่เปลี่ยนแปลงเช่นกัน

ที่เวลา t = 2.1: Arrival of customer 3 จะแสดงให้เห็นในรูป 2.14 ซึ่งเซิร์ฟเวอร์ก็ยังคงไม่ว่างอยู่เช่นเคย และคิวจะถูกเพิ่มขึ้นมาอีก 1 โดยที่เวลาของผู้ใช้บริการเข้ามาถึงจะถูกเก็บไว้ในตำแหน่งที่สองของอาร์เรย์ Next arrival จะถูกเปลี่ยนแปลงไปเป็น $t + A_4$ เท่ากับ 2.1 รวมกับ 1.7 นั้นก็คือ 3.8 แต่ Next departure ก็ยังคงไม่เปลี่ยนแปลง เนื่องจากยังต้องรอให้ระบบบริการผู้ใช้บริการคนที่ 1 ให้เรียบร้อยก่อน ส่วน Delay counters ยังมีค่าเท่าเดิม ไม่เปลี่ยนแปลง

ที่เวลา t = 2.4: Departure of customer 1 จะแสดงให้เห็นในรูป 2.15 ในขณะนี้ Main Program จะก่อให้เกิดชุดคำสั่งของ departure ดังรูป 2.10 และเซิร์ฟเวอร์ก็ยังคงไม่ว่างอยู่เช่นเคย เนื่องจากผู้ใช้บริการคนที่ 2 ได้ออกจากตำแหน่งแรกของคิวไปและเข้าไปรับการบริการ ดังนั้นจึงทำให้คิวลดลง 1 และเวลาผู้ใช้บริการที่เข้ามาถึงจะถูกเลื่อนขึ้นไป 1 ตำแหน่งซึ่งจะแสดงให้เห็นว่าผู้ใช้บริการคนที่ 3 จะถูกย้ายไปที่ตำแหน่งแรกแล้ว ในขณะนี้ผู้ใช้บริการคนที่ 2 จะเข้ามารับการบริการโดยจะถูกกำหนดหน่วยของเวลาที่ S_2 เท่ากับ 0.7 ดังนั้นเวลาของ Departure ถัดไปใน event list (ผู้ใช้บริการคนที่ 2) จะถูกเปลี่ยนเป็น 3.1 (2.4+0.7) ในขณะที่เวลาของ Arrival ถัดไป (ผู้ใช้บริการคนที่ 4) จะไม่เปลี่ยนแปลง

ที่เวลา t = 3.1: Departure of customer 2 จะแสดงให้เห็นในรูป 2.16 การเปลี่ยนแปลงที่ Departure นี้ จะเหมือนกับที่ Departure ของผู้ใช้บริการคนที่ 1 ณ เวลา 2.4 หลังจากที่เหตุการณ์นี้ดำเนินการเสร็จเรียบร้อยแล้ว คิวจะว่างอีกครั้งหนึ่งแต่เซิร์ฟเวอร์จะยังคงไม่ว่างอยู่เช่นเคย

ที่เวลา t = 3.3: Departure of customer 3 จะแสดงให้เห็นในรูป 2.17 การเปลี่ยนแปลงจะเหมือน Departure ทั้ง 2 ตัวที่ผ่านมา ซึ่งในขณะนี้คิวจะว่างอยู่ ส่วนเซิร์ฟเวอร์จะยังไม่ทำงานอะไร ดังนั้นจึงต้องกำหนดเวลาของ departure ถัดไปใน event list เป็น ∞ อีกครั้งหนึ่งเหมือนในตอนแรก

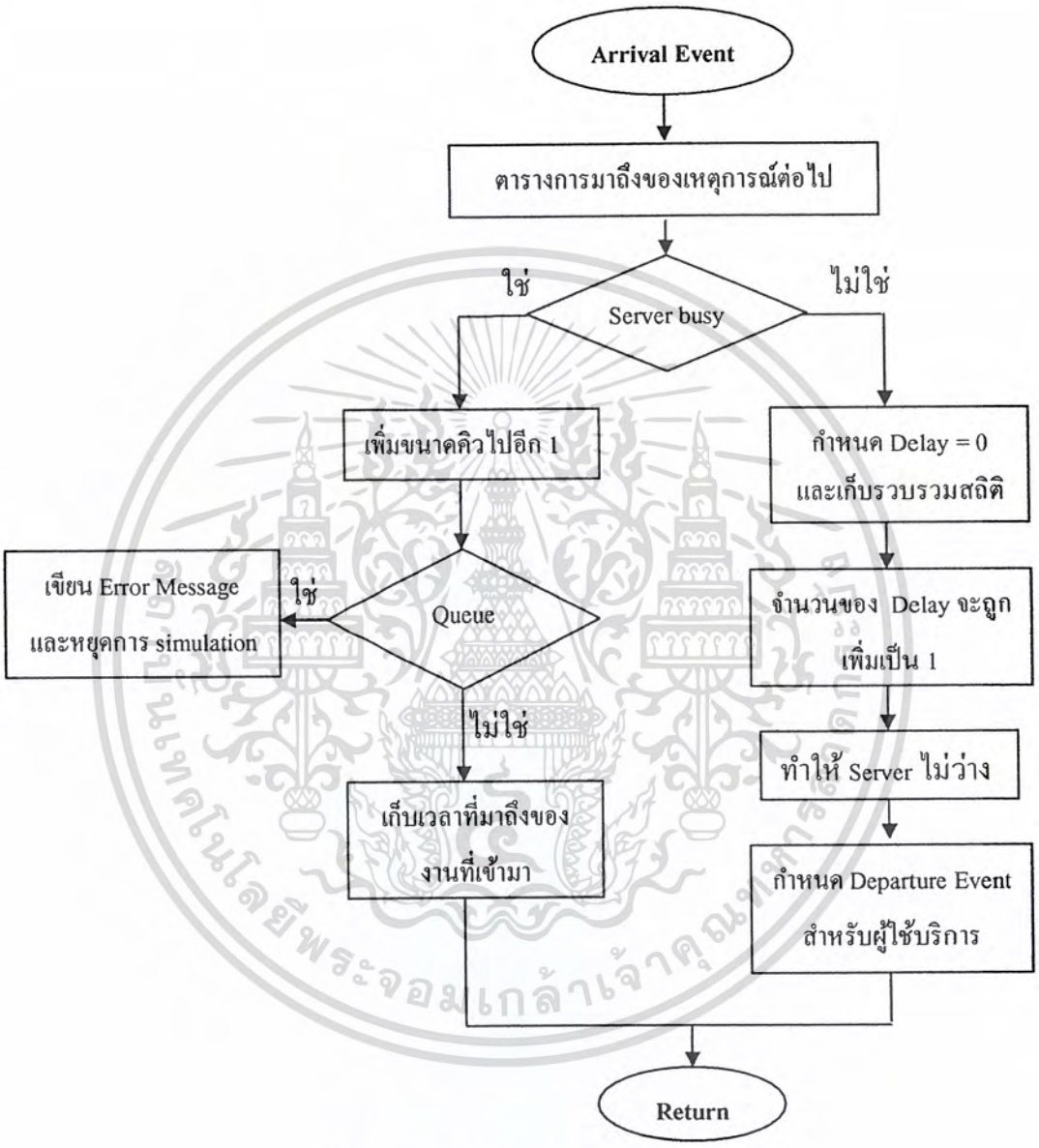
ที่เวลา t = 3.8: Arrival of customer 4 จะแสดงให้เห็นในรูป 2.18 เนื่องจากผู้ใช้บริการรายนี้เข้ามาถึงจะพบว่าเซิร์ฟเวอร์ยังคงว่างอยู่ ผู้ใช้บริการรายนี้จะมีดีเลย์เป็น 0 และจะเข้าสู่การรับบริการ ด้วยเหตุนี้การเปลี่ยนแปลงที่เวลานี้จะมีค่าคล้ายกับ arrival ของผู้ใช้บริการรายแรกที่เวลา 0.4

2.4.4 PROGRAM ORGANIZATION AND LOGIC

2.4.4.1 Arrival Event

เป็น Logic ที่เป็น Event routine โดยเวลาในการมาถึงจะดูจาก Event List จากนั้นก็จะทำการตรวจสอบเซิร์ฟเวอร์ว่าขณะนั้นพร้อมให้ใช้งานหรือไม่ ถ้าไม่ว่างจำนวนของคิวจะเพิ่มไป 1 จากนั้นก็จะตรวจสอบอีกว่าถ้าคิวเต็มให้ส่ง Error Message ไปบอกและทำการ Drop Event นั้นทิ้งไป แต่ถ้าคิวยังไม่เต็มให้ทำการรอเวลาจนกว่าเซิร์ฟเวอร์จะว่างและเข้าใช้งาน

เซิร์ฟเวอร์ ในกรณีเซิร์ฟเวอร์ว่างพร้อมใช้งานแล้วเมื่อมีงานเข้ามาในเซิร์ฟเวอร์ค่าดีเลย์จะถูกตั้งให้เท่ากับศูนย์และทำการเก็บค่าสถิติของงานที่เข้ามา จากนั้นเซิร์ฟเวอร์ก็จะไม่ว่างจนกว่างานนั้นจะทำงานเสร็จสิ้น เพื่อความเข้าใจจะสามารถดูได้จากผังการทำงานรูป 2.25

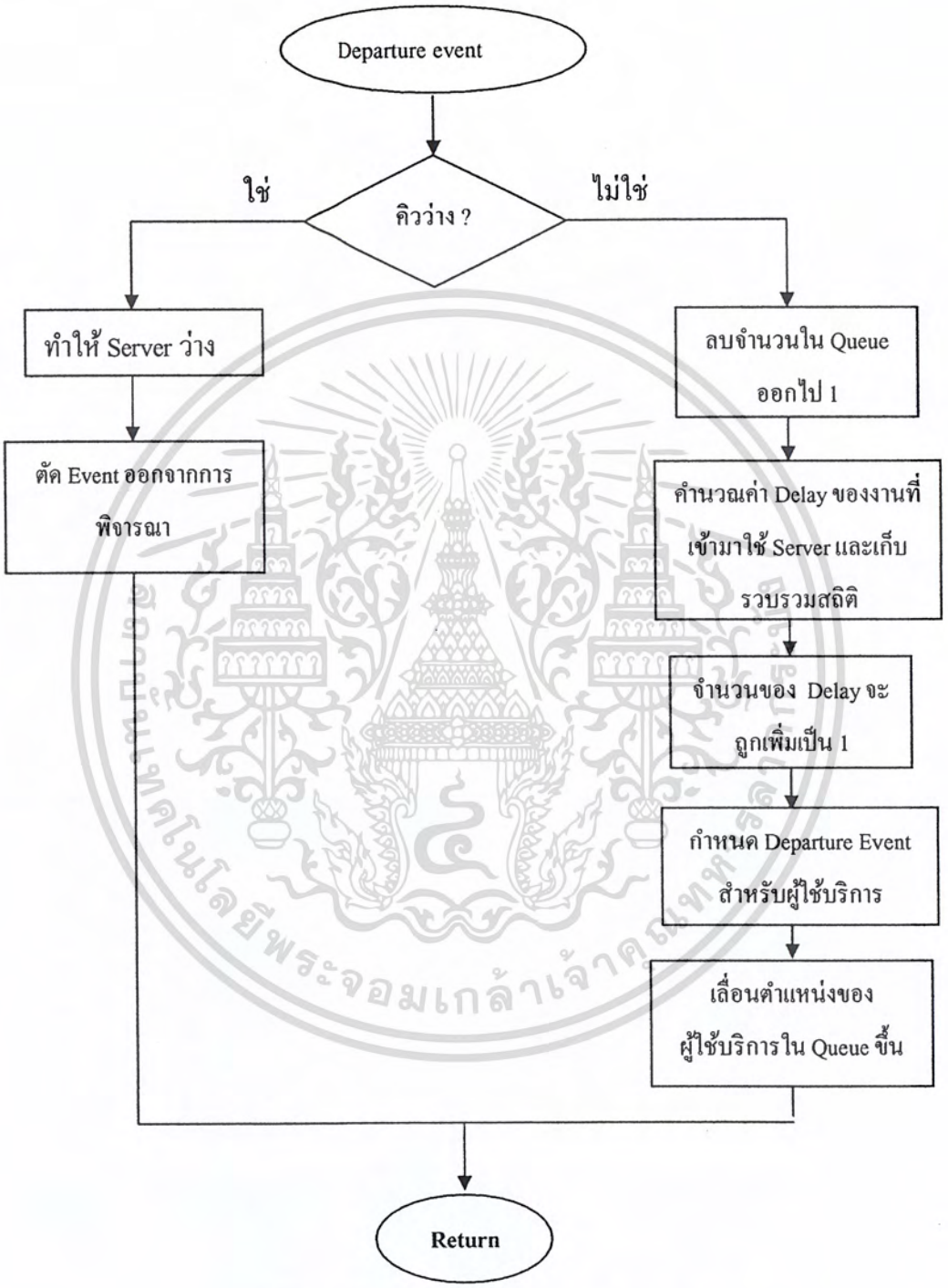


รูป 2.25 ผังการทำงานของ Arrival Event

2.4.4.2 Departure Event

เป็น Logic ที่เป็น Event routine เมื่อมีงานที่ใช้ เซิร์ฟเวอร์เสร็จสิ้นแล้ว ระบบจะตรวจสอบว่าตอนนั้นมีงานอื่นๆ ที่ต้องการใช้เซิร์ฟเวอร์อีกหรือไม่ (รออยู่ในคิว) ถ้าไม่มีก็จะทำให้สถานะของเซิร์ฟเวอร์ว่างและระบบจะตัดงานนั้นออกจากการพิจารณา แต่ถ้ายังมีงานอื่นๆ อยู่ในคิวระบบก็จะให้งานนั้นเข้ามาใช้งานเซิร์ฟเวอร์จากนั้นก็จะลบจำนวนคิวออกไป 1 จากนั้น

ระบบจะคำนวณค่าดีเลย์ใหม่และเก็บรวบรวมสถิติ เพื่อความเข้าใจจะสามารถดูได้จากผังการทำงาน รูป 2.26



รูป 2.26 ผังการทำงานของ Departure Event

2.5 C PROGRAM

ในส่วนนี้จะเป็นการแสดงถึงโปรแกรมที่เขียนด้วยภาษาซีสำหรับระบบจำลองการทำงานของ M/M/1 ซึ่งจะแสดงดังต่อไปนี้

โปรแกรม 2.1 โค๊ดภาษาซีที่ใช้ในการประกาศค่าต่างๆ ที่ใช้

```
#include <stdio.h>
#include <math.h>
#include "lcgrand.h"/*Header file for random-number
generator. */
#define Q_LIMIT 100      /*Limit on queue length. */
#define BUSY    1/*Mnemonics for server's being busy */
#define IDLE    0   /* and idle. */
int  next_event_type, num_custs_delayed,
     num_delays_required, num_events, num_in_q,
     server_status;
float     area_num_in_q, area_server_status,
     mean_interarrival, mean_service, sim_time,
     time_arrival[Q_LIMIT + 1], time_last_event,
     time_next_event[3], total_of_delays;
FILE *infile, *outfile;

void initialize(void);
void timing(void);
void arrive(void);
void depart(void);
void report(void);
void update_time_avg_stats(void);
float expon(float mean);
```

ส่วนนี้ จะเป็นส่วนที่ประกาศ Header, variable และฟังก์ชันต่างๆ ที่ใช้ในโปรแกรมSimulation แบบ M/M/1 ซึ่งฟังก์ชันที่ใช้เป็นฟังก์ชันพื้นฐานที่ใช้ในการทำ Simulation ซึ่งเราจะใช้ฟังก์ชันเหล่านี้ เป็นพื้นฐานในการเขียนโปรแกรม Simulation

โปรแกรม 2.2 โค๊ดภาษาซีสำหรับฟังก์ชันหลัก

```
main()    /* Main function */
{
     /* Open input and output files. */
     infile    = fopen("mm1.in", "r");
     outfile   = fopen("mm1.out", "w");
/* Specify the number of events for the timing function.
*/   num_events = 2;
     /* Read input parameters. */
```

โปรแกรม 2.2 โค๊ดภาษาซีสำหรับฟังก์ชันหลัก (ต่อ)

```
fscanf(infile, "%f %f %d", &mean_interarrival,
     &mean_service,&num_delays_required);
     /* Write report heading and input parameters. */
     fprintf(outfile,"Single-server queueing
     system\n\n");
     fprintf(outfile, "Mean interarrival time%11.3f
     minutes\n\n" ,mean_interarrival);
     fprintf(outfile, "Mean service time%16.3f
     minutes\n\n", mean_service);
     fprintf(outfile, "Number of customers%d\n\n",
     num_delays_required);
     /* Initialize the simulation. */
     initialize();
     /* Run the simulation while more delays are still
     needed. */
     while (num_custs_delayed < num_delays_required)
     {
     /* Determine the next event. */
          timing();
     /* Update time-average statistical accumulators. */
          update_time_avg_stats();
     /* Invoke the appropriate event function. */
          switch (next_event_type){
               case 1:
                    arrive();
                    break;
               case 2:
                    depart();
                    break;
                  }
            }
     /* Invoke the report generator and end the
     simulation. */
     report();
     fclose(infile);
     fclose(outfile);
     return 0;
}
```

ภายในฟังก์ชันหลักนี้ จะมีการเปิดไฟล์เพื่ออ่าน 1 ไฟล์ เพื่อกำหนดค่าให้กับตัวแปรโดยนำค่าจากไฟล์ โดยกำหนดค่าให้ทั้งหมด 3 ตัวคือ mean_interarrival ,mean_service, num_delays_required ซึ่งค่า mean_interarrival กับ mean_service จะใช้ในการหาค่าเพิ่มเติมในฟังก์ชัน expo เพื่อกำหนดค่าให้กับ time_next_event[i] แต่ละตัว ส่วนค่า num_delays_required จะเป็นตัวกำหนดว่า ถ้าจำนวนการใช้งานของผู้ใช้งานในระบบเกินค่า num_delays_required ก็จะทำ

การจบลูปการทำงาน แล้วไปทำการเขียนรายงานเพื่อจบการทำ Simulation ซึ่งในตอนเริ่มต้นของฟังก์ชันหลักก็จะทำการเรียกฟังก์ชัน initialize() ซึ่งฟังก์ชันนี้ จะทำการกำหนดค่าเริ่มต้นให้กับตัวแปรที่ใช้ในโปรแกรม

ส่วนภายในลูป While นั้นจะมีเงื่อนไขคือ เมื่อมีจำนวนผู้ใช้งาน น้อยกว่า จำนวนดีเลย์ที่กำหนดไว้ (อ่านจากในไฟล์) ก็จะยังคงทำงานในลูป ซึ่งภายในลูปจะมีการเรียกฟังก์ชัน timing() แล้วทำการเรียกฟังก์ชัน update_time_avg_stats() และใช้ Switch ในการเลือกว่าผู้ใช้งานเข้ามาเป็นแบบ arrive() หรือ depart() ซึ่งถ้าหลุดลูปนี้ไป ก็จะทำการเรียกฟังก์ชัน report() แล้วจบการทำ Simulation

โปรแกรม 2.3 โค๊ดภาษาซีสำหรับฟังก์ชัน Initialize

```
void initialize(void)     /* Initialization function. */
{
    /* Initialize the simulation clock. */
    sim_time = 0.0;
    /* Initialize the state variables. */
    server_status = IDLE;
    num_in_q = 0;
    time_last_event = 0.0;
    /* Initialize the statistical counters. */
    num_custs_delayed = 0;
    total_of_delays   = 0.0;
    area_num_in_q  = 0.0;
    area_server_status = 0.0;
    time_next_event[1]=sim_time+expon(mean_interarrival);
    time_next_event[2] = 1.0e+30;
}
```

ส่วนนี้เป็นฟังก์ชัน initialize () ซึ่งใช้กำหนดค่าเริ่มต้นให้กับตัวแปรต่างๆที่จำเป็นในการทำ Simulation ซึ่งจะมีการกำหนดตัวแปร

1) sim_time จะเริ่มที่เวลา 0.0 แล้วก็จะทำการเลื่อนเวลาไปตามเหตุการณ์ที่เข้ามาทำงาน

2) server_status ในการเริ่มทำงานตอนแรกเราจะกำหนดให้เซิร์ฟเวอร์ว่างพร้อมใช้งานได้ คือ สถานะ IDLE ซึ่งถ้า เซิร์ฟเวอร์ไม่สามารถใช้งานได้ก็จะอยู่ในสถานะ BUSY

3) num_in_q เป็นตัวนับจำนวนว่ามีจำนวนผู้ใช้งานกี่คนที่รออยู่ในคิว

4) time_last_event เป็นตัวชี้ว่าเหตุการณ์ที่เกิดขึ้นล่าสุดอยู่ที่เวลาเท่าไหร่

5) num_custs_delayed ตัวแปรที่ใช้นับจำนวนผู้ใช้งานซึ่งได้กล่าวไปใน main ฟังก์ชันด้านบน

6) total_of_delays ตัวนับดีเลย์ทั้งหมดที่เกิดขึ้นภายในระบบ

7) time_next_event[1] กำหนดค่าให้ค่า time_next_event ของ arrive

8) time_next_event[2] กำหนดค่าให้ค่า time_next_event ของ depart

โปรแกรม 2.4 โค๊ดภาษาซีสำหรับฟังก์ชัน Timing

```
void timing(void)
/* Timing function. */
{
     int  i;
     float     min_time_next_event = 1.0e+29;
     next_event_type = 0;
 /* Determine the event type of the next event to occur.
*/
     for(i = 1; i <= num_events; ++i)
          if(time_next_event[i] < min_time_next_event){
          min_time_next_event   =  time_next_event[i];
          next_event_type  = i;
     }
     if(next_event_type == 0){
 /* The event list is empty, so stop the simulation. */
     fprintf(outfile,"\n Event list empty at time %f",
     sim_time);
          exit(1);
     }
  /* The event list is not empty, so advance the
     simulation clock. */
     sim_time = min_time_next_event;
}
```

ส่วนกำหนดค่าเวลา และ next_event_type ว่าจะเป็นการทำงานแบบ arrive หรือ depart ซึ่งมีการกำหนดค่า min_time_next_event ไว้ในกรณีที่ไม่มีเหตุการณ์ที่ไวกว่าค่านี้ ก็จะขยับค่า sim_time ไป ซึ่งในตอนเริ่มจะกำหนดไว้ว่า next_event_type เป็น 0 ไว้ก่อน

ภายในลูป for จะทำการเทียบค่า time_next_event[i] โดย i จะดูจากทุกตัวที่เป็นเหตุการณ์ที่สามารถเกิดขึ้นได้ (ในที่นี้มี arrive กับ depart ดังนั้น num_events จึงมีค่าเป็น 2 ดังใน main ฟังก์ชัน) ซึ่งถ้าค่าตัวไหนน้อยที่สุด ก็จะเลือกให้ next_event_type ตัวนั้นทำงานก่อน และในกรณีที่ไม่มีเหตุการณ์ใดที่ทำงานไวกว่าค่า min_time_next_event ที่กำหนดไว้เริ่มต้นค่า next_event_type จะยังคงเป็น 0 ซึ่งก็คือภายใน Event list ไม่มีเหตุการณ์รอการทำงานอยู่ แล้วเมื่อทำการเลือก next_event_type แล้ว ตอนสุดท้ายก็จะทำการเลื่อน sim_time ไปอยู่ที่ตำแหน่ง min_time_next_event ที่ทำการเปรียบเทียบมา

โปรแกรม 2.5 โค๊ดภาษาซีสำหรับฟังก์ชัน Arrive

```
void arrive(void)   /*Arrival event function */
{
     float     delay;
   /* Schedule next arrival. */
time_next_event[1] = sim_time + expon(mean_interarrival);
   /* Check to see whether server is busy. */
     if(server_status == BUSY)
     {
   /* Server is busy, so increment number of customers in
queue. */
          ++num_in_q;
   /* Check to see whether an overflow condition exists.
*/
          if(num_in_q > Q_LIMIT) {
   /* The queue has overflowed, so stop the simulation.
*/
          fprintf(outfile, "\nOverflow of the array time
          arrival at");
          fprintf(outfile, "time %f", sim_time);
          exit(2);
       }
   /* There is still room in the queue, so store the time
of arrival of the arriving customer at the (new) end of
time_arrival. */
          time_arrival[num_in_q] = sim_time;
      }
      else{
   /* Server is idle, so arriving customer has a delay of
zero. */
          delay  = 0.0;
          total_of_delays +=  delay;
   /* Increment the number of customers delays, and make
server busy. */
           ++num_custs_delayed;
          server_status  = BUSY;
   /* Schedule a departure (services completion). */
          time_next_event[2]  =  sim_time +
expon(mean_service);
      }
}
```

ส่วนนี้จะเป็นส่วนที่รับผู้ใช้เข้ามาทำงาน ซึ่งในตอนเริ่มต้นก็จะกำหนดค่าให้กับ time_next_event[1] ใหม่เพราะเมื่อ sim_time มีการเปลี่ยนค่าตัวนี้ ก็จะเปลี่ยนด้วย และจะทำการเช็คว่า server_status BUSY หรือ IDLE

1) server_status เป็น BUSY จะทำการเพิ่มจำนวนผู้ใช้ที่รอในคิว และถ้าเกิดว่าคิวที่รอเกินกว่าลิมิตของคิวที่กำหนดไว้ ก็จะทำการเขียนแจ้งไว้ในไฟล์ว่ามีการ overflow ที่เวลาใดของการทำ simulation และกำหนดค่า time_arrival[num_in_q] ไว้ด้วย
2) server_status เป็น IDLE ให้ค่าดีเลย์ที่เข้ามาเป็น 0.0 แล้วทำการคำนวณค่า total_of_delays และเพิ่มจำนวนผู้ใช้งาน และให้ server_status เป็น BUSY เนื่อง จากมีผู้ใช้งานอยู่ แล้วทำการกำหนด time_next_event[2] ด้วย

โปรแกรม 2.6 โค๊ดภาษาซีสำหรับฟังก์ชัน Depart

```
void depart(void)  /* Departure event fuction.*/
{
     int  i;
     float  delay;
     /* Check to see whether the queue us empty. */
     if(num_in_q == 0){
          server_status = IDLE;
          time_next_event[2] = 1.0e+30;
     }
     else{
          --num_in_q;
          delay =  sim_time - time_arrival[1];
          total_of_delays     +=   delay;
          ++num_custs_delayed;
          time_next_event[2]   =  sim_time +
     expon(mean_service);
          for(i = 1; i <= num_in_q ; ++i)
               time_arrival[i]     =  time_arrival[i+1];
     }
}
```

ส่วนนี้เป็นส่วนของการ Departure ซึ่งจะทำการเช็คว่าถ้าในคิวเป็น 0 หรือไม่

1) ถ้าคิวเป็น 0 ให้ปรับ server_status เป็น IDLE พร้อมใช้งาน และตั้งค่าให้กับ time_next_event[2] ด้วย
2) ถ้าคิวไม่เป็น 0 จะไปทำการลบ num_in_q ไป 1 และกำหนดค่าให้กับดีเลย์และคำนวณค่า total_of_delays เพิ่มตัวแปร num_custs_delayed ไป 1 และทำการกำหนดค่า time_next_event[2] ใหม่อีกครั้งเนื่องจาก sim_time มีการเปลี่ยนแปลง และสุดท้ายคือกำหนดค่า time_arrival[i] ให้กับแต่ละตัวในคิว

โปรแกรม 2.7 โค๊ดภาษาซีสำหรับฟังก์ชัน Update_time_avg_stats

```
void update_time_avg_stats(void) /* update area
accumulators for time-average statistic. */
{
     float     time_since_last_event;
     time_since_last_event =     sim_time -
time_last_event;
     time_last_event         =     sim_time;
     area_num_in_q       +=  num_in_q *
time_since_last_event;
     area_server_status  +=  server_status *
time_since_last_event;
}
```

ส่วนนี้เป็นการอัพเดทค่าต่างๆ ก่อนทำการเข้า Switch case ในฟังก์ชันหลัก ซึ่งจะมีการกำหนดค่า time_since_last_event , time_last_event , area_num_in_q และ area_server_status

โปรแกรม 2.8 โค๊ดภาษาซีสำหรับฟังก์ชันเอ็กซ์โปแนนเชียล

```
float     expo(float mean) /* Exponential variate
generation function. */
{
          return     -mean * log(lcgrand(1));
}
```

ส่วนนี้เป็นส่วนที่ใช้คำนวณค่า mean_interarrival กับค่า mean_service ที่ใช้ในการคำนวณค่าต่างๆ ที่ใช้ค่า log ซึ่งฟังก์ชันนี้ต้องใส่ #include <math.h> และ #include "lcgrand.h" ไว้บน header ด้วยจึงจะสามารถทำงานได้

2.6 ทฤษฎีทีซีพี

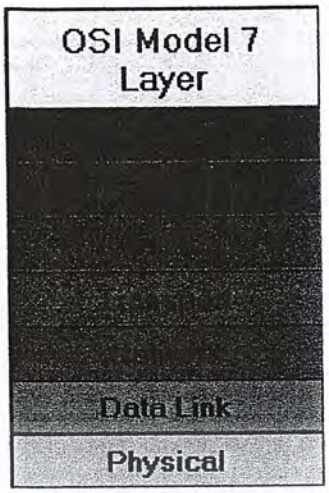
Transmission Control Protocol/Internet Protocol (TCP/IP) เป็นชุดโพรโทคอลที่ใช้ในอินเตอร์เน็ต ซึ่งเป็นที่นิยมใช้กันแพร่หลายมากในปัจจุบัน ในที่นี้จะได้อธิบายถึงพื้นฐานของโพรโทคอลชุดนี้ไว้บ้าง จุดเริ่มต้นของชุดโพรโทคอลนี้มาจากกลุ่มงานวิจัยของ Defense Advanced Research Project Agency (DARPA) ซึ่งได้ตั้งเครือข่าย ARPANET แนวความคิดในการทำ internetwork ของ ARPANET นี้มีมาก่อนที่จะเกิดโอเอสไอโมเดล แต่เนื่องจากมาตรฐานของไอเอสโอ ก็มีบทบาทมากในการอ้างอิงทางด้าน computer network จึงควรพิจารณาเปรียบเทียบชุดทีซีพีไอพีนี้กับ โอเอสไอ โมเดล ด้วยตามแนวความคิดของโอเอสไอนั้นอาจมองเป็นภาพรวมได้ว่าในระดับชั้นที่ 1-3 เป็น local procedures และในระดับชั้นที่ 4-7 เป็น end-to-end procedures ถ้าสังเกตจะเห็นว่า ในระดับชั้นที่ 2 และ 4 นั้น จะมีหน้าที่จัดการกับข้อผิดพลาดเพิ่มความเชื่อถือได้

ซึ่งอาจมองว่าซ้ำซ้อนกัน ทั้งนี้เพราะในระดับชั้นที่ 3 อาจทำให้เกิดการสูญหายของข้อมูลในช่วงที่อยู่ในคิวที่โหนดเองแทนที่จะไปหายบน line หรืออีกสาเหตุหนึ่งคือ อาจมีความจำเป็นต้องละทิ้งข้อมูลบางส่วนไปเพราะเกิด Congestion จึงทำให้ดูเหมือนว่าระดับชั้นที่ 2 และ 4 นี้มีหน้าที่ซ้ำซ้อนกันในบางส่วน ที่อธิบายถึงแนวความคิดนี้ก่อนเพราะต้องการให้เข้าใจว่าในการออกแบบโพรโทคอลแต่ละชั้นนั้น จะต้องพิจารณาสภาพแวดล้อมของโพรโทคอลระดับล่างคือผู้ให้บริการด้วย

ทีซีพีเป็นโพรโทคอลในระดับชั้นที่ 4 เมื่อเทียบกับ โอเอสไอ มีลักษณะการทำงานเป็น Virtual Circuit คือจะมีการทำวงจรเสมือนขึ้นมาก่อนที่จะรับส่งข้อมูลกัน นั่นคือแต่ละโหนดต้องมีตารางของแอดเดรสและ destination route เพื่อให้รู้ว่าจะต่อกับใครจึงจะได้วงจรเสมือนตามต้องการ เมื่อทำ connection setup เสร็จแล้วก็จะรับส่งข้อมูลกันโดยใช้เส้นทางนี้ตลอด ดังนั้นจะไม่มีปัญหาเรื่องการเรียงลำดับของชุดข้อมูลผิดพลาด หรือ เกิดการซ้ำซ้อนของข้อมูล การส่งผ่านข้อมูลบนทีซีพีเป็น byte stream-oriented สำหรับหน้าที่ของทีซีพีนี้ก็คือจัดการเรื่องตรวจสอบ error , ทำ flow control , ทำการ multiplex หรือ demultiplex application layer connection นอกจากนี้ก็ยังทำหน้าที่ควบคุมแลกเปลี่ยนสถานะและทำ Synchronization ด้วย

2.6.1 แบบจำลองการสื่อสารขององค์กรมาตรฐานสากล ISO

เป็นแบบจำลองของการสื่อสารข้อมูลที่กำหนดขึ้นมาโดยองค์การมาตรฐานสากล (International Standard Organization-ISO) และมีชื่อเรียกว่า Open System interconnection Model (OSI Model) ซึ่งก็อาจเรียกรวมกันแบบย่อๆ เป็น ไอเอสโอ/โอเอสไอ โมเดล แบบจำลองนี้ได้แบ่งระบบการทำงานในการสื่อสารออกเป็นชั้นย่อยๆ จำนวน 7 ชั้น (หรือ 7 เลเยอร์) เหตุผลที่ทำให้ต้องมีการแบ่งออกเป็น 7 ชั้นก็เพื่อกำหนดมาตรฐานและแบบจำลองที่จะใช้ในการอ้างอิงในแต่ละชั้นของการทำงาน, ช่วยลดขนาดของปัญหาในการสื่อสารให้เล็กลงเพื่อที่จะสามารถจัดการได้ง่ายขึ้น และสนับสนุนให้ผู้ผลิตรายต่างๆ สามารถพัฒนาผลิตภัณฑ์ที่สามรถทำงานร่วมกันได้ นอกจากนี้ยังช่วยในการพัฒนาระบบการสื่อสารไม่จำเป็นต้องเริ่มต้นจากศูนย์และต้องทำให้ครบทุกองค์ประกอบ แต่สามารถพัฒนาขึ้นมาเพียงชั้นเดียวจากจำนวน 7 ชั้นแล้วนำไปใช้งานร่วมกับชั้นอื่นที่มีการพัฒนาไว้แล้ว



รูป 2.27 โอเอสไอเลเยอร์

โดยหลักการแล้วแต่ละชั้นจะสื่อสารกับชั้นในระดับเดียวกันที่อยู่บนเครื่องอีกเครื่องหนึ่งแต่ในทางปฏิบัติแต่ละชั้นที่อยู่ติดกันทั้งที่อยู่บนหรือล่างเท่านั้น จะยกเว้นก็แต่ชั้นล่างสุดคือชั้น Physical ที่จะติดต่อกับชั้น Physical ของอีกเครื่องหนึ่งได้ มีข้อสังเกตว่าในทางปฏิบัติจริงมีบางโอกาสเครื่องคอมพิวเตอร์อาจจะมีการใช้ชั้นใดชั้นหนึ่งพร้อมๆ กันหลายๆ ครั้ง โดยแต่ละครั้งมีความเป็นอิสระต่อกัน เช่น กรณีเครื่องคอมพิวเตอร์ที่ติดตั้งการ์ดเครือข่ายจำนวนหลายการ์ด โดยอาจจะเป็นอีเทอร์เน็ต (Ethernet) ทั้งหมด หรืออีเทอร์เน็ตผสมกับโทเคนริง (Token Ring) และอื่นๆ ก็ได้ในโลกแห่งจินตนาการของแบบจำลอง ไอเอสโอ/โอเอสไอ นั้น ควรจะได้ถูกกำหนดตัวสถาปัตยกรรมนี้ขึ้นมาก่อน จากนั้นชิ้นส่วนต่างๆ ทางด้านเครือข่ายจึงตามมาทีหลัง โดยหน่วยงานทางการค้า, หน่วยงานวิจัย และองค์กรมาตรฐาน แต่ละชิ้นส่วนที่ถูกสร้างขึ้นมาจะต้องตรงตามมาตรฐานชั้นใดชั้นหนึ่งของสถาปัตยกรรมนี้อย่างสมบูรณ์ อย่างไรก็ตามในโลกแห่งความเป็นจริงนั้นมีเทคโนโลยีจำนวนไม่น้อยที่ถูกพัฒนาขึ้นก่อนที่จะมีแบบจำลองนี้ และในบางกรณีเทคโนโลยีที่เกิดขึ้นภายหลังจากนี้บางอย่างก็ไม่ได้เดินตามแบบจำลองนี้อย่างสมบูรณ์ แม้ว่าโลกแห่งความเป็นจริงจะดูห่างไกลจากโลกแห่งจินตนาการ แต่ผลที่ได้รับจากแบบจำลองนี้ก็ช่วยให้เกิดแรงผลักดันไปสู่การพัฒนาระบบที่สามารถทำงานร่วมกันได้ของผู้ผลิตรายการต่างๆรายละเอียดของแต่ละชั้นในแบบจำลองมีดังนี้

2.6.1.1 ชั้น Physical

ชั้น Physical เป็นการอธิบายคุณสมบัติทางกายภาพ เช่น คุณสมบัติทางไฟฟ้า และกลไกต่างๆ ของวัสดุที่ใช้เป็นสื่อกลาง ตลอดจนสัญญาณที่ใช้ในการส่งข้อมูล คุณสมบัติที่กำหนดไว้ในชั้นนี้ประกอบด้วยคุณลักษณะทางกายภาพของสาย, อุปกรณ์เชื่อมต่อ (Connector), ระดับความต่างศักย์ของไฟฟ้า (Voltage) และอื่นๆ เช่น อธิบายถึงคุณสมบัติของสาย Unshielded Twisted Pair (UTP)

2.6.1.2 ชั้น Data-Link

ชั้น Data-Link เป็นชั้นที่อธิบายถึงการส่งข้อมูลไปบนสื่อกลาง ชั้นนี้ยังได้ถูกแบ่งออกเป็นชั้นย่อย (Sub Layer) คือ Logical Link Control (LLC) และ Media Access Control (MAC) การแบ่งแยกเช่นนี้จะทำให้ชั้น LLC ชั้นเดียวสามารถจะใช้ชั้น MAC ที่แตกต่างกันออกไปได้หลายชั้น ชั้น MAC นั้นเป็นการดำเนินการเกี่ยวกับแอดเดรสทางกายภาพอย่างที่ใช้ในมาตรฐานอีเทอร์เน็ตและโทเคนริง แอดเดรสทางกายภาพนี้จะถูกฝังมาในการ์ดเครือข่ายโดยบริษัทผู้ผลิตการ์ดนั้น แอดเดรสทางกายภาพนั้นเป็นคนละอย่างกับแอดเดรสทางตรรกะ เช่น IP Address ที่จะถูกใช้งานในชั้น Network เพื่อความชัดเจนครบถ้วนสมบูรณ์ของการใช้ชั้น Data-Link นี้

2.6.1.3 ชั้น Network

ในขณะที่ชั้น Data-Link ให้ความสนใจกับแอดเดรสทางกายภาพ แต่การทำงานในชั้น Network จะให้ความสนใจกับแอดเดรสทางตรรกะ การทำงานในชั้นนี้จะเป็นการเชื่อมต่อและการเลือกเส้นทางนำพาข้อมูลระหว่างเครื่องสองเครื่องในเครือข่ายชั้น Network ยังให้บริการเชื่อมต่อในแบบ “Connection Oriented” อย่างเช่น X.25 หรือบริการแบบ “Connectionless” เช่น Internet Protocol (IP) ซึ่งใช้งานโดยชั้น Transport ตัวอย่างของบริการหลักที่ชั้น Network มีให้คือ การเลือกเส้นทางนำพาข้อมูลไปยังปลายทาง
ที่เรียกว่า Routing ตัวอย่างของโปรโตคอลในชั้นนี้ประกอบด้วย Internet Protocol (IP) และ Internet Control Message Protocol (ICMP)

2.6.1.4 ชั้น Transport

ในชั้นนี้มีบางโพรโทคอลจะให้บริการที่ค่อนข้างคล้ายกับที่มีในชั้น Network โดยมีบริการด้านคุณภาพที่ทำให้เกิดความน่าเชื่อถือ แต่ในบางโพรโทคอลที่ไม่มีการดูแลเรื่องคุณภาพดังกล่าวจะอาศัยการทำงานในชั้น Transport นี้เพื่อเข้ามาช่วยดูแลเรื่องคุณภาพแทน เหตุผลที่สนับสนุนการใช้งานชั้นนี้ก็คือ ในบางสถานการณ์ของชั้นในระดับล่างทั้งสาม (คือชั้น Physical, Data-Link และ Network) ดำเนินการโดยผู้ให้บริการโทรคมนาคม การจะเพิ่มความมั่นใจในคุณภาพให้กับผู้ใช้บริการก็ด้วยการใช้ชั้น Transport นี้ "Transmission Control Protocol (TCP) เป็นโปรโตคอลในชั้น Transport ที่มีการใช้งานกันมากที่สุด"

2.6.1.5 ชั้น Session

ชั้น Session ทำหน้าที่สร้างการเชื่อมต่อ, การจัดการระหว่างการเชื่อมต่อและการตัดการเชื่อมต่อคำว่า "เซสชัน" (Session) นั้นหมายถึงการเชื่อมต่อกันในเชิงตรรกะ (Logic) ระหว่างปลายทางทั้งสองด้าน (เครื่อง 2 เครื่อง) ชั้นนี้อาจไม่จำเป็นต้องถูกใช้งานเสมอไป อย่างเช่น ถ้าการสื่อสารนั้นเป็นไปในแบบ "Connectionless" ที่ไม่จำเป็นต้องเชื่อมต่อ เป็นต้น ระหว่างการสื่อสารในแบบ "Connection-less" ทุกๆ แพ็คเกต (Packet) ของข้อมูลจะมีข้อมูลเกี่ยวกับเครื่องปลายทางที่เป็นผู้รับติดอยู่อย่างสมบูรณ์ในลักษณะของจดหมายที่มีการจ่าหน้าซองอย่างถูกต้อง

ครบถ้วน ส่วนการสื่อสารในแบบ "Connection Oriented" จะต้องมีการดำเนินการบางอย่างเพื่อให้เกิดการเชื่อมต่อ หรือเกิดเป็นวงจรในเชิงตรรกะขึ้นมาก่อนที่การรับ/ส่งข้อมูลจะเริ่มต้นขึ้น แล้วเมื่อการรับส่งข้อมูลดำเนินไปจนเสร็จสิ้นก็ต้องมีการดำเนินการบางอย่างเพื่อที่จะตัดการเชื่อมต่อลง ตัวอย่างของการเชื่อมต่อแบบนี้ได้แก่การใช้โทรศัพท์ที่ต้องมีการกดหมายเลขปลายทาง จากนั้นก็ต้องมีการดำเนินการบางอย่างของระบบจนกระทั่งเครื่องปลายทางมีเสียงดังขึ้น การสื่อสารจะเริ่มขึ้นจริงเมื่อมีการทักทายกันของคู่สนทนา จากนั้นเมื่อคู่สนทนาฝ่ายใดฝ่ายหนึ่งวางหูก็ต้องมีการดำเนินการบางอย่างที่จะตัดการเชื่อมต่อลงชั้น Session นี้มีระบบการติดตามด้วยว่าฝั่งใดที่ส่งข้อมูลซึ่งเรียกว่า "Dialog Management" Simple Mail Transport Protocol (SMTP), File Transfer Protocol (FTP) และ Telnet เป็นตัวอย่างของโปรโตคอลที่นิยมใช้ และมีการทำงานครอบคลุมในชั้น Session, Presentation และ Application

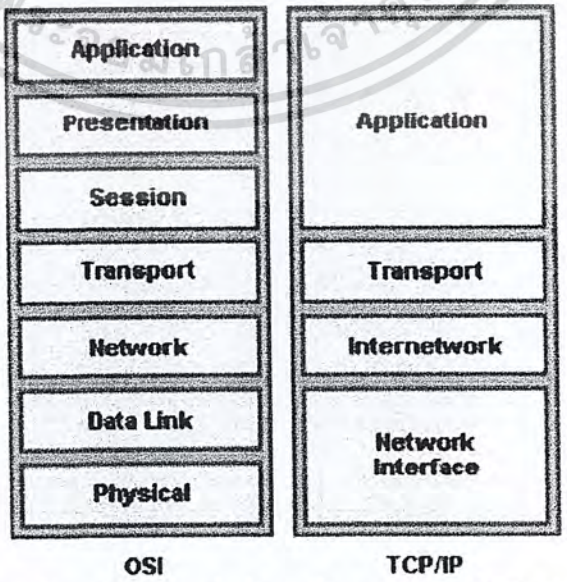
2.6.1.6 ชั้น Presentation

ชั้น Presentation ให้บริการทำการตกลงกันระหว่างสองโปรโตคอลถึงไวยากรณ์ (Syntax) ที่จะใช้ในการรับส่งข้อมูล เนื่องจากว่าไม่มีการรับรองถึงไวยากรณ์ที่จะใช้ร่วมกัน การทำงานในชั้นนี้จึงมีบริการในการแปลข้อมูลตามที่ได้รับการร้องขอด้วย

2.6.1.7 ชั้น Application

ชั้น Application เป็นชั้นบนสุดของแบบจำลอง ไอเอสโอ/โอเอสไอ เป็นชั้นที่ใช้บริการของชั้น Presentation (และชั้นอื่นๆ ในทางอ้อมด้วย) เพื่อประยุกต์ใช้งานต่างๆ เช่น การทำ E-mail Exchange (การรับส่งอีเมล์), การโอนย้ายไฟล์ หรือการประยุกต์ใช้งานทางด้านเครือข่ายอื่นๆ

2.6.2 เปรียบเทียบระหว่างโมเดลโอเอสไอกับทีซีพีไอพี



รูป 2.28 เปรียบเทียบระหว่างโมเดลโอเอสไอกับทีซีพีไอพี

เนื่องจาก โอเอสไอ เกิดขึ้นมาหลังจากที่ซีพีไอพีได้มีการใช้งานกันอย่างแพร่หลายไปแล้ว โดยซีพีไอพีใช้ในเครือข่าย ARPANET เป็นเครือข่ายแรก ซึ่งต่อมาได้ขยายการเชื่อมต่อไปทั่วโลกเป็นเครือข่ายอินเตอร์เน็ต ทำให้มาตรฐานของทีซีพีไอพีเป็นที่ยอมรับ กันอย่างกว้างขวาง และการที่ทีซีพีไอพีเป็นโปรโตคอลชนิดที่ให้ใช้โดยไม่ต้องจ่ายค่าลิขสิทธิ์ การใช้งานซีพีไอพีก็ยังมีจำนวนผู้ใช้เพิ่มมากขึ้นไปอีกจนถือเป็นมาตรฐานที่มีผู้ใช้รับส่งข้อมูลมากที่สุดในปัจจุบัน เมื่อซีพีไอพีเป็นมาตรฐานที่เกิดขึ้นก่อน OSI 7-Layer Model มาตรฐานของซีพีไอพีจึงไม่ใช่มาตรฐานเดียว กันกับของ โอเอสไอ โดยซีพีไอพีจะมีการแบ่งจำนวนขั้นตอนที่ใช้รับส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์สองระบบออกเป็น 4 ชั้น เท่านั้น หรือ เรียกว่าเป็นTCP/IP Stack โดยมีชื่อเรียกแตกต่างกันดังนี้

ชั้นบนคือ Process Layer จะเป็น แอพพลิเคชันโพรโทคอล ที่ทำหน้าที่เชื่อมต่อกับผู้ใช้และให้บริการต่าง ๆ เช่น FTP, Telnet, SNMP ฯลฯ ชั้นถัดมาคือ Host-to-Host Layer จะควบคุมการรับส่งข้อมูลจากด้านส่งถึงด้านรับข้อมูล

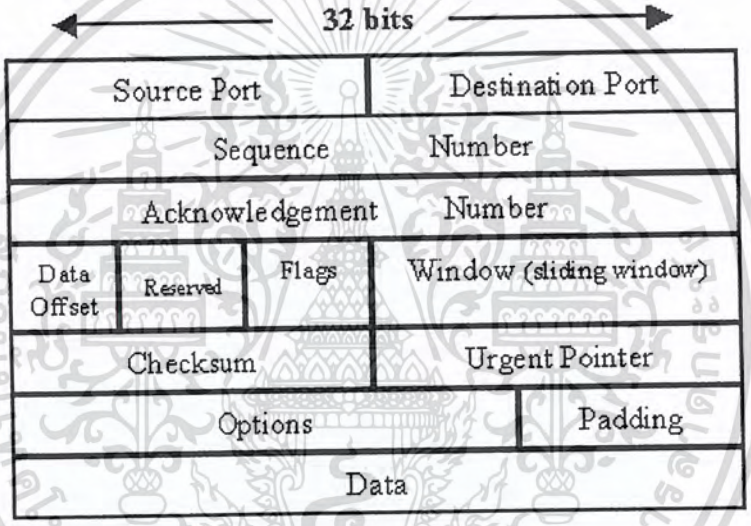
ชั้นถัดลงมาคือ Internetwork Layer ได้แก่ส่วนของโปรโตคอล IP ทำหน้าที่เชื่อต่อคอมพิวเตอร์เข้ากับระบบ เครือข่ายที่อยู่ชั้นล่างลงไป และทำหน้าที่เลือกเส้นทางการรับส่งข้อมูลผ่านอุปกรณ์เครือข่ายต่าง ๆ จนไปถึงผู้รับส่งข้อมูล

ส่วนชั้นสุดท้ายคือ Network Interface จะทำหน้าที่เชื่อมต่อกับฮาร์ดแวร์ และควบคุมการรับส่งข้อมูลในระดับ ฮาร์ดแวร์ของเครือข่าย ซึ่งที่ใช้กันอยู่จะเป็นตามมาตรฐานของ IEEE เช่น IEEE 802.3 จะเป็นการเชื่อมต่อผ่าน LAN เป็นต้น

2.6.3 โพรโทคอลทีซีพี

โพรโทคอลทีซีพี(Transmission Control Protocol) เป็นโพรโทคอลที่มีการรับส่งข้อมูลแบบ stream oriented protocol หมายความว่า การรับส่งข้อมูลจะไม่คำนึงถึงปริมาณข้อมูลที่จะส่งไป แต่จะแบ่งข้อมูลเป็นส่วนย่อย ๆ ก่อน แล้วจึงจะส่งไปยังปลายทางอย่างต่อเนื่องเป็นลำดับข้อมูล ในกรณีที่ข้อมูลส่วนใดส่วนหนึ่งสูญหายไป ก็จะส่งข้อมูลส่วนนั้นใหม่อีกครั้ง สำหรับปลายทางก็จะทำหน้าที่จัดเรียงส่วนของข้อมูล datagram ใหม่ให้ต่อเนื่องกันและประกอบกลับเป็นข้อมูลทั้งหมดได้ ซึ่งจะแยกข้อมูลที่ไม่ถูกต้องออก ดังนั้นแอพพลิเคชันหรือโพรเซสใดที่อาศัยการส่งผ่านข้อมูลด้วยโพรโทคอลทีซีพี จะต้องใช้หน่วยความจำและขนาดของช่องสัญญาณ (bandwidth) มากกว่ายูดีพี การติดต่อระหว่างกันจะต้องเป็นแบบ connection-oriented คือต้องมีการสร้างการติดต่อกันเป็น session ทั้ง 2 ด้านเสียก่อน แล้วจึงจะรับส่งข้อมูลไปได้พร้อมกัน (full duplex) เหมือนกับการใช้โทรศัพท์ติดต่อกัน เมื่อผู้ติดต่อต้นทางเรียกให้ฝ่ายตรงข้ามรับสายแล้ว จึงเริ่มการสนทนา เช่น พูดคำว่า "สวัสดี" หรือ "ฮัลโหล" กันก่อนเพื่อให้แน่ใจว่าฝ่ายตรงข้ามพร้อมจะติดต่อด้วย จากนั้นจึงเริ่มต้นติดต่อกัน และเมื่อต้องการจะเลิกการติดต่อก็จะมีการพูดคำว่า "สวัสดี" ให้ฝ่ายตรงข้ามทราบว่าจะเลิกการติดต่อและวางสายไป ซึ่งในระหว่างการติดต่อกันนั้น แม้ว่าฝ่ายใด

ฝ่ายหนึ่งหรือทั้งสองฝ่ายจะเงียบไป คือไม่พูดอะไรเป็นเวลานาน ๆ แต่การเชื่อมโยงระหว่างทั้งสองด้านยังคงมีอยู่ไม่ขาดไปจนกว่าฝ่ายใดฝ่ายหนึ่งจะวางสาย เช่นเดียวกับการติดต่อกันด้วยกลไกโพรโทคอลทีซีพี เมื่อแอพพลิเคชันต้องการส่งผ่านข้อมูลจะใช้โพรโทคอลที่เหมาะสมในชั้น Process layer ติดต่อไปและมีการสร้างช่องส่งข้อมูลผ่านพอร์ทที่กำหนดเพื่อส่งผ่านข้อมูลไปยังโพรโทคอลทีซีพีในระหว่างการรับส่งข้อมูลนี้โพรโทคอลทีซีพี จะเพิ่มขบวนการสอบทานข้อมูลเพื่อให้ข้อมูลมีความถูกต้องไม่ผิดพลาดไปจากเดิม โดยการส่งสัญญาณสอบทานข้อมูล (acknowledgement) และส่งข้อมูลให้ใหม่อีกครั้ง ถ้าปลายทางไม่ได้รับหรือเกิดความผิดพลาดขึ้น ความน่าเชื่อถือของการส่งผ่านข้อมูลโดยโพรโทคอลทีซีพีจะมีมากกว่า แต่ก็ต้องอาศัยทรัพยากรของระบบมากกว่าในการทำงานเช่นกัน



รูป 2.29 ทีซีพีเซกเมนต์

จากภาพข้างต้นสามารถอธิบายรายละเอียดดังนี้

2.6.3.1 Source Port Number

หมายถึง พอร์ตที่โฮสต์ต้นทางใช้ในการสื่อสารกันของเซสชันนี้ และ TCP/IP จะใช้พอร์ตนี้ตลอดไป โดยทั่วไปพอร์ตนี้จะเรียกว่า "ไคล์เอนต์พอร์ต" คือ พอร์ตที่ไคลเอนต์เปิดขึ้นมาเพื่อรอการตอบรับจากเซิร์ฟเวอร์ ไคลเอนต์พอร์ต จะมีหมายเลขไม่แน่นอนและเปลี่ยนไปทุกครั้งที่มีการเริ่มการเชื่อมต่อใหม่เป็นพอร์ตที่ถูกเปิดไว้ในระยะเวลาสั้นๆ ค่าที่เป็นไปได้ของพอร์ตนี้ขึ้นอยู่กับการจัดสรรของระบบปฏิบัติการ ในการกำหนดขอบเขตของพอร์ตเหล่านี้ส่วนใหญ่จะมีค่าอยู่ในช่วง 1024 - 5000

2.6.3.2 Destination Port Number

หมายถึง หมายเลขพอร์ตบนโฮสต์ปลายทางที่โฮสต์ต้นทางต้องการติดต่อด้วย โดยนัยแล้วจะหมายถึง แอพพลิเคชั่นที่ให้บริการอยู่พอร์ตนั้นที่โฮสต์ปลายทางนั่นเอง พอร์ต

นั้นจะเรียกอีกอย่างว่า "เซิร์ฟเวอร์พอร์ต"หมายเลขพอร์ตที่เปิดไว้จะขึ้นอยู่กับแอพพลิเคชั่นที่ให้บริการ โดยทั่วไปแอพพลิเคชั่นแต่ละประเภทจะมีหมายเลขพอร์ตเป็นมาตรฐานสำหรับให้ไคลเอนต์ได้เรียกใช้บริการ

2.6.3.3 Sequence Number

เป็นฟิลด์ที่ระบุถึงหมายเลขลำดับที่ใช้อ้างอิงในการสื่อสารข้อมูลแต่ละครั้ง เพื่อให้ทั้ง 2 ฝ่ายจะได้รับทราบตรงกันว่าเป็นข้อมูลของชุดใด การนำไปใช้งานจะได้ไม่ปะปนกัน และมีลำดับที่ถูกต้อง เนื่องจากการสื่อสารข้อมูลผ่าน TCP นั้นจังหวะและลำดับเป็นส่วนสำคัญของโปรโตคอลไม่ได้น้อยไปกว่าข้อมูลใน TCP

2.6.3.4 Header

รวมไปถึงการที่ข้อมูลในแต่ละ TCP Segment อาจจะถูกทำการแฟรกเมนต์ในเลเยอร์ของ IP ถัดลงไปทำให้ข้อมูลถูกแบ่งออกและส่งไปในลำดับที่ไม่เรียงกัน หากไม่มีจุดอ้างอิงของข้อมูลก็จะไม่สามารถอ่านข้อมูลกลับใหม่ได้อย่างสมบูรณ์และถูกต้อง การส่งข้อมูลและการตอบรับจะใช้ฟิลด์นี้เป็นตัวยืนยันระหว่างกันเสมอ

2.6.3.5 Acknowledge Number

ทำหน้าที่เช่นเดียวกับ Sequence Number ต่างกันตรงที่เป็น Sequence Number ซึ่งในการตอบรับกล่าวคือ เนื่องจาก Sequence Number ที่ใช้ในการอ้างอิงนั้นผู้ที่เริ่มส่งข้อมูลจะเป็นผู้กำหนดเลขขึ้นมาและส่งไปพร้อมกับการสร้างการเชื่อมต่อครั้งใหม่แต่สำหรับฝ่ายที่ถูกติดต่อก็จำเป็นต้องกำหนดหมายเลขสำหรับใช้อ้างอิง ในการตอบรับเช่นกัน ค่าที่อยู่ใน ACK Number ก็คือหมายเลขที่ใช้อ้างอิงในการตอบรับนี้

2.6.3.6 Header Length

โดยปกติความยาวของ TCP Header จะเท่ากับ 20 ไบต์ แต่ถ้าหากมีการใช้ Option อาจจะทำให้ขนาดของ Header ยาวขึ้นตามข้อมูลที่ต้องเพิ่มมาจาก Option นั้น แต่ทั้งหมดแล้วจะไม่เกิน 60 ไบต์ เป็นข้อมูลในระดับบิตที่ใช้เป็นตัวบอกคุณสมบัติของ TCP Segment ที่กำลังส่งอยู่นั้น และใช้เป็นตัวควบคุมจังหวะ การรับส่งข้อมูลด้วย Flag ทั้งหมดมีอยู่ 6 บิต แต่ละบิตจะมีชื่อและมีความหมาย ดังนี้

1) URG ใช้บอกความหมายว่าเป็นข้อมูลด่วน และมีข้อมูลพิเศษมาด้วย
2) ACK แสดงว่าข้อมูลในฟิลด์ Acknowledge Number นำมาใช้งานได้
3) PSH เพื่อแจ้งให้ผู้รับข้อมูลทราบว่า ควรจะส่งข้อมูล Segment นี้ไปยัง Process ที่กำลังรออยู่ทันที
4) RST ใช้ในกรณีที่เกิดการสับสนขึ้นด้วยเหตุผลต่างๆ เช่น โฮสต์มีปัญหา
5) SYN ใช้ในการเริ่มต้นขอติดต่อกับปลายทาง
6) FIN ใช้ส่งเพื่อแจ้งให้ปลายทางทราบว่ายุติการติดต่อ

7) Window Size เป็นขนาดของการรับ - ส่งข้อมูลในแต่ละครั้งที่ทางฝ่ายผู้รับจะสามารถรับได้ เนื่องจากในการรับข้อมูลนั้น ทางผู้รับจะต้องจัดเตรียมหน่วยความจำในการพักข้อมูลที่มาจาก TCP และทำการ Demultiplex ออกมา หากไม่มีการตกลง ถึงขนาดที่ทางฝ่ายรับสามารถรับได้ ก็จะทำให้การสื่อสารข้อมูลไม่สมดุล และฝ่ายรับอาจจะประมวลผลทัน ซึ่งจะส่งผลให้ต้องส่งข้อมูลซ้ำหลายครั้ง

8) Check sum

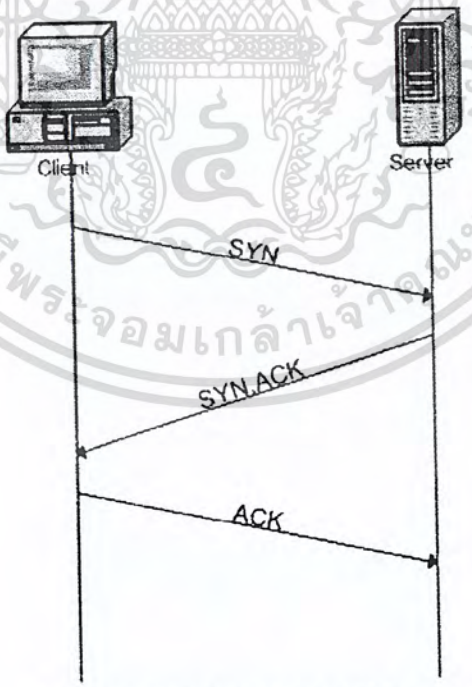
คือ ฟิลด์ที่ใช้ในการตรวจสอบความถูกต้องของข้อมูลใน TCP Segment ล่าสุดที่อยู่ในโหมด Urgent

9) Urgent pointer

คือ ข้อมูลที่เพิ่มเติมซึ่งจะอยู่ใน TCP Header เมื่อมีการตั้งค่า Option บางอย่าง ต้องการข้อมูลเพิ่มเติม ซึ่งไม่มีใน TCP Header เช่น MSS, Strict Route

2.6.4 การทำงานของโพรโทคอลทีซีพี

2.6.4.1 Connection Establishment

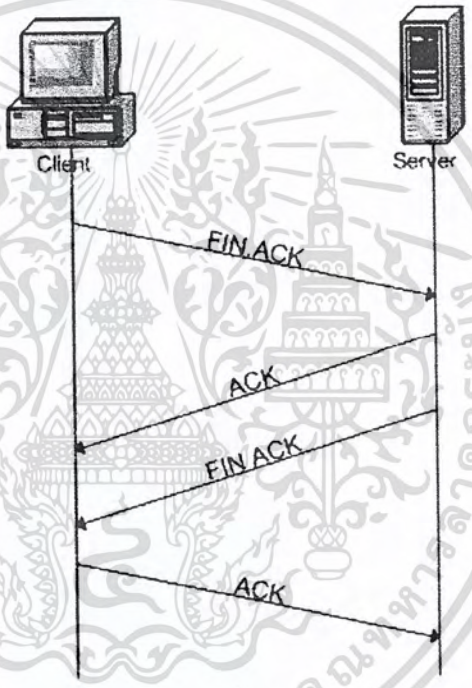


รูป 2.30 Connection Establishment

ก่อนที่จะเริ่มต้นการสื่อสาร จะต้องมีการส่งสัญญาณเพื่อบอกโฮสต์อีกฝั่งหนึ่งให้เตรียมตัวติดต่อ ซึ่งกระบวนการที่ใช้มีชื่อเรียกว่า 3-Way Hand Shake มีขั้นตอนคือ เครื่อง

ไคลเอนต์จะทำการส่งเซกเมนต์ โดยเปิด SYN Flag ระบุหมายเลขพอร์ตที่ต้องการติดต่อบนเซิร์ฟเวอร์และระบุหมายเลข ลำดับของข้อมูล (ISN - Initial Sequence Number) เครื่องเซิร์ฟเวอร์เมื่อได้รับข้อมูลเซกเมนต์จากข้อ 1 ก็จะตอบกลับด้วยการเพิ่มค่า ISN ที่ได้รับขึ้นอีก 1 พร้อมทั้งระบุหมายเลขลำดับ (ISN) ของตนเอง และเปิด SYN กับ ACK Flag ไคล์เอนต์เมื่อได้รับการตอบกลับจากเซิร์ฟเวอร์ตามข้อ 2 ก็จะทำการตอบรับกลับไป โดยการเพิ่มค่า ISN ของเซิร์ฟเวอร์ขึ้นอีก 1 และเปิด ACK Flag เมื่อผ่านการสร้าง connection ทั้ง 3 ขั้นตอนแล้ว ตอนนี้ทั้งไคล์เอนต์ และเซิร์ฟเวอร์เปรียบเสมือนมีการเชื่อมต่อถึงกันแล้ว สถานะของการเชื่อมต่อในขณะนี้เรียกว่า Established

2.6.4.2 Connection Termination



รูป 2.31 Connection Termination

เมื่อการสื่อสารของทั้งสองฝั่งจบลง และไม่ต้องการรับส่งข้อมูลอีกต่อไป จะต้องทำตามขั้นตอนการยุติการสื่อสารเพื่อให้การสื่อสารจบลงอย่างสมบูรณ์ ซึ่งมีอยู่ 4 ขั้นตอนคือ

1) ไคลเอนต์ทำการส่ง ISN พร้อมกับ FIN ACK Flag ไปยังเซิร์ฟเวอร์

2) เซิร์ฟเวอร์ทำการตอบรับ ISN และบวกค่า ISN อีก 1 พร้อม ACK Flag

3) เซิร์ฟเวอร์ทำการส่ง ISN พร้อมกับ FIN ACK Flag ไปยังไคลเอนต์

4) ไคลเอนต์ทำการตอบรับ ISN และบวกค่า ISN อีก 1 พร้อม ACK Flag

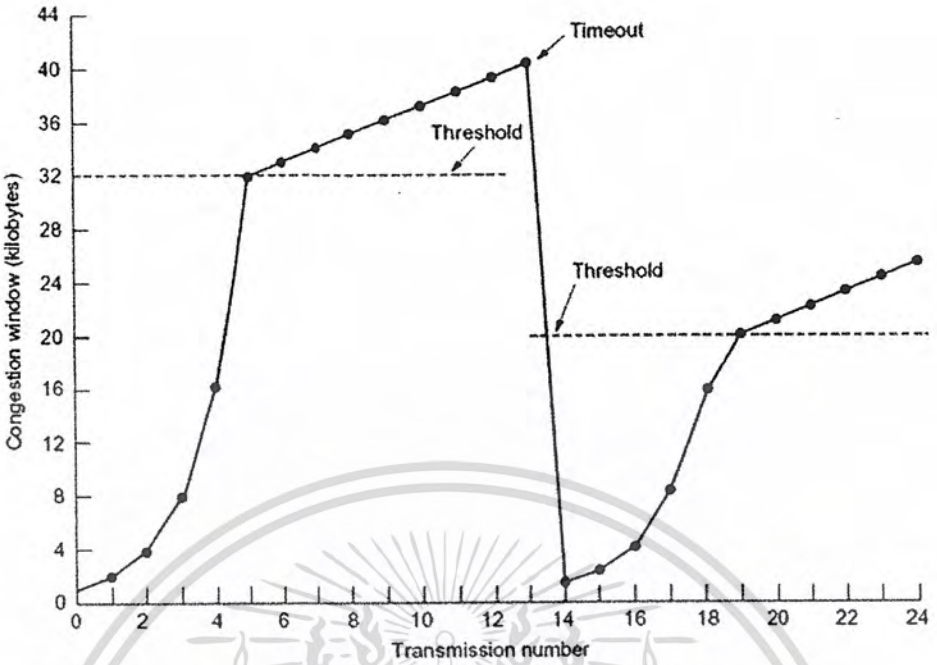
การยุติการเชื่อมต่อ โดยส่ง FIN ACK ออกไปมีความหมายคือ โฮสต์ที่ส่งไม่มีข้อมูลจะส่งไปอีก มิใช่ต้องการปิดการสื่อสารทั้งหมดในทันที ดังนั้นจึงต้องทำทั้งสองทาง การ

สื่อสารจึงจะยุติลงอย่างสมบูรณ์ ในการใช้งานจริง อาจมีการยุติการสื่อสารเพียงด้านเดียว คือหยุดส่งข้อมูล แต่ยังคงเปิดพอร์ตไว้รอรับข้อมูลจากอีกด้านหนึ่ง ทั้งนี้ขึ้นอยู่กับลักษณะการใช้งาน การปิดพอร์ตสื่อสารเพียงด้านเดียวเช่นนี้ เรียกว่า Half-Close

2.6.4.3 Congestion Control

คือการควบคุมความแออัดของข้อมูล เนื่องมาจากเส้นทางในเครือข่ายรองรับปริมาณดาต้าที่วิ่งอยู่บนเครือข่ายไม่ได้ เช่น อินพุทเข้ามาเยอะ แต่ส่งออกไม่ทันเมื่อเกิดความแออัด (Congestion) ทำให้รับ-ส่งข้อมูลไม่ทัน ดังนั้นแพ็คเกตส่วนใหญ่จะถูกทิ้งไป คิดเป็น 99% เมื่อ Packet ถูกทิ้งไป ผู้ส่งไม่ได้รับ acknowledge ก็จะพยายามส่งใหม่ เพราะเมื่อไม่มี acknowledge กลับมาภายในเวลาที่กำหนดทีซีพีจะตีความว่าแพ็คเกตนั้นสูญหาย และจะทำการส่งใหม่เราไม่สามารถแก้ปัญหาได้โดยตรง แต่สามารถหลีกเลี่ยงได้โดยใช้วิธีของ Jacobson's algorithm มีอยู่ 2 ส่วนคือ

1) Slow start เริ่มต้นส่งทีละน้อยแล้วค่อยๆ เพิ่มขึ้น Set congestion window (cwnd) ที่ 1 segment เพิ่ม cwnd ครั้งละเท่าตัวแบบ Exponential จนถึงระดับของ window size ที่ต้องการ เช่นต้องการส่งข้อมูลขนาด 64 กิโลไบต์ จะลดลงมาที่ 1 และไล่ขึ้นไป 1,2,4,8,16,32,64 หากเพิ่มขนาดข้อมูลเกิน Threshold จะเข้าสู่สถานะ Congestion avoidance
2) Congestion avoidance ทำการเพิ่มขนาดของวินโดว์การส่งครั้งละ 1 ไปเรื่อยๆจนกว่าจะเกิด Time Out



รูป **2.32 Congestion Control**

เมื่อเกิด Time Out จะทำการลดขนาดของวินโดว์การส่งเป็น 1 ใหม่ และขนาดของ Threshold ลดลงเหลือครึ่งหนึ่งของขนาดของวินโดว์ที่เกิด Time Out ขณะนั้น

2.6.4.4 Flow Control

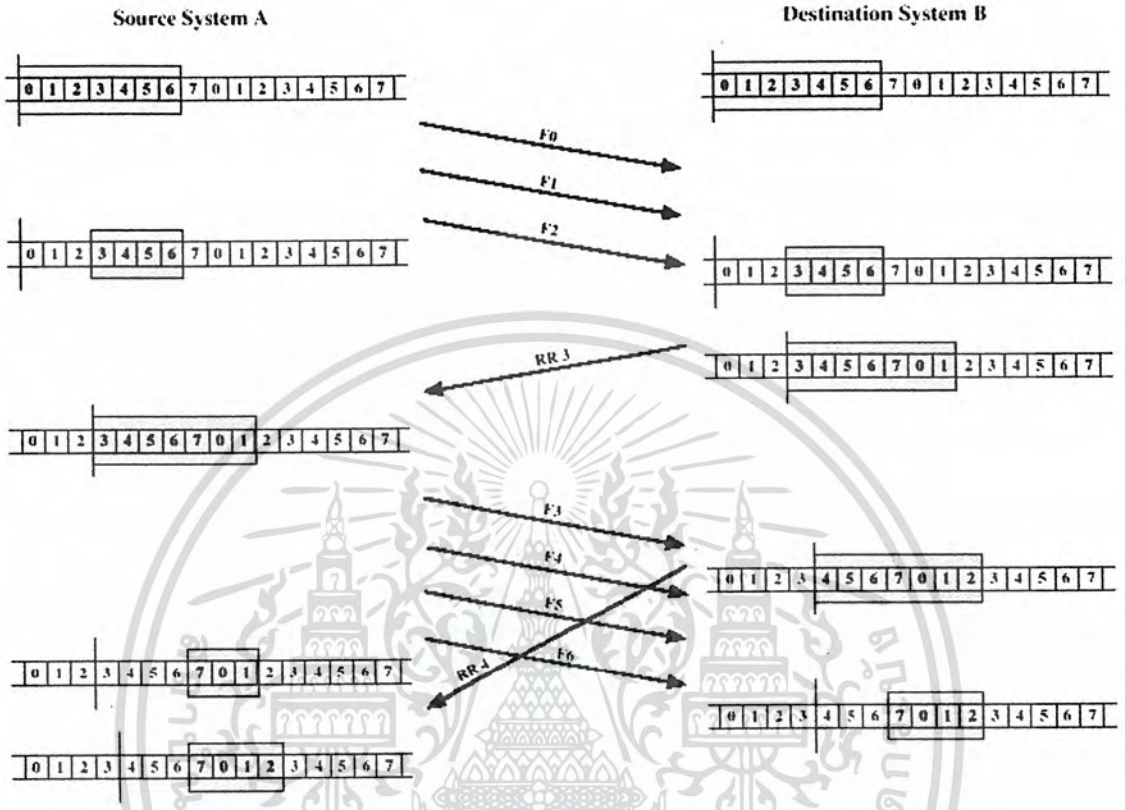
เป็นการจัดการ การไหลของข้อมูล เพื่อให้การไหลของข้อมูลมีประสิทธิภาพในการทำงานสูงและเมื่อส่งข้อมูลจำนวนมากข้อมูลจะได้ไม่ตกหล่นหรือสูญหาย

Sliding Windows คือวิธีการควบคุมการไหลของข้อมูลแบบเลื่อนหน้าต่างผู้ส่งสามารถส่งเฟรมข้อมูลจำนวนมากๆก่อนที่จะได้รับการตอบกลับ acknowledge ของผู้รับเพื่อรับเฟรมถัดไป แต่ผู้รับจะมีการตอบกลับแค่เพียงบางเฟรมเท่านั้น

กำหนดให้มีข้อมูลจำหนวน 7 เฟรม คือ เฟรมที่ F0 - F6

ในการเริ่มต้นของการส่ง ผู้ส่งอาจส่งเฟรมทั้งหมด 7 เฟรม นั้นก็คือ F0-F6 ไปยังผู้รับ แต่ในขณะที่ผู้ส่งได้ทำการส่งเฟรมที่ F0 F1และ F2 ไปได้ยังไม่มีการตอบกลับ ACK ผู้ส่งจะหดตารางเหลือ 4 เฟรม โดยมีการคัดลอกเฟรมที่ส่งไป 3 เฟรม ก็คือ F0 F1และ F2ก่อนหน้านั่นลงไปในบัฟเฟอร์ ของผู้รับ และเมื่อผู้รับคัดลอกเสร็จแล้วผู้รับก็จะทำการตอบกลับ RR 3 (Receive Ready) นั้นก็คือ ผู้รับพร้อมจะรับข้อมูลตั้งแต่เฟรมเลขที่ 3 และ Bar จะทำการหดไปอยู่ที่เลขที่ 3 และ จะบวกเพิ่มไปอีก 3 เฟรม ตารางของผู้ส่งก็จะเพิ่มไปอีก 3 เช่นเดียวกัน และทำการส่งเฟรมหมายเลขที่ 3-6 ไปยังผู้รับ ผู้รับเมื่อได้รับเฟรมเลขที่ 3 แล้วก็จะทำการหดและบวกเพิ่มไปอีก 1 เฟรม และทำการ RR4 เพื่อ

แสดงว่า ผู้รับพร้อมจะรับเฟรมหมายเลขที่ 4 แล้ว ผู้ส่งก็จะหดเฟรมลง 1 เฟรมและ บวกเพิ่มอีก 1 เฟรมและทำการส่งเฟรมเลขที่ 4 ต่อไป



รูป 2.33 Flow Control

2.7 Action Script

Flash ใช้ Action Script เป็นภาษาในการเพิ่มความสามารถเชิงโต้ตอบ interactive ให้กับการแสดงผล สิ่งที่ Action Script ทำได้ เช่น ควบคุมหัวอ่านเฟรมของมูฟวี่, สร้างเว็บไซท์ เชิงโต้ตอบกับผู้ใช้, เกม, การเพิ่มเทคนิคพิเศษ ให้กับการเคลื่อนไหว, การสร้างฟอร์มรับข้อมูลจากผู้ใช้, การติดต่อกับฐานข้อมูล รวมไปถึงการสร้างระบบ chat แบบ real time หรือ multiplayer online game

Action script จัดเป็นภาษาแบบ script language มีลักษณะเชิงออบเจ็ค (object-oriented programming language) เหมือนกับ JavaScript ซึ่งจัดเป็นมาตรฐาน ชนิดเดียวกันเรียกว่า ECMA-262 ถูกกำหนดขึ้นโดย European Computer Manufacturers Association (ECMA) เพื่อใช้เป็นมาตรฐานของภาษา java script ดังนั้นหากผู้ศึกษา action script มีความสามารถในการเขียน JavaScript อยู่แล้วจะสามารถเข้าใจได้ง่ายขึ้นมาก

ในยุคแรกของ Flash นั้น คำสั่งต่างๆที่มีให้ใช้สำหรับควบคุมTimeline และกลไกต่างๆถูกเรียกว่า Actions ซึ่งสามารถเรียกใช้ได้จากการลากคำสั่งที่ต้องการจากรายการคำสั่งที่มีมาให้มาจัดวางตามลำดับที่ต้องการให้ทำงาน ต่อมาในปี2000 เมื่อมีการออก flash 5 ชุดคำสั่งนั้นถูกพัฒนาให้มี

การใช้ในลักษณะการเขียนโปรแกรมแบบภาษาคอมพิวเตอร์เชิงสคริปและถูกเรียกใหม่ว่า Action script ซึ่งมีลักษณะคล้ายกับ JavaScript ดังที่กล่าวมาแล้วข้างต้น

จนกระทั่งปี 2003 เมื่อ Flash ถูกพัฒนาเป็นเวอร์ชัน MX2004 ทำงานคู่กับ flash player7 ได้มีการพัฒนารูปแบบ Action Script จากรูปแบบเดิมให้มีความใกล้เคียงกับมาตรฐาน ECMA Script Fourth Edition draft specification เพื่อรองรับการเขียนโปรแกรมใน flash ที่ใหญ่และซับซ้อนมากยิ่งขึ้น โดยเพิ่มส่วนการกำหนดชนิดของตัวแปรในการประกาศ (type annotation)และเพิ่มรูปแบบการสร้าง class ให้อยู่ในรูปแบบ class –based syntax และถูกเรียกว่า Action Script 2.0 แต่ยังคงถูกแปลง (compile) ให้อยู่ในรูป Action Script1.0 และ2.0 ปะปนกันได้

ในปี 2006 ได้มีการประกาศ Action Script 3.0 เป็นครั้งแรก เพื่อรองรับการทำงานที่เข้าถึงการควบคุมการแสดงผลโดยตรง และรูปแบบการทำงานที่ซับซ้อนมากขึ้น โดยมีรูปแบบทางภาษาที่เปลี่ยนไปจากเดิมมากพอสมควร ทำให้โค้ดถูกแปลงอยู่ในรูปแบบไบต์โค้ดที่แตกต่างจากเดิม และถูกประมวลผลโดย virtual machine ที่ต่างกัน คือ AVM2 (Action Script Virtual Machine 2) ที่มาพร้อมกับ Flash Player 9 ทำให้ Flash CS3 แยก virtual machine เป็นสองส่วน คือ AVM1 สำหรับประมวลผลโค้ดในรูปแบบ Action Script 1.0 กับ Action Script 2.0 และ AVM2 สำหรับ Action Script 3.0 เท่านั้น

2.8 แอนิเมชัน

แอนิเมชัน (Animation) หมายถึง "การสร้างภาพเคลื่อนไหว" ด้วยการนำภาพนิ่งมาเรียงลำดับกัน และแสดงผลอย่างต่อเนื่องทำให้ดวงตาเห็นภาพที่มีการเคลื่อนไหวในลักษณะภาพติดตา (Persistence of Vision) เมื่อตามนุษย์มองเห็นภาพที่ฉาย อย่างต่อเนื่อง เรตินาระรักษาภาพนี้ไว้ในระยะสั้นๆ ประมาณ 1/3 วินาที หากมีภาพอื่นแทรกเข้ามาในระยะเวลาดังกล่าว สมองของมนุษย์จะเชื่อมโยงภาพทั้งสองเข้าด้วยกันทำให้เห็นเป็นภาพเคลื่อนไหว ที่มีความต่อเนื่องกัน แม้ว่าแอนิเมชันจะใช้หลักการเดียวกับวิดิโอ แต่แอนิเมชันสามารถนำไปประยุกต์ใช้กับงานต่างๆได้มากมาย เช่น งานภาพยนตร์ งานโทรทัศน์ งานพัฒนาเกมส์ งานสถาปัตย์ ซึ่งในทุกวันนี้ นิยามความเข้าใจของแอนิเมชันต่อคนทั่วไปดูจะเจาะจงเฉพาะภาพการ์ตูนหรือภาพที่เกิดจากการวาดหรือสร้างสรรค์ของมนุษย์ที่ สามารถเคลื่อนไหวได้เท่านั้น

ปิยกุล เลาวัณย์ศิริ (2532 : 931-932) ได้สรุปหลักการและคุณสมบัติของภาพยนตร์แอนิเมชันเอาไว้ดังนี้

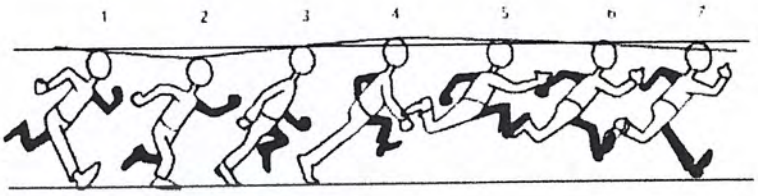
1) สามารถใช้จินตนาการได้อย่างไม่มีขอบเขต

2) สามารถอธิบายเรื่องที่ซับซ้อนและเข้าใจยากให้ง่ายขึ้น

3) ใช้อธิบายหรือแสดงความคิดเห็นที่เป็นนามธรรมให้เป็นรูปธรรมได้

4) ใช้อธิบายหรือเน้นส่วนสำคัญให้ชัดเจนและกระจ่างขึ้นได้

ดังนั้นเราจึงมีการนำแอนิเมชันมาใช้อธิบายเรื่องที่ยุ่งยากให้เห็นเป็นรูปธรรมมากขึ้น ทำให้เพิ่มความเข้าใจที่มากขึ้นของผู้เรียน มากกว่าการอ่านที่มองไม่เห็นภาพเคลื่อนไหว



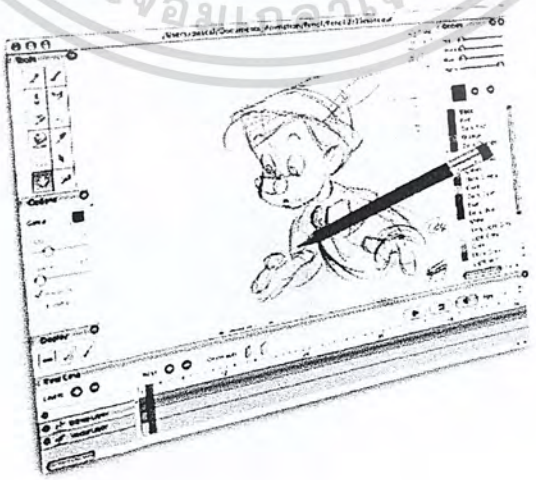
รูป 2.34 ตัวอย่างการ์ตูนแอนิเมชัน

2.8.1 ประเภทของแอนิเมชันที่พบเห็นกันได้ทั่วไป

แอนิเมชันนั้นมีด้วยกันหลายประเภท แต่ในที่นี้ขอกล่าวถึง 3 ประเภทที่พบเห็นกันทั่วไปได้แก่

2.8.1.1 Traditional Animation / Hand Drawing Animation / 2D Animation

เป็นงานแอนิเมชันสมัยแรกเริ่มมักจะใช้การวาดด้วยมือ ซึ่งงานประเภทนี้พบเห็นได้ทั่วไป ในการทำแอนิเมชันยุคแรกๆ โดยใช้เทคนิคการวาดด้วยมือทีละแผ่น แล้วใช้วิธี Flip เพื่อตรวจดูท่าทางของตัวละครที่เราได้ทำการ animate ไปแล้ว หรือที่เราเรียกกันว่า In Between (IB) โดยทั่วไปแล้วในงานแอนิเมชันแบบนี้ ถ้าเป็นงานแอนิเมชันจากฝั่งตะวันตก หรือเป็นหนังโรงจะกำหนดให้ 1 วินาที ใช้รูป 24 เฟรม แต่ถ้าเป็นพวกซีรี่ย์การ์ตูนญี่ปุ่น จะกำหนดไว้ที่ 1 วินาที ใช้รูป 12 เฟรม หรือ อาจมากกว่านั้น



รูป 2.35 ตัวอย่าง Hand Drawing Animation

2.8.1.2 Stop-motion หรือ Clay Animation

งานแอนิเมชันประเภทนี้ ผู้จัดทำแอนิเมชันจะต้องเข้าไปทำการเคลื่อนไหวโดยตรงกับโมเดล และทำการถ่ายภาพเอาไว้ทีละเฟรมๆ การทำ Stop Motion ถือเป็นเรื่องยากพอสมควร เพราะ ต้องแม่นในเรื่องของ Timing และ Pose มากๆ แม้การทำจะไม่ต้องอาศัยการวาดรูปเป็นหลัก แต่ก็ต้องทำ IB เองทั้งหมดด้วยมือ การทำ IB ในงาน Animation ประเภทนี้ ต้องอาศัยความชำนาญในการคำนวณล่วงหน้า เพราะ ถึงแม้จะมีอุปกรณ์ต่างๆ ช่วยในการ Flip แล้วก็ตาม (เช่น โปรแกรมต่างๆ ที่ช่วยในการ Capture รูป แล้ว Play ดูได้ทันที) แต่การจัดแสง และการควบคุมความต่อเนื่องระหว่างเฟรม ต้องอาศัยความรอบคอบ และความอดทนสูงมาก

รูป 2.36 ตัวอย่าง Clay Animation

2.8.1.3 Computer Animation / 2D Animation on computer / 3D Animation

เป็นงานแอนิเมชันที่มักพบกันได้บ่อยในยุคปัจจุบัน เนื่องจากการเข้าถึงโปรแกรมเป็นไปได้ง่าย และการนำหลักการแบบ 2D เข้ามาผสมผสานกับตัวโปรแกรม ทำให้เข้าใจได้ง่าย แถมยังสะดวกในการแก้ไข และแสดงผล จึงเป็นที่นิยมกันมาก ผู้จัดทำแอนิเมชันงานประเภทนี้ จึงมีเกิดขึ้นมาในยุคปัจจุบันอย่างมากมาย พร้อมด้วยความต้องการของวงการบันเทิงในยุคนี้ ที่เน้นการทำ CG Animation มากขึ้น ดูได้จากเมืองไทย ที่มีสถาบันสอนการทำแอนิเมชันเกิดขึ้นอย่างมากมาย และ Studio ที่ทำงานแอนิเมชันในบ้านเราก็มีมากขึ้น เราจะเห็นได้ว่างานต่างๆ

ในวงการบันเทิงไทย ไม่ว่าจะเป็นภาพยนตร์ ภาพยนตร์โฆษณา การ์ตูนซีรีย์ต่างๆ ล้วนล้วนแต่ มีงาน CG Animation แฝงอยู่ด้วยแทบทั้งนั้น

รูป 2.37 ตัวอย่าง 3D Animation

2.8.2 ประเภทของแอนิเมชันที่แบ่งตามลักษณะของ Flash Animation

ประเภทของการเคลื่อนไหวใน Adobe Flash แบ่งออกเป็น 2 ประเภทดังนี้

2.8.2.1 การเคลื่อนไหวชนิดภาพต่อภาพ (Frame By Frame Animation)

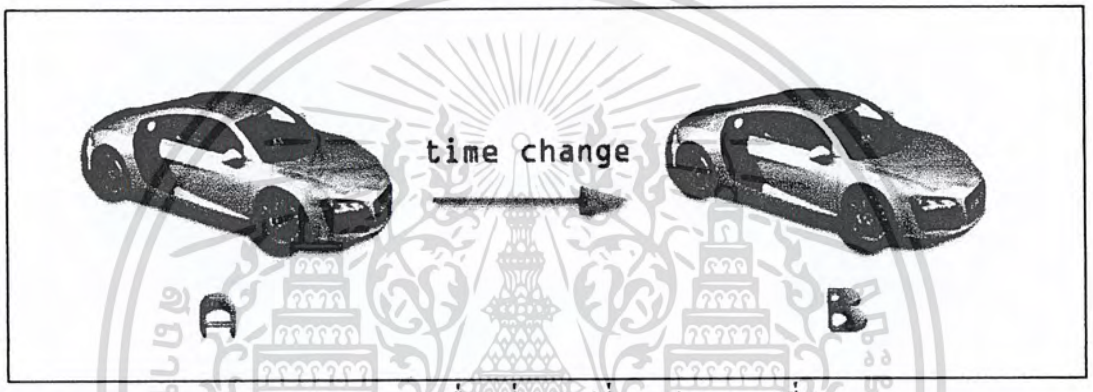
เป็นการเคลื่อนไหวชนิด ภาพที่ 1 ไปภาพที่ 2 ไปภาพที่ 3 และไปภาพสุดท้าย หรือเป็นลักษณะการเคลื่อนไหวของการ์ตูนนั่นเอง การเปลี่ยนแปลงของภาพแต่ละภาพที่เรียงอย่างต่อเนื่อง กันลักษณะนี้เหมาะกับการทำแอนิเมชันที่ซับซ้อน เช่น แอนิเมชันที่มีการเคลื่อนไหวลักษณะท่าทางมาก เป็นต้น ซึ่งจะใช้ภาพจำนวนมาก โดยแต่ละ Frame จะใส่ภาพในลักษณะท่าทางต่างๆ 1 ภาพทำให้เสียเวลา แต่จะให้ภาพที่มีการเคลื่อนไหวที่เหมือนกับความเป็นจริง

2.8.2.2 การเคลื่อนไหวชนิดกำหนดจุดเริ่มต้นและจุดสิ้นสุด (Tweened Animation)

การเคลื่อนไหวของแอนิเมชันลักษณะนี้จะมีการกำหนดจุดเริ่มต้นในการแสดงภาพเคลื่อนไหวและใช้วิธีการคำนวณของ Flash ในการแสดงภาพต่างๆ โดยที่เราไม่ต้องไปหาภาพมาเรียงต่อกัน เราสามารถแบ่งการเคลื่อนไหวลักษณะนี้เป็น 2 ลักษณะด้วยกันคือ

2.8.2.2.1 การเคลื่อนที่แบบเปลี่ยนแปลงสถานที่ (Motion Tween)

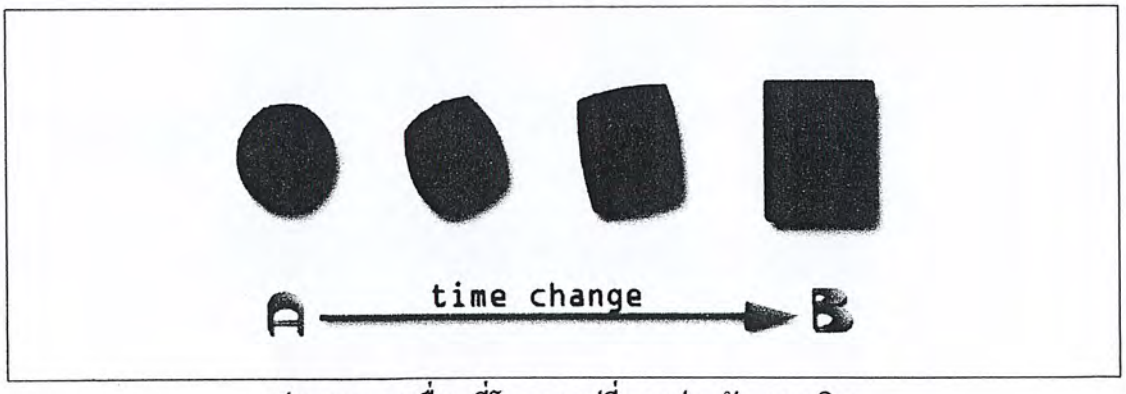
เป็นการเคลื่อนที่แบบมีการย้ายสถานที่ โดยไม่มีการเปลี่ยนแปลงรูปร่างของวัตถุ แต่สามารถเปลี่ยนสีหรือขนาดได้ คือ การเปลี่ยนแปลงจากจุด A ไปจุด B นั่นเองโดยที่ระหว่างการเคลื่อนที่จาก A ไป B สามารถมีการเปลี่ยนแปลง ขนาด เปลี่ยนสี หรือค่อยๆจางหายไป แต่ไม่สามารถเปลี่ยนจากรูปหนึ่งไปยังอีกรูปหนึ่งได้ เป็นต้น การทำแอนิเมชันลักษณะนี้ Flash Movie จะมีขนาดเล็กกว่าการเคลื่อนไหวชนิดภาพต่อภาพ (Frame by Frame Animation) เพราะใช้การคำนวณการเคลื่อนไหวแทนการใช้ภาพจริงหลายๆ ภาพมาแสดงต่อๆ กัน



รูป 2.38 การเคลื่อนที่แบบเปลี่ยนแปลงสถานที่

2.8.2.2.2 การเคลื่อนที่โดยการเปลี่ยนแปลงลักษณะเดิม (Shape Tween)

เป็นการเคลื่อนไหวโดยมีการเปลี่ยนแปลงรูปร่างหรือเปลี่ยนจากวัตถุหนึ่งไปเป็นอีกวัตถุหนึ่ง โดยเราจะใช้ Flash ให้ทำการคำนวณเปลี่ยนแปลงรูปร่างให้ ซึ่งวิธีนี้จะใช้ภาพ 2 ภาพ เป็นจุดเริ่มต้นและจุดปลาย และกำหนดให้ Flash ทำการเปลี่ยนแปลงระหว่างภาพทั้ง 2 เพื่อให้ภาพหนึ่งกลายเป็นอีกภาพหนึ่ง



รูป 2.39 การเคลื่อนที่โดยการเปลี่ยนแปลงลักษณะเดิม

บทที่ 3
การออกแบบและพัฒนา

3.1 ภาพรวมของระบบ

ในการออกแบบระบบในขั้นตอนแรก จะทำการจำลองระบบการให้บริการของเราท์เตอร์เป็นรูปแบบ M/M/1 คือมีผู้ให้บริการซึ่งในที่นี้ก็คือเราท์เตอร์เพียงแค่ 1 ตัว มีการเข้าคิวเพื่อรอรับบริการหากเราท์เตอร์ยังไม่พร้อมที่จะให้บริการ นอกจากนี้ระบบจะจำลองการเกิดเหตุการณ์แบบ Event-Drive เพื่อบอกเวลาที่จะเกิดเหตุการณ์ต่างๆขึ้นด้วย

ในการออกแบบขั้นตอนต่อไปจะเป็นการเพิ่มจำนวนผู้ให้บริการกับจำนวนของผู้รับบริการให้มีจำนวนมากขึ้น โดยสามารถที่จะกำหนดเส้นทางในการส่งแพ็คเกตจากต้นทางไปยังปลายทางได้ โดยการเพิ่มให้ระบบมีการทำงานแบบโพรโทคอลทีซีพีคือจะมีการทำ Connection-Oriented create_TCP_connection, Close Connection, Congestion Control, Flow Control, หมายเลขของแพ็คเกต เป็นต้น อีกทั้งยังสามารถส่งจากต้นทางไปหาปลายทางได้หลายคู่ โดยผ่านเราท์เตอร์ซึ่งมีการทำงานของเราท์ติ้งเทเบิ้ลอยู่

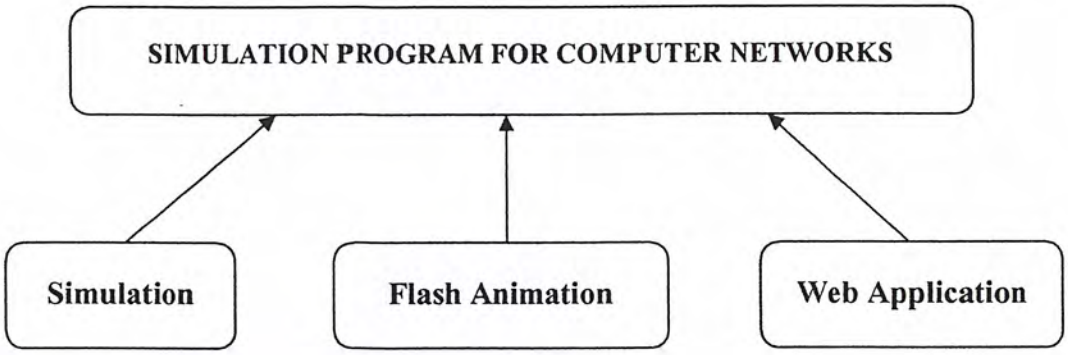
จากนั้นทำการเพิ่มส่วนที่ใช้ในการปรับค่าต่างๆที่จำเป็นสำหรับซิมมูเลชันเช่น อัตราการเข้ามาของแพ็คเกต (Arrival Rate), อัตราการให้บริการของเซิร์ฟเวอร์ (Mean_Service) และหน่วยเวลาในการทำซิมมูเลชัน (Simulation Time) และได้เก็บค่าสถิติต่างๆเพื่อใช้วัดประสิทธิภาพของระบบโดยนำเสนอในรูปแบบของกราฟและนำเสนอการทำงานออกมาในรูปแบบของอนิเมชันเพื่อให้ผู้ศึกษามีความเข้าใจมากยิ่งขึ้น

3.2 เครื่องมือที่ใช้ในการพัฒนา Simulation & Animation

1) Adobe Flash CS 5 (โดยใช้ภาษาแอคชั่นสคริปต์ในส่วนของการออกแบบส่วน Simulation สำหรับการออกแบบใส่ส่วนของผู้ใช้งานและส่วนของการทำแอนิเมชัน)
2) Adobe Dreamweaver CS4 (ใช้สำหรับการทำหน้าเว็บไซต์)
3) Adobe Photoshop CS5 (ใช้สำหรับตกแต่งรูปทำแอนิเมชันและเว็บไซต์)

3.3 ส่วนประกอบของระบบ

ส่วนประกอบของระบบจะถูกแบ่งออกเป็น 3 ส่วนการออกแบบส่วน Simulation Code, ส่วน Flash Animation และส่วนของเว็บแอพลิเคชัน จากนั้นจะนำทั้ง 3 ส่วนที่ได้พัฒนาไว้มารวมเป็น Application Program ดังรูป 3.1



รูป 3.1 ส่วนประกอบของระบบ

3.3.1 ส่วนซิมมูเลชัน (Simulation)

1) เขียนโค้ดโปรแกรมบน Adobe Flash CS5

2) ออกแบบส่วน Simulation โดยใช้ภาษาแอคชันสคริปต์และแสดงผล Application Program โดย Adobe Flash

3) ออกแบบให้รองรับการจำลองการทำงานด้วยจำนวนอุปกรณ์บนเครือข่ายที่มากขึ้น และมีการปรับค่าเริ่มต้นที่จำเป็นต่อระบบ

4) สามารถปรับตั้งค่าต่างๆที่จำเป็นต่อการใช้งานเช่น เวลาที่ทำการซิมมูเลชัน

5) มีการทำงานแบบ Connection-Oriented และการควบคุม Flow Control, Congestion Control ตามหลักการของโพรโทคอลทีซีพี

6) มีแม่แบบ (Template) การวางอุปกรณ์สำเร็จรูป เพื่อง่ายและสะดวกต่อการใช้งาน

3.3.2 ส่วนแฟลชแอนิเมชัน (Flash Animation)

1) ออกแบบโดยใช้โปรแกรม Adobe Flash CS 5

2) ออกแบบให้มีการรองรับการเลือกใช้อุปกรณ์ตามความต้องการ

3) แสดงผลการรับส่งแพ็คเกต

4) แสดงผลกราฟเพื่อวัดประสิทธิภาพของระบบ

5) สามารถแสดงรายละเอียดของแพ็คเกตในแต่ละเลเยอร์ได้

6) แสดงผลเอาท์พุทของการทำงานทั้งหมด

7) แสดงแอนิเมชันของการทำงาน

3.3.3 ส่วนการแสดงเว็บแอพลิเคชัน (Web Application)

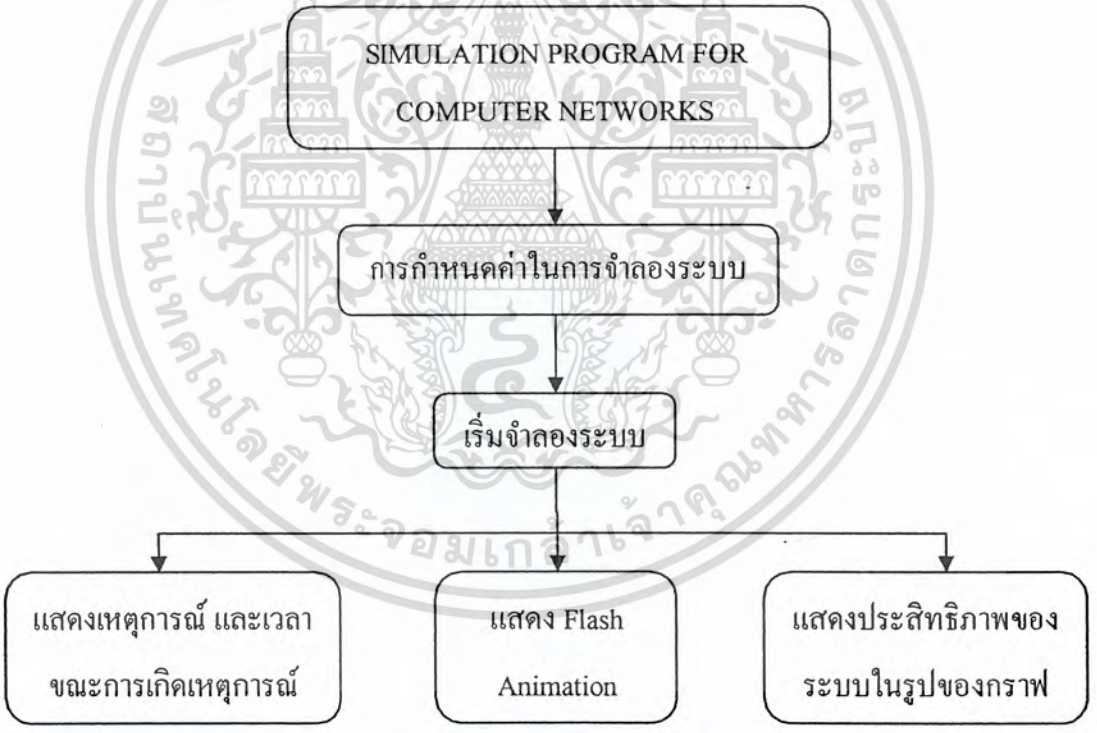
1) แสดงวิดีโอสาธิตการใช้งาน

2) วิธีการใช้

3) มีบทเรียนประกอบเพื่อศึกษาเพิ่มเติม

3.4 กระบวนการทำงานของระบบ

กระบวนการทำงานของระบบจะเริ่มที่ Application Program ซึ่งผู้ใช้งานจะต้องทำการกำหนดค่าในการจำลองระบบ ซึ่งจะเป็นค่าที่ต้องใช้ในการทำการจำลองระบบและเมื่อทำการตั้งค่าเกี่ยวกับการจำลองระบบเสร็จแล้ว ผู้ใช้งานจะต้องทำการเลือกหรือสร้างโทโปโลยีที่จะต้องการใช้ในการทำการจำลองระบบ และเลือกต้นทางและปลายทางที่ต้องการจะส่งข้อมูล เมื่อทำการกำหนดค่าเรียบร้อยแล้ว ระบบก็จะเริ่มทำการจำลองขึ้น โดยที่การแสดงเหตุการณ์จะแบ่งออกเป็น 3 ส่วน คือ ส่วนการแสดง Flash Animation ซึ่งจะแสดงผลออกมาเป็นภาพเคลื่อนไหวและแสดงการทำงานในแต่ละส่วนเพื่อให้ผู้ศึกษามีความเข้าใจมากยิ่งขึ้น ส่วนที่สองคือการแสดงเหตุการณ์ที่เกิดขึ้นของระบบ เช่น บอกว่าเหตุการณ์ที่จะเกิดขึ้นคืออะไร ที่เวลาเท่าไหร่ และส่วนที่สามคือส่วนที่แสดงประสิทธิภาพของระบบในรูปแบบของกราฟต่างๆเช่น กราฟแสดงการดรอปของแพ็คเกต, กราฟความสำเร็จของการส่งแพ็คเกต, กราฟ Queuing Delay และกราฟทรูพุต ซึ่งแสดงให้เห็นดังรูป 3.2



รูป 3.2 กระบวนการทำงานของระบบ

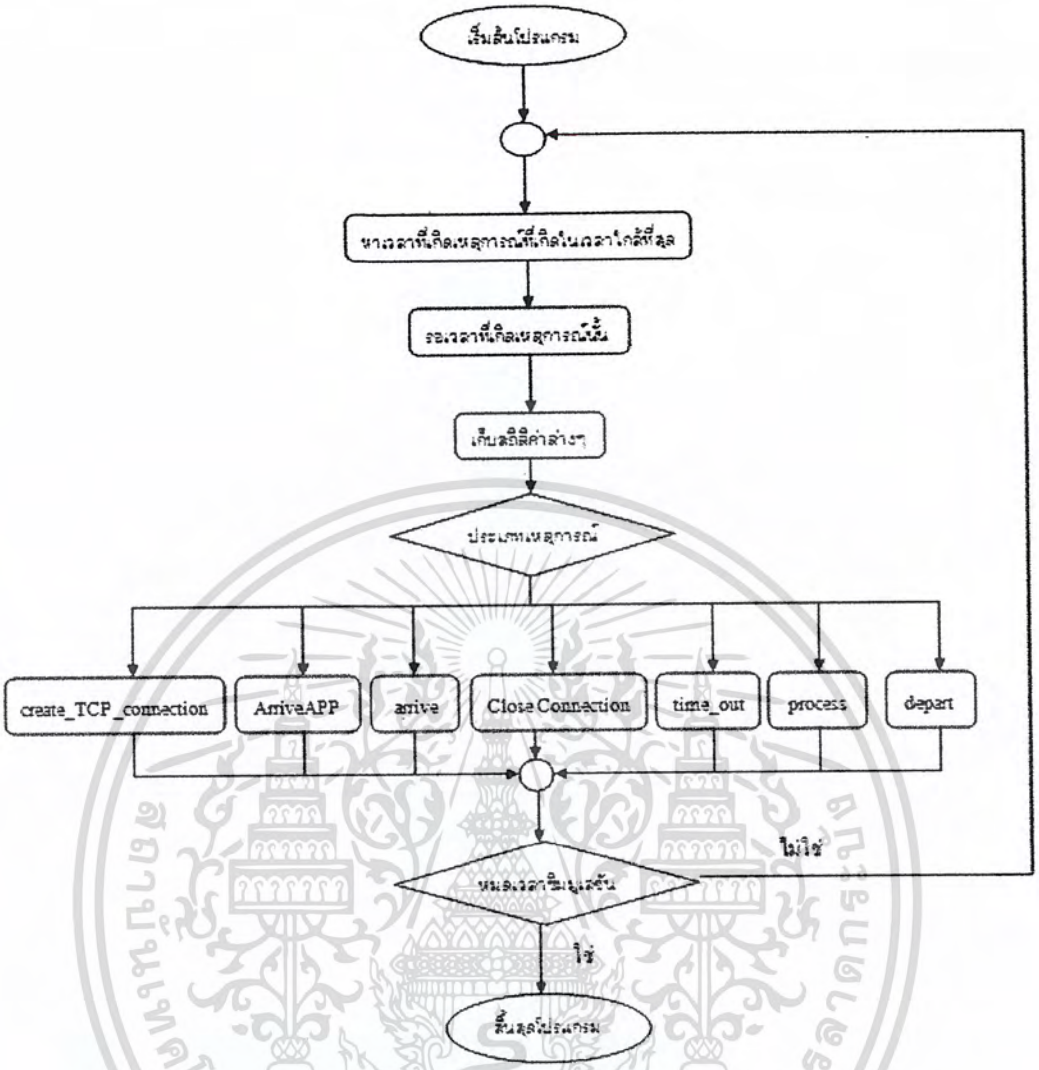
3.5 ส่วนการออกแบบส่วนซิมมูเลชัน

3.5.1 การออกแบบเหตุการณ์

ในการออกแบบส่วนซิมมูเลชัน เราได้ออกแบบอ้างอิงตามโมเดล Discrete time simulation แบบ Event driven เป็นแบบจำลองที่จะสนใจต่อเหตุการณ์ที่เกิดขึ้น ดังนั้นการเลื่อนไปของเวลาที่ใช้ในแบบจำลองจะขึ้นอยู่กับการเกิดเหตุการณ์ต่างๆ หากมีเหตุการณ์ใดเหตุการณ์หนึ่งเกิดขึ้นจึงจะมีการเลื่อนไปของเวลา หากไม่มีเหตุการณ์ใดๆ เกิดขึ้น สถานะของระบบใน discrete event simulation ก็จะไม่เปลี่ยนแปลง ผู้จัดทำได้ออกแบบของระบบให้มีเหตุการณ์อยู่ 7 แบบ คือ

1) เหตุการณ์ arrive เป็นเหตุการณ์การเข้ามาของแพ็คเกต
2) เหตุการณ์ depart เป็นเหตุการณ์การออกไปของแพ็คเกต
3) เหตุการณ์ create_TCP_connection เป็นเหตุการณ์ที่ใช้เปิดการเชื่อมต่อระหว่างต้นทางและปลายทาง
4) เหตุการณ์ close_connection เป็นเหตุการณ์ที่ใช้ปิดการเชื่อมต่อระหว่างต้นทางและปลายทาง
5) เหตุการณ์ arriveAPP เป็นเหตุการณ์ที่เกิดขึ้นที่เครื่องต้นทางเพื่อจะส่งแพ็คเกตออกไปยังปลายทาง โดยในเหตุการณ์นี้จะสร้างแพ็คเกตที่จะส่งขึ้นมา
6) เหตุการณ์ time_out เป็นเหตุการณ์ที่เกิดขึ้นเมื่อเกิด Time Out
7) เหตุการณ์ process เป็นเหตุการณ์ที่เกิดขึ้นเมื่อปลายทางได้รับแพ็คเกตและนำแพ็คเกตนั่นไปใช้งาน

ซึ่งในการทำงานของ Discrete event simulation จะทำงานเรียงตามเหตุการณ์ที่เกิดขึ้นมาก่อนหลังผู้จัดทำได้จึงจำเป็นที่จะต้องออกแบบ list ของ event เพื่อเก็บเหตุการณ์ที่เกิดขึ้นของระบบในขณะนั้น โดยผู้จัดทำได้สร้าง event list เป็นตัวแปร Array ที่ชื่อ time_next_event ขึ้นมา ซึ่งจะประกอบไปด้วยเหตุการณ์ เวลาที่เกิดเหตุการณ์ ออปเจคของอุปกรณ์และออปเจคของแพ็คเกตที่เกิดเหตุการณ์นั้น ซึ่งตัวอย่างการเกิดเหตุการณ์ต่างๆสามารถเกิดได้ดังนี้



รูป 3.3 การเกิดเหตุการณ์

3.5.2 การออกแบบดีเลย์

ในการทำงานของส่วนดีเลย์ของระบบ ซึ่งในปกติดีเลย์จะมีทั้งหมด 4 ประเภทคือ

1) Processing delay ดีเลย์ที่เกิดจากการประมวลผลของเซิร์ฟเวอร์

2) Queuing delay เป็นดีเลย์ในส่วนของการเข้าคิวของแพ็คเกตเพื่อรอการประมวลผลจากเซิร์ฟเวอร์

3) Transmission delay ดีเลย์ที่เกิดจากอัตราการส่งข้อมูล ค่านี้จะมีความสัมพันธ์กับแบนด์วิดธ์

4) Propagation delay ดีเลย์ของสื่อที่ใช้ส่งข้อมูล เป็นคุณสมบัติเฉพาะตัวของสื่อนั้นๆ ซึ่งค่าดีเลย์คำนวณได้จากอัตราส่วนระหว่างระยะทางกับความเร็วภายในสื่อ

ผู้จัดทำได้ออกแบบให้มีค่าของ Transmission delay ซึ่งค่าดีเลย์คำนวณได้จากอัตราส่วนระหว่างขนาดของแพ็คเกตกับ Bandwidth ซึ่งส่วนนี้ฟังก์ชันที่เกิดเหตุการณ์ depart จะทำ

การคำนวณขึ้นซึ่งค่าที่ได้จะเป็นเหตุการณ์ arrive ของเครื่องถัดไป ส่วน queing delay เราคำนวณหาค่าได้จากเวลาที่ packet process ลบด้วยเวลาที่แพ็คเกตเข้ามา ซึ่งจะเกิดขึ้นเมื่อเซิร์ฟเวอร์ยังไม่พร้อมให้บริการกับแพ็คเกตนั้นซึ่งเราจะเก็บแพ็คเกตที่ยังไม่พร้อมให้บริการไว้ในคิวของเซิร์ฟเวอร์ซึ่งมีชื่อว่า time_arrive ซึ่งในนี้จะประกอบไปด้วย เวลาที่แพ็คเกตเข้ามาและออปเจคของแพ็คเกตส่วน processing delay , propagation delay เราสมมติว่าให้มีค่าน้อยมาก

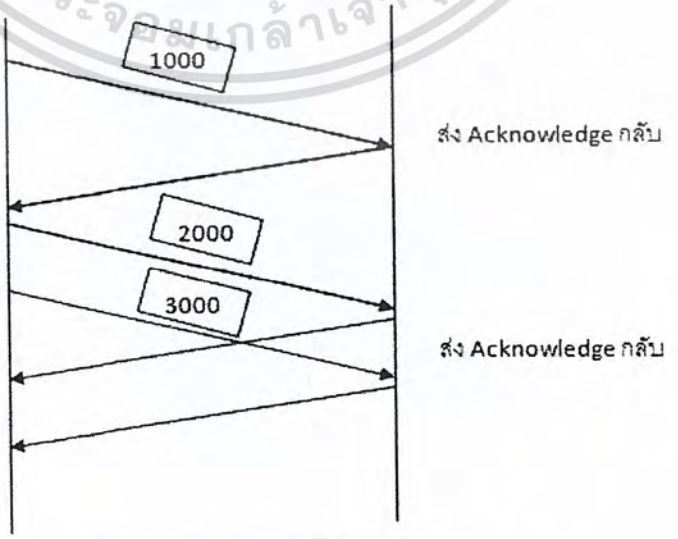
3.5.3 การออกแบบขนาดและหมายเลขของแพ็คเกต

ออกแบบให้ภายในแพ็คเกตเก็บค่าไอพีแอดเดรสของเครื่องต้นทางและเครื่องปลายทางเอาไว้และให้แพ็คเกตมีขนาด 1000 ไบต์ และไม่ได้อิงตามประเภทแอพลิเคชัน เพื่อให้ผู้ศึกษาสามารถศึกษาและทำความเข้าใจได้ง่ายและกำหนดให้ขนาดของดาต้าในชั้นแอพลิเคชันมีขนาด 40000 ไบต์ ซึ่งหมายความว่าการส่งดาต้าครั้งนี้จะส่งแพ็คเกตทั้งหมด 40 แพ็คเกต โดยหมายเลขของแพ็คเกตเป็นไปตามขนาดของแพ็คเกต เช่น เริ่มต้นที่หมาย 0 ซึ่งหมายความว่าแพ็คเกตนี้เก็บข้อมูล 0-999 ซึ่งรวมแล้วมีขนาด 1000 ไบต์ไว้ ซึ่งแพ็คเกตต่อไปก็จะมีหมายเลขเท่ากับ 1000 เป็นต้น

ส่วนการออกแบบหมายเลขแพ็คเกต Acknowledge เรากำหนดให้ปลายทางสร้างแพ็คเกต Acknowledge ที่มีขนาดของแพ็คเกตเท่ากับ 1 ไบต์ส่งกลับมา

3.5.4 การออกแบบการทำงานการส่งแพ็คเกต

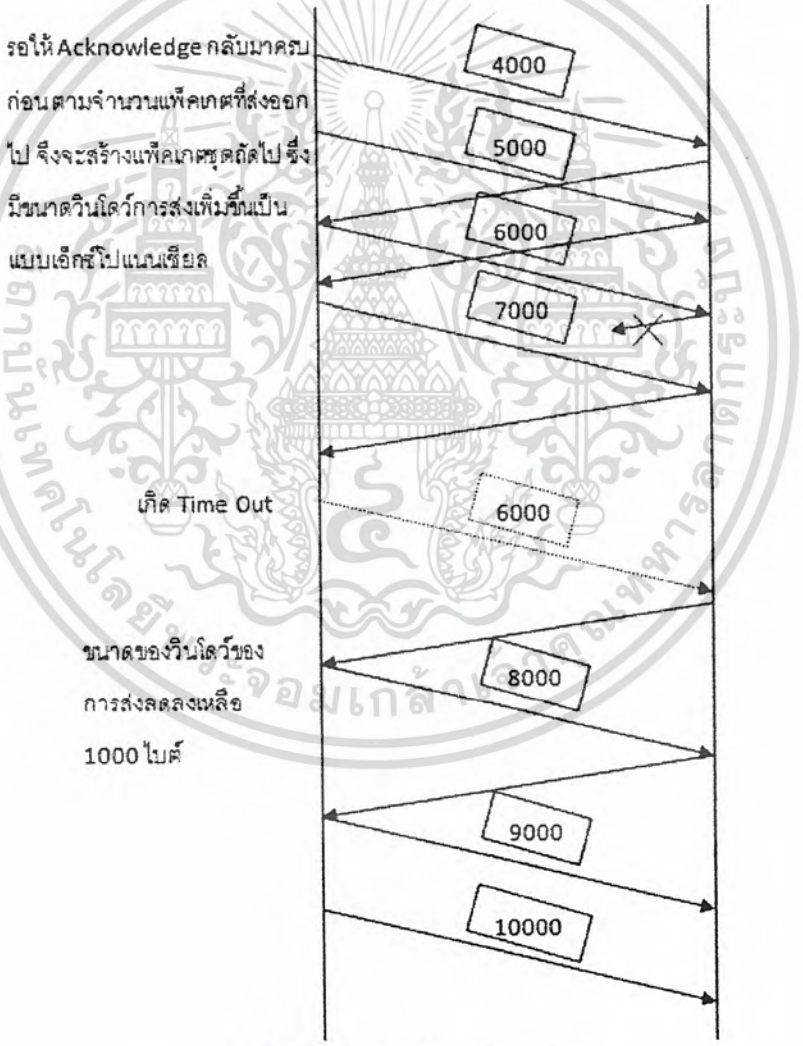
ออกแบบการทำงานการส่งแพ็คเกต โดยเมื่อต้นทางส่งไปยังปลายทาง ปลายทางจะส่งแพ็คเกต Acknowledge กลับมาทุกครั้ง เพื่อให้ง่ายต่อการศึกษาและทำความเข้าใจและการส่งข้อมูลจะส่งจากต้นทางไปยังปลายทางเท่านั้น ไม่มีการส่งข้อมูลจากปลายทางมายังต้นทาง ซึ่งก็คือปลายทางมีหน้าที่ส่ง Acknowledge กลับอย่างเดียว



รูป 3.4 การทำงานการส่งแพ็คเกต

3.5.5 การออกแบบ Congestion Control

ออกแบบให้ระบบมีการทำงานของ Congestion Control เป็นการควบคุมการส่งแพ็คเกตโดยดูความหนาแน่นของเครือข่าย เริ่มต้นให้ขนาดของวินโดว์ของการส่ง (Sender Window) มีค่าเท่ากับ 1000 ไบต์และค่า Threshold เริ่มต้นให้มีขนาด 64 กิโลไบต์ เมื่อต้นทางส่งแพ็คเกตสำเร็จก็จะเพิ่มขนาดของวินโดว์ของการส่งขึ้น โดยอัตราการเพิ่มนั้นจะเป็นแบบเอ็กซ์โปแนนเชียลและเมื่อถึงค่า Threshold จะเพิ่มขนาดของวินโดว์การส่งขึ้นทีละ 1000 ไบต์ จนกว่าจะเกิด Time Out เมื่อเกิด Time Out จะลดค่าของขนาดของวินโดว์ของการส่งลงเหลือ1000 ไบต์ใหม่ และค่า Threshold จะลดลงเหลือครึ่งหนึ่งของขนาดวินโดว์การส่ง ณ ขณะที่เกิด Time Out ดังรูปต่อไปนี้



รูป 3.5 Congestion Control

3.5.6 การออกแบบ Flow Control

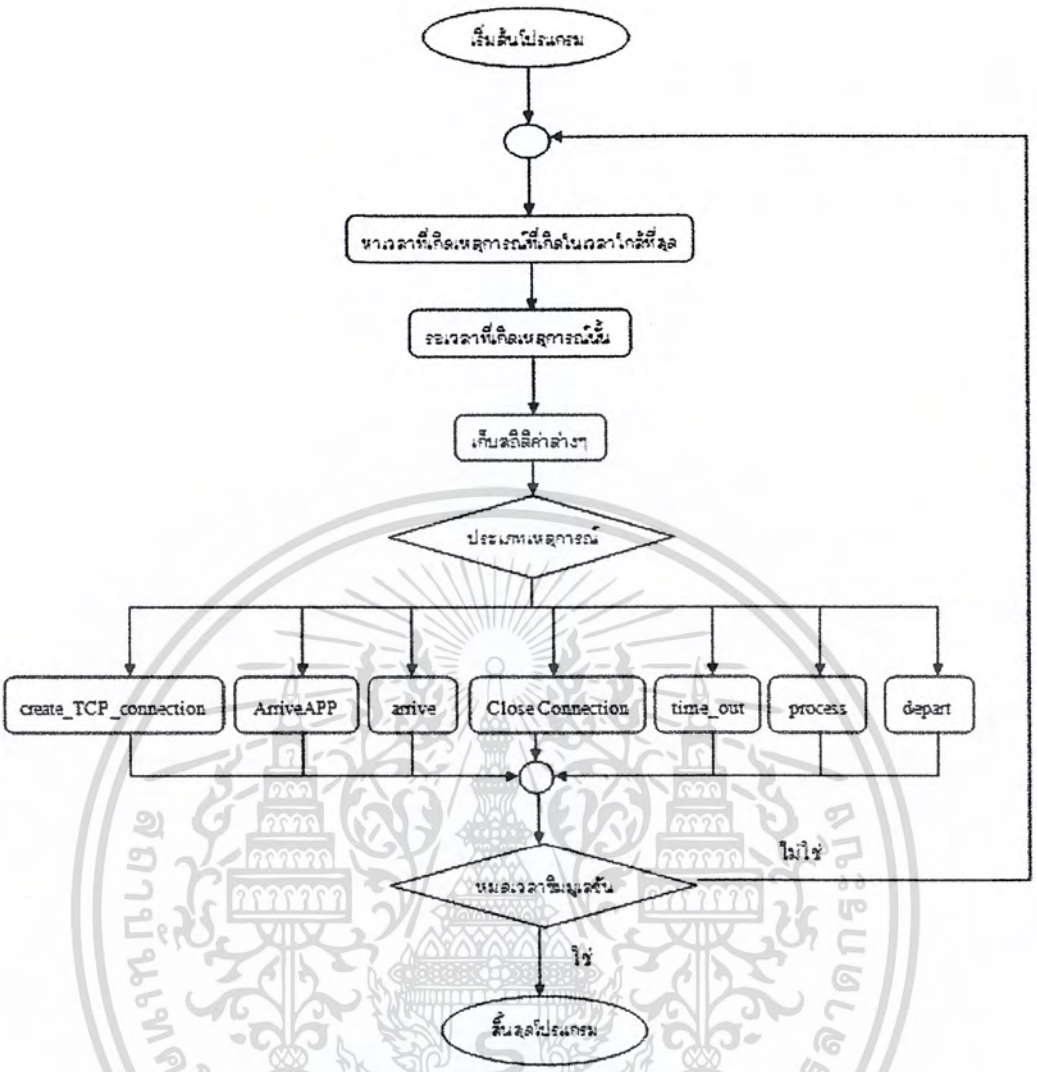
ออกแบบให้ระบบมีการทำงานของ Flow Control เป็นการควบคุมการส่งแพ็คเกตโดยดูสถานะของเครื่องปลายทางว่าพร้อมที่จะรับแพ็คเกตเท่าไร โดยกำหนดจากขนาดของวินโดว์ของการรับ (Receive Window) ของปลายทาง หากวินโดว์ของการรับเหลือพื้นที่มาก ต้นทางก็สามารถที่จะส่งแพ็คเกตไปได้มากตาม แต่หากขนาดของวินโดว์ของการรับเต็มต้นทางจะส่งแพ็คเกตออกไปทีละ 1 ไบต์จนกว่าวินโดว์ของการรับของปลายทางจะพร้อมรับ ซึ่งต้นทางจะรู้ได้ว่าปลายทางเหลือขนาดวินโดว์ของการรับเท่าไรได้จากการบอกผ่านมาในแพ็คเกตก่อนหน้า จะเห็นได้ว่าการทำงานของ Flow Control และ Congestion Control มีความสัมพันธ์กัน หากต้นทางเห็นว่าขนาดของวินโดว์ของการส่งมีมากก็ใช่ว่าจะสามารถส่งได้ตามขนาดของวินโดว์การส่ง แต่จำเป็นที่จะต้องดูวินโดว์การรับของปลายทางด้วย

3.5.7 การทำงานของฟังก์ชันในระบบ

1) timing() คือฟังก์ชันที่ทำการเรียงลำดับค่าในตัวแปรอาเรย์ซึ่งเก็บค่าของเหตุการณ์ที่จะเกิดขึ้น โดยจะเอาเหตุการณ์ที่จะเกิดขึ้นเร็วที่สุดมาไว้เป็นตัวแรก
2) update_time_avg_stats() คือฟังก์ชันที่ใช้คำนวณเวลา delay ของทั้งระบบ เพื่อนำไปคำนวณประสิทธิภาพของระบบ
3) arrive() คือฟังก์ชันที่นำแพ็คเกตที่เข้ามาไปประมวลผลหรือเข้าคิวรอ
4) process() คือฟังก์ชันที่นำแพ็คเกตเข้ามาเพื่อประมวณผลต่อไป
5) depart() คือฟังก์ชันที่ส่งแพ็คเกตออกไป โดยจะดูเส้นทางจากฟังก์ชัน routing_table()
6) Queue () คือฟังก์ชันที่ทำหน้าที่จัดการและบริหารแพ็คเกตที่อยู่ในคิว
7) routing_table() คือฟังก์ชันที่นำแพ็คเกตเข้ามาตรวจสอบเส้นทางต่อไป ซึ่งในที่นี้เราได้ออกแบบเราท์เตอร์ให้มี routing protocol แบบ RIP1
8) Timeout() เป็นฟังก์ชันที่จัดการกับระบบเมื่อเกิด time out ขึ้น

3.5.8 กระบวนการซิมมูเลชัน

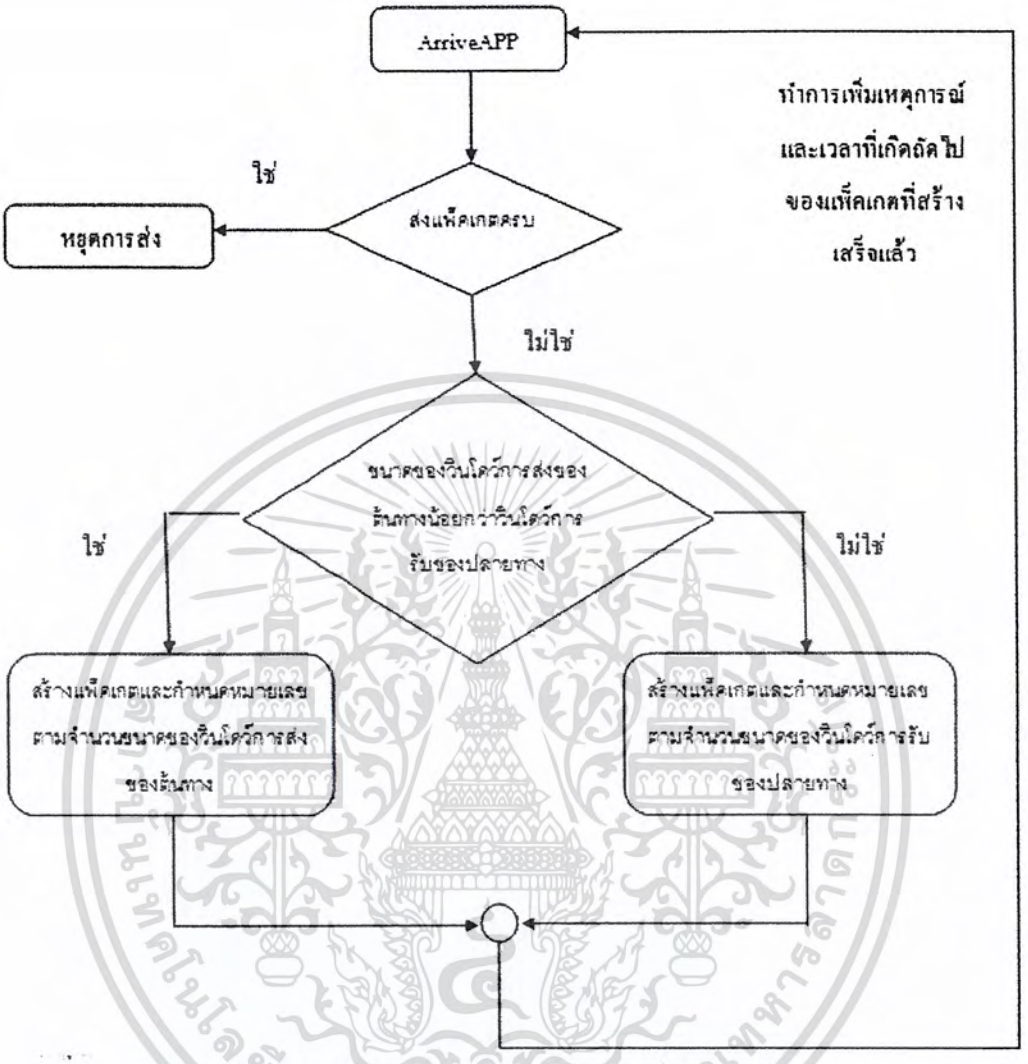
แสดงแผนผังทำงานของในส่วนของซิมมูเลชัน แสดงดังรูป 3.6



รูป 3.6 กระบวนการซิมมูเลชัน

1) เริ่มต้นจะทำการหาเวลาที่จะเกิดเหตุการณ์ที่ใกล้ที่สุด เพื่อนำมาประมวลผลเหตุการณ์นั้น
2) รอเวลาที่จะเกิดเหตุการณ์ที่เลือกเอาไว้
3) ทำการเก็บค่าสถิติข้อมูลต่างๆของระบบเช่นค่า Queuing delay เป็นต้น
4) เมื่อถึงเวลาที่เกิดเหตุการณ์จะเลือกทำขั้นตอนต่างๆตามเหตุการณ์นั้นๆ
5) เมื่อทำขั้นตอนต่างๆตามเหตุการณ์เสร็จแล้ว จะทำการตรวจสอบว่าหมดเวลาซิมมูเลชันแล้วหรือยัง หากยังไม่หมดจะทำการหาเหตุการณ์ที่เกิดในเวลาที่ใกล้เคียงกับเวลาปัจจุบันมากที่สุดแล้วดำเนินการแบบเดิมต่อไป แต่หากหมดเวลาซิมมูเลชันก็จะเป็นการสิ้นสุดโปรแกรม

3.5.8.1 กระบวนการทำงานของเหตุการณ์ ArriveAPP

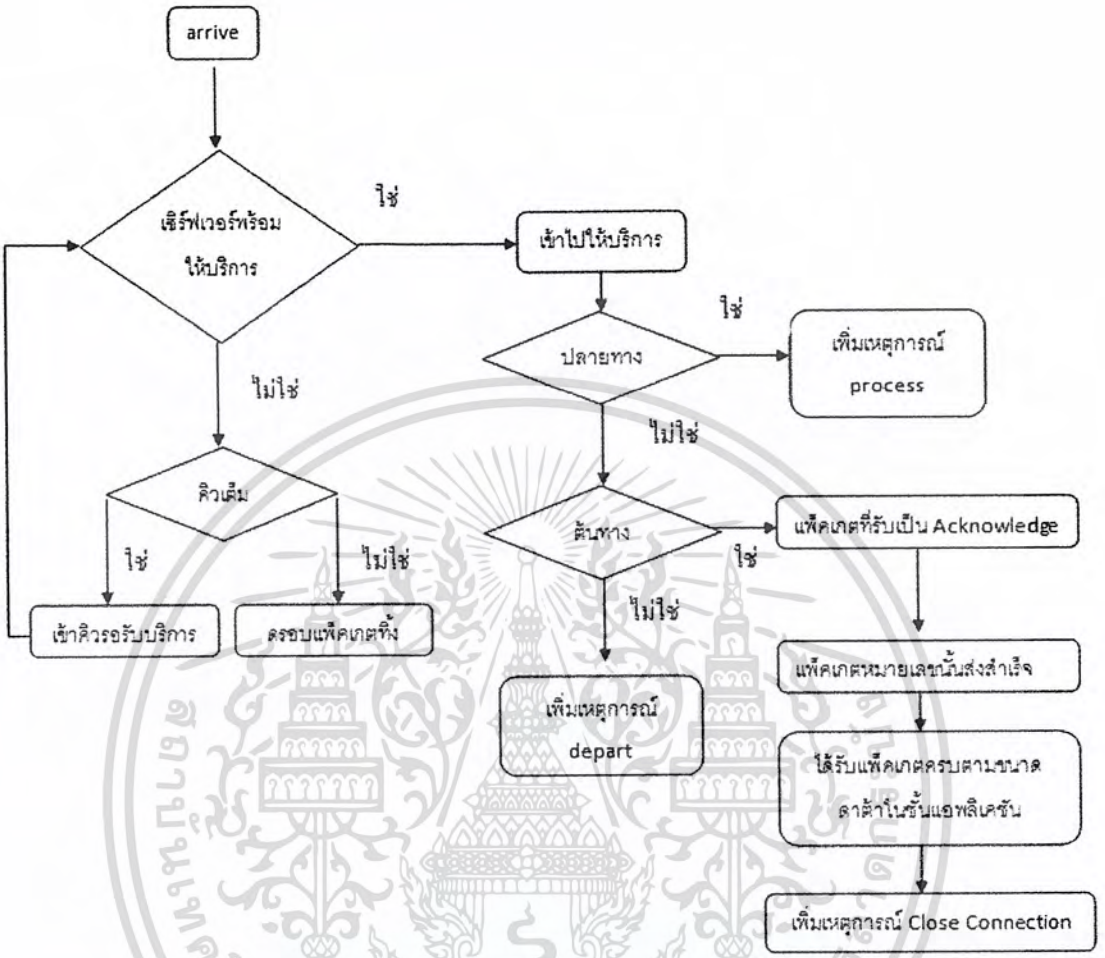


รูป 3.7 กระบวนการทำงานของเหตุการณ์ ArriveAPP

จากรูป 3.7 เป็นกระบวนการทำงานของเหตุการณ์ ArriveAPP ซึ่งเกิดขึ้นเมื่อต้นทางต้องการส่งแพ็คเกตไปยังเครื่องปลายทาง ในขั้นตอนแรกจะทำการตรวจสอบว่าส่งแพ็คเกตครบตามจำนวนทั้งหมดตามขนาดของดาต้าชั้นแอพลิเคชันหรือไม่เช่น ดาต้ามีขนาด 40000 ไบต์ จะทำการส่งแพ็คเกตทั้งหมด 40 ครั้ง หากครบแล้วจะทำการหยุดการสร้างแพ็คเกตและการส่ง

จากนั้นจะทำการตรวจสอบว่าขนาดนี้วินโดว์ของการส่งในต้นทางมีขนาดเท่าไหร่และเปรียบเทียบกับวินโดว์ฝั่งรับของปลายทางว่าในขณะนี้ปลายทางสามารถรับแพ็คเกตได้จำนวนเท่าไหร่แล้วหากค่าใดมีค่าน้อยกว่าจะทำการสร้างแพ็คเกตตามขนาดของวินโดว์นั้นๆ ในการสร้างแพ็คเกตนั้นจะกำหนดหมายเลขลำดับของแพ็คเกตด้วย แล้วจะทำการเพิ่มเหตุการณ์ของแพ็คเกตที่สร้างขึ้นใหม่ลงไปเพื่อรอให้เวลาของการซิมมูเลชันเลื่อนมาถึง

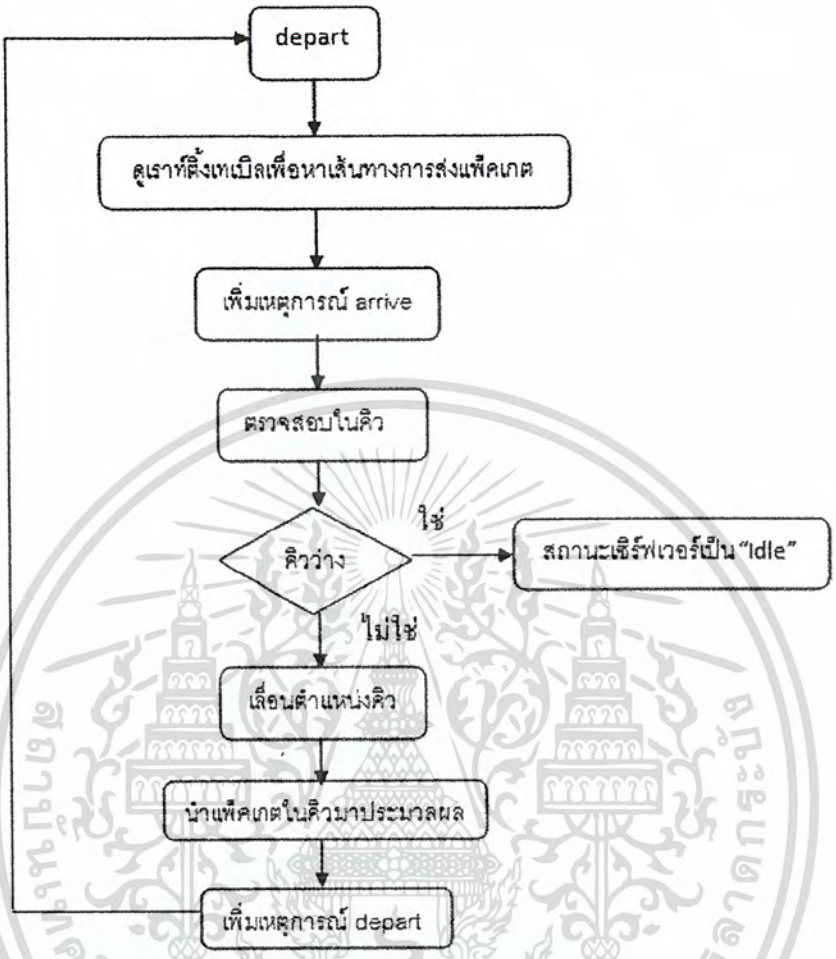
3.5.8.2 กระบวนการทำงานของเหตุการณ์ arrive



รูป 3.8 กระบวนการทำงานของเหตุการณ์ arrive

จากรูป 3.8 แสดงการทำงานของเหตุการณ์ arrive ซึ่งเกิดขึ้นเมื่อมีแพ็คเกตเข้ามายังอุปกรณ์เราท์เตอร์ เมื่อเข้ามาแล้วจะทำการตรวจสอบว่าเราท์เตอร์สามารถให้บริการแก่แพ็คเกตได้ตอนนี้หรือไม่ หากไม่พร้อมจะต้องเข้าคิวรอรับบริการแต่ในกรณีที่คิวเต็ม แพ็คเกตนั้นจะถูกดรอปทิ้ง หากเราท์เตอร์พร้อมที่จะให้บริการจะตรวจสอบว่าอุปกรณ์นั้นเป็นอุปกรณ์ปลายทางหรือไม่ หากใช่จะทำการเพิ่มเหตุการณ์ process เข้าไปเพื่อนำแพ็คเกตนั้นไปประมวลผลต่อไป แต่หากอุปกรณ์นั้นเป็นต้นทาง มีความเป็นไปได้เดียวคือแพ็คเกตนั้นต้องเป็น Acknowledge ที่ส่งจากปลายทางซึ่งจะแสดงให้เห็นว่าแพ็คเกตหมายเลขนั้นส่งสำเร็จและหากต้นทางได้รับแพ็คเกต Acknowledge ครบตามจำนวนขนาดของดาต้าในชั้นแอพลิเคชันก็จะทำการเพิ่มเหตุการณ์ Close Connection ลงไปเพื่อจบการเชื่อมต่อ แต่หากอุปกรณ์นั้นไม่ใช่ทั้งต้นทางและปลายทาง แสดงว่าเป็นเราท์เตอร์ระหว่างทาง ซึ่งจะทำการเพิ่มเหตุการณ์ depart ลงไปเพื่อจะส่งออกไปยังอุปกรณ์ต่อไป

3.5.8.3 กระบวนการทำงานของเหตุการณ์ depart



รูป 3.9 กระบวนการทำงานของเหตุการณ์ depart

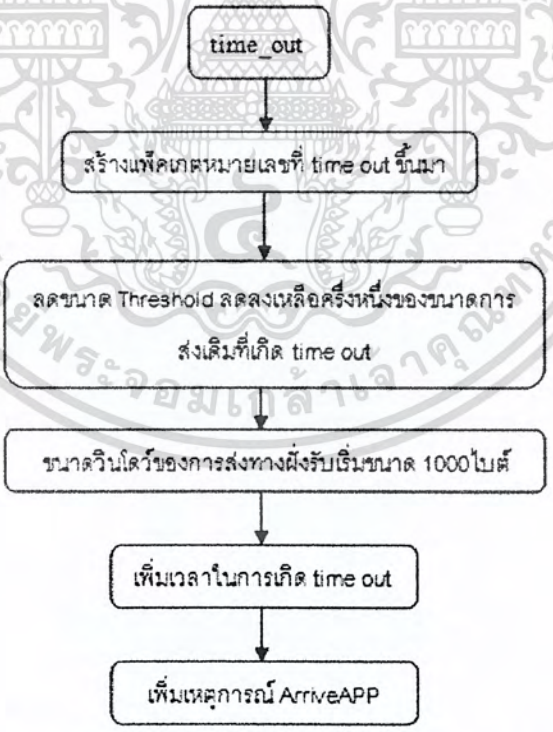
เมื่อเกิดเหตุการณ์ depart จะทำการส่งแพ็คเกตออกไปจากเครื่อง โดยจะทำการเช็คเราท์ติ้งเทเบิลของเราท์เตอร์เพื่อดูหมายเลขไอพีแอดเดรสของแพ็คเกตนั้นว่าต้องการส่งไปที่ใด จากนั้นจะทำการเพิ่มเหตุการณ์ arrive โดยเวลาที่จะเกิดเหตุการณ์ arrive คือเวลาซิมมูเลชันปัจจุบันบวกกับเวลาของ Transmission delay จากนั้นจะทำการตรวจสอบในคิวว่า คิวว่างแล้วหรือยัง หากว่างแล้วจะตั้งค่าสถานะเซิร์ฟเวอร์เป็นสถานะ Idle หากยังไม่ว่างจะทำการเลื่อนตำแหน่งคิวขึ้นมาหนึ่งลำดับ เพื่อแพ็คเกตตัวแรกในคิวมาทำการประมวลผลเพื่อกำหนดเหตุการณ์ depart ต่อไป

3.5.8.4 กระบวนการทำงานของเหตุการณ์ process

รูป 3.10 กระบวนการทำงานของเหตุการณ์ process

กระบวนการทำงานของเหตุการณ์ process จะเกิดขึ้นเมื่อปลายทางได้รับแพ็คเกตคาด้าจากต้นทาง ซึ่งปลายทางจะทำการสร้างแพ็คเกต Acknowledge เพื่อที่จะกลับไปยังต้นทาง เมื่อสร้างเสร็จจะทำการเพิ่มเหตุการณ์ depart ลงไป

3.5.8.5 กระบวนการทำงานของเหตุการณ์ time_out



รูป 3.11 กระบวนการทำงานของเหตุการณ์ time_out

เมื่อเกิดเหตุการณ์ time_out จะทำการสร้างแพ็คเกตที่เกิด time out ขึ้นมาใหม่ซึ่งมีหมายเลขเดิม และทำการลดขนาดของ Threshold ลงเหลือขนาดครึ่งหนึ่งของขนาดวินโดว์

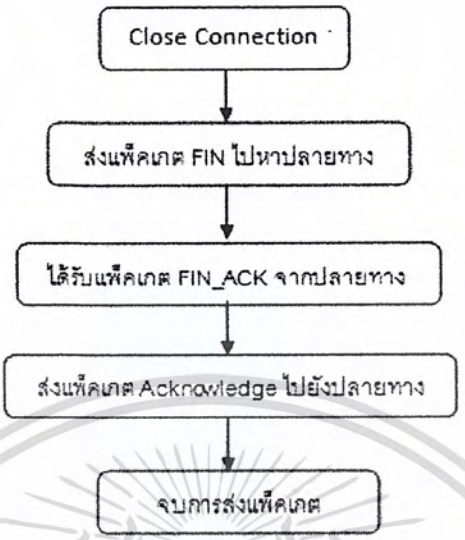
การส่ง ณ ขณะที่เกิด time out เช่น หากตอนนั้นขนาดวินโดว์การส่งกำหนดให้สามารถส่งแพ็คเกตได้ขนาด 8000 ไบต์ ซึ่งขณะนั้นสามารถส่งแพ็คเกตได้ 8 แพ็คเกตติดต่อกัน แต่ถ้าเกิด time out ขึ้นมา ในรอบถัดไปขนาดของ Threshold จะลดลงเหลือ 8000/2=4000ไบต์ และขนาดวินโดว์ของการส่งจะลดลงเหลือ 1000 ไบต์เหมือนตอนที่ส่งแพ็คเกตแรกสุด แล้วจะทำการปรับเวลาในการเกิด time out ขึ้น โดยการเพิ่มเพื่อลดการเกิด time out และสุดท้ายจะเพิ่มเหตุการณ์ ArriveAPP เพื่อรอส่งแพ็คเกตที่สร้างขึ้นใหม่

3.5.8.6 กระบวนการทำงานของเหตุการณ์ create_TCP_connection

รูป 3.12 กระบวนการทำงานของเหตุการณ์ create_TCP_connection

ในการเริ่มการเชื่อมต่อระหว่างต้นทางกับปลายทางเพื่อส่งข้อมูล ฝ่ายต้นทางจะเกิดเหตุการณ์ create_TCP_connection โดยการส่งแพ็คเกต SYN ไปหาปลายทาง เมื่อปลายทางส่งแพ็คเกต SYN_ACK กลับมา ต้นทางจะส่งแพ็คเกต ACK กลับไปยืนยันและทำการเพิ่มเหตุการณ์ ArriveAPP เพื่อสร้างแพ็คเกตที่ต้องการจะส่งต่อไป

3.5.8.7 กระบวนการทำงานของเหตุการณ์ Close Connection



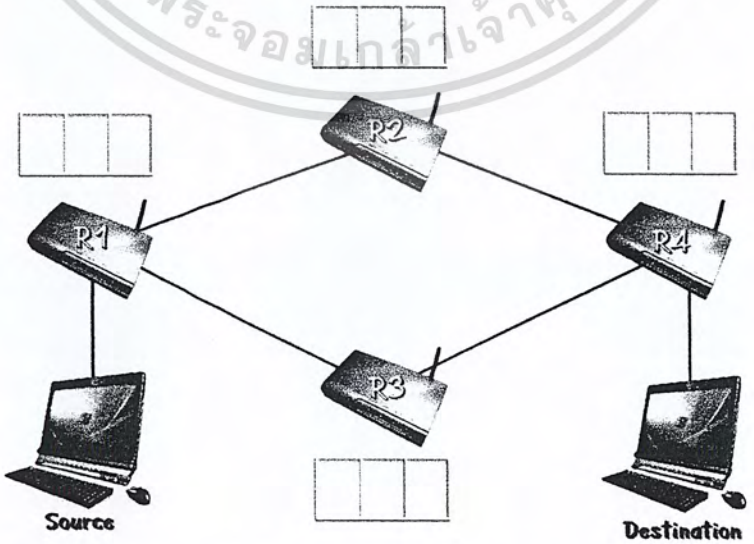
รูป 3.13 กระบวนการทำงานของเหตุการณ์ Close Connection

การทำงานคล้ายกับเหตุการณ์ create_TCP_connection เริ่มจากการส่งแพ็คเกต FIN ไปยังปลายทาง เมื่อปลายทางได้รับจะทำการส่งแพ็คเกต FIN_ACK กลับมา เมื่อต้นทางได้รับ ก็จะส่งแพ็คเกต ACK กลับไปเป็นอันจบการส่งข้อมูล

3.5.9 จำลองสถานการณ์ซิมมูเลชัน

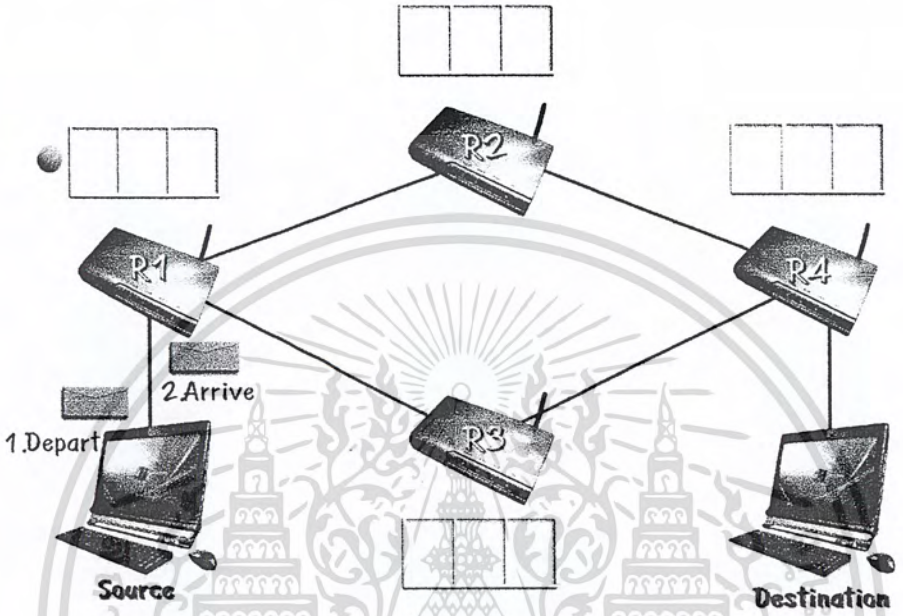
3.5.9.1 จำลองการรับส่งแพ็คเกตในกรณีไม่ต้องรอคิว

ก่อนเริ่มการทำงาน ยังไม่มีเหตุการณ์ใดๆเกิดขึ้น ดังรูป 3.14



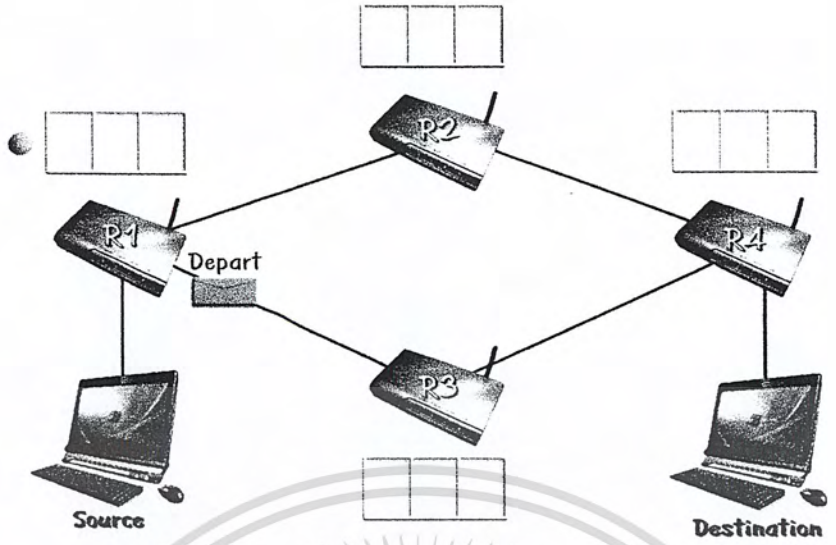
รูป 3.14 สถานะก่อนเริ่มทำงาน

โดยเมื่อระบบเริ่มทำงานจะเกิดเหตุการณ์แรกคือ depart ซึ่งเกิดที่เครื่องต้นทางทำการสร้างแพ็คเกตที่บรรจุไอพีแอดเดรสของเครื่องต้นทางและปลายทางออกมาเพื่อส่งไปยังปลายทาง ดังรูป 3.15

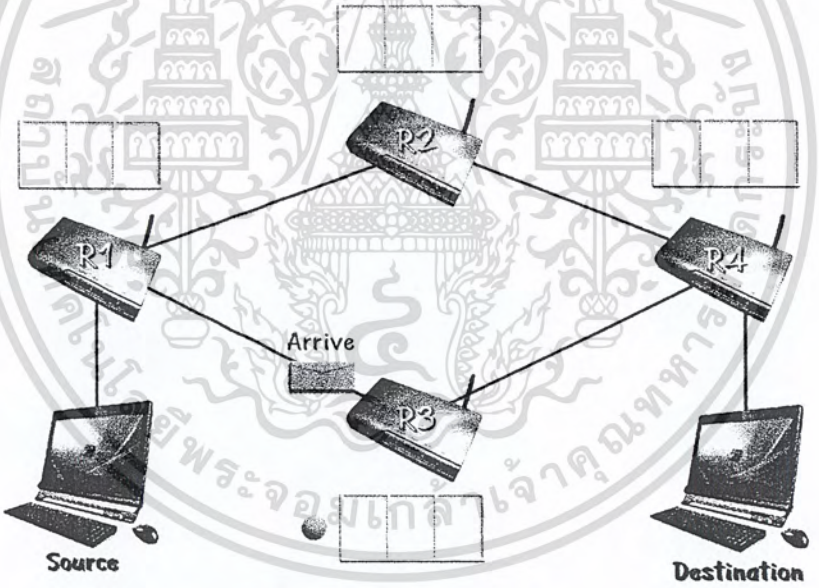


รูป 3.15 เกิดเหตุการณ์ depart จากเครื่องต้นทางและเหตุการณ์ arrive ที่เครื่องปลายทาง

เราท์เตอร์ R1 จะทำการตรวจสอบว่าสถานะของเซิร์ฟเวอร์เป็น idle หรือ busy ในกรณีนี้สถานะเป็น idle ซึ่งก็คือสามารถนำแพ็คเกตนั่นมาโพรเซสได้ทันทีจากนั้นเปลี่ยนสถานะของเซิร์ฟเวอร์เป็น busy เมื่อโพรเซสจะนำแพ็คเกตไปตรวจสอบกับ routing table จะเกิดเหตุการณ์ depart ออกไปดังรูป 3.16 และเกิดเหตุการณ์ arrive ที่เครื่องปลายทางดังรูป 3.17

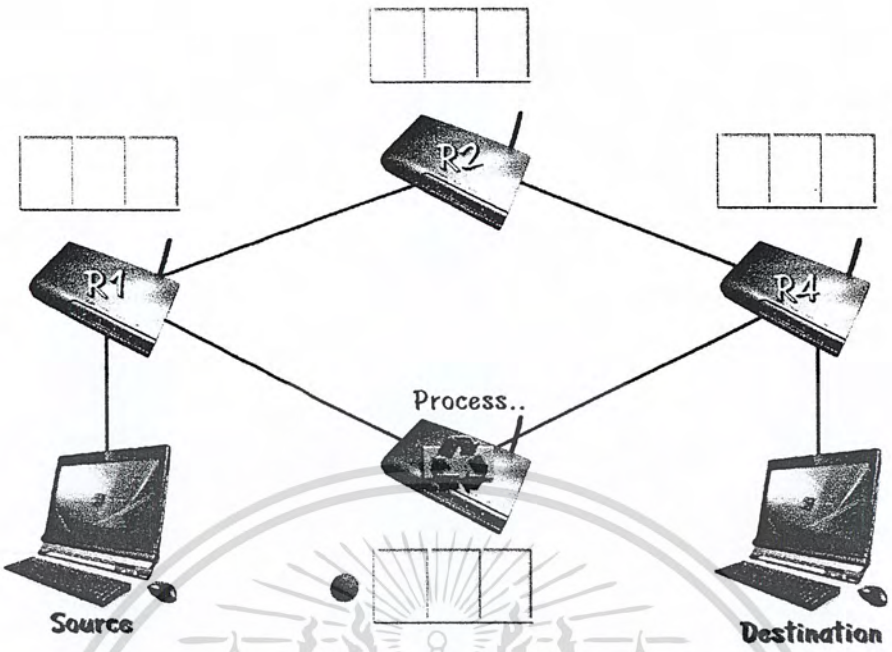


รูป 3.16 เกิดเหตุการณ์ depart ของเราท์เตอร์ R1



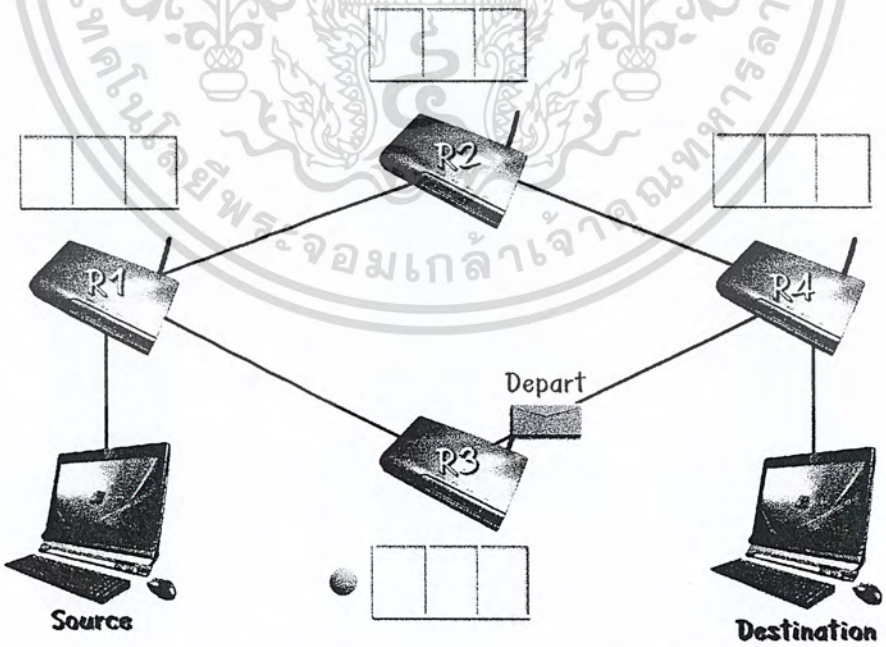
รูป 3.17 เราท์เตอร์ R3 เกิดเหตุการณ์ arrive เข้ามา

เราท์เตอร์ R3 จะทำการตรวจสอบว่าสถานะของเซิร์ฟเวอร์เป็น idle หรือ busy ในกรณีนี้สถานะเป็น idle ซึ่งก็คือสามารถนำแพ็คเกตนั่นมาโพรเซสได้ทันที และเปลี่ยนสถานะของเซิร์ฟเวอร์เป็น busy ดังรูป 3.18

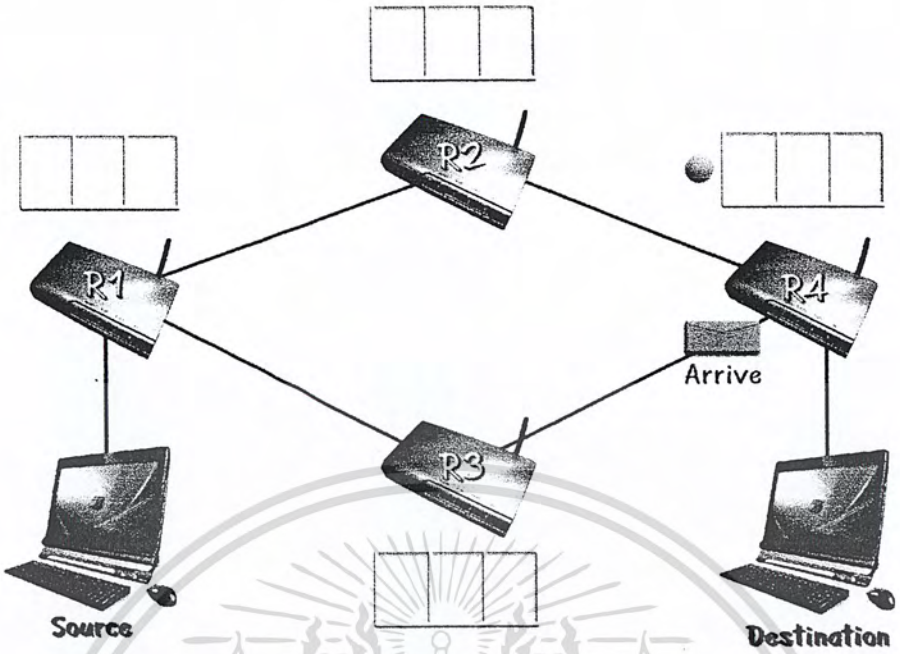


รูป 3.18 เราท์เตอร์ R3 ทำการโพรเซสแพ็คเกตที่เข้ามา

เมื่อทำการโพรเซสแล้ว เราท์เตอร์ R3 จะเกิดเหตุการณ์ depart ออกไปดังรูป 3.19 และเกิดเหตุการณ์ arrive ที่เครื่องปลายทาง ดังรูป 3.20

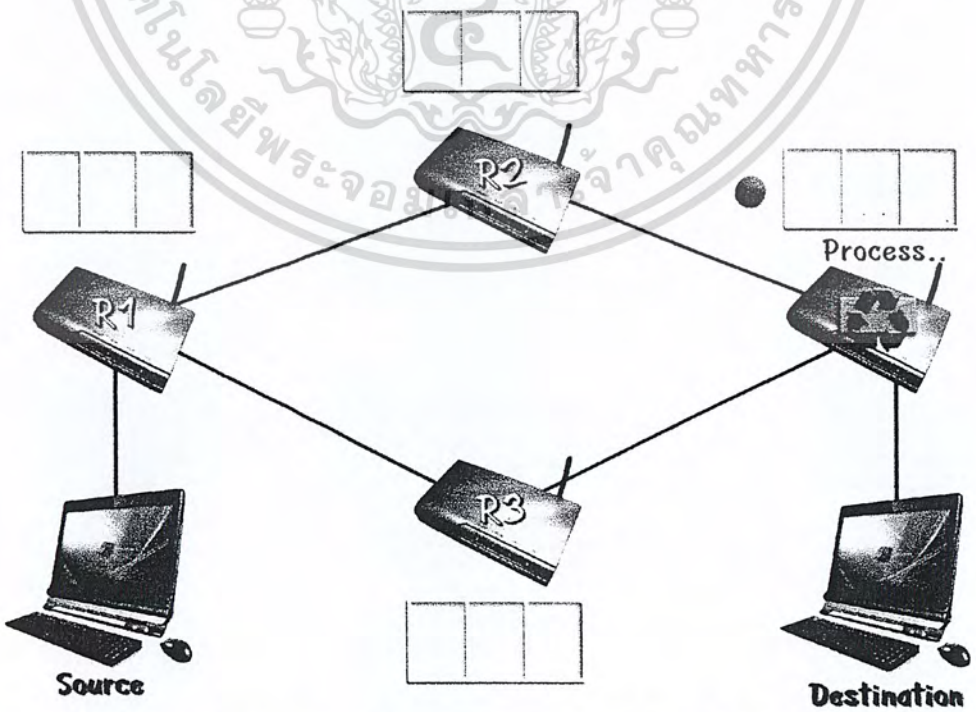


รูป 3.19 เกิดเหตุการณ์ depart ของเราท์เตอร์ R3



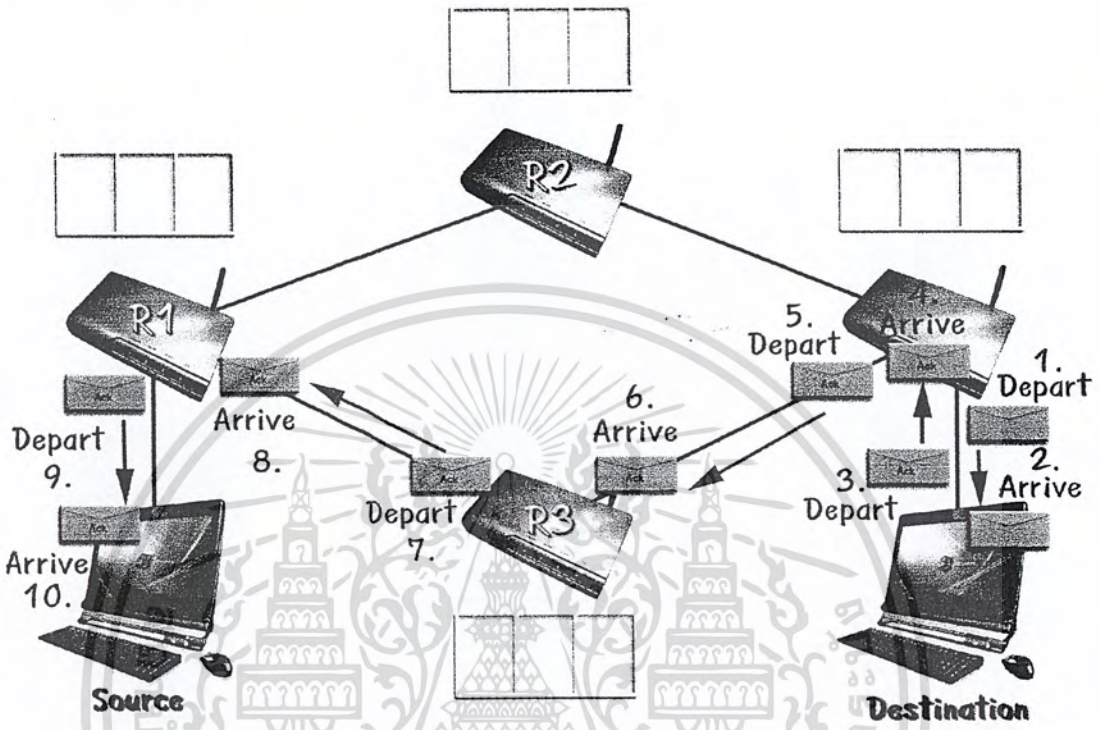
รูป 3.20 เกิดเหตุการณ์ arrive ของเราท์เตอร์ R4

เราท์เตอร์ R4 จะทำการตรวจสอบว่าสถานะของเซิร์ฟเวอร์เป็น idle หรือ busy ในกรณีนี้สถานะเป็น idle ซึ่งก็คือสามารถนำแพ็คเกตนั่นมาโพรเซสได้ทันที และเปลี่ยนสถานะของเซิร์ฟเวอร์เป็น busy ดังรูป 3.21



รูป 3.21 เราท์เตอร์ R4 ทำการโพรเซสแพ็คเกตที่เข้ามา

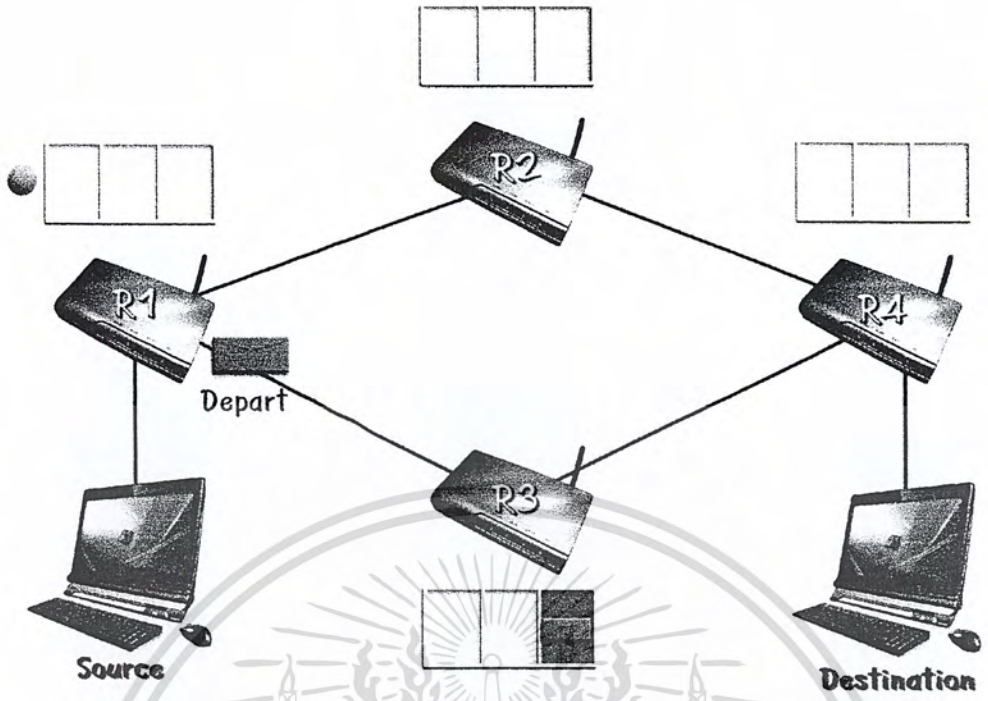
เมื่อทำการโพรเซสแล้ว เราท์เตอร์ R4 จะเกิดเหตุการณ์ depart ออกไปเกิดเหตุการณ์ arrive ที่เครื่องปลายทาง จากนั้นเป้าหมายปลายทางจะทำการสร้างแพ็คเกต ACK ส่งกลับตามเส้นทางเดิมดังรูป



รูป 3.22 เครื่องปลายทางส่งแพ็คเกต ACK กลับ

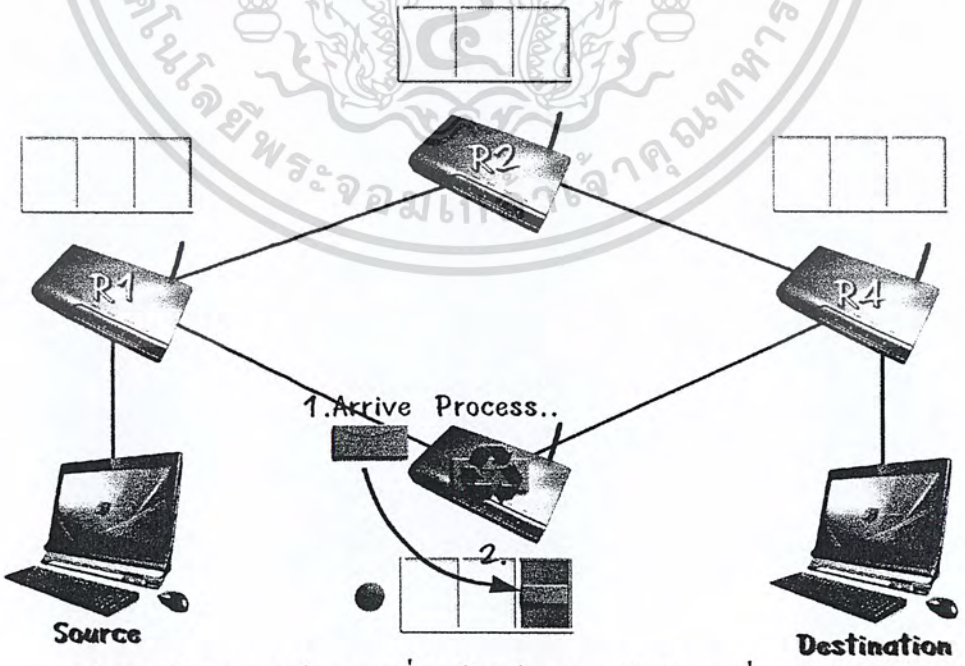
3.5.9.2 จำลองการรับส่งแพ็คเกตในกรณีที่ต้องรอคิว

เครื่องต้นทางส่งแพ็คเกตไปยังเราท์เตอร์ R1 ซึ่ง idle อยู่จึงจะทำการโพสเซสได้เลย และเกิดเหตุการณ์ depart จากเราท์เตอร์ต้นทาง R1 ไปยังเครื่องปลายทาง และมีการเราท์ติ้งเส้นทางไปยังเราท์เตอร์ R3 ก่อน ตามเราท์ติ้งเทเบิ้ล และจะเห็นว่า ในคิวของเราท์เตอร์ R3 มีแพ็คเกตรออยู่ ดังรูป 3.23



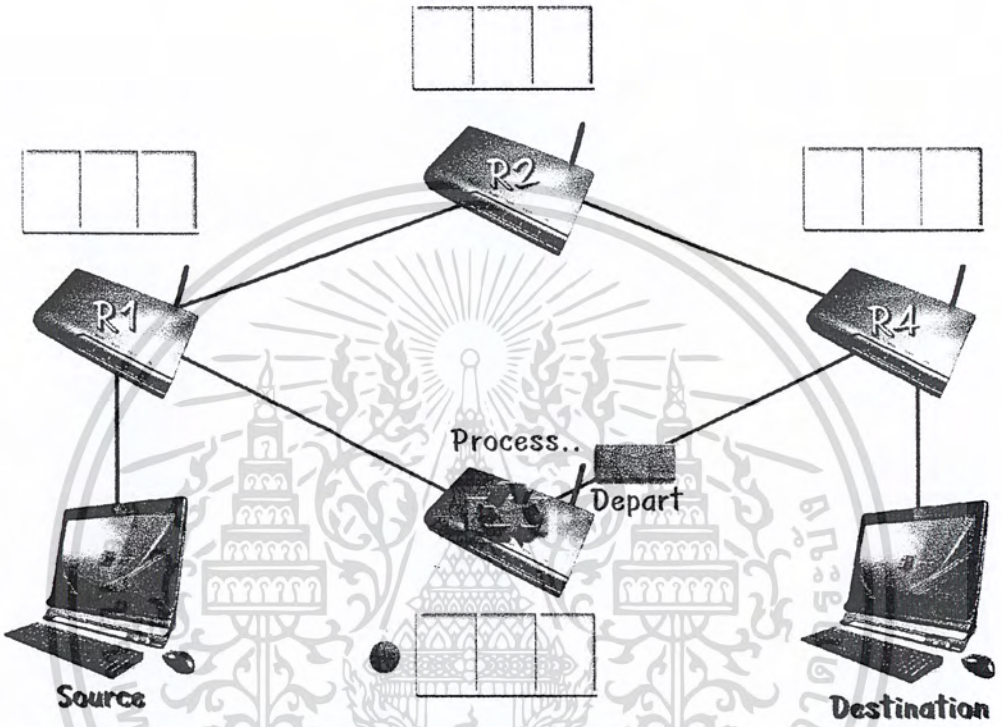
รูป 3.23 เกิดเหตุการณ์ depart จากเราท์เตอร์ R1

จากรูป 3.24 เมื่อเกิดเหตุการณ์ arrive ณ เราท์เตอร์ R3 สถานะของอุปกรณ์ เป็น busy (วงกลมสีแดงจากรูป 3.24) เนื่องจากมีการให้บริการแพ็คเกตที่เข้ามาก่อนหน้านี้ (สีฟ้า) อยู่แพ็คเกตที่เราพิจารณา (สีแดง) จึงต้องเข้าไปรอในคิว



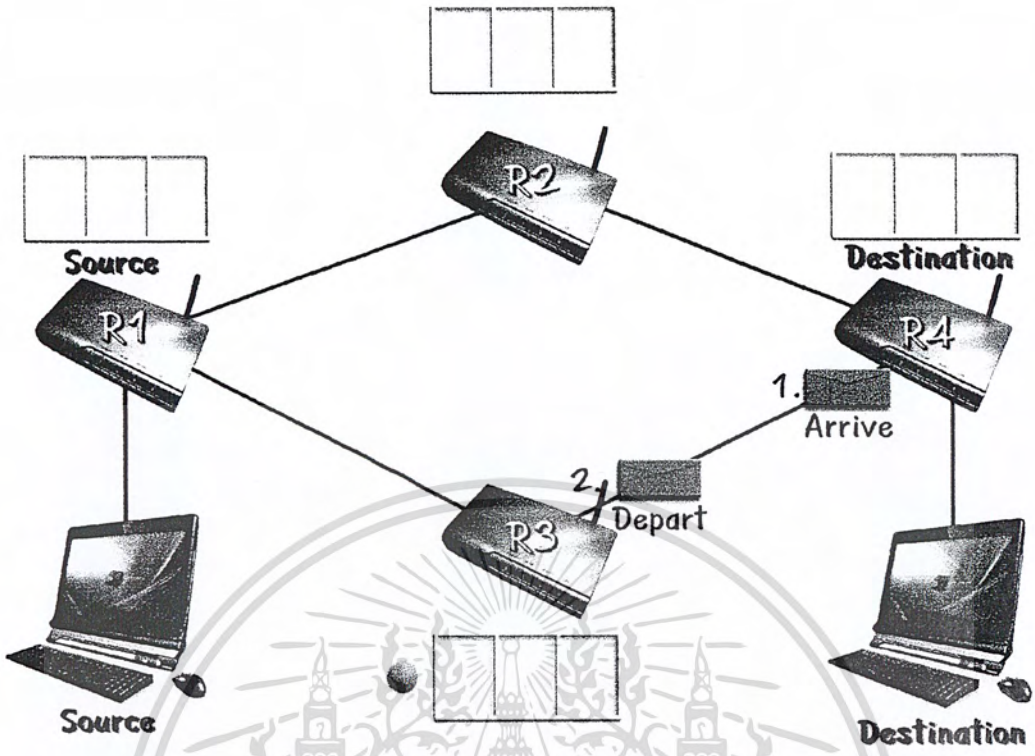
รูป 3.24 เกิดเหตุการณ์ arrive ที่เราท์เตอร์ R3 และมีแพ็คเกตที่ถูกให้บริการอยู่

จากรูป 3.25 เมื่อแพ็คเกตสีฟ้าได้รับบริการเสร็จแล้วและเกิดเหตุการณ์ depart เพื่อไปยังเครื่องปลายทาง และจะมีการตรวจสอบว่าในคิวมีแพ็คเกตค้างอยู่หรือไม่ ซึ่งในคิวยังมีแพ็คเกตสีแดงค้างอยู่ จึงนำแพ็คเกตสีแดงมาให้บริการแล้วทำการเลื่อนคิวเข้ามาด้วย



รูป 3.25 เกิดการโพรเซสที่เราท์เตอร์ R3

จากรูป 3.26 เกิดเหตุการณ์ Depart ขึ้นที่เราท์เตอร์ R3 เพื่อส่งแพ็คเกตสีแดงไปยังเครื่องปลายทาง และตรวจสอบว่าในคิว มีแพ็คเกตเหลืออยู่หรือไม่ ซึ่งในคิวไม่มีแพ็คเกตเหลืออยู่เลย สถานะของอุปกรณ์จึงเป็นสถานะ idle (วงกลมสีเขียวจากรูป 3.26)



รูป 3.26 เกิดเหตุการณ์ depart ที่เราท์เตอร์ R3 ไปสู่เครื่องปลายทางและคิวว่าง

บทที่ 4
ผลการดำเนินงาน

4.1 ภาพรวมหน้าต่างโปรแกรม

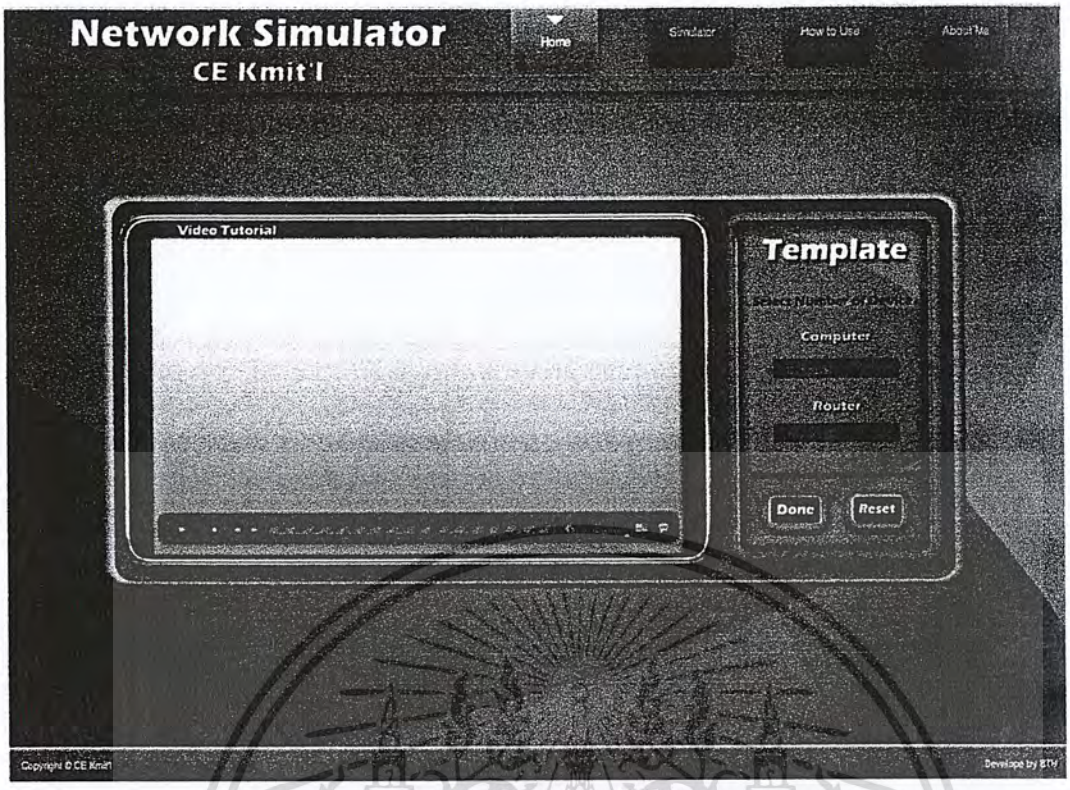
เมื่อเปิดโปรแกรมโดยเว็บเบราว์เซอร์จะพบหน้าต่างโปรแกรม ดังรูป 4.1 ประกอบด้วย ชื่อโปรแกรม, Credit และ แถบเมนูไว้สำหรับเปลี่ยนหน้าต่างโปรแกรมซึ่งประกอบด้วย

1) Home เป็นหน้าต่างเริ่มต้นโปรแกรม
2) Simulator คือส่วนแสดงผลการจำลองระบบเครือข่าย ตัวโปรแกรมหลักจะอยู่ในส่วนนี้
3) How to Use แสดงวิธีการใช้งานโปรแกรม
4) About Me รายละเอียดเกี่ยวกับเวอร์ชั่นโปรแกรม และทีมผู้พัฒนา

4.2 หน้าต่าง Home

เป็นแทบเมนูเริ่มต้นของโปรแกรม แถบนี้จะพบว่ามีหน้าต่างโปรแกรมดัง รูป 4.1 โดยแบ่งออกเป็นสองส่วนดังนี้

1) A: Video Tutorial จะเป็นส่วนวีดีโอสาธิตการใช้งานโปรแกรมจำลองการทำงานสำหรับเครือข่ายคอมพิวเตอร์
2) B: Template จะเป็นส่วนที่ใช้ในการเลือกแม่แบบโทโปโลยีสำเร็จรูป โดยผู้สร้างไม่ต้องทำการ สร้างโทโปโลยีเอง



รูป 4.1 หน้าหลักของโปรแกรม

4.3 หน้าต่างซิมมูเลเตอร์

4.3.1 หน้าต่างซิมมูเลเตอร์ส่วนกำหนดค่า

เป็นเมนูหลักในการจำลองเครือข่ายโปรแกรม โดยเมื่อเปิดแถบเมนูส่วนแรกจะเป็นส่วนในการกำหนดค่า จะพบว่ามีหน้าต่าง โปรแกรมดัง รูป 4.2 โดยแบ่งออกเป็นส่วนย่อยๆดังนี้

1) A: เป็นส่วนสร้าง และ แสดงผลโทโปโลยี ซึ่งผู้ใช้สามารถสร้างโทโปโลยี ได้จากการลากเครื่องมือจาก Toolbar ด้านล่างมาใส่

2) B: Toolbar เป็นแถบที่ไว้เก็บเครื่องมือในการสร้าง และ เครื่องมือในการ Debug โทโปโลยี

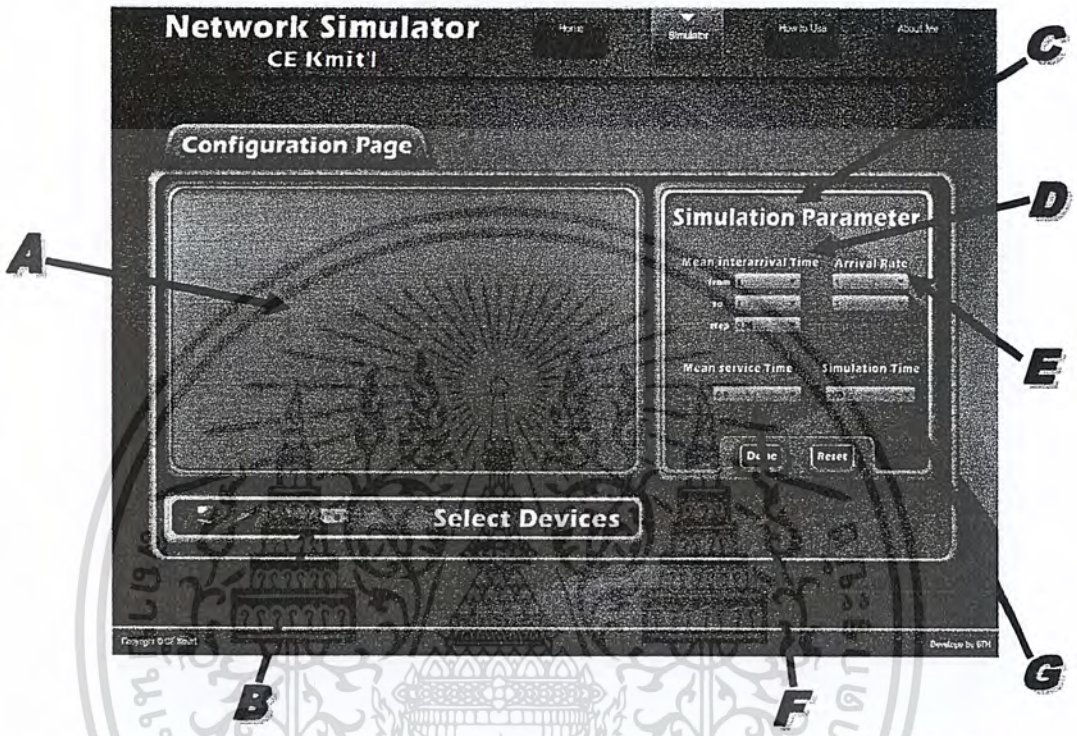
3) C: Simulation Parameter แบ่งออกเป็นสี่ส่วนย่อยๆ ดังนี้

3.1) D: Mean Interarrival Time คือ การตั้งค่าเวลาของความถี่ของการเข้ามาของผู้ใช้งาน หรือ ความถี่ของการส่งแพ็คเกต ซึ่งจะแปรผกผันกับ Arrival Rate หรืออัตราการเข้ามาของผู้ใช้งานต่อหน่วยเวลา โดยช่อง from คือค่าเริ่มต้นของ Mean Interarrival ส่วน to คือค่าสุดท้ายของ Mean Interarrival โดยจะทำการลดค่าต่อครั้งตามจำนวนค่าที่ตั้งไว้ใน step ซึ่งยิ่งค่า Mean Interarrival เยอะเท่าไหร่ความถี่ในการเข้ามาของผู้ใช้งานก็จะยิ่งน้อยลงเท่านั้น

3.2) E: Arrival Rate เป็นส่วนกลับของ Mean Interarrival Time

3.3) F: Mean Service Time คือ การตั้งค่าหน่วยเวลาของเวลาที่เซิร์ฟเวอร์จะใช้ทำงานต่อ 1 แพ็คเกต

3.4) G: Simulation Time คือ การตั้งค่าหน่วยเวลาที่จะใช้ในการจำลองระบบเครือข่าย



รูป 4.2 หน้าต่างเมนูซิมมูเลเตอร์ส่วนกำหนดค่า

4.3.2 หน้าต่างซิมมูเลเตอร์ส่วนแสดงผล

เมื่อทำการตั้งค่าและสร้างโทโปโลยีเสร็จแล้วให้ผู้ใช้งานกดปุ่ม Done เพื่อเข้าสู่ส่วนแสดงผล จะพบว่ามีหน้าต่างโปรแกรมดัง รูป 4.3 โดยแบ่งออกเป็นส่วนย่อยๆดังนี้

1) A: เป็นส่วนแสดงผลโทโปโลยี ที่เราได้ทำการสร้างไว้แล้วในหน้ากำหนดค่า

2) B: Output เป็นส่วนแสดงผลข้อความเมื่อทำการเริ่มต้นจำลองการทำงาน

3) C: Animation เป็นส่วนแสดงแอนิเมชัน โดยจะทำการแสดงผลเมื่อผู้ใช้ทำการคลิกเมาส์ลงบนอุปกรณ์ในโทโปโลยี

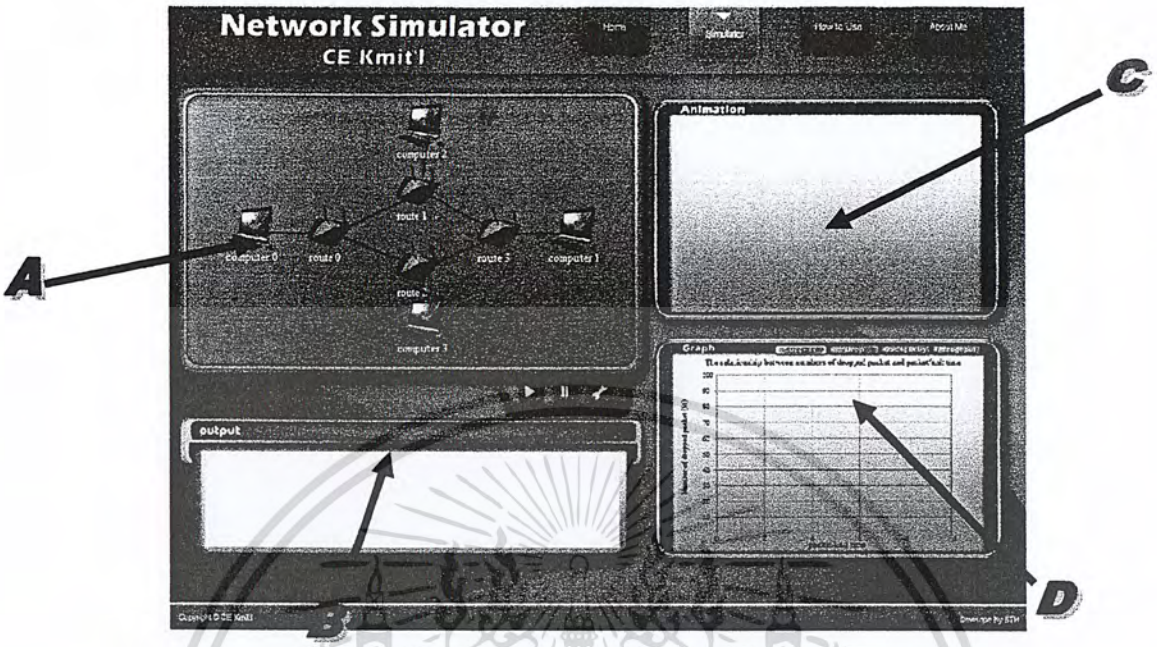
4) D: Graph เป็นส่วนแสดงผล ประสิทธิภาพของระบบ โดยจะแยกออกเป็น 4 ส่วน ดังนี้

4.1) Success คือ กราฟที่แสดงจำนวนแพ็คเกตที่สำเร็จ

4.2) Drop คือ กราฟแสดงจำนวนแพ็คเกตที่ถูกดรอป

4.3) Queuing Delay กราฟดีเลย์เฉลี่ยของการรอในคิว

4.4) Throughput คือ กราฟทรูพุทแสดงปริมาณข้อมูลที่เข้ามา

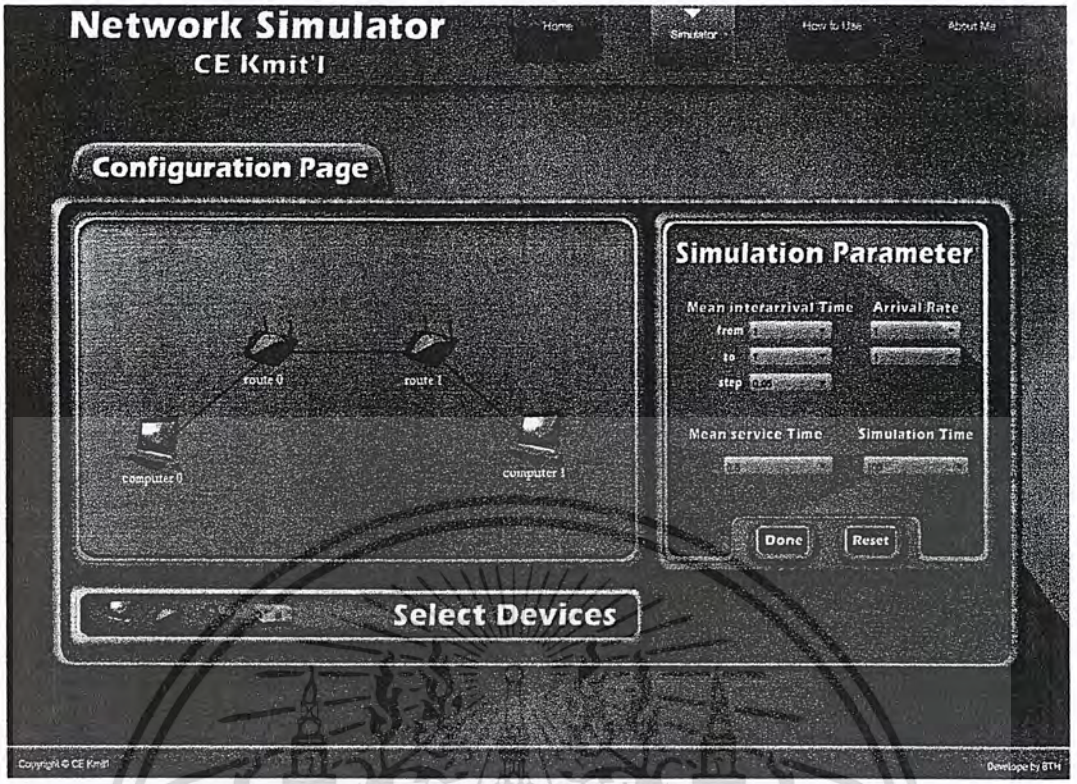


รูป 4.3 หน้าต่างเมนูซิมมูเลเตอร์ส่วนแสดงผล

4.4 การจำลองระบบ

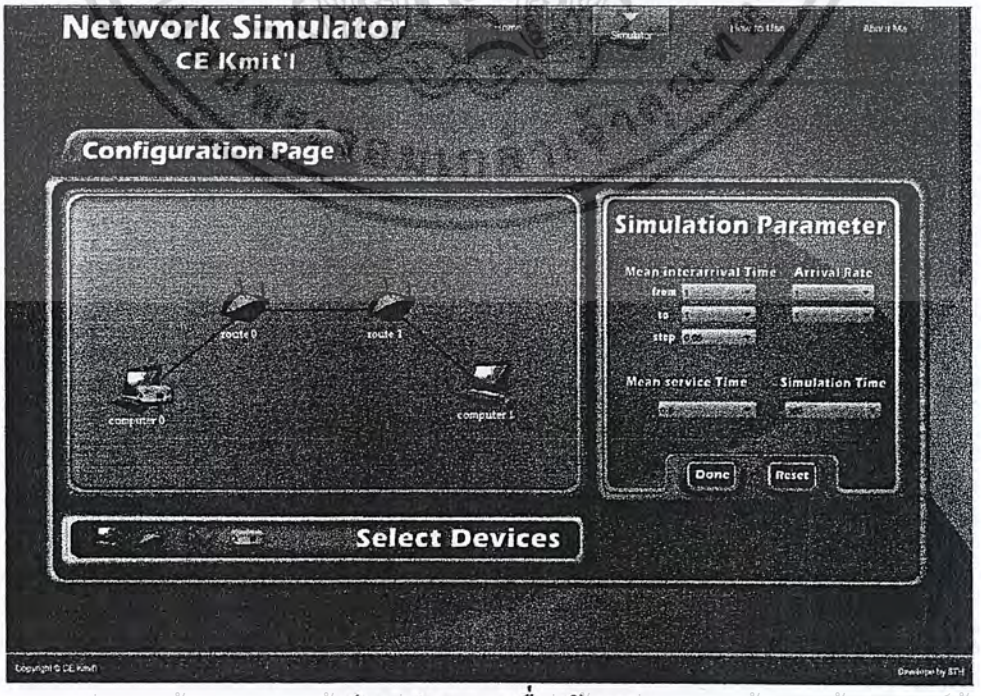
4.4.1 ขั้นตอนการสร้างและส่งแพ็คเกต

ในส่วนนี้ เราจะแสดงการส่งแพ็คเกตจากเครื่องต้นทางไปยังเครื่องปลายทาง โดยอิงหลักการ Discrete model แบบ Event driven ขั้นตอนแรกเราทำการสร้างโทโปโลยี โดยลากเครื่องมือจาก Toolbar ในที่นี้เราจะเลือก คอมพิวเตอร์จำนวน 2 ตัว เราท์เตอร์จำนวน 2 ตัว และลากสายติดต่อระหว่างเราท์เตอร์ ดังรูป 4.4

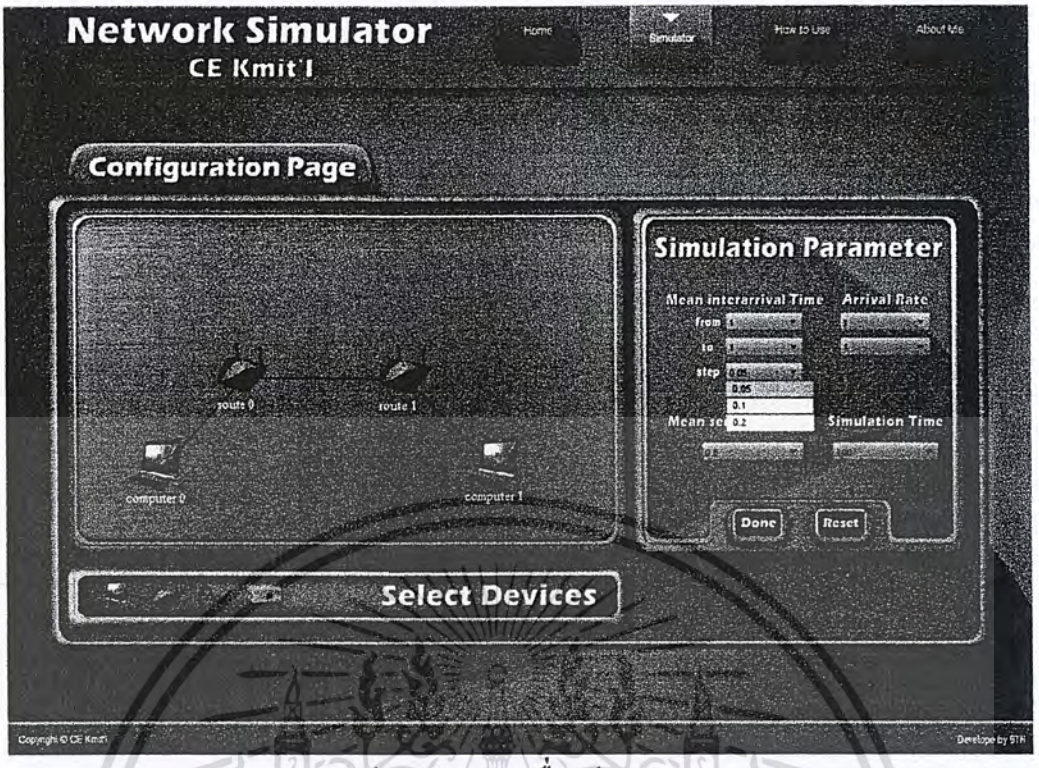


รูป 4.4 ทำการสร้างโทโปโลยี

กำหนดการส่งแพ็คเกต โดยคลิกเลือกรูปแพ็คเกตจาก Toolbar และเลือกเครื่องต้นทางและปลายทางจากโทโปโลยี ที่เราสร้างเอาไว้และทำการปรับค่า Simulation Parameter ดัง รูป 4.5 และ 4.6

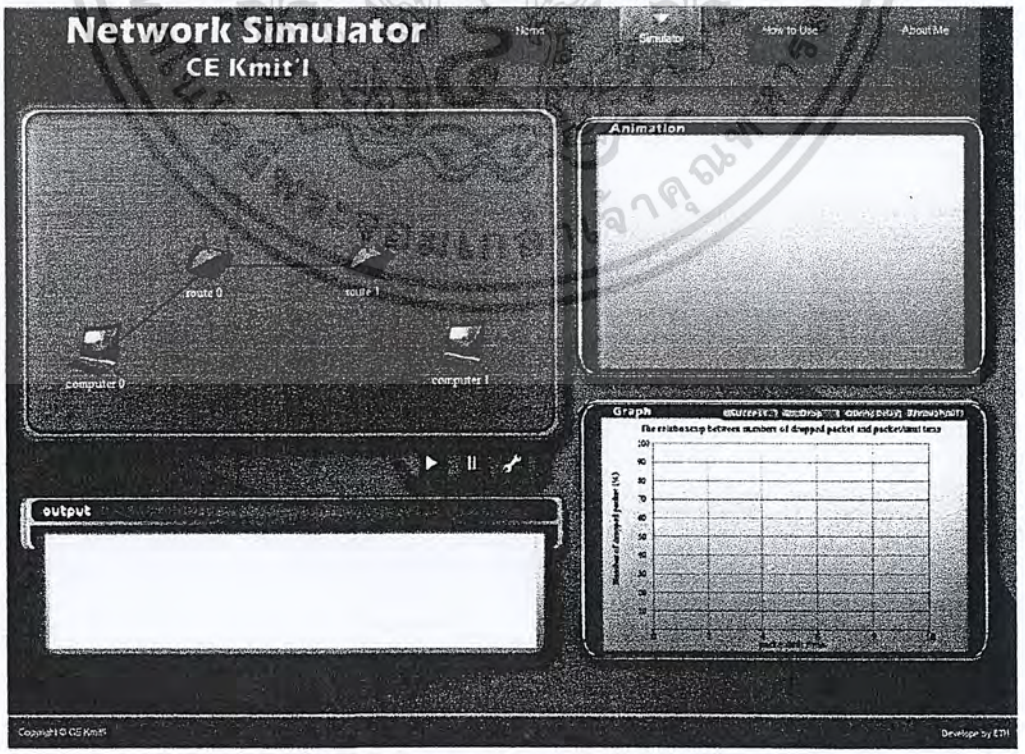


รูป 4.5 กำหนดเครื่องต้นทาง



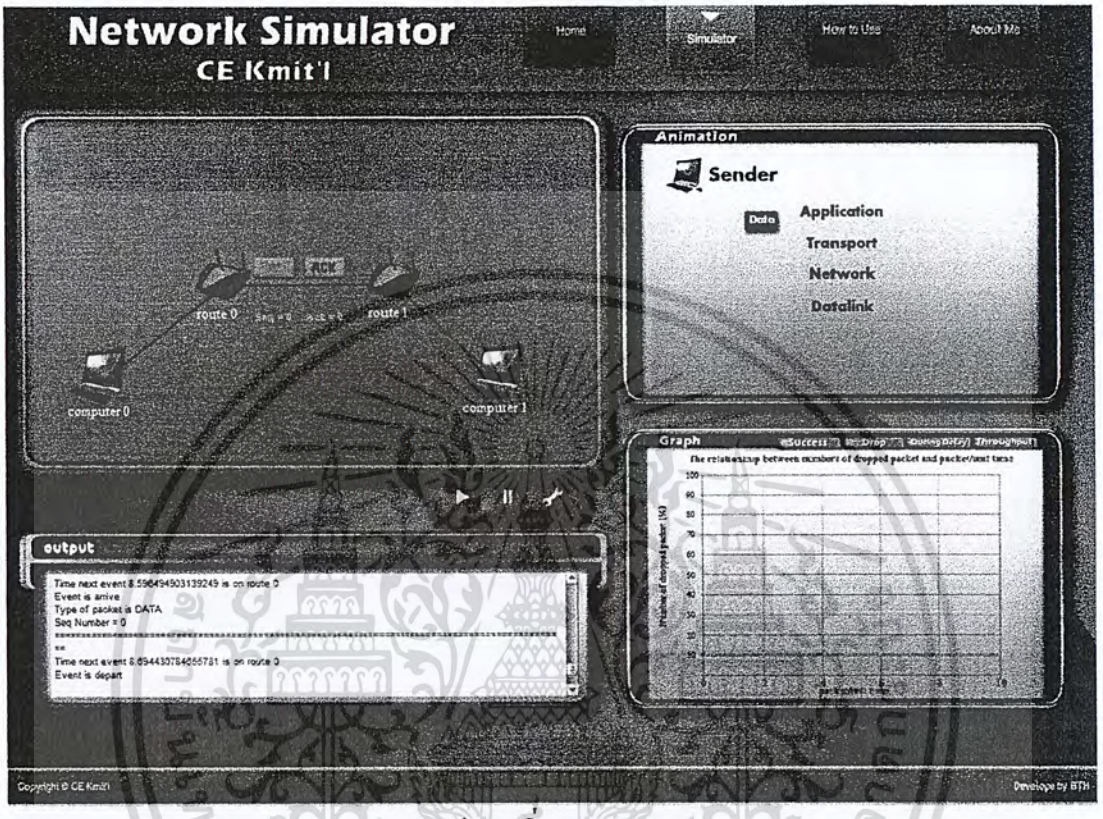
รูป 4.6 กำหนดเครื่องปลายทาง

เมื่อเลือกเครื่องต้นทางและปลายทางได้แล้ว ให้กดปุ่ม Done เพื่อเข้าสู่หน้าต่างเมนู Simulator ส่วนแสดงผล ดังรูป 4.7



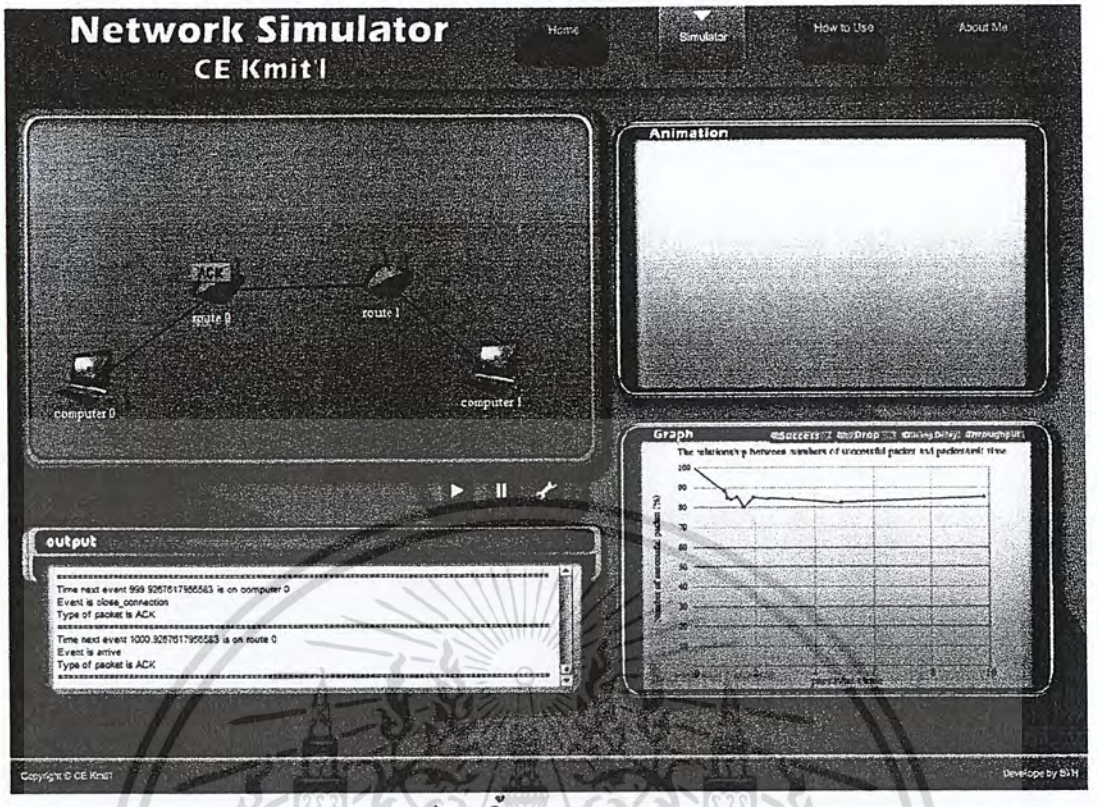
รูป 4.7 หน้าต่างเมนูซิมมูเลเตอร์ส่วนแสดงผล

หลังจากนั้น เริ่มทำการจำลองการทำงาน โดยให้กดปุ่ม Play บริเวณ ขวาล่างของโทโปโลยี จะเห็นว่ามีแอนิเมชันการเคลื่อนที่ของแพ็คเกตจากเครื่องต้นทางไปเครื่องปลายทาง และเมื่อมีการคลิกที่ตัวอุปกรณ์จะมีการแสดงแอนิเมชันประกอบเพื่อความเข้าใจ ดังรูป 4.8



รูป 4.8 เริ่มการทำงาน

เมื่อการจำลองสถานการณ์สิ้นสุดแล้ว จะได้ผลลัพธ์ดังรูป 4.9



รูป 4.9 สิ้นสุดการทำงาน

4.5 การทดสอบระบบ

การจำลองระบบเครือข่ายผ่านทาง Application Program นั้นจัดทำขึ้นเพื่อวัดประสิทธิภาพของระบบและขั้นตอนการทำงานของระบบ เพื่อนำประสิทธิภาพที่ได้นั้นไปใช้ในการศึกษาพิจารณาและวิเคราะห์ระบบก่อนนำไปออกแบบระบบจริงจากอุปกรณ์ ซึ่งประสิทธิภาพของระบบวัดได้จาก

1) Output ซึ่งจะบอกรายละเอียดต่างๆ เมื่อเกิดเหตุการณ์ขึ้นในระบบ
2) ค่าประสิทธิภาพที่ได้เมื่อการจำลองระบบเสร็จสิ้น เช่น จำนวนแพ็คเกตที่ดรอปจำนวน แพ็คเกต ที่ส่งสำเร็จ จำนวนแพ็คเกตที่คอมพิวเตอร์ส่งเข้ามาในระบบทรูพุทค่าเฉลี่ยของการรอในคิวของแต่ละเราเตอร์ และค่าดีเลย์รวมของการจำลองระบบ
3) กราฟวัดประสิทธิภาพของระบบเมื่อจำลองระบบเครือข่ายเสร็จสิ้น

4.5.1 Output

เป็นส่วนบอกรายละเอียดต่างๆที่เกิดขึ้นภายในระบบดังแสดงตัวอย่างในรูป 4.10

Time next event 4.143348652249325 **is on** route 0

Event is depart

Type of packet is SYN_ACK

==========

Time next event 21.709464089212947 **is on** route 1

Event is process

Type of packet is DATA

Seq Number = 5000

รูป 4.10 รายละเอียดที่เกิดขึ้นในระบบ

1) Time next event คือ เวลาต่อไปที่จะมีเหตุการณ์เกิดขึ้น

2) on คือ เหตุการณ์ นั้นๆ จะเกิดขึ้นที่อุปกรณ์ตัวไหน

3) Event คือ ชื่อของเหตุการณ์ที่จะเกิดขึ้น ประกอบด้วย arrive, depart,create_TCP_connection, close_connection, arriveAPP, time_out, process

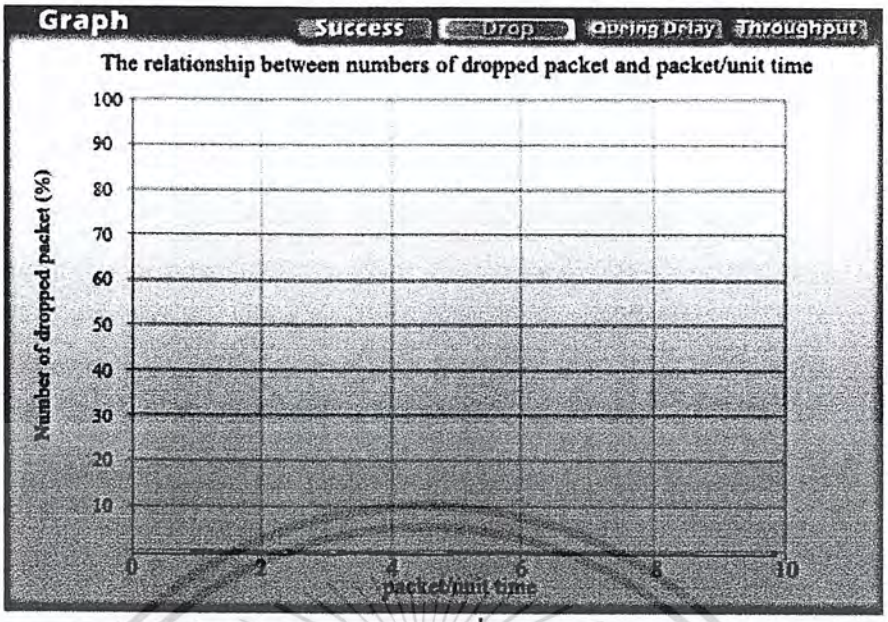
4) Type of packet คือ ชนิดของ แพ็คเกตที่จะเกิดขึ้น ประกอบด้วย DATA, ACK, SYN, SYN_ACK, FIN, FIN_ACK

5) Seq Number คือ เลข Sequence Number ของ แพ็คเกตที่มีชนิดข้อมูลเป็น DATA

4.5.2 กราฟวัดประสิทธิภาพของระบบ

4.5.2.1 กราฟแสดงจำนวนแพ็คเกตที่ถูกดรอป

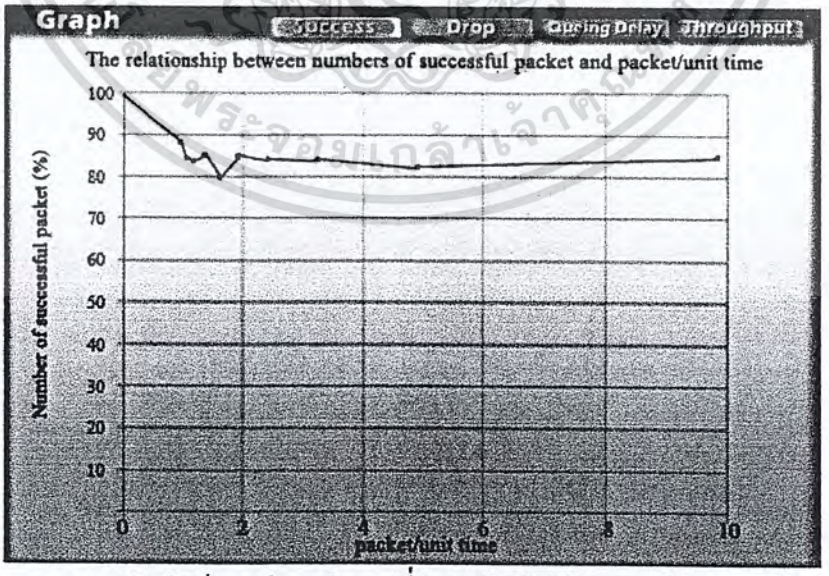
จากรูป 4.11 จะเห็นได้ว่าเมื่ออัตราการเข้ามาของแพ็คเกตเพิ่มขึ้นจำนวนของแพ็คเกต ที่ดรอปก็จะสูงขึ้นตาม (กราฟอาจขึ้นกับค่าอื่นด้วย เช่น เวลาการให้บริการของเซิร์ฟเวอร์ ค่าเวลาทั้งหมดที่ใช้ในการจำลองระบบ เป็นต้น)



รูป 4.11 ความสัมพันธ์ระหว่างจำนวนแพ็คเกตที่ถูกครอปกับอัตราการเข้ามาของแพ็คเกต

4.5.2.2 กราฟแสดงจำนวนแพ็คเกตที่ส่งสำเร็จ

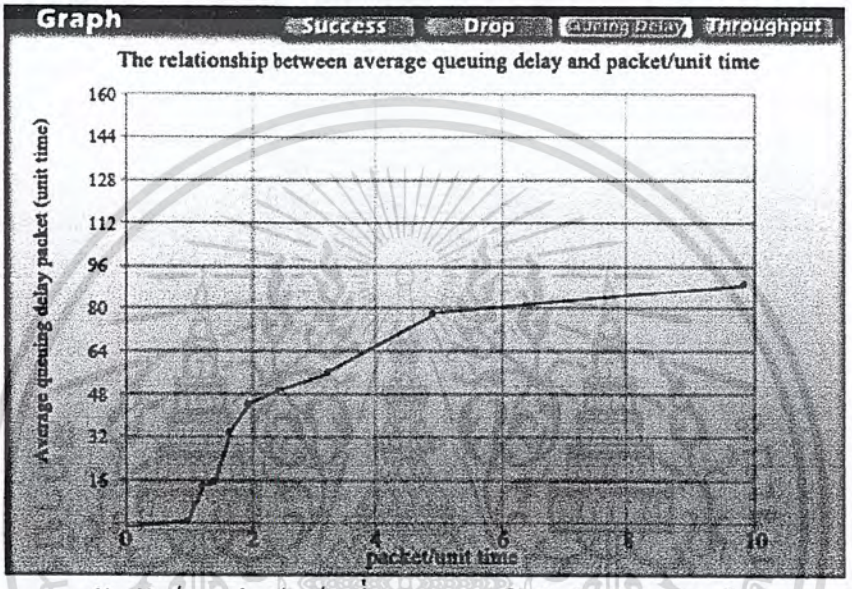
จากรูป 4.12 จะเห็นได้ว่าเมื่ออัตราการเข้ามาของแพ็คเกตเพิ่มขึ้นจำนวนของแพ็คเกตที่ส่งสำเร็จก็จะสูงขึ้นตาม เนื่องจากมีแพ็คเกตเข้ามาในระบบมากขึ้นแต่เมื่อถึงจุดที่ระบบทำงานอย่างเต็มประสิทธิภาพ จำนวนแพ็คเกตที่ส่งสำเร็จก็จะไม่เพิ่มขึ้นตามอัตราการเข้ามาของแพ็คเกตแล้ว (กราฟอาจขึ้นกับค่าอื่นด้วย เช่น เวลาการให้บริการของเซิร์ฟเวอร์ ค่าเวลาทั้งหมดที่ใช้ในการจำลองระบบ เป็นต้น)



รูป 4.12 ความสัมพันธ์ระหว่างแพ็คเกตที่ส่งสำเร็จกับอัตราการเข้ามาของแพ็คเกต

4.5.2.3 กราฟดีเลย์เฉลี่ยของการรอในคิว

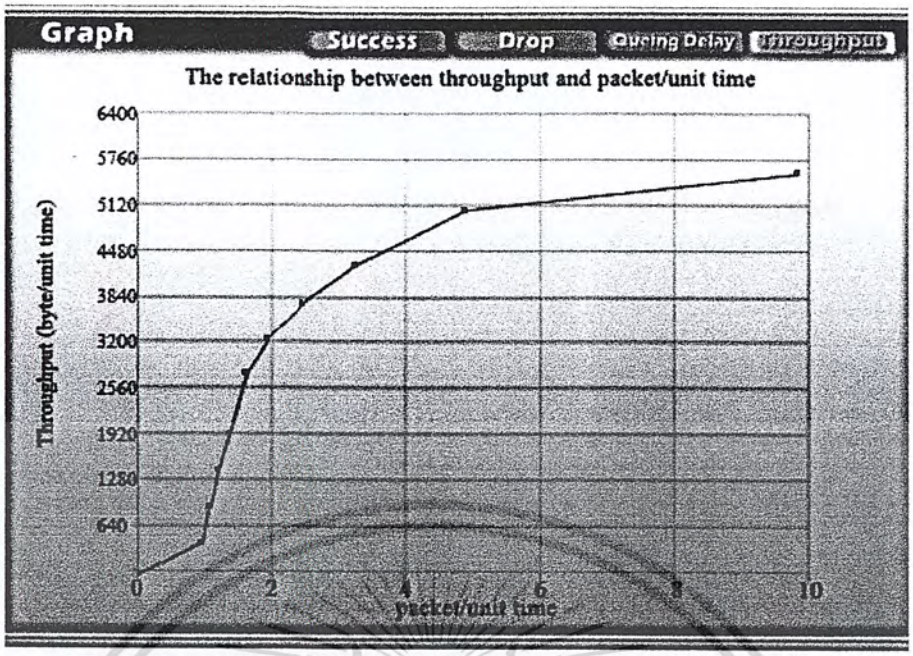
จากรูป 4.13 จะเห็นได้ว่ากราฟดีเลย์เฉลี่ยของการรอในคิวกับอัตราการเข้ามาของแพ็คเกตจะแบ่งกราฟออกให้แสดงแยกแต่ละเราเตอร์ โดยจะเห็นว่าเมื่อมีอัตราการเข้ามาของแพ็คเกตมากๆ ดีเลย์เฉลี่ยของการรอในคิวก็จะเพิ่มขึ้น เมื่อมีการรอในคิวของแต่ละแพ็คเกตนานขึ้น ค่าของกราฟก็จะเพิ่มขึ้นด้วย (กราฟอาจขึ้นกับค่าอื่นด้วย เช่น เวลาการให้บริการของเซิร์ฟเวอร์ ค่าเวลาทั้งหมดที่ใช้ในการจำลองระบบ เป็นต้น)



รูป 4.13 ความสัมพันธ์ระหว่างดีเลย์เฉลี่ยของการรอในคิวกับอัตราการเข้ามาของแพ็คเกต

4.5.2.4 กราฟทรูพุท

กราฟทรูพุทกับอัตราการเข้ามาของแพ็คเกตก็เช่นกันเมื่ออัตราการเข้ามาของแพ็คเกตเพิ่มขึ้นค่าทรูพุทก็จะเพิ่มขึ้นด้วย ซึ่งจะแสดงให้เห็นได้จากรูป 4.14 (กราฟอาจขึ้นกับค่าอื่นด้วย เช่น เวลาการให้บริการของเซิร์ฟเวอร์ ค่าเวลาทั้งหมดที่ใช้ในการจำลองระบบ เป็นต้น)



รูป 4.14 ความสัมพันธ์ระหว่างทรูพุทกับอัตราการเข้ามาของแพ็คเกต

บทที่ 5
บทสรุป

5.1 บทสรุป

โปรแกรมจำลองสำหรับเครือข่ายคอมพิวเตอร์ มีจุดประสงค์หลักคือเพื่อต้องการจำลองการทำงานของเครือข่ายคอมพิวเตอร์และใช้สำหรับเป็นสื่อการสอนในวิชา Computer Network ซึ่งโปรแกรมจะจำลองการรับส่งแพ็คเกตจากต้นทางไปยังปลายทางโดยใช้หลักการของโพรโทคอลทีซีพีไอพี ซึ่งเป็นโพรโทคอลที่มีการใช้กันมากในปัจจุบัน เพื่อให้ระบบมีความเสมือนจริงมากยิ่งขึ้น เช่น การติดต่อแบบ Connection-oriented ระหว่างต้นทางกับปลายทางก่อนที่จะรับหรือส่งข้อมูลหากัน หรือมี Congestion Control ไว้สำหรับควบคุมทราฟฟิคบนระบบเครือข่าย เป็นต้น และเพื่อให้โปรแกรมมีความอิสระในการใช้งาน ผู้ใช้งานสามารถเลือกอุปกรณ์และลักษณะการวางของอุปกรณ์ได้อย่างอิสระ อีกทั้งยังมีแม่แบบสำหรับการวางอุปกรณ์เพื่อให้ผู้ใช้งานมีความสะดวกมากยิ่ง นอกจากนี้ระบบยังมีการเก็บข้อมูลต่าง ๆ เพื่อนำมาแสดงเป็นประสิทธิภาพของระบบ ได้แก่ เปอร์เซ็นต์การส่งแพ็คเกตสำเร็จ, Queuing Delay ของระบบ, เปอร์เซ็นต์การดรอปของแพ็คเกต, ทรูพุทอีกทั้งยังทำการแสดงผลออกมาในรูปแบบของอนิเมชันเพื่อให้ง่ายต่อการศึกษาและเพิ่มความน่าสนใจให้กับโปรแกรม

5.2 วิจารณ์สิ่งที่ได้จากโครงงาน

จากการทำโครงงานชิ้นนี้ทำให้ทราบว่า การที่จะพัฒนาระบบใดๆขึ้นมาสักระบบหนึ่ง การวิเคราะห์และการออกแบบ เป็นขั้นตอนที่มีความสำคัญมาก และจะต้องมองระบบโดยภาพรวมให้ครอบคลุมกระบวนการทำงานทั้งหมด และต้องมีความละเอียดรอบคอบในการออกแบบ เพื่อลดการเกิดปัญหาต่างๆที่จะเกิดขึ้นในภายหลัง

5.3 ปัญหาและอุปสรรค

1) เนื่องจากโครงงานนี้ ค่อนข้างมีขอบเขตที่ค่อนข้างกว้างมาก ทำให้ผู้พัฒนาไม่สามารถที่จะพัฒนาได้ครบถ้วนตามหลักการของโพรโทคอลทีซีพีไอพี
2) เนื่องจากโครงงานนี้เป็นการจำลองการทำงานของระบบเครือข่ายคอมพิวเตอร์ ผลลัพธ์บางอย่างที่ได้ออกมาเป็นเพียงค่าโดยประมาณเท่านั้น ไม่สามารถนำมาใช้อ้างอิงในงานที่ต้องการความละเอียดได้

5.4 แนวทางการพัฒนาต่อ

1) เพิ่มการทำงานโพรโทคอลทีซีพีไอพีให้ครอบคลุมและสมบูรณ์มากที่สุด

2) เพิ่มประเภทของอุปกรณ์และใส่รายละเอียดในอุปกรณ์ต่างๆมากขึ้น

3) เพิ่มประเภทของกราฟการแสดงประสิทธิภาพให้หลากหลายมากขึ้น

บรรณานุกรม

Law, Averill M. 2007. **Simulation modeling and analysis.** Boston : McGraw-Hill

เอกสิทธิ์ วิริยจารี. 2548. **เรียนรู้ระบบเน็ตเวิร์กจากอุปกรณ์ของ Cisco.** กรุงเทพฯ : ซีเอ็ดยูเคชั่น

กำพล ลีลาภรณ์. 2551. **Flash Actionscript.** กรุงเทพฯ : โปรวิชั่น

ไพบูลย์ สัวสดิ์ปัญญาโชติ. 2552. **รู้ลึกเรียนลัด ActionScript 3.0 for Adobe Flash CS4 Professional.**กรุงเทพฯ: ฟิวเจอร์ เกมเมอร์

บุญญาดา ช้อนขุนทด. 2550. **Insight Flash CS3.** กรุงเทพฯ : โปรวิชั่น

บรรยง ใจซื่อ, ฝนทิพย์ ดุรงค์พันธ์ และ วรนันท์ บรรเจิดประยูร. 2552. "ระบบจำลองการทำงานสำหรับเครือข่ายคอมพิวเตอร์." ปริญญานิพนธ์วิศวกรรมศาสตร์มหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

Adobe Systems Incorporated. 2008. **ActionScript 3.0 Language and Components Reference** [Online].Available : http://www.adobe.com/livedocs/flash/9.0/ActionScriptLangRefV3/