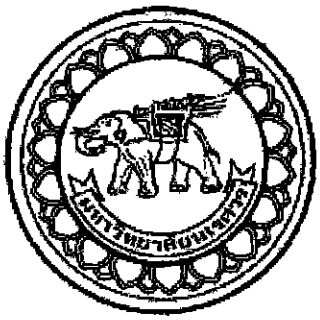


โปรแกรมจำลองระบบเครือข่ายคอมพิวเตอร์เบื้องต้น

Network Simulation Software

นางสาวพัชนก คำก้อน รหัส 43360486

นายอัฐราวุฒิ เดชผล รหัส 43360676

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2546

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	โปรแกรมจำลองระบบเครือข่ายคอมพิวเตอร์เบื้องต้น
ผู้ดำเนินโครงงาน	นางสาวพัชนก คำก้อน รหัส 43360486
	นายอัฐราวุฒิ เดชผล รหัส 43360676
อาจารย์ที่ปรึกษา	ดร. ยงยุทธ ชนบดีเฉลิมรุ่ง
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2546

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ดร. ยงยุทธ ชนบดีเฉลิมรุ่ง)

.....กรรมการ

(ดร. สุรเชษฐ์ กานต์ประชา)

.....กรรมการ

(ดร. พนมขวัญ ริยะมงคล)

หัวข้อโครงงาน	โปรแกรมจำลองระบบเครือข่ายคอมพิวเตอร์เบื้องต้น
ผู้ดำเนินโครงงาน	นางสาวพัชนก คำก้อน รหัส 43360486
	นายอัฐราวุฒิ เดชผล รหัส 43360676
อาจารย์ที่ปรึกษา	ดร. ยงยุทธ ชนบดีเฉลิมรุ่ง
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2546

.....

บทคัดย่อ

โครงงานนี้เป็นการศึกษาและพัฒนาโปรแกรมเพื่อออกแบบและประยุกต์ใช้โปรแกรมแบบวิชวล (Visual Programing) ให้ได้มาซึ่งโปรแกรมที่สามารถจำลองการทำงานของระบบเครือข่ายอย่างง่ายได้ โดยจำลองเครือข่ายที่มีการจัดการภายในเป็นแบบระบบ M/M/1 Queue โปรแกรมนี้ใช้โปรแกรมบอร์แลนด์ซีพลัสพลัสบิลเดอร์ เวอร์ชัน 5 (C++ Builder 5.0) ในการสร้างและออกแบบโปรแกรมต่างๆ

ผลที่ได้จากโครงงานนี้ คือ โปรแกรมที่สามารถรองรับการจำลองการทำงานของระบบเครือข่ายที่มีระบบการจัดการภายในแบบ M/M/1 Queue และสามารถหาค่า Time Delay ที่เกิดขึ้นได้

Projec Tile	Network Simulation Software	
Name	Miss Patchanok Khomkon	ID.43360486
	Mr. Auttarawut Dedpun	ID.43360676
Project Advisor	Mr. Yongyut Chanabodeechalermrung	
Major	Computer Engineering	
Department	Electrical and Computer Engineering	
Academic Year	2003	

.....

.

ABSTRACT

This project is to study and develop software for designing and applying the Visual Programing for a program that can simulate a simple network by modeling a network which has M/M/1 Queue i-side. This project use Boland C++ Builder 5.0 to construct and design the program.

As a result of the project , there will be a program that can support the simulation of a simple network which M/M/1 queue , and can find time delay of the network.

กิตติกรรมประกาศ

การจัดทำโครงงานวิศวกรรมครั้งนี้ สำเร็จลุล่วงด้วยดี เนื่องมาจากการแนะนำและช่วยเหลือจากอาจารย์ที่ปรึกษา ท่านอาจารย์ ยงยุทธ ชนบดีเฉลิมรุ่ง และขอขอบคุณอาจารย์ภาควิศวกรรมไฟฟ้าและคอมพิวเตอร์ทุกท่าน ที่สละเวลาในการให้คำปรึกษาชี้แนวทางที่เป็นประโยชน์อย่างสูงในการจัดทำโครงงานครั้งนี้ และเจ้าหน้าที่สำนักงานคณะวิศวกรรมศาสตร์ทุกท่านที่ช่วยประสานงานในการเบิกค่าใช้จ่ายในการทำโครงงาน

นางสาวพัชนก คำก้อน

นายอัฐราวุฒิ เดชผล

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

ในปัจจุบัน โครงข่ายอินเทอร์เน็ต (Internet) ประกอบขึ้นจากโครงข่ายคอมพิวเตอร์ขององค์กรต่างๆ จำนวนมาก ที่ได้รับการเชื่อมต่อเข้าด้วยกันจนกลายเป็นระบบโครงข่ายขนาดใหญ่ สามารถรองรับความต้องการการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ทุกเครื่องที่อาจอยู่ใกล้หรือห่างไกลกันได้อย่างไร้ขอบเขตและมีประสิทธิภาพ รูปแบบการให้บริการสื่อสารบนอินเทอร์เน็ตได้รับการพัฒนาอย่างต่อเนื่องและแตกต่างไปจากในช่วงเริ่มต้นอย่างมาก จากเดิมที่เน้นเฉพาะการสื่อสารข้อมูล (Data Communication) ตั้งแต่การส่งผ่านไฟล์ (File Transfer) การรับส่งไปรษณีย์อิเล็กทรอนิกส์ (e-mail) จน ถึงยุคของการพัฒนาโพรโทคอล HTTP(Hypertext Transfer Protocal) ที่ก่อให้เกิดโปรแกรมเว็บบราวซิง (Web browsing) การติดต่อสื่อสารทางเสียงผ่านอินเทอร์เน็ตก็เริ่มมีการใช้งานกันมากขึ้น และแน่นอนว่ารูปแบบการให้บริการจะเพิ่มขึ้นอีกมากมายในอนาคต จนนำไปสู่ระบบสื่อสารมัลติมีเดีย (Multimedia) อย่างเต็มรูปแบบ ความสำเร็จของการสื่อสารบนอินเทอร์เน็ตสามารถพิจารณาได้จากจำนวนโฮสต์ (host) หรือเครื่องคอมพิวเตอร์ที่ต่อเชื่อมกับอินเทอร์เน็ต ได้ขยายตัวอย่างรวดเร็วในทศวรรษที่ผ่านมา ดูสถิติจำนวนโฮสต์ที่ประมาณ โดยกลุ่มของ Internet software consortium ที่ผ่านมาจำนวนโฮสต์ได้เพิ่มขึ้นอย่างรวดเร็วในปี ค.ศ. 2001 อินเทอร์เน็ตโฮสต์เชื่อมต่ออยู่มาก ถึงในหลักของร้อยล้านเลยทีเดียว ที่น่าสังเกตเป็นพิเศษคือ จำนวนโฮสต์ได้เพิ่มขึ้นอย่างรวดเร็วโดยเฉพาะในปีหลังๆ ฉะนั้นถ้าอัตราการเพิ่มของจำนวนโฮสต์ยังคงมีแนวโน้มเดิมไม่เปลี่ยนแปลง อุปกรณ์สื่อสารที่ต่อเชื่อมอยู่กับระบบอินเทอร์เน็ตจะมีอยู่จำนวนมากกว่าจำนวนโทรศัพท์ทั้งหมดบนโลกใบนี้ในไม่ช้า

จะเห็นได้ว่าระบบเครือข่ายคอมพิวเตอร์มีบทบาทมากในสังคมที่มีการแข่งขันด้านข้อมูลข่าวสาร มีการเชื่อมโยงเครือข่ายมากขึ้นทำให้เรามีความจำเป็นที่จะต้องมีการพัฒนาโปรแกรมต่างๆ เพื่อที่จะนำมาใช้และอำนวยความสะดวกในการวางระบบเครือข่ายทางคอมพิวเตอร์ การจำลองระบบเครือข่ายคอมพิวเตอร์เป็นอีกทางเลือกหนึ่งที่ช่วยให้เราสามารถเข้าใจถึงการวางระบบเครือข่ายได้อย่างถูกต้องและง่ายต่อการเข้าใจ

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อที่จะเลียนแบบหรือจำลองการทำงานของระบบเครือข่ายคอมพิวเตอร์อย่างง่ายได้
2. เพื่อศึกษา ความรู้เกี่ยวกับเครือข่ายคอมพิวเตอร์
3. เพื่อเข้าใจถึงการเชื่อมโยงเครือข่ายคอมพิวเตอร์แบบต่างๆ ได้
4. เพื่อศึกษาการรับและส่งข้อมูลในเครือข่ายคอมพิวเตอร์

1.3 ขอบข่ายของโครงงาน

1. เลียนแบบหรือจำลองการทำงานของเครือข่ายคอมพิวเตอร์อย่างง่ายบนเครื่องคอมพิวเตอร์ PC ได้
2. สามารถสร้าง Network Simulation ได้
3. สามารถสร้าง Node Model ได้
4. สามารถสร้างตัวกำเนิดแพ็คเกจ (Generator) ได้
5. สามารถหาค่า Time Delay ได้
6. สามารถหาค่า Throughput ได้

1.4 ขั้นตอนการดำเนินโครงงาน

ตารางที่ 1.1 ตารางการศึกษาข้อมูลต่างๆ

กิจกรรม	เดือน – ปี				
	ต.ค. 45	พ.ย. 45	ธ.ค. 45	ม.ค. 46	ก.พ. 46
1. เลือกหัวข้อ โครงงาน	←				
2. ปรึกษาอาจารย์เรื่องหัวข้อโครงงาน	↔				
3. ศึกษาความรู้ที่เกี่ยวข้องกับโครงงาน		↔	↔	↔	
4. ทำการศึกษาโปรแกรมที่ใช้ในการทำโครงงาน			↔	↔	↔

ตารางที่ 1.2 ตารางการปฏิบัติงาน

กิจกรรม	เดือน - ปี						
	มี.ค.46	เม.ย.46	พ.ค.46	มิ.ย. 46	ก.ค. 46	ส.ค.46	ก.ย. 46
1. วิเคราะห์และออกแบบระบบ	←	—	→				
2. พัฒนาโปรแกรมพร้อมกับศึกษาปัญหาที่เกิดขึ้น		←	—	—	→		
3. ทดสอบ และแก้ไขปัญหาที่เกิดขึ้นกับโปรแกรม				←	—	→	
4. สรุปและรวบรวมข้อมูลจัดทำรายงาน						←	→

1.5 ผลที่คาดว่าจะได้รับ

1. เข้าใจการทำงานของระบบเครือข่ายคอมพิวเตอร์อย่างง่ายได้
2. ความรู้ทางด้านเครือข่าย (Network) คอมพิวเตอร์
3. ความรู้ทางด้านการเขียนโปรแกรมคอมพิวเตอร์
4. รู้จักพัฒนาและนำโปรแกรมต่างๆ มาประยุกต์ใช้

1.6 งบประมาณที่ใช้

1. ค่าถ่ายเอกสาร	1000	บาท
2. ค่าวัสดุสำนักงาน	500	บาท
3. ค่าวัสดุคอมพิวเตอร์	500	บาท
รวมทั้งสิ้น	2000	บาท

บทที่ 2
ทฤษฎีและโปรแกรมที่ใช้

2.1 การจำลองระบบเครือข่ายคอมพิวเตอร์

การจำลองระบบเครือข่ายคอมพิวเตอร์นั้นจะประกอบไปด้วยโนด (Node) ต่างๆ ซึ่งเชื่อมต่อกันเป็นระบบเครือข่าย แต่ละโนดนั้นจะให้แทนคอมพิวเตอร์แต่ละเครื่อง โนดที่เชื่อมต่อกันจนกลายเป็นระบบเครือข่ายเราจะให้เสมือนว่าเป็นระบบเครือข่ายคอมพิวเตอร์อย่างง่าย องค์ประกอบภายในของโนดนั้นจะลดหลั่นลงไปเป็นชั้นๆ คือ

Network Model เป็นชั้นแรกสุดซึ่งจะประกอบไปด้วยโนดแต่ละโนด ซึ่งเชื่อมโยงกัน โนดจะติดต่อสัมพันธ์กัน จนเป็นระบบเครือข่าย ซึ่งแต่ละโนดจะมีการรับและส่งข้อมูล

Node Model ภายใน แต่ละโนดจะประกอบไปด้วยหน่วยต่าง ๆ ซึ่งทำหน้าที่แตกต่างกันไป เช่น หน่วยรับข้อมูล(Receiver), หน่วยส่งข้อมูล(Transmitter), หน่วยประมวลผล(Processor) และหน่วยสร้างแพ็คเกจ(Generator) เป็นต้น เราเรียกหน่วยต่างๆ นี้ว่าโมดูล (Module)

2.2 การออกแบบโมเดล (Designing the Model)

การออกแบบโมเดล เป็นการจัดการแบบเป็นลำดับขั้น

ขั้นแรกสุด เป็นเขียนอัลกอริทึม (Algorithm) หรือนำอัลกอริทึมมาใช้ โดยใช้ C++Builder

ขั้นที่สอง เป็นการแยกหน้าที่ (Functions) ของการทำงาน เช่น แยกบัฟเฟอริ่ง(Buffering) , แยกกระบวนการ (Processing) , แยกการส่ง (Transmitting) , และแยกการรับข้อมูลแพ็คเกจ (Receiver Data packet) เป็นต้น

ขั้นสุดท้าย เป็นการเชื่อมโยงแต่ละโนด ให้เป็นเครือข่าย (Network Model)

2.3 การจัดการแพ็คเกจ (packet) ภายในโนด (Node)

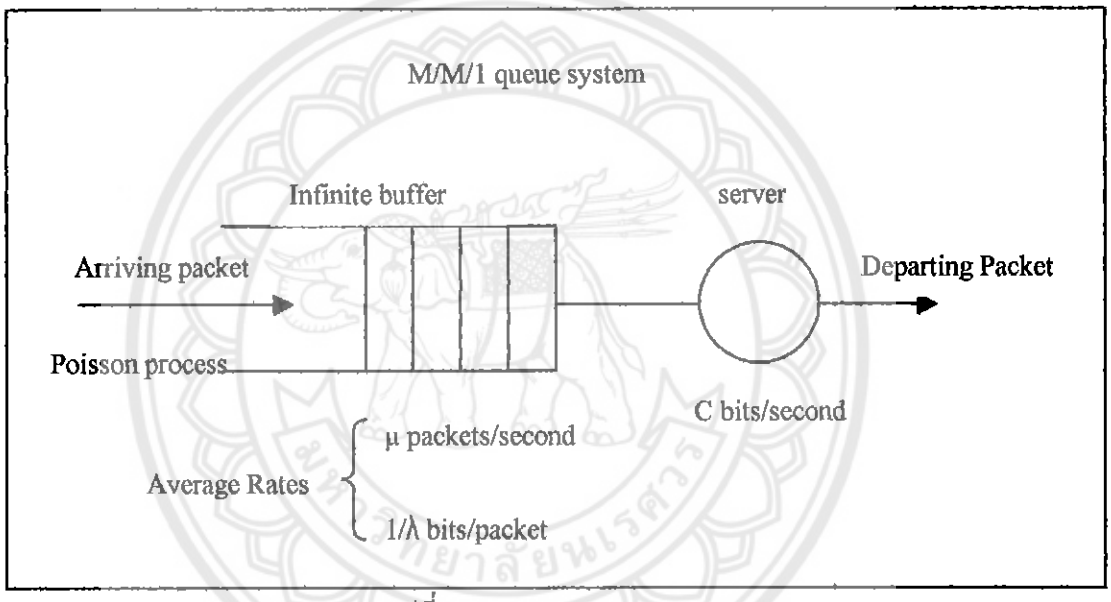
การจัดการแพ็คเกจ (Packet) ภายในโนด (Node) เป็นการเลียนแบบการทำงานของระบบ M/M/1 Queue ระบบ M/M/1 Queue มีการทำงาน แบบ First-In-First-Out (FIFO) ซึ่งเป็นการจัดการการทำงานของระบบโครงสร้างข้อมูลแบบคิว (Queue)

สำหรับแพ็คเกจที่ถูกส่งออกมาจากตัวกำเนิดแพ็คเกจ(Generator) นั้น จะถูกส่งออกมาอย่างสม่ำเสมอตามที่กำหนด แต่ไม่เสมอไปที่แพ็คเกจจะถูกส่งออกไปจากโนดได้ทันที ซึ่งโนดอาจจะมีอัตราการส่งแพ็คเกจออกไปได้ช้ากว่าที่แหล่งกำเนิดแพ็คเกจ(Generator) สร้างแพ็คเกจ แล้วส่ง

ออกมา จึงทำให้เกิดการรอคอย ซึ่งระบบ M/M/1 Queue นั้นจะเป็นระบบที่เข้ามาช่วยในการจัดการกับการจัดส่งแพ็คเกจ การจัดการลำดับแพ็คเกจ และคอยรองรับแพ็คเกจ ที่ถูกส่งออกมา

การดำเนินการของระบบ M/M/1 queue จะขึ้นอยู่กับ 3 ตัวแปร คือ ขนาดของแพ็คเกจ (Packet size) เวลาการมาถึงของแพ็คเกจ(Interarrival time) และความสามารถในการรองรับแพ็คเกจ(Queue size) ถ้าเกิดว่าเวลาการมาถึงของแพ็คเกจ และขนาดของแพ็คเกจ(Packet size) ใหญ่เกินกว่าความจุของการให้บริการ(Service rate) ของระบบ M/M/1 queue จะทำให้ขนาดของคิวสามารถขยายไปได้ไม่จำกัด

จุดประสงค์การสร้าง Node Model ด้วยการเลียนแบบระบบ M/M/1 Queue มีการทำงานดังรูป ที่ 2.1



รูปที่ 2.1 M/M/1 queue system

λ ,$1/\mu$ และ C คือสัญลักษณ์ที่ใช้แทนอัตราเฉลี่ยการเข้ามาของแพ็คเกจ(Packet arrival rate) ขนาดของแพ็คเกจ(Packet size) และความสามารถในการให้บริการของคิว (Service capacity) ตามลำดับ

ขั้นแรก แบบจำลองจะกำหนดความหมายของตัวกำเนิด(Generator) , ตัวจัดการแพ็คเกจ(Queue Packet) และตัวรองรับแพ็คเกจ(Serving Packet) โดยจะกำหนดความสามารถระดับขั้นโมดูล (Module) เมื่อเรามองย่อยลงไปในระดับของ Module จะเห็นได้ว่าการทำงานแบ่งได้เป็น 3 ส่วน ดังนี้

- Queue Module จะประกอบไปด้วยQueue process ย่อย ที่จะทำหน้าที่คอยพิจารณาว่าแพ็คเกจที่ได้รับ จะถูกส่งออกไปหรือเก็บไว้ก่อน ตามลำดับในระบบ FIFO นอกจากนี้จะพิจารณา แพ็คเกจที่ถูกส่งเข้ามาภายในโนด ว่าจะจัดการอย่างไรกับแพ็คเกจที่ถูกส่งเข้ามา

- Transmitter Module เป็นส่วนที่ทำหน้าที่ส่งแพ็คเกจ ออกไปภายนอก Node Model และรับแพ็คเกจที่ถูกส่งจากโนดอื่นๆ ภายใน Transmitter Module จะประกอบด้วยโมดูลที่ทำหน้าที่รับแพ็คเกจ และส่งแพ็คเกจออกไปภายนอก
- Generator Module นั้นจะประกอบไปด้วยโมดูล ที่ทำงานเพื่อสร้างหรือให้กำเนิดแพ็คเกจ

Queue Module เป็นส่วนจัดเรียงแพ็คเกจ จะทำหน้าที่รับแพ็คเกจที่ตัวกำเนิดแพ็คเกจ (Generator Module) สร้างขึ้น แล้วนำมาจัดเรียงและกำหนดเวลาในการส่งตามลำดับไปให้ Transmitter Module

แพ็คเกจที่ถูกส่งมายัง Queue Module นั้นจะมีอัตราการมาถึงของแพ็คเกจ(Packet arrival rate) มีหน่วยเป็น แพ็คเกจ/วินาที ซึ่งนอกจากแพ็คเกจที่มาจากตัวกำเนิดแพ็คเกจ (Generator Module) แล้วแพ็คเกจที่มาถึง Queue Module นั้นยังมีแพ็คเกจที่มาจากภายนอกของโนดอีก ซึ่งการที่จะจัดการอย่างไรกับ แพ็คเกจนี้เป็นหน้าที่ของ Queue Module อาจจะนำส่งออกไปอีก หรือจะเก็บไว้ภายในโนดก็ได้ หรือจะส่งกลับออกไป ซึ่งแล้วแต่คุณสมบัติที่กำหนดให้ Queue Module

2.4 การพัฒนาโปรแกรมแบบวิชวลด้วย C++ Builder 5.0

การพัฒนาโปรแกรมแบบวิชวล คือ การพัฒนาโดยเห็นผลที่จะเกิดขึ้นเมื่อรันโปรแกรมได้ตั้งแต่ในขณะที่กำลังสร้าง โดยนำชิ้นส่วนต่างๆ ที่ต้องการ ได้แก่ ปุ่ม(Botton) ข้อความ (Label) รูปภาพ (Image) ฯลฯ ซึ่งเหล่านี้เรียกโดยรวมๆ ว่า คอมโพเนนต์ (Component) นำมาวางบนวินโดวส์ที่เรียกว่า ฟอร์ม (Form) ปรับขนาดและตำแหน่งรวมทั้งคุณสมบัติต่างๆ ของคอมโพเนนต์ และแม้แต่ฟอร์มเองให้ได้ผลตามที่ต้องการ โดยการเปลี่ยนคุณสมบัติเหล่านี้จะมีผลตั้งแต่ในขณะที่กำลังออกแบบ และเมื่อรันโปรแกรมก็จะได้ผลลัพธ์เหมือนกับที่เห็นในขณะที่ออกแบบ

วิธีเขียนโปรแกรมมีดังนี้

1. สร้างซอร์สโค้ด ซอร์สโค้ดเป็นภาษาอังกฤษที่เขียนขึ้นตามกฎเกณฑ์ของภาษา C++ ผู้ที่รู้ภาษา C++ จะเข้าใจความหมายของซอร์สโค้ดว่าเป็นการสั่งให้คอมพิวเตอร์ทำอะไร ทำอย่างไร และทำเมื่อไร การสร้างซอร์สโค้ดใน C++ บิลเดอร์สามารถทำได้ง่ายและรวดเร็วเพราะ C++ บิลเดอร์จะสร้างซอร์สโค้ดส่วนหนึ่งให้โดยอัตโนมัติ ซึ่งเป็นส่วนที่ทำให้โปรแกรมสามารถทำงานได้ในระดับหนึ่ง ส่วนของซอร์สโค้ดที่จะทำให้โปรแกรมทำงานได้สมบูรณ์เราอาจป้อนเพิ่มเติมเข้าทางคีย์บอร์ดหรือโดย การอ่าน (open) จากไฟล์ในดิสก์ก็ได้

2. คอมไพล์ เมื่อมีซอร์สโค้ดอยู่ในคอมพิวเตอร์เรียบร้อยแล้ว ขั้นต่อไปก็คือการคอมไพล์ ซึ่งเป็นการตรวจสอบว่าโปรแกรมเขียนถูกต้องตามกฎหรือไม่ ถ้ามีที่ผิด คอมไพเลอร์จะแสดงข้อความระบุสาเหตุของความผิด (Compiler Error Message) เราจะต้องแก้ไขและคอมไพล์ใหม่จนถูกต้อง เมื่อไม่พบความผิดแล้วก็ถือว่าโปรแกรมนั้นผ่านการคอมไพล์

3. ลิงก์ สำหรับโปรแกรมที่ผ่านการคอมไพล์แล้ว C++ บิลเดอร์จะนำส่วนประกอบอื่นๆ เช่น โปรแกรมย่อย ข้อมูล ซึ่งโปรแกรมของเราจะต้องใช้เข้ามารวมเพื่อให้โปรแกรมสามารถทำงานได้ตามต้องการ ขั้นตอนนี้เรียกว่าการลิงก์ ซึ่ง C++ บิลเดอร์จะดำเนินการให้แบบอัตโนมัติทันทีที่คอมไพล์ผ่าน เมื่อผ่านการลิงก์แล้วเราจะได้ออบเจ็กต์โค้ดอยู่ในไฟล์ชนิด .EXE

4. รัน เมื่อเป็นออบเจ็กต์โค้ดแล้วโปรแกรมนั้นก็สามารถทำงานได้ การสั่งให้โปรแกรมทำงานเรียกว่าการรัน (Run) ขณะรันโปรแกรมอาจมีความผิดเกิดขึ้นได้อีกเราเรียกความผิดกลุ่มนี้ว่า รันไทม์เออเรอร์ (Runtime Error) ซึ่งโปรแกรมจะหยุดทำงานทันทีและแสดงสาเหตุของความผิดนั้น

2.5 โครงสร้างข้อมูลชนิดอาร์เรย์

อาร์เรย์ (array) เป็นข้อมูลประเภทสตรักเจอร์ (Structure) ซึ่งหมายถึงข้อมูลชุดหนึ่งที่มีจำนวนแน่นอนและเป็นข้อมูลชนิดเดียวกัน ข้อมูลแต่ละรายการเรียกว่า สมาชิกของอาร์เรย์ (Element of array) แต่ละสมาชิกของอาร์เรย์มีอินเด็กซ์ (index) สำหรับใช้เพื่อการอ้างถึงอินเด็กซ์เป็นข้อมูลที่มีลำดับ

วิธีกำหนดให้ข้อมูลเป็นอาร์เรย์ 2 มิติ มีวิธีดังนี้

ชนิดของข้อมูล ชื่ออาร์เรย์ [จำนวนสมาชิกชุดที่ 1][จำนวนสมาชิกชุดที่ 2]

ตัวอย่างเช่น อาร์เรย์เก็บข้อมูลชนิดอินทีเจอร์ซึ่งเป็นลำดับของแพ็คเกจที่ถูกเก็บ 10,000 ค่าคุณสมบัติของแพ็คเกจ 6 เขียนสเตตเมนต์ได้ดังนี้

```
int Packet[10,000][6];
```

บทที่ 3
การออกแบบและพัฒนาระบบ

3.1 การออกแบบ

เริ่มจากลักษณะของการใช้งานโปรแกรม จะเป็นการที่ผู้ใช้มีความประสงค์ที่จะทราบ Time Delay (เวลาที่ระบบเครือข่ายใช้ส่งกลุ่มข้อมูล 1 กลุ่ม ตั้งแต่เริ่มสร้างกลุ่มข้อมูลนั้นจนถึงส่งกลุ่มข้อมูลนั้นจนเสร็จสิ้น) ที่เกิดขึ้นในระบบเครือข่าย ผู้ใช้ (User) จะส่งคุณสมบัติของระบบเครือข่ายที่ต้องการจำลองการทำงานให้กับโปรแกรม จากนั้น โปรแกรมจะจำลองการทำงานตามคุณสมบัติที่ผู้ใช้กำหนด เมื่อได้ผลการจำลองการทำงาน โปรแกรมจะส่งผลการจำลองการทำงานให้ผู้ใช้ได้ทราบดังรูปที่ 3.1

รูปที่ 3.1 ความสัมพันธ์ระหว่างโปรแกรมกับผู้ใช้

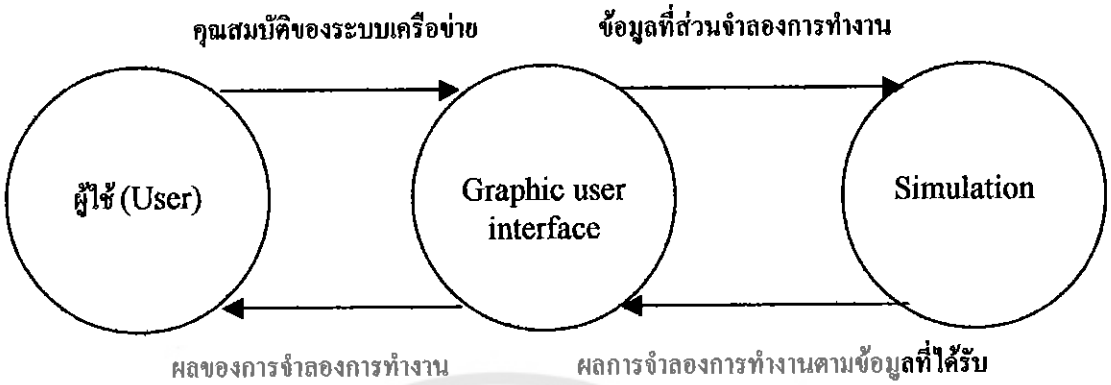
โปรแกรมจำลองการทำงาน จะแบ่งการทำงานออกเป็น 2 ส่วน ที่สัมพันธ์กัน ดังนี้

1. ส่วน Graphic user interface คือ ส่วนที่ติดต่อระหว่างผู้ใช้ (User) กับโปรแกรม (โดยส่วนนี้จะรับคุณสมบัติของระบบเครือข่ายที่ผู้ใช้กำหนด และรับเอาข้อมูลที่โปรแกรมทำการประมวลผลได้ไปแสดงให้ผู้ใช้เข้าใจ)

2. ส่วนการจำลองการทำงาน (Simulation) คือ ส่วนที่จำลองการทำงานตามคุณสมบัติที่กำหนด และหาผลลัพธ์ที่ได้จากการจำลองการทำงาน

ทั้งสองส่วนจะทำงานสัมพันธ์กัน คือ ส่วน Graphic user interface จะรับคุณสมบัติต่างๆ ของระบบเครือข่ายจากผู้ใช้ แปลเป็นข้อมูลที่ทำให้ส่วนการจำลองการทำงานเข้าใจและนำข้อมูลที่ได้ไปใช้ในการจำลองการทำงาน เมื่อส่วนการจำลองการทำงานได้รับข้อมูลแล้วก็จะทำการจำลองการทำงานของระบบเครือข่ายตามข้อมูลที่ได้ เมื่อจำลองการทำงานเสร็จเรียบร้อยแล้วก็จะส่งผลของการจำลองการทำงาน ไปให้กับส่วน Graphic user interface จากนั้นส่วน Graphic user

interface จะแปลผลการทำงานที่ได้รับให้เป็นข้อมูลที่ผู้ใช้เข้าใจ และแสดงให้ผู้ใช้ได้ทราบผลการจำลองการทำงานดังรูปที่ 3.2



รูปที่ 3.2 ความสัมพันธ์ระดับย่อยภายในโปรแกรม

การออกแบบการจำลองการทำงานของระบบ M/M/1 Queue ประกอบด้วยส่วนต่างๆ ดังนี้

Packet คือ กลุ่มของข้อมูลที่ถูกสร้างขึ้น โดยในการออกแบบจะให้แพ็คเกจ แบ่งออกเป็น 6 ส่วน คือ

1. ส่วนที่เก็บเวลาที่แพ็คเกจเกิดขึ้น
2. ส่วนที่เก็บเวลาขณะที่แพ็คเกจเคลื่อนที่ไป
3. ส่วนที่เก็บตำแหน่งที่เกิดแพ็คเกจ
4. ส่วนที่เก็บตำแหน่งที่แพ็คเกจเคลื่อนที่ไป
5. ส่วนที่เก็บตำแหน่งที่แพ็คเกจจะถูกส่งไป
6. ส่วนที่เก็บความยาวทั้งหมดของแพ็คเกจ

Generator คือ ส่วนที่ทำหน้าที่สร้างแพ็คเกจ

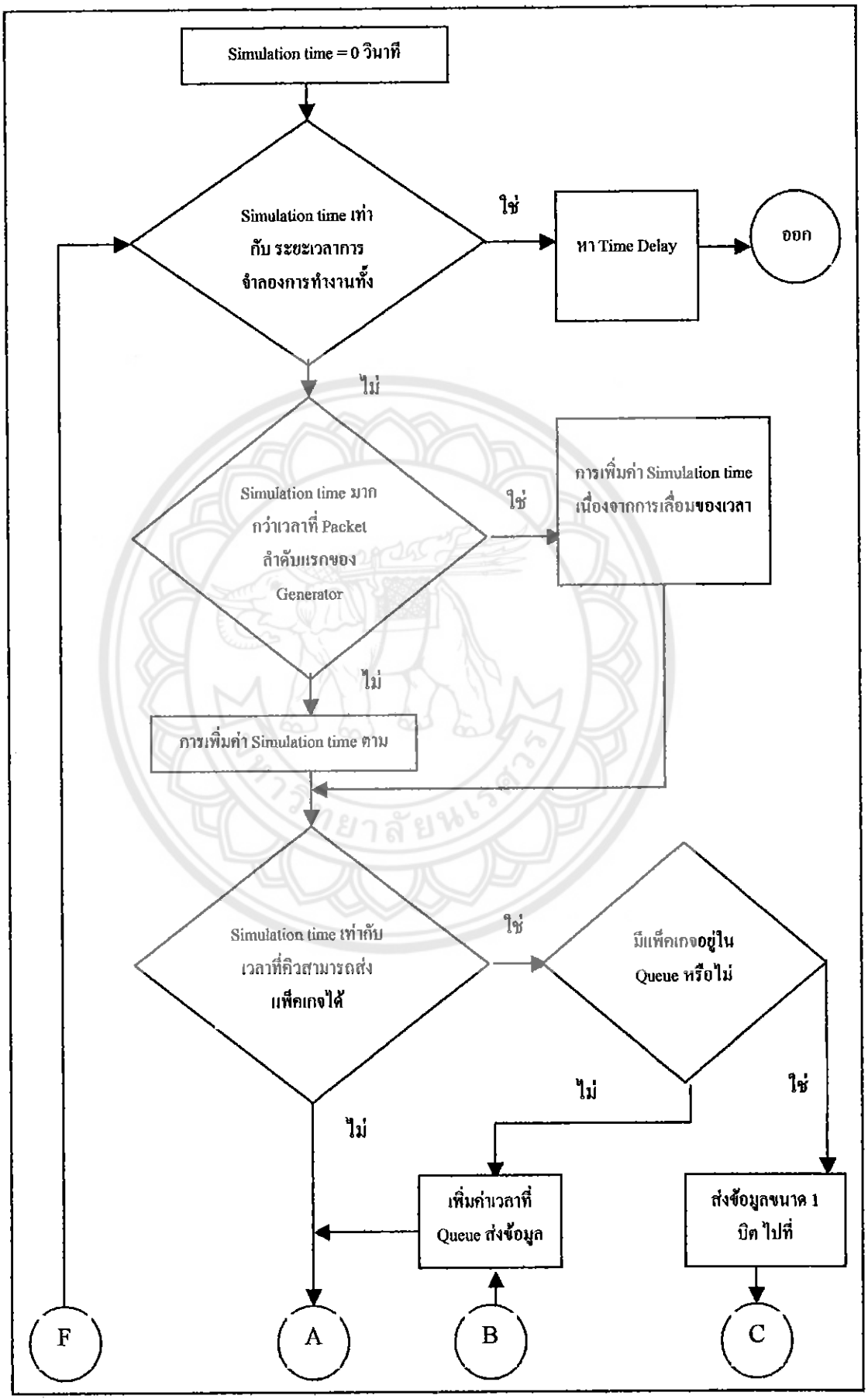
Queue คือ ส่วนที่ทำหน้าที่จัดการการทำงานภายในเครือข่าย ลักษณะการทำงานจะเป็นแถวคอยแบบ First In/First Out (FIFO) คือ เมื่อแพ็คเกจที่เข้ามาในคิว แพ็คเกจที่ถูกส่งมาก่อนก็จะถูกส่งออกไปจากคิว (Queue) ก่อน

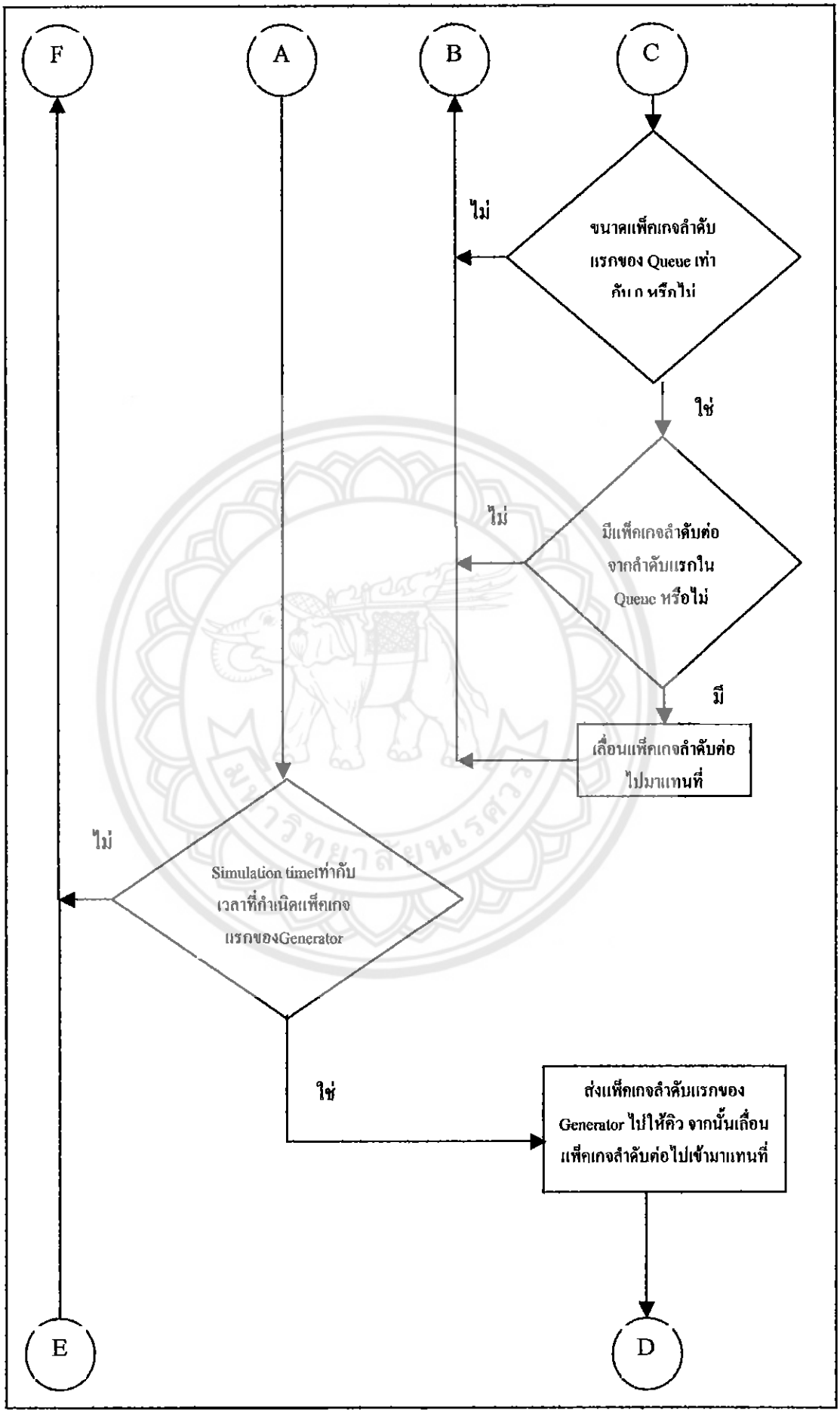
Transmitter คือ ส่วนที่ทำหน้าที่ส่งแพ็คเกจออกจากระบบเครือข่าย

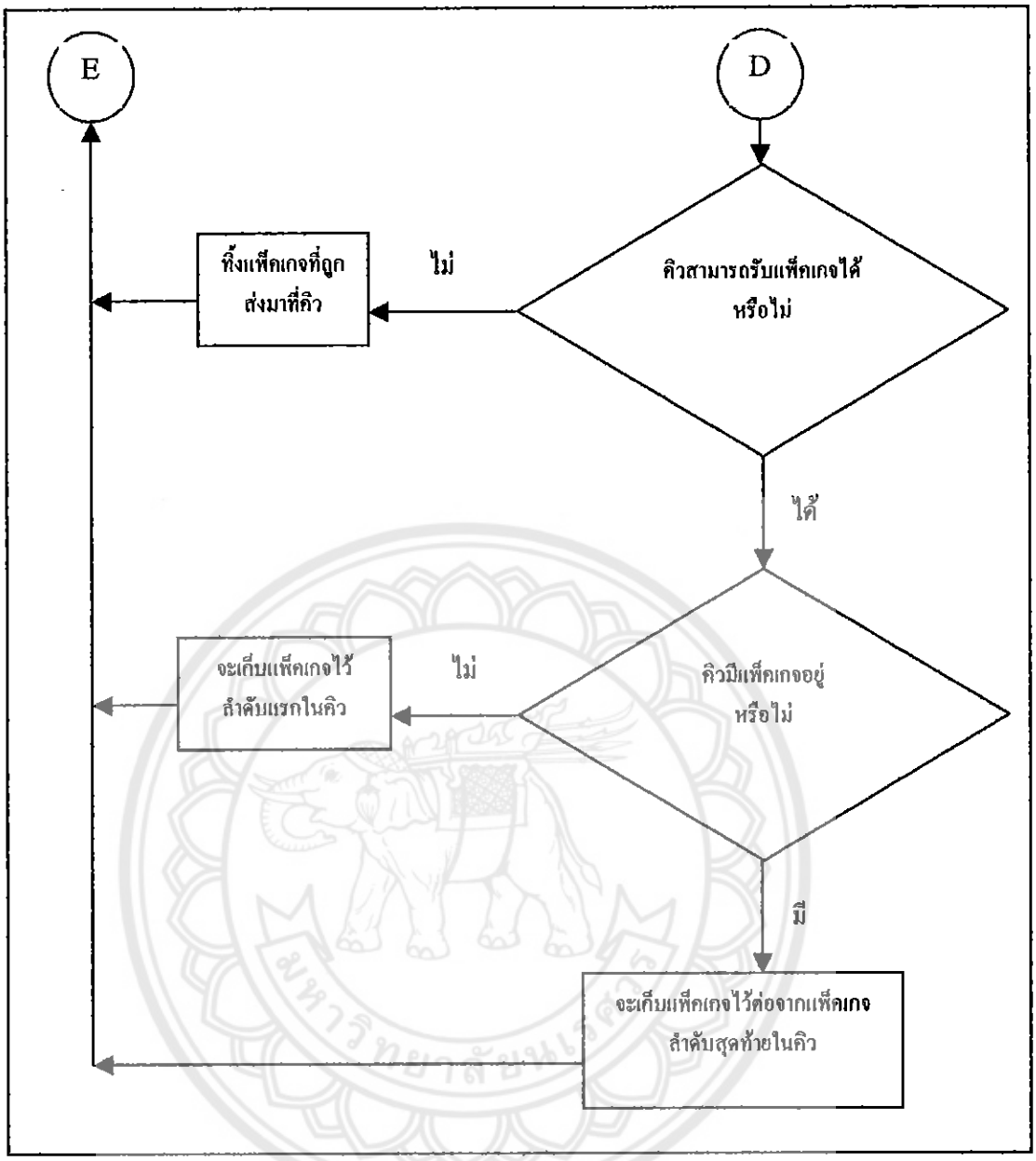
Time คือ เวลาที่สิ้นสุดการจำลองการทำงาน

Simulation Time คือ เวลาในการจำลองการทำงานที่สมมุติขึ้น โดยเมื่อเริ่มจำลองการทำงานจะกำหนดให้มีค่าเท่ากับศูนย์ ขณะที่มีการจำลองการทำงานจะถูกเพิ่มค่าขึ้นไปเรื่อยๆ จนกระทั่งสิ้นสุดการจำลองการทำงาน

รูปแบบของโปรแกรมจำลองระบบเครือข่ายคอมพิวเตอร์เป็นไปตาม Flowchart ดังนี้

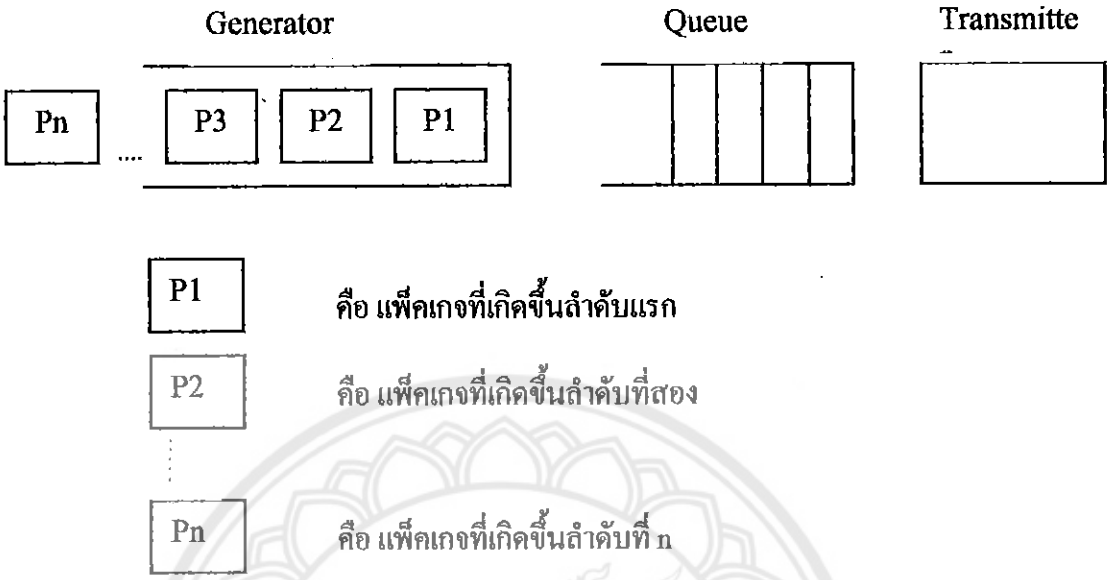



F
A
B
C
ไม่
ขนาดแพ็คเกจลำดับแรกของ Queue เท่ากับ 0 หรือไม่
ใช่
ไม่
มีแพ็คเกจลำดับต่อจากลำดับแรกใน Queue หรือไม่
มี
เลื่อนแพ็คเกจลำดับต่อไปมาแทนที่
ไม่
Simulation timeเท่ากับเวลาที่กำเนิดแพ็คเกจแรกของGenerator
ใช่
ส่งแพ็คเกจลำดับแรกของ Generator ไปให้คิว จากนั้นเลื่อนแพ็คเกจลำดับต่อไปเข้ามาแทนที่
E
D



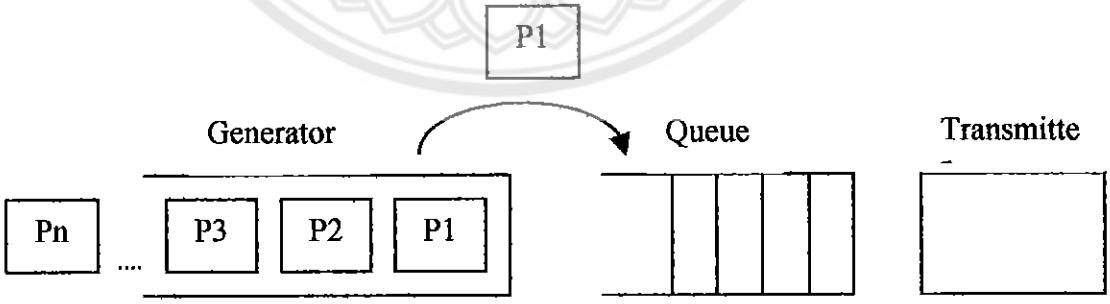
รูปที่ 3.3 แผนผังลำดับการทำงานของโปรแกรม

เมื่อเริ่มการจำลองการทำงานตัวกำเนิดแพ็คเกจจะสร้างแพ็คเกจขึ้นมา โดยเรียงลำดับตามเวลาที่แพ็คเกจเกิดขึ้น แพ็คเกจที่เกิดก่อนเรียงกันไปเรื่อยๆ จนถึงแพ็คเกจสุดท้าย ดังรูปที่ 3.4



รูปที่ 3.4 การสร้างแพ็คเกจของ Generator

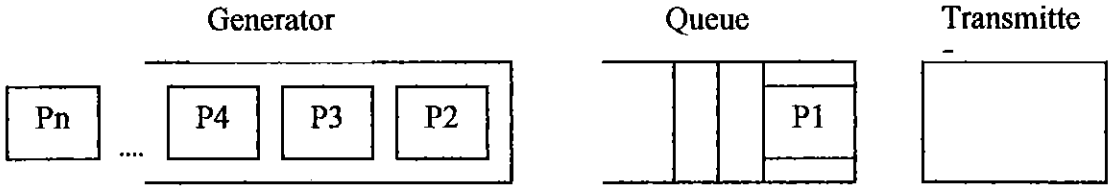
เมื่อ Simulation Time เท่ากับเวลาที่แพ็คเกจลำดับแรกของตัวกำเนิดแพ็คเกจเกิดขึ้น ตัวกำเนิดแพ็คเกจก็จะส่งแพ็คเกจอันดับแรกนั้นไปที่คิว จากนั้นจะเลื่อนแพ็คเกจลำดับต่อไปเข้ามาแทนที่แพ็คเกจเดิมที่ถูกส่งไป ซึ่งแพ็คเกจลำดับต่อไป จากแพ็คเกจที่ถูกเลื่อนไปก็จะเลื่อนมาแทนที่แพ็คเกจที่เลื่อนไปแล้ว ทำแบบนี้ไปเรื่อยๆ จนถึงแพ็คเกจลำดับสุดท้ายดังรูป 3.5



รูปที่ 3.5 การส่งแพ็คเกจจาก Generator สู่ Queue

จากนั้นคิว จะมีการจัดการกับแพ็คเกจที่ถูกส่งเข้ามาเป็น 3 กรณี ดังนี้ คือ

กรณีที่ 1 กรณีที่คิวไม่มีแพ็คเกจอยู่เลย คิวจะนำแพ็คเกจที่ได้รับไปไว้ในตำแหน่งแรกของคิวเพื่อพร้อมที่จะส่งออกไป ดังรูป 3.6



รูปที่ 3.6 เมื่อ Queue รับแพ็คเกจเข้ามา ในกรณีที่ Queue ไม่มีแพ็คเกจอยู่เลย

กรณีที่ 2 กรณีที่คิวมีแพ็คเกจอยู่แต่คิวยังสามารถรับแพ็คเกจอีกได้ คิวจะนำแพ็คเกจที่ได้รับไปต่อท้ายแพ็คเกจลำดับสุดท้ายที่คิวมีอยู่ เช่น เมื่อแพ็คเกจ P2 ถูกส่งมา ที่คิวในขณะที่แพ็คเกจ P1 ยังส่งออกจากคิวไม่เสร็จสิ้น คิวจะนำแพ็คเกจ P2 ไปต่อท้ายแพ็คเกจ P1 ดังรูปที่ 3.7

รูปที่ 3.7 เมื่อ Queue รับแพ็คเกจเข้ามา ในกรณีที่ Queue มีแพ็คเกจอยู่

กรณีที่ 3 กรณีที่คิวมีแพ็คเกจอยู่ และไม่สามารถรับแพ็คเกจได้อีก คิวจะได้รับแพ็คเกจที่ส่งจากตัวกำเนิดและคงสถานะภายในคิวไว้ ส่วนแพ็คเกจที่ถูกส่งมาจะถูกทิ้งไป

เมื่อ Simulation time เท่ากับเวลาที่คิวส่งแพ็คเกจไปยัง Transmitter แล้วคิวจะจัดส่งข้อมูลขนาด 1 บิต ภายในแพ็คเกจอันดับแรกที่อยู่ในคิวไปให้ Transmitter แล้วคิวจะจัดการภายในคิวเอง ซึ่งแบ่งเป็น 3 กรณี คือ

กรณีที่ 1 คิวไม่มีแพ็คเกจอยู่เลย คิวก็จะไม่ส่งข้อมูลให้ Transmitter

กรณีที่ 2 คิวมีแพ็คเกจอยู่เพียงแพ็คเกจเดียว เมื่อส่งข้อมูลไปให้ Transmitter แล้ว แพ็คเกจที่อยู่ในคิวก็จะหมดไป ไม่เหลือแพ็คเกจอยู่ในคิวอีก คิวก็จะไปรอแพ็คเกจที่จะส่งมาจากตัวกำเนิดแพ็คเกจ

กรณีที่ 3 คิวมีแพ็คเกจอยู่มากกว่า 1 แพ็คเกจ เมื่อข้อมูลในแพ็คเกจแรกของคิวถูกส่งไป Transmitter หมดแล้ว คิวจะเลื่อนเอาแพ็คเกจที่อยู่ในลำดับต่อไปมาแทนที่ และจะเลื่อนแพ็คเกจในลำดับต่อจากแพ็คเกจที่เลื่อนไป มาแทนที่แพ็คเกจที่เลื่อนไป ทำอย่างนี้ไปเรื่อยๆ จนถึงแพ็คเกจสุดท้ายภายในคิว

Time Delay คือ เวลาที่แพ็คเกจใช้ไปทั้งหมด ตั้งแต่แพ็คเกจถูกสร้างขึ้นจนถึงเวลาที่แพ็คเกจถูกส่งออกจากโนด หรือถึงปลายทางที่แพ็คเกจถูกกำหนดให้ส่ง

เราจะเริ่มคิด Time Delay ตั้งแต่เวลาที่ถูกส่งออกไปจากตัวกำเนิดแพ็คเกจจนกระทั่งบิตสุดท้ายของแพ็คเกจนั้นถูกส่งไปที่ Transmitter

3.2 การพัฒนาโปรแกรม

จากหลักการทำงานของระบบ M/M/1 Queue ทำให้เรานำมาเป็นแนวคิดและเขียนโปรแกรม เพื่อจำลองการทำงานโดยแบ่งลักษณะของการทำงานออกเป็น 2 ส่วน ดังนี้

1. ส่วนของผู้ใช้โปรแกรม (Graphic user interface)
2. ส่วนการจำลองการทำงาน (Simulation)

1. ส่วนของผู้ใช้โปรแกรม (Graphic user interface)

ส่วนของผู้ใช้โปรแกรม (Graphic user interface) จะเป็นส่วนที่ติดต่อระหว่างผู้ใช้กับโปรแกรม ส่วนนี้จะเป็นส่วนบอกให้ทราบว่า แบบจำลองมีลักษณะเป็นแบบจำลองแบบระบบ M/M/1 Queue ซึ่งจะประกอบไปด้วยโนด (Node) จำนวน 1 โนด และภายในโนดนั้นจะประกอบไปด้วยโมดูล (Module) 3 โมดูล เชื่อมต่อกัน ดังรูปที่ 3.8

ก. Generator ข. Queue Process ค. Transmitter

รูปที่ 3.8 ลักษณะแบบจำลองแบบ M/M/1 Queue

1.1 คุณสมบัติของ Generator

รูปที่ 3.9 Generator

Generator คือ ตัวกำเนิดหรือสร้างแพ็คเกจ (Packet) ซึ่งจะมีคุณสมบัติต่างๆ ดังนี้

1. name คือ กำหนดชื่อของ Generator

2. interarrival pdf* เป็นการกำหนดให้ระยะห่างของการเกิดแพ็คเกจนั้น มีลักษณะอยู่ภายใต้เงื่อนไขตามฟังก์ชันต่างๆ ซึ่งจะมีให้เลือกเป็น 2 ลักษณะ

   - ค่าคงที่ (Constant) ซึ่งหากกำหนดเป็น Constant จะทำให้ตัวกำเนิดแพ็คเกจ (Generator) สร้างแพ็คเกจที่มีระยะห่างของเวลาที่แพ็คเกจเกิดขึ้นนั้นเป็นเวลาที่คงที่มีค่าเท่ากับ ค่าของ interarrival args** ที่กำหนด

---

* pdf ย่อมาจาก Probability density function

** args ย่อมาจาก argument

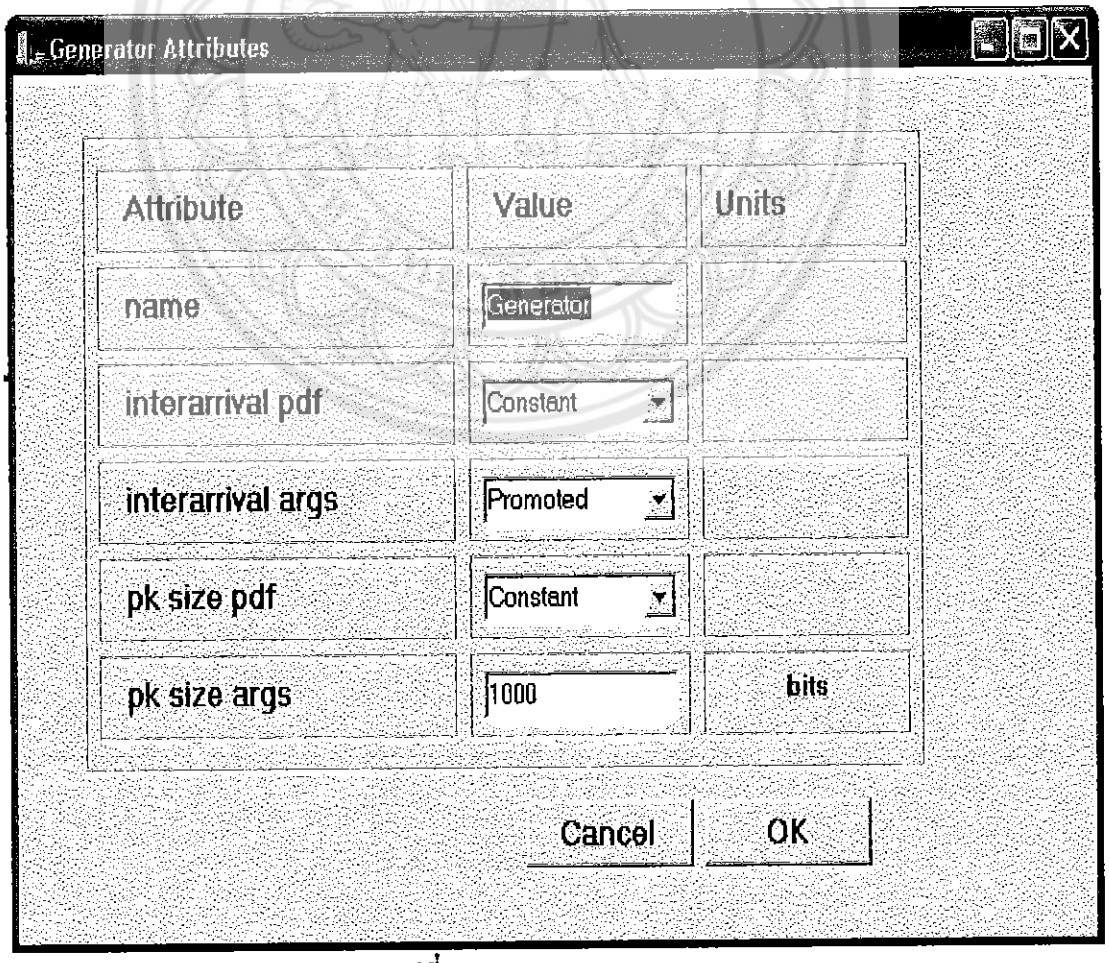
- ค่าเอ็กซ์โพแนนเชียล (Exponential) ซึ่งหากกำหนดเป็น Exponential แล้วตัวกำเนิดแพ็คเกจจะสร้างแพ็คเกจ ที่มีระยะห่างของเวลาที่แพ็คเกจเกิดขึ้นนั้นสอดคล้องกับสมการฟังก์ชัน Exponential

3. interarrival args** เป็นการกำหนดขนาดของระยะห่างระหว่างแพ็คเกจแต่ละแพ็คเกจที่เกิดขึ้น ซึ่งค่าที่กำหนดนี้จะเป็นค่าเฉลี่ยในที่นี้จะกำหนดเป็น Promoted ซึ่งจะกำหนดให้เติมค่า ก่อนการ Simulation

4. packet size pdf* เป็นการกำหนดค่าของขนาดความยาวของแพ็คเกจนั้น มีลักษณะเป็นไปตามฟังก์ชันต่างๆ ซึ่งจะมีให้เลือกเป็น 2 ลักษณะ

- ค่าคงที่ (Constant) ซึ่งหากกำหนดเป็น Constant จะทำให้ตัวกำเนิดแพ็คเกจสร้างแพ็คเกจที่มีขนาดคงที่ตลอดทุกแพ็คเกจที่ตัวกำเนิดแพ็คเกจ สร้างขึ้น
- ค่าเอ็กซ์โปแนนเชียล (Exponential) ซึ่งหากกำหนดเป็น Exponential แล้วตัวกำเนิดแพ็คเกจจะสร้างแพ็คเกจที่มีขนาดสอดคล้อง กับสมการฟังก์ชัน Exponential

5. packet size args** กำหนดขนาดของแพ็คเกจเฉลี่ย

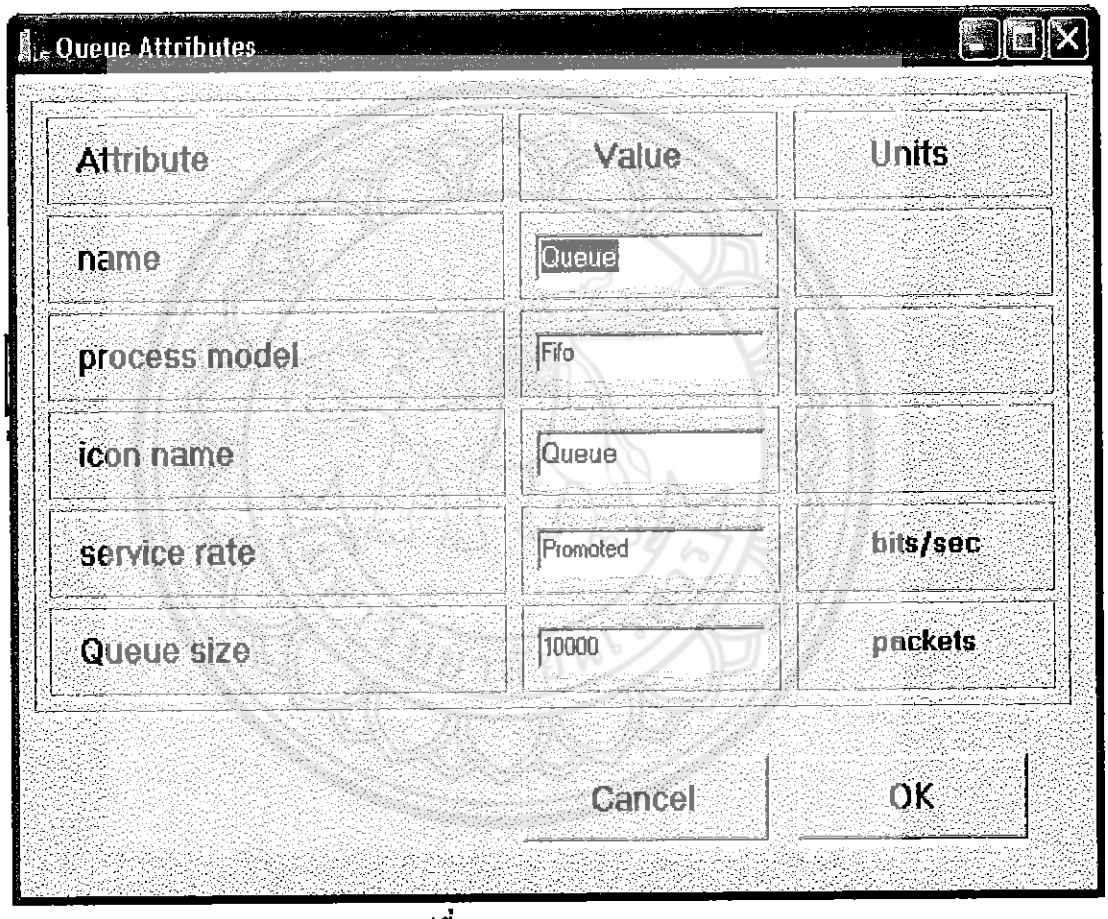


รูปที่ 3.10 Generator Attribute

1.2 คุณสมบัติของ Queue Process

รูปที่ 3.11 Queue

Queue Process เป็น ส่วนที่ทำหน้าที่ในการจัดการภายในระบบ M/M/1 Queue คุณสมบัติของคิว (Queue) มีดังนี้



รูปที่ 3.12 Queue Attributes

1. name กำหนดชื่อของ Queue

2. process model จะกำหนดให้คิวมีลักษณะการทำงานแบบใด ในที่นี้จะกำหนดให้เป็น FIFO (First In /First Out) ซึ่งเป็นลักษณะการทำงานของคิวแบบหนึ่ง มีลักษณะการทำงานดังนี้ คือ ถ้าแพ็คเกจที่เข้ามาสู่ คิวก่อนก็จะถูกส่งออกไปก่อน ส่วนแพ็คเกจที่ตามมาก็จะเรียงต่อกันไป โดยจะไม่มีการแซงคิว

3. icon name คือ ชื่อของลักษณะ โมดูล Module เช่น กำหนดให้เป็น Queue เป็นต้น

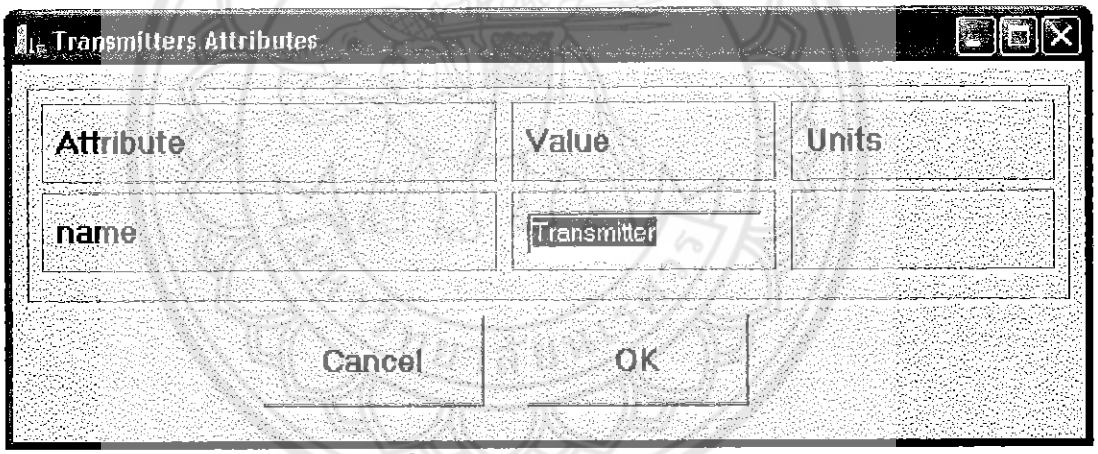
4. service rate คือ อัตราการส่งข้อมูลออกไปจากคิว กำหนดให้เป็น Promoted ซึ่งจะกำหนดให้เติมค่าก่อนการ Simulation

5. Queue size เป็นการกำหนดขนาดของคิว ซึ่งจะเป็นการกำหนดว่าคิวนี้สามารถรองรับ แพ็คเกจได้ทั้งหมดกี่แพ็คเกจ

1.3 คุณสมบัติของ Transmitter

รูปที่ 3.13 Transmitter

Transmitter เป็นส่วนที่ทำหน้าที่ส่งแพ็คเกจออกไปจากโนด โดยจะรับแพ็คเกจต่อจากคิว ซึ่ง Transmitter มีคุณสมบัติดังนี้



รูปที่ 3.14 Transmitter Attributes

1. name คือ กำหนดชื่อของ Transmitter

2. ส่วนการจำลองการทำงาน (Simulation)

การจำลองการทำงาน (Simulation) จะเป็นการนำเอาหลักการของระบบ M/M/1 Queue มาทำการจำลองการทำงาน หลังจากได้รับข้อมูลตามที่กำหนดในส่วนของ Graphic user interface แล้วจะเข้าสู่การจำลองการทำงาน ซึ่งในส่วนการจำลองการทำงาน จะแบ่งออกเป็นส่วนๆ ดังนี้

2.1 ส่วน Generator

รูปที่ 3.15 Generator

ส่วน Generator จะสร้างแพ็คเกจขึ้นมาโดยแพ็คเกจนี้จะมีลักษณะเป็นโครงสร้างข้อมูลชนิดอาร์เรย์ 2 มิติ ซึ่งมิติแรกจะบอกให้ทราบถึงลำดับของแพ็คเกจ มิติที่ 2 จะบอกถึงคุณสมบัติต่างๆ ของแพ็คเกจ ดังนี้

intPacket[n][1] คือ แพ็คเกจลำดับที่ n คุณสมบัติที่ 1 คือ เวลาที่แพ็คเกจเกิดขึ้นเป็นวินาที

intPacket[n][2] คือ แพ็คเกจลำดับที่ n คุณสมบัติที่ 2 คือ เวลาขณะที่แพ็คเกจอยู่ในขณะนั้น

intPacket[n][3] คือ แพ็คเกจลำดับที่ n คุณสมบัติที่ 3 คือ ตำแหน่งที่แพ็คเกจนั้นเกิด

intPacket[n][4] คือ แพ็คเกจลำดับที่ n คุณสมบัติที่ 4 คือ ตำแหน่งที่แพ็คเกจอยู่ในขณะนั้น

intPacket[n][5] คือ แพ็คเกจลำดับที่ n คุณสมบัติที่ 5 คือ ตำแหน่งสุดท้ายของแพ็คเกจ เป็นจุดที่แพ็คเกจถูกส่งไป

intPacket[n][6] คือ แพ็คเกจลำดับที่ n คุณสมบัติที่ 6 คือ ความยาวทั้งหมดของแพ็คเกจ

จากฟังก์ชัน intPacket[n][m]

n คือจำนวนเต็มบวก 0,1,2,3,…,10,000

m คือจำนวนเต็มบวกตั้งแต่ 1 ถึง 6

การสร้างแพ็คเกจของ Generator นั้นจะมีอยู่หลายลักษณะ ขึ้นอยู่กับคุณสมบัติที่กำหนด

ถ้าค่าของ interarrival pdf* มีค่าเป็น Constant จะทำให้ระยะเวลาที่แพ็คเกจเกิดขึ้นนั้นมีความห่างกันในแต่ละแพ็คเกจ เท่ากับ ค่า interarrival args**

ถ้าค่าของ interarrival pdf* มีค่าเป็น Exponential จะทำให้ระยะเวลาที่แพ็คเกจเกิดขึ้นนั้นมีความห่างกันตามสมการ

interarrival time = -(mean interarrival time) ln(1- u)

interarrival time = ระยะเวลาที่แพ็คเกจเกิดห่างกัน มีหน่วยเป็นวินาที

mean interarrival time = interarrival args**

u = ค่าที่สุ่มขึ้นมา มีค่าอยู่ระหว่าง 0-1

ถ้าค่า packet size pdf* มีค่าเป็น Constant จะทำให้ความยาวของแพ็คเกจนั้นคงที่ และจะมีค่าเท่ากับค่าของ packet size args**

ถ้าค่าของ packet size pdf* มีค่าเป็น Exponetial จะทำให้ความยาวของแพ็คเกจนั้นจะเกิดขึ้น สอดคล้องกับสมการ Exponential ดังนี้

packetsize = - (mean packet) ln(1- u)

packetsize = ขนาดของแพ็คเกจ มีหน่วยเป็นบิต (bit)

mean packet = ขนาดของ packet size args** หน่วยเป็นบิต (bit)

u = ค่าที่เกิดจากการสุ่ม มีค่าอยู่ระหว่าง 0-1

2.2 ส่วน Queue

รูปที่ 3.16 Queue

ส่วน Queue จะเป็น อาร์เรย์ขนาด 2 มิติ เช่นเดียวกับ Generator ดังนี้

intPacketQ[n][1] คือ ตำแหน่งรับแพ็คเกจ ลำดับที่ n ซึ่งเก็บค่าเวลาที่แพ็คเกจนั้นเกิดขึ้น

intPacketQ[n][2] คือ ตำแหน่งรับแพ็คเกจลำดับที่ n ซึ่งเก็บค่าเวลาในการจำลองระบบ (Simulation time) ในขณะนั้นไว้

intPacketQ[n][3] คือ ตำแหน่งรับแพ็คเกจลำดับที่ n ซึ่งเก็บค่าของตำแหน่งที่ แพ็คเกจนั้นเกิดขึ้น

intPacketQ[n][4] คือ ตำแหน่งรับแพ็คเกจลำดับที่ n ซึ่งเก็บค่าของตำแหน่งที่ แพ็คเกจนั้นอยู่ในขณะนั้น

intPacketQ[n][5] คือ ตำแหน่งรับแพ็คเกจลำดับที่ n ซึ่งเก็บค่าของตำแหน่งที่ แพ็คเกจนั้นถูกส่งไป

intPacketQ[n][6] คือ ตำแหน่งรับแพ็คเกจลำดับที่ n ซึ่งเก็บค่าความยาวของแพ็คเกจในขณะนั้น

สำหรับคุณสมบัติของอัตราการให้บริการ (service rate) ของคิว นั้นจะเป็นคุณสมบัติที่ทำให้คิว ส่งข้อมูลออกไปจากคิวเป็นจำนวนกี่บิตใน 1 วินาที ส่วน Queue size จะเป็นคุณสมบัติของจำนวนแพ็คเกจที่คิวสามารถรองรับได้

2.3 ส่วน Transmitter

รูปที่ 3.17 Transmitter

ส่วน Transmitter จะมีคุณสมบัติคล้ายกับคิว โดยจะเป็นอาร์เรย์ 2 มิติ คอยรับ แพ็คเกจที่ส่งมาแล้วส่งแพ็คเกจออกไปจากโนด โดยมีคุณสมบัติต่างๆ ดังนี้

intPacketS[n][1] คือ แพ็คเกจที่ส่งออกตัวที่ n เวลาที่เกิดของแพ็คเกจ

intPacketS[n][2] คือ แพ็คเกจที่ส่งออกตัวที่ n เวลาที่ส่งออกไป

intPacketS[n][3] คือ แพ็คเกจที่ส่งออกตัวที่ n ตำแหน่งที่แพ็คเกจเกิด

intPacketS[n][4] คือ แพ็คเกจที่ส่งออกตัวที่ n ตำแหน่งที่แพ็คเกจอยู่ขณะนั้น

intPacketS[n][5] คือ แพ็คเกจที่ส่งออกตัวที่ n ตำแหน่งปลายทางของแพ็คเกจ

intPacketS[n][6] คือ แพ็คเกจที่ส่งออกตัวที่ n ความยาวแพ็คเกจ

สำหรับ Transmitter จะส่งแพ็คเกจออกตลอดเวลา ไม่มีการพักรอเหมือนกับคิว

ประเภทของการจำลองการทำงาน (Simulation)

การจำลองการทำงาน (Simulation) จะแบ่งเป็น 2 แบบ ดังนี้

1. การจำลองการทำงาน (Simulation) ในช่วงเวลา
2. การจำลองการทำงาน (Simulation) จนได้จำนวนแพ็คเกจตามที่ต้องการ

1. การจำลองการทำงาน (Simulation) ในช่วงเวลา

การจำลองการทำงาน (Simulation) ในช่วงเวลา คือ การจำลองการทำงาน โดยกำหนดระยะเวลาการจำลองการทำงาน เมื่อกำหนดระยะเวลาจำลองการทำงาน แล้วตัวกำเนิดแพ็คเกจ (Generator) จะนำเอาช่วงเวลานี้ไปสร้าง แพ็คเกจขึ้นมา เช่น ถ้าระยะเวลาการ จำลองการทำงานเท่ากับ 100 วินาที ตัวกำเนิดแพ็คเกจจะสร้างแพ็คเกจขึ้นมาตามคุณสมบัติที่กำหนด โดยการสุ่มเวลาที่แพ็คเกจแรกเกิดขึ้น จากนั้นก็บวกด้วยค่า interarrival args** จะได้เวลาที่แพ็คเกจตัวที่ 2 เกิดขึ้น ส่วนแพ็คเกจตัวที่ 3 ก็เอาเวลาที่เกิดแพ็คเกจที่ 2 บวกกับ interarrival args** ทำแบบนี้ไปเรื่อยๆ จน

ได้แพ็คเกจที่มีเวลาที่เกิดแพ็คเกจมากกว่าหรือเท่ากับ 100 วินาทีหรือ ระยะเวลาการจำลองการทำงาน จากนั้นการจำลองการทำงานก็จะเริ่มขึ้น

โดยจะเริ่มกำหนดเวลาในการจำลองการทำงานที่สมมุติขึ้น (Simulation Time) ให้มีค่าเท่ากับ 0 แล้วจะเริ่มต้นตามขั้นตอนดังนี้

ขั้นตอนที่ 1 เพิ่มค่าให้กับ Simulation Time โดยการเพิ่มค่าของ Simulation Time มีอยู่ 2 ลักษณะดังนี้

- การเพิ่มค่า Simulation Time ตามปกติ จะเป็นการนำเอาค่า Simulation Time ไปบวกกับเวลาในการส่งข้อมูลขนาด 1 บิต ออกจากคิวในการเพิ่มค่าตามปกตินี้ จะเกิดขึ้นเมื่อการ Simulation Time ก่อนหน้าที่จะนำมาเพิ่มค่า มีค่าน้อยกว่าหรือเท่ากับค่าของเวลาการเกิดของแพ็คเกจลำดับแรก ของตัวกำเนิดแพ็คเกจ ดังตัวอย่างต่อไปนี้

  Ex.1 ให้ Simulation Time เท่ากับ 10 วินาที เวลาในการส่งข้อมูลขนาด 1 บิต ออกจากคิว เท่ากับ 0.3 วินาที เวลาที่แพ็คเกจแรกของตัวกำเนิดแพ็คเกจ เท่ากับ 11 วินาที จากค่าที่ได้ จะเพิ่มค่า Simulation Time เท่ากับ 10 + 0.3 = 10.3 วินาที

- การเพิ่มค่า Simulation Time เนื่องจากการเลื่อมของเวลา จะเกิดขึ้นเมื่อเวลา Simulation Time มีค่ามากกว่า เวลาการเกิดของแพ็คเกจแรกของตัวกำเนิดแพ็คเกจ ค่า Simulation Time จะถูกลดลงให้เท่ากับค่าของเวลาการเกิดของแพ็คเกจแรก ของตัวกำเนิดแพ็คเกจ จากนั้นก็จะทำการจำลองการทำงานต่อไป แล้ว Simulation Time จะถูกเพิ่มขึ้นเท่ากับค่าเวลา Simulation Time ก่อนที่จะถูกลดค่าลง ดังตัวอย่างต่อไปนี้

  Ex.2 ให้ Simulation Time เท่ากับ 11.2 วินาที เวลาในการส่งข้อมูลขนาด 1 บิต ออกจากคิวเท่ากับ 0.3 วินาที เวลาที่แพ็คเกจแรกของตัวกำเนิดแพ็คเกจเท่ากับ 11 วินาที จากค่าที่ได้ ค่าSimulation Time จะถูกลดลงเท่ากับ 11 วินาที และเวลา SimulationTime ครั้งต่อไปจะเท่ากับ 11.2 วินาที เหมือนเดิม

ขั้นตอนที่ 2 เวลาในการจำลองการทำงาน (Simulation time) เท่ากับเวลาที่คิวสามารถส่งแพ็คเกจได้

รูปที่ 3.18 คิวส่งแพ็คเกจให้ Transmitter

คิวจะตรวจดูว่าที่ลำดับแรกของคิวมีแพ็คเกจอยู่หรือไม่ ถ้ามีก็ให้ส่งข้อมูล 1 บิตในแพ็กเกจนั้นไปให้ Transmitter

เวลาในการที่คิวสามารถส่งแพ็คเกจได้จะเพิ่มขึ้น โดยบวกกับเวลาที่คิวใช้ในการส่งข้อมูล 1 บิต ถ้าลำดับแรกของคิวมีขนาดแพ็คเกจเท่ากับศูนย์แล้ว คิวจะเลื่อนแพ็คเกจลำดับต่อไปมาแทนที่ลำดับแรก แพ็คเกจลำดับต่อๆ ไป ก็จะถูกเลื่อนเข้ามาเรื่อยๆ จนถึงแพ็คเกจลำดับสุดท้ายภายในคิว แล้วก็จะไปทำงานขั้นตอนต่อไปแทน แต่ถ้าในคิวไม่มีแพ็คเกจ อยู่เลย คิวก็จะไม่เกิดการส่งข้อมูลไปให้ Transmitter

ขั้นตอนที่ 3 ถ้าเวลาในการจำลองการทำงาน (Simulation time) เท่ากับเวลาการกำเนิดแพ็คเกจลำดับแรก

รูปที่ 3.19 ตัวกำเนิดแพ็คเกจส่งแพ็คเกจให้คิว

ตัวกำเนิดแพ็คเกจจะทำการเช็คว่าคิวมีที่ว่างหรือไม่ โดยเช็คจากลำดับแรกของคิวไปจนถึงลำดับสุดท้ายของคิว ถ้าคิวว่าง ตัวกำเนิดแพ็คเกจจะส่งแพ็คเกจลำดับแรกของตัวกำเนิดแพ็คเกจ ไปที่คิว ตำแหน่งที่ว่างอยู่ แล้วจะเลื่อนแพ็คเกจลำดับต่อไปในตัวกำเนิดแพ็คเกจเข้ามาแทนที่ลำดับแรก แล้วเลื่อนแพ็คเกจลำดับต่อๆ ไป เข้ามาเรื่อยๆ จนถึงแพ็คเกจลำดับสุดท้ายในตัวกำเนิดแพ็คเกจ ถ้าคิวไม่ว่างตัวกำเนิดแพ็คเกจก็จะทิ้งแพ็คเกจนั้นไปแล้วเลื่อนแพ็คเกจลำดับต่อไปในตัวกำเนิดแพ็คเกจเข้ามาแทนที่ลำดับแรก แล้วเลื่อน แพ็คเกจลำดับต่อๆ ไป เข้ามาเรื่อยๆ จนถึงแพ็คเกจลำดับสุดท้ายในตัวกำเนิดแพ็คเกจ

ขั้นตอนที่ 4 Time Delay จะหาจากเวลาในการเกิดแพ็คเกจจนกระทั่งเวลาที่แพ็คเกจถูกส่งออกไป (Transmitter) นั่นคือ ตรวจสอบดูว่า เวลาในการจำลองการทำงาน (Simulation time) มีค่ามากกว่าหรือเท่ากับระยะเวลาของการจำลองการทำงานหรือไม่? ถ้าน้อยกว่าให้ไปทำขั้นตอนที่ 1,2 และ 3 ตามลำดับ ถ้าเท่ากับ จะเป็นการสิ้นสุดการจำลองการทำงาน

intPacketS[n][2] - intPacketS[n][1]

n คือ ลำดับที่แพ็คเกจมาถึง

2. <u>การจำลองการทำงาน (Simulation) จนได้จำนวนแพ็คเกจที่ต้องการ</u>

การจำลองการทำงาน จนได้จำนวนแพ็คเกจที่ต้องการ คือ การจำลองการทำงาน โดยการนับจำนวนแพ็คเกจที่ส่งได้สำเร็จ ลักษณะการทำงานโดยทั่วไปจะเหมือนกับแบบการจำลองการทำงาน ในช่วงเวลา แต่แตกต่างกันดังนี้

1. ตัวกำเนิดแพ็คเกจจะสร้างแพ็คเกจขึ้นมา 1000 แพ็คเกจ จากนั้นทำการจำลองการทำงานไปเรื่อยๆ จน แพ็คเกจเหลือเพียง 5 แพ็คเกจ ตัวกำเนิดแพ็คเกจก็จะสร้างแพ็คเกจเพิ่มขึ้นต่อจากแพ็คเกจ ลำดับที่ 5 อีก 1000 แพ็คเกจ ทำแบบนี้ไปจนกระทั่งได้แพ็คเกจที่ส่งสำเร็จตามต้องการ

2. เวลาในการจำลองการทำงานนั้น จะเพิ่มขึ้นเรื่อย ๆ จนกระทั่งได้แพ็คเกจที่ส่งสำเร็จตามต้องการแล้วจึงจะหยุดการทำงาน

3.3 การหาค่า Throughput

การหาค่า Throughput นั้น คือ อัตราส่วนของแพ็คเกจที่ถูกส่งสำเร็จ ต่อจำนวนแพ็คเกจที่ถูกสร้างและส่งออกจากตัวกำเนิดแพ็คเกจทั้งหมด ส่วนค่าแพ็คเกจที่ถูกส่งสำเร็จนั้นคือ แพ็คเกจที่มาถึง Transmitter นั้นเอง

บทที่4

ผลการทดลองและผลการวิเคราะห์

4.1 จุดประสงค์ของการทดลอง

4.1.1 เพื่อทดสอบการทำงานของโปรแกรม

4.1.2 เพื่อวัดประสิทธิภาพการทำงานของโปรแกรม

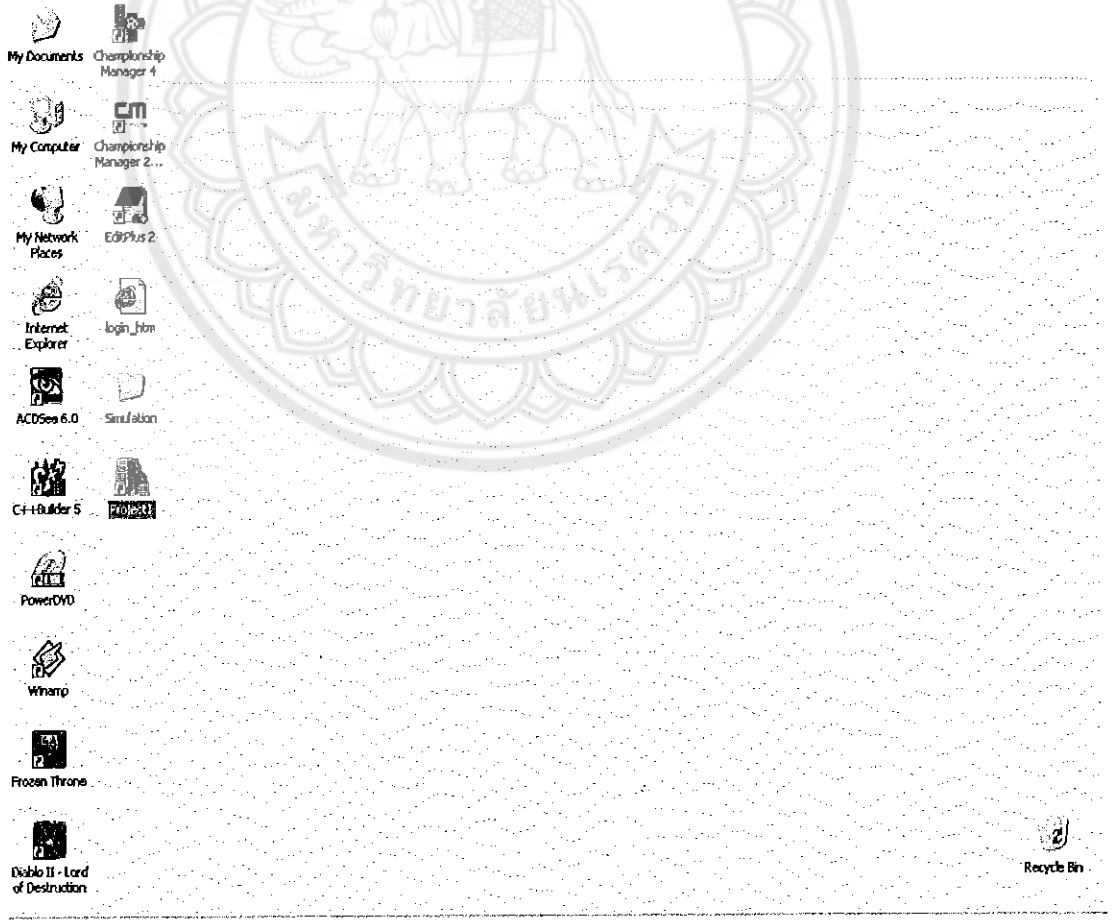
4.2 ขั้นตอนการทดลอง

4.2.1 ทดสอบการทำงานของโปรแกรม

วิธีการทดสอบ

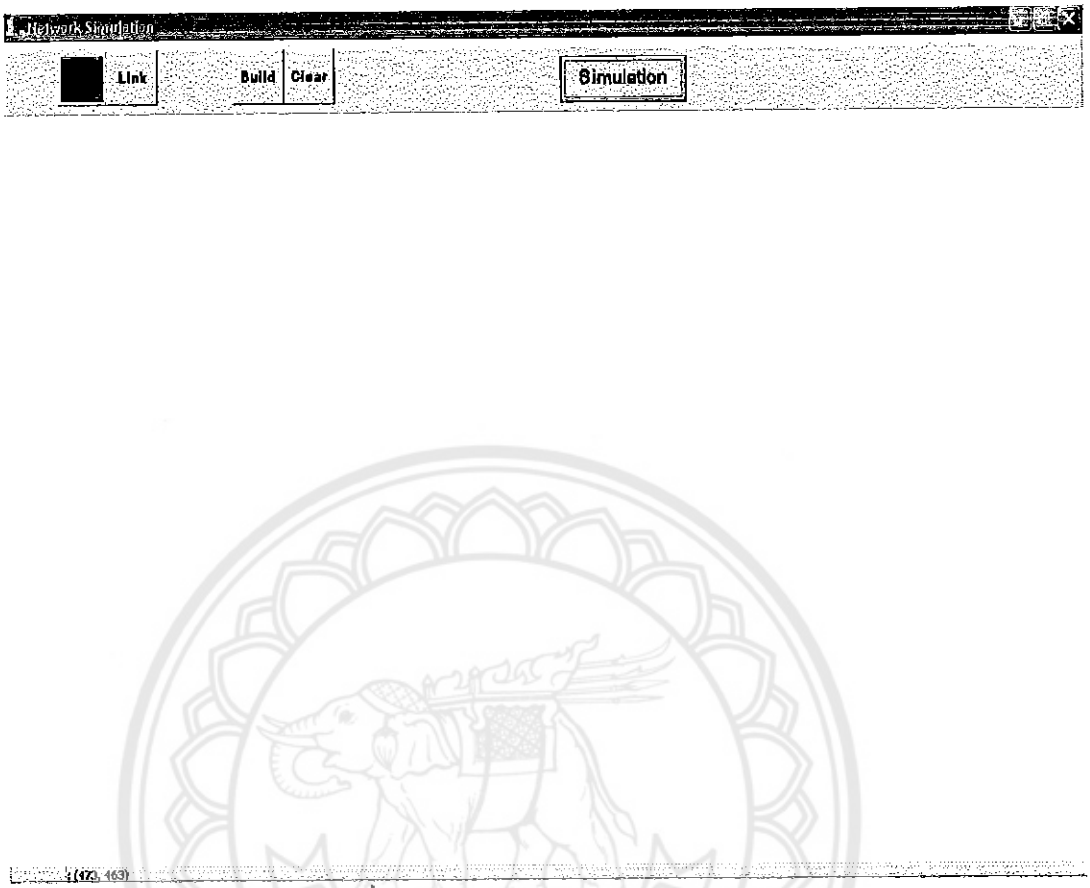
การทดสอบโปรแกรม Network Simulation จะมีขั้นตอนดังนี้

1. เปิดโปรแกรม Project1.exe ดังรูปที่ 4.1



รูปที่ 4.1 เปิดโปรแกรม Network Simulation

2. จะเข้าสู่โปรแกรม Network Simulation ดังรูปที่ 4.2



รูปที่ 4.2 โปรแกรม Network Simulation

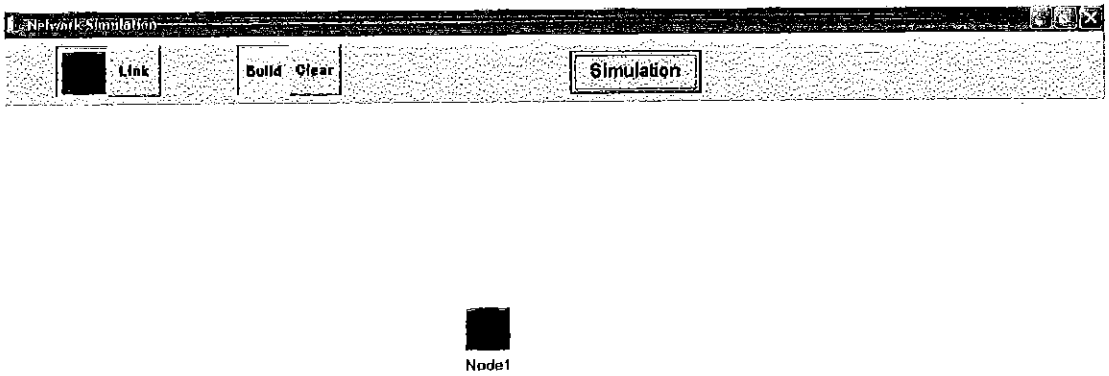
3. เครื่องมือที่ควบคุมการทำงานจะแบ่งเป็น 2 ส่วน คือ

- ส่วนที่บอกลักษณะ คือ ปุ่มNode และ ปุ่มLink Link
- ส่วนที่บอกลักษณะการกระทำคือ ปุ่มBuild Build และ ปุ่มClear Clear

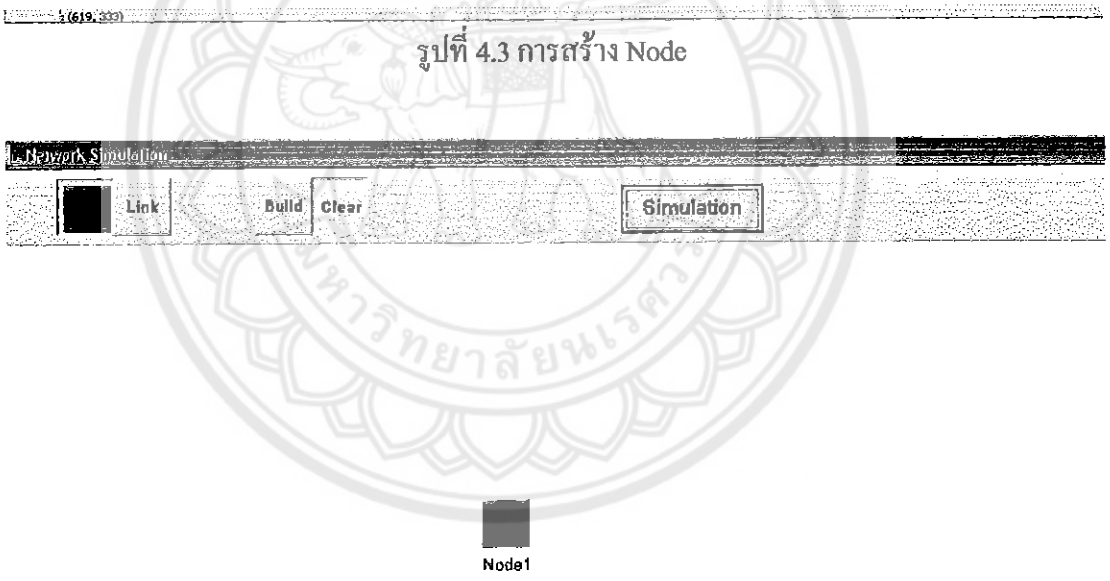
ถ้าจะสร้าง Node เริ่มจากการเลือกคลิกปุ่มNode แล้วตามด้วยคลิกปุ่มBuild แล้วคลิกซ้ายในบริเวณสีขาว ดังรูปที่ 4.3

ถ้าจะลบหรือยกเลิก Node เดิมที่สร้างไว้แล้ว เริ่มจากการเลือกคลิกปุ่มNode แล้วตามด้วยคลิกปุ่มClear แล้วคลิกซ้ายในบริเวณ Node ที่ต้องการลบหรือยกเลิก ดังรูปที่ 4.4

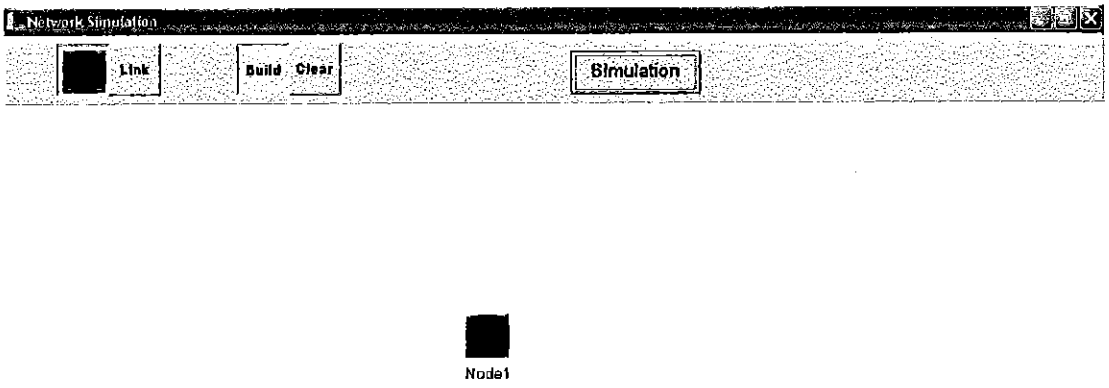
ถ้าจะเลื่อน Node เริ่มจากการเลือกคลิกปุ่มNode แล้วตามด้วยคลิกปุ่มBuild แล้วคลิกซ้ายในบริเวณ Node ที่ต้องการเลื่อน ดังรูปที่ 4.5 จากนั้นเลื่อนเมาส์ (Mouse) ไปคลิกจุดที่ต้องการเลื่อน Nodeไปที่จุดนั้น Nodeที่ต้องการเลื่อนก็จะไปอยู่ที่ตำแหน่งนั้น ดังรูปที่ 4.6



รูปที่ 4.3 การสร้าง Node



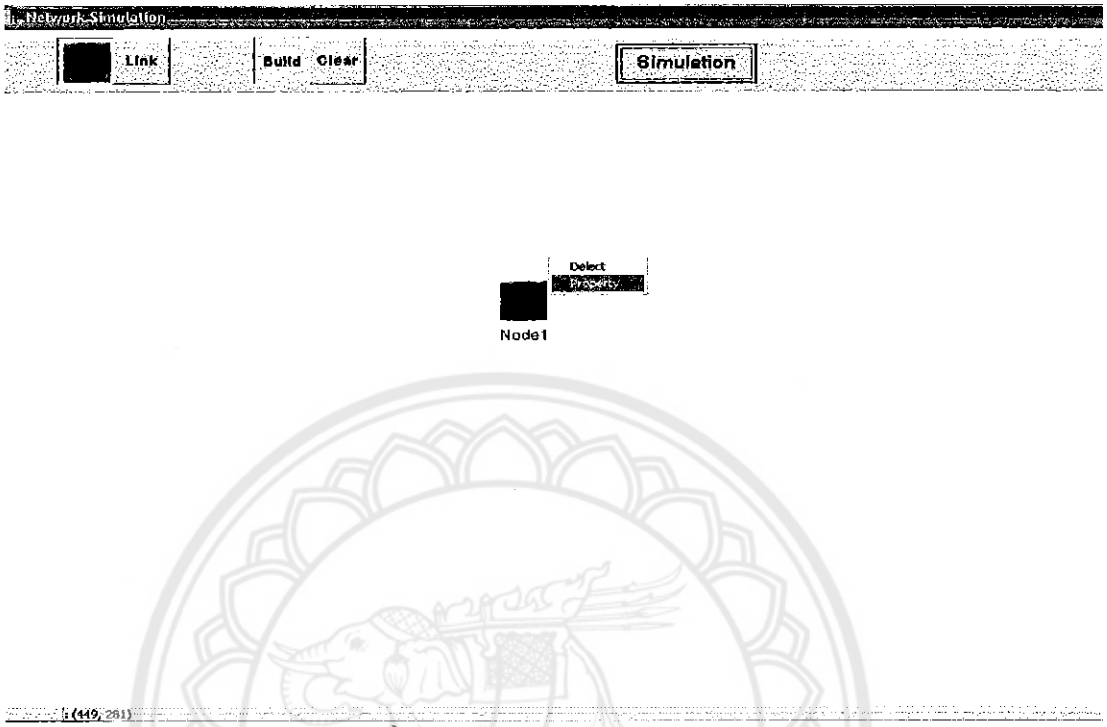
รูปที่ 4.4 การลบ หรือยกเลิก Node ที่สร้างขึ้น



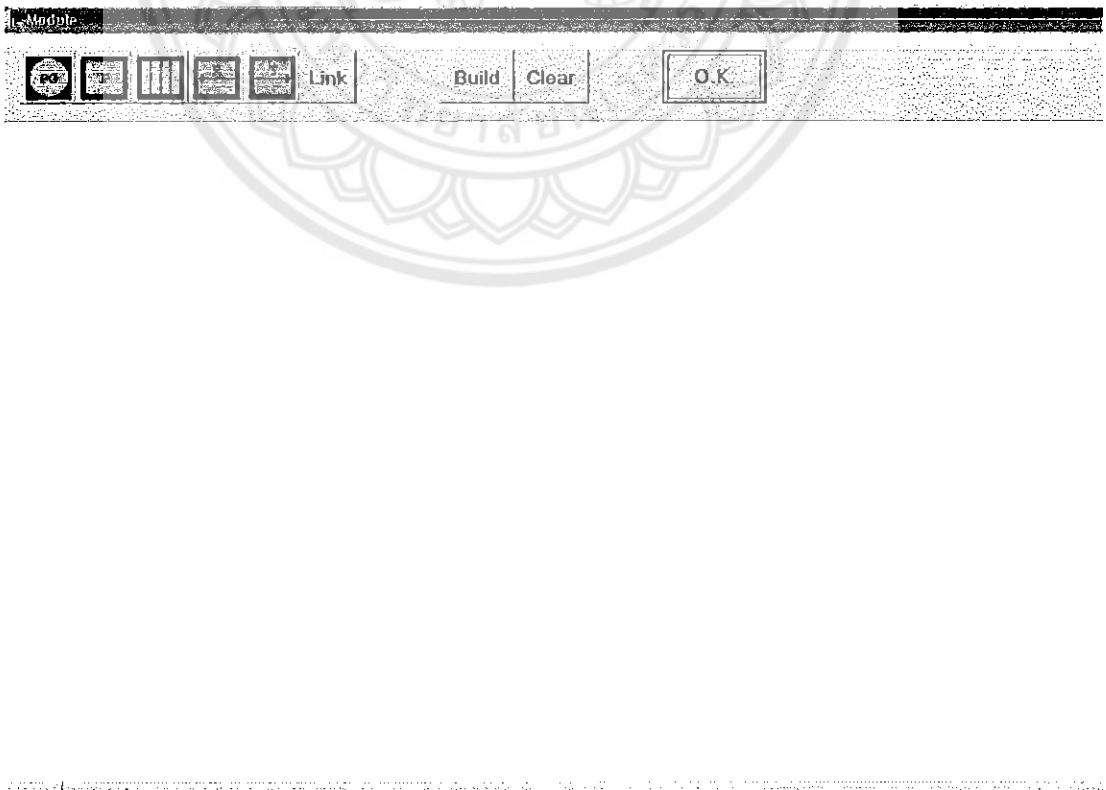
รูปที่ 4.5 การเลื่อน Node ที่สร้างขึ้นโดยเริ่มนำเมาส์ (Mouse) ไปคลิกที่ Node

รูปที่ 4.6 หลังการเลื่อน Node ที่สร้างขึ้น

3. กำหนดคุณสมบัติของ Node โดยคลิกขวาที่ Node จะมีเมนูให้เลือก Property ดังรูปที่ 4.7 และเข้าสู่คุณสมบัติของ Node นั้น (หน้าต่าง Module) ดังรูปที่ 4.8



รูปที่ 4.7 กำหนดคุณสมบัติของ Node



รูปที่ 4.8 หน้าต่างระดับ Module ของ Node

5. เครื่องมือที่ควบคุมการทำงานภายใน Node (หน้าต่าง Module) จะแบ่งเป็น 2 ส่วน คือ

- ส่วนที่บอกลักษณะคือ ปุ่มGenerator  ,ปุ่ม Processors  ,ปุ่มQueue  , ปุ่ม Receiver  ,ปุ่มTransmitter  และ ปุ่มLink  (Link คือ การเชื่อมโยงเข้าด้วยกัน)
- ส่วนที่บอกลักษณะการกระทำคือ ปุ่มBuild  และปุ่มClear

ถ้าจะสร้าง Generator, Processors, Queue, Receiver, Transmitter หรือ Link เริ่มจากการเลือกคลิกที่ปุ่ม Generator, ปุ่ม Processors, ปุ่ม Queue, ปุ่ม Receiver, ปุ่ม Transmitter หรือ ปุ่มLink แล้วตามด้วยคลิกปุ่มBuild แล้วคลิกซ้ายในบริเวณสีขาว เหมือนกับลักษณะการสร้าง Node

ถ้าจะลบหรือยกเลิก Generator, Processors, Queue, Receiver, Transmitter และ Link เดิมที่สร้างไว้แล้ว เริ่มจากการเลือก ปุ่ม Generator,ปุ่ม Processors,ปุ่ม Queue,ปุ่ม Receiver,ปุ่ม Transmitter หรือปุ่ม Link แล้วตามด้วยคลิกปุ่ม Clear แล้วคลิกซ้ายในบริเวณ Generator, Processors, Queue, Receiver, Transmitter หรือ Link ที่ต้องการลบ เหมือนกับลักษณะการสร้าง Node

ถ้าจะเลื่อน Generator, Processors, Queue, Receiver หรือ Transmitter เริ่มจากการเลือกปุ่ม Generator,ปุ่ม Processors,ปุ่ม Queue,ปุ่ม Receiver หรือปุ่ม Transmitter แล้วตามด้วยคลิกปุ่ม Build แล้วคลิกซ้ายในบริเวณ Generator, Processors, Queue, Receiver และTransmitter ที่ต้องการเลื่อน จากนั้นเลื่อน Mouse ไปคลิกจุดที่ต้องการเลื่อนไป Generator, Processors, Queue, Receiver และTransmitter ที่ต้องการเลื่อนก็จะไปอยู่ที่ตำแหน่งนั้น

5.1 การสร้าง Generator จะเริ่มจากการเลือกคลิกที่ปุ่ม Generator  แล้วตามด้วยคลิกปุ่ม Build  แล้วคลิกบริเวณที่ต้องการสร้าง Generator ดังรูปที่ 4.9 แล้วคลิกขวาที่ Generator และเลือก Property เพื่อที่จะกำหนดคุณสมบัติของ Generator (หน้าต่าง Generator Attributes) ดังรูปที่ 4.10

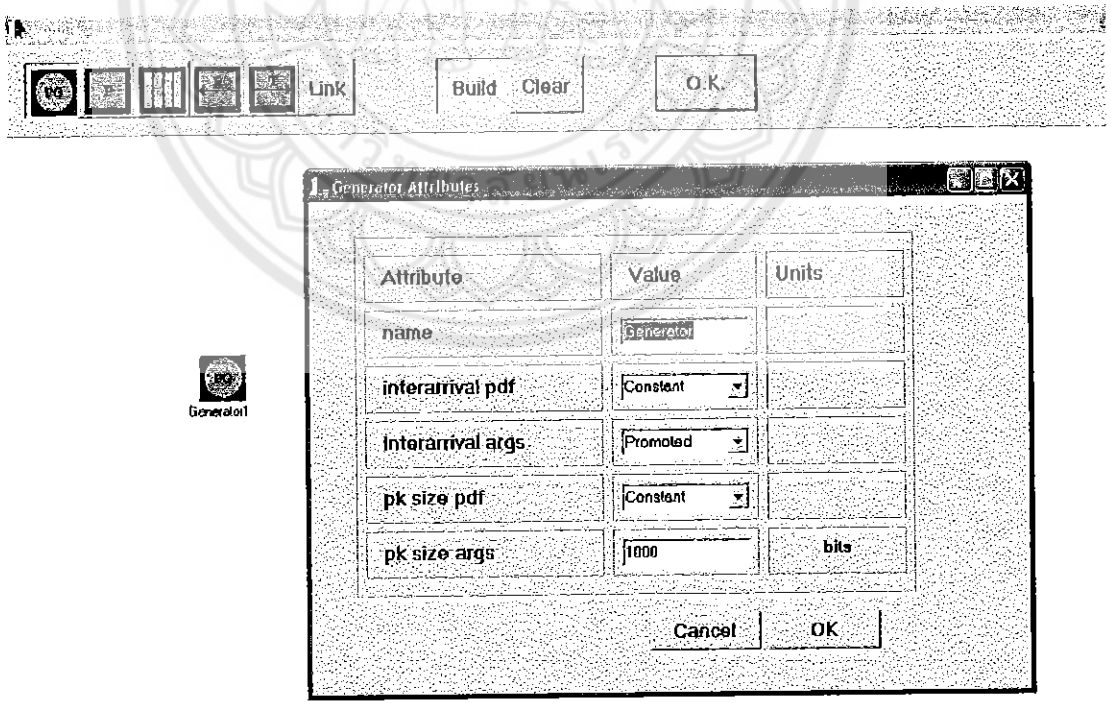
สามารถกำหนดคุณสมบัติของ Generator ได้ดังนี้

1. name คือ ให้เติมชื่อของ Generator

2. interarrival pdf *คือ ให้เลือกคุณสมบัติ Interarrival Probability Density Function จะมีให้เลือก 2 แบบ คือ

- Constant(ค่าคงที่)
- Exponential(ค่าเอ็กซ์โพเนนต์เชียล)

รูปที่ 4.9 การสร้าง Generator



รูปที่ 4.10 การกำหนดคุณสมบัติของ Generator

3. interarrival args** คือ ให้กำหนดคุณสมบัติ interarrival args** ซึ่งจะกำหนดให้เป็น Promoted

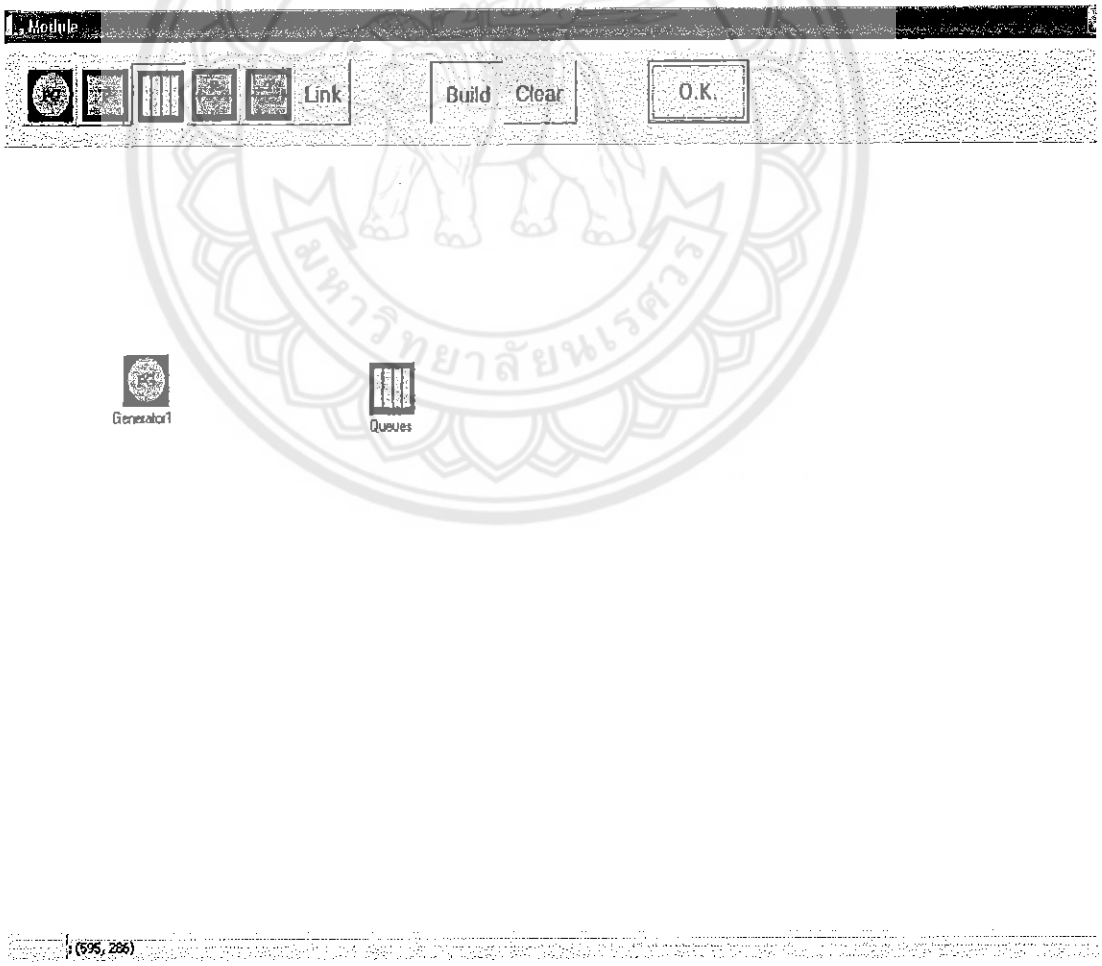
4. pk size pdf* คือกำหนดคุณสมบัติ Packet Size Probability Density Function จะมีให้เลือก 2 แบบคือ

- Constant (ค่าคงที่)
- Exponential (ค่าเอ็กซ์โพเนนต์เชียล)

5. pk size args** คือ ให้กำหนดคุณสมบัติ Packet Size Argument ซึ่งจะเป็นการกำหนดความยาวเฉลี่ยของแพ็คเกจที่ Generator สร้างขึ้น ซึ่งจะกำหนดค่าได้ตั้งแต่ 0-10,000

เมื่อกำหนดคุณสมบัติเรียบร้อยแล้วก็กดปุ่ม OK จะเป็นการสิ้นสุดการกำหนดคุณสมบัติของ Generator

5.2 การสร้าง Queue Process จะเริ่มจากการเลือกคลิกที่ปุ่มQueue แล้วตามด้วยที่คลิกปุ่มBuild แล้วคลิกบริเวณที่ต้องการสร้าง Queue ดังรูปที่ 4.11



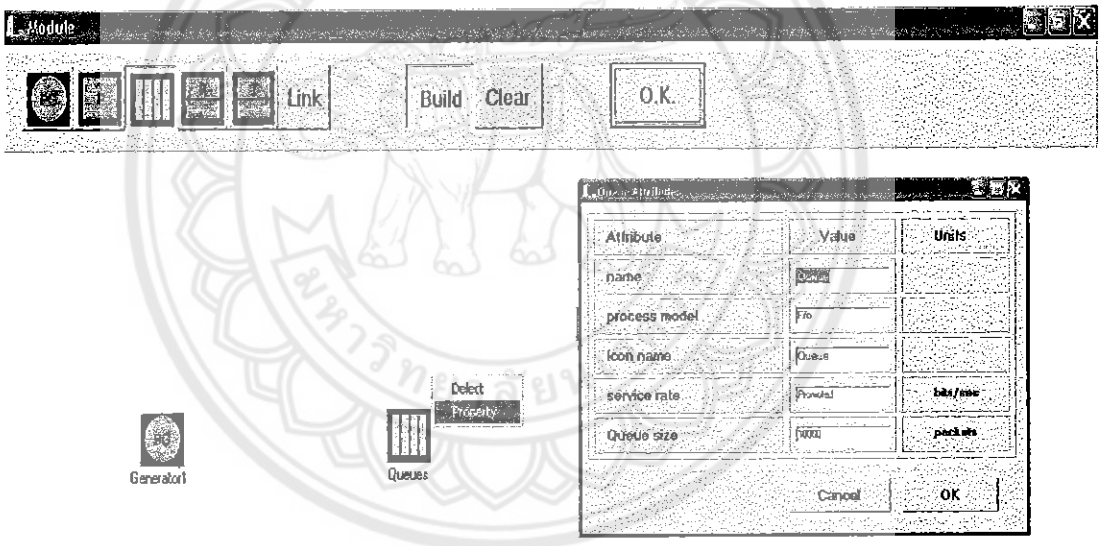
รูปที่ 4.11 การสร้าง Queue

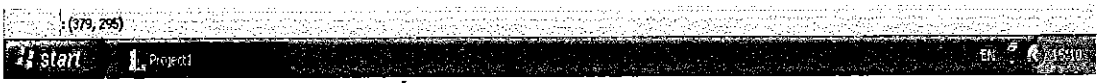
คลิกขวาที่ Queue และเลือก Property เพื่อที่จะกำหนดคุณสมบัติของ Queue (หน้าต่าง Queue Attribute) ดังรูปที่ 4.12

สามารถกำหนดคุณสมบัติของ Queue ได้ดังนี้

1. name คือ ให้เติมชื่อของ Queue
2. process model คือ ให้เติมคุณสมบัติลักษณะการทำงานภายใน Queue ซึ่งจะกำหนดให้เป็น FIFO คือ แบบ First In/First Out
3. icon name คือ ชื่อของ icon ซึ่งจะกำหนดให้เป็น Queue
4. service rate คือ ให้เติมอัตราการส่งแพ็คเกจ ออกจาก Queue ซึ่งจะกำหนดให้เป็น Promoted
5. Queue size คือ ให้เติมค่าจำนวนแพ็คเกจที่ Queue สามารถรับได้

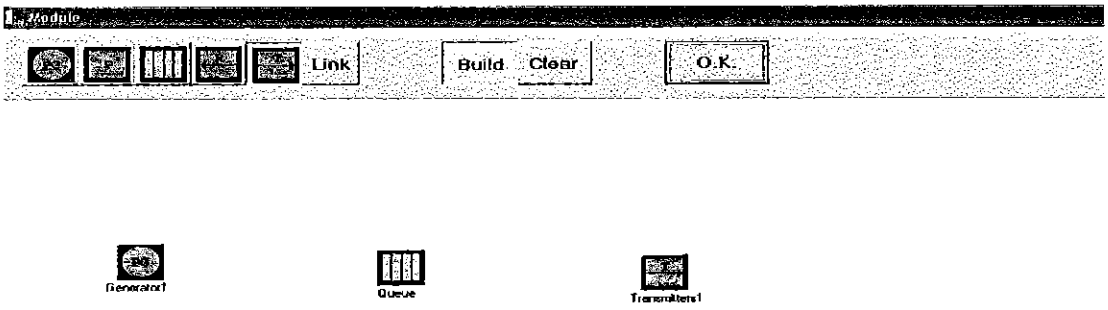
เมื่อกำหนดคุณสมบัติเรียบร้อยแล้วก็กดปุ่ม OK จะเป็นการสิ้นสุดการกำหนดคุณสมบัติของ Queue





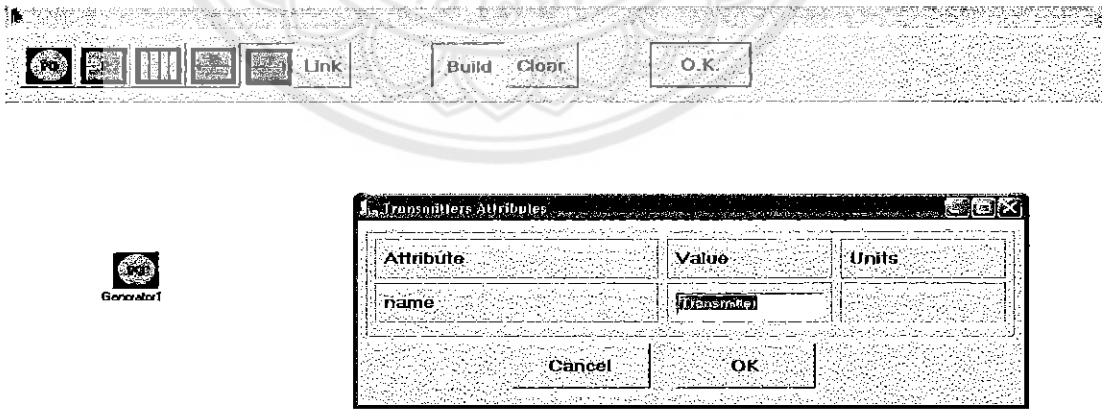
รูปที่ 4.12 การกำหนดคุณสมบัติของ Queue

5.3 การสร้าง Transmitter จะเริ่มจากการเลือกคลิกปุ่มTransmitter  แล้วตามด้วยคลิกที่ปุ่มBuild  แล้วคลิกบริเวณที่ต้องการสร้าง Transmitter ดังรูปที่ 4.13



รูปที่ 4.13 การสร้าง Transmitter

คลิกขวาที่ Transmitter และเลือก Property เพื่อที่จะกำหนดคุณสมบัติของ Transmitter ดังรูป ที่ 4.14



รูปที่ 4.14 การกำหนดคุณสมบัติของ Transmitter

สามารถกำหนดคุณสมบัติของ Transmitter ได้ดังนี้

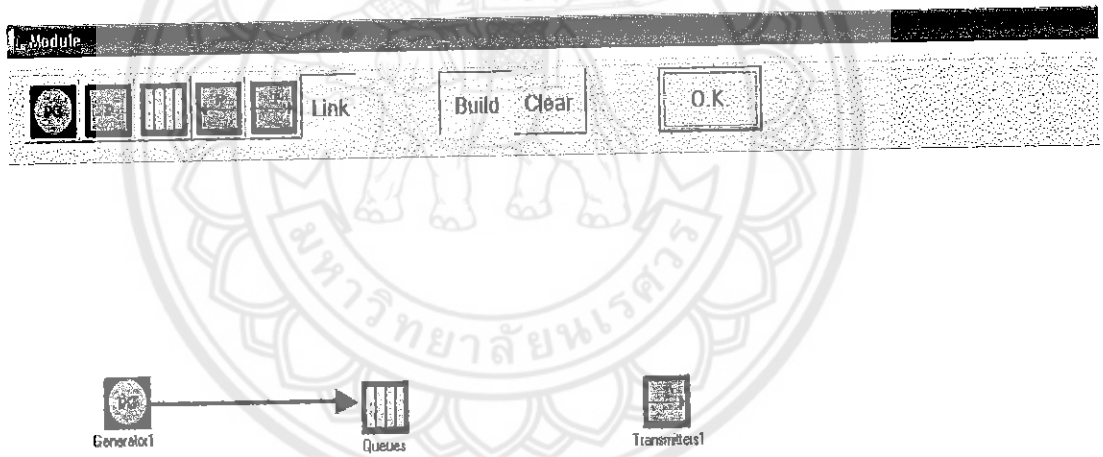
1. name คือ ให้เติมชื่อของ Transmitter

เมื่อกำหนดคุณสมบัติเรียบร้อยแล้วก็กดปุ่ม OK OK จะเป็นการสิ้นสุดการกำหนดคุณสมบัติของ Transmitter

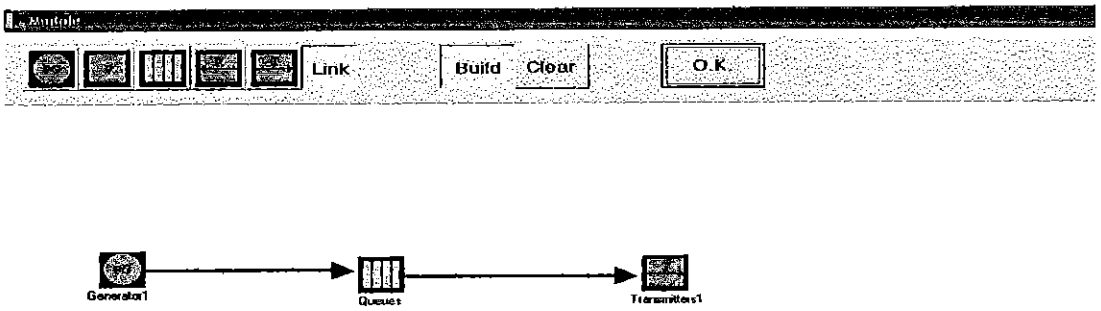
5.4 ทำการ Link แต่ละรูปเข้าด้วยกัน ตามแบบของระบบ M/M/1 Queue System ดังนี้

1. ทำการ Link Generator และ Queue เข้าด้วยกัน โดยเริ่มจากเลือกคลิกปุ่ม Link Link แล้วตามด้วยคลิกปุ่ม Build Build จากนั้นนำเมาส์ (Mouse) ไปคลิกที่ Generator แล้วไปคลิกที่ Queue ดังรูปที่ 4.15

2. ทำการ Link Queue และ Transmitter เข้าด้วยกัน โดยเริ่มจากเลือกคลิกปุ่ม Link Link แล้วตามด้วยคลิกปุ่ม Build Build จากนั้นนำ Mouse ไปคลิกที่ Queue แล้วไปคลิกที่ Transmitter ดังรูปที่ 4.16



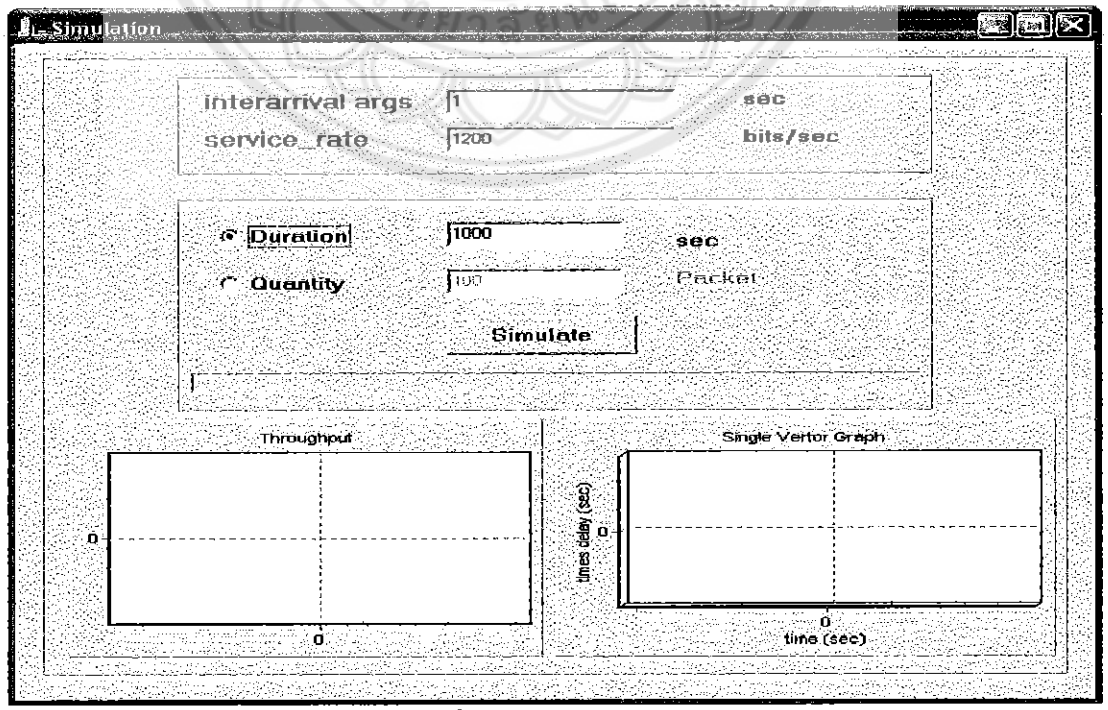
รูปที่ 4.15 การ Link Generator กับ Queue



รูปที่ 4.16 การ Link Queue กับ Transmitter

6. หลังจากนั้นก็กดปุ่ม OK ก็จะกลับไปที่ระดับ Node (หน้าต่าง Network Simulation) หากต้องการแก้ไขคุณสมบัติของ Node ให้คลิกขวาที่ Node แล้วเลือก Property ก็จะสามารถเข้าไปแก้ไขได้

7. หลังจากทำการกำหนดคุณสมบัติของ Node เรียบร้อยแล้ว จะทำการ simulation โดยคลิกที่ปุ่ม simulation จะปรากฏหน้าต่างการ simulation ดังรูปที่ 4.17

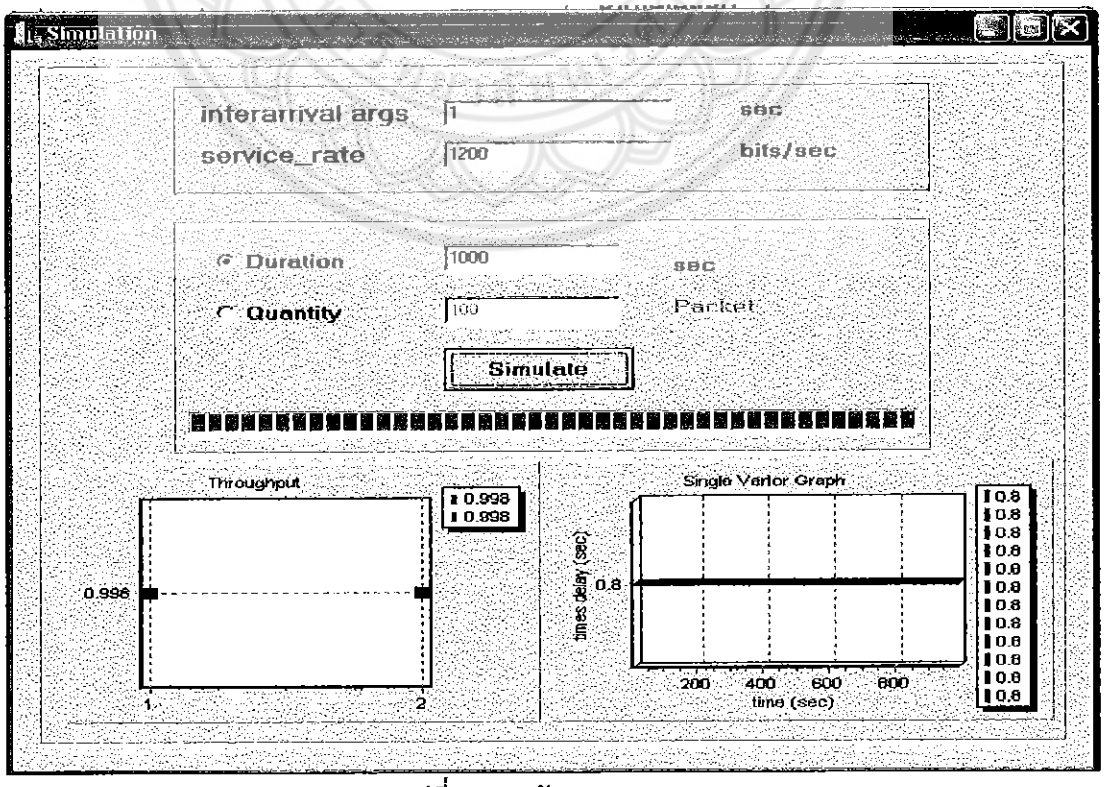


รูปที่ 4.17 การ simulation

คุณสมบัติการ simulation มีดังนี้

1. interarrival args** คือ คุณสมบัติเดียวกับ interarrival args** ของ Generator เพียงแต่ว่าที่ Generator Attribute กำหนดเป็น Promoted เพื่อที่จะมาเติมที่นี่ (หน้าต่าง Simulation) เช่น ถ้าเรากำหนดให้เป็น 1 วินาที จะทำให้แพ็คเกจที่เกิดขึ้นก่อนมีเวลาการเกิดห่างจากแพ็คเกจต่อไป 1วินาที เป็นต้น
2. service rate คือ คุณสมบัติเดียวกับ service rate ของ Queue เพียงแต่ว่าที่ Queue กำหนดเป็น Promoted เพื่อที่จะมาเติมที่นี่ (หน้าต่าง Simulation) เช่น ถ้ากำหนดให้เท่ากับ 1000 bit/sec Queue ใน 1 วินาที จะส่งข้อมูลออกไป 1000 bit
3. Duration คือ การ simulation โดยกำหนดช่วงเวลา มีหน่วยเป็นวินาที เช่น ถ้ากำหนดให้เท่ากับ 1000 วินาที จะ simulate ในเวลา 1000 วินาที
4. Quantity คือ การ simulation โดยกำหนดจำนวน Packet ที่ส่งได้สำเร็จ มีหน่วยเป็นแพ็คเกจ เช่น ถ้ากำหนดให้เท่ากับ 1000 แพ็คเกจ จะ simulate จนส่งแพ็คเกจได้ 1000 แพ็คเกจ
5. simulate คือ ทำการประมวลผล ตามคุณสมบัติที่กำหนด

8. กดปุ่ม simulate โปรแกรมจะทำการ simulate แล้วจะแสดงผลการ simulate เป็น 2 อย่าง
   - Time Delay ที่เกิดขึ้น ดังรูปที่ 4.18
   - Throughput ที่ได้ ดังรูปที่ 4.18



รูปที่ 4.18 หลังการ simulate

4.2.2 เพื่อวัดประสิทธิภาพการทำงานของโปรแกรม

จากการจำลองการทำงาน จะได้กราฟที่มีค่า Time Delay ออกมาค่าหนึ่ง ซึ่งเราสามารถตรวจสอบค่าความคลาดเคลื่อนของกราฟได้จากสูตร ดังต่อไปนี้

วิธีการหาค่าของ mean delay ดังนี้

<u>แบบที่1</u>

mean arrival rate : $\lambda = \frac{1}{\text{mean interarrival time}}$ (packet/second)

Packet Length : $\frac{1}{\mu}$ (bit/packet)

service capacity : C (bit/second)

mean delay : $\bar{w} = \frac{1}{\mu C - \lambda}$ (second).....สมการที่ 4.2.1

<u>แบบที่2</u>

mean delay : $\bar{w} = \frac{\text{(ผลรวมของ time delay ทั้งหมด)}}{\text{(จำนวนครั้งของ time delay ทั้งหมด)}}$ .....สมการที่ 4.2.2

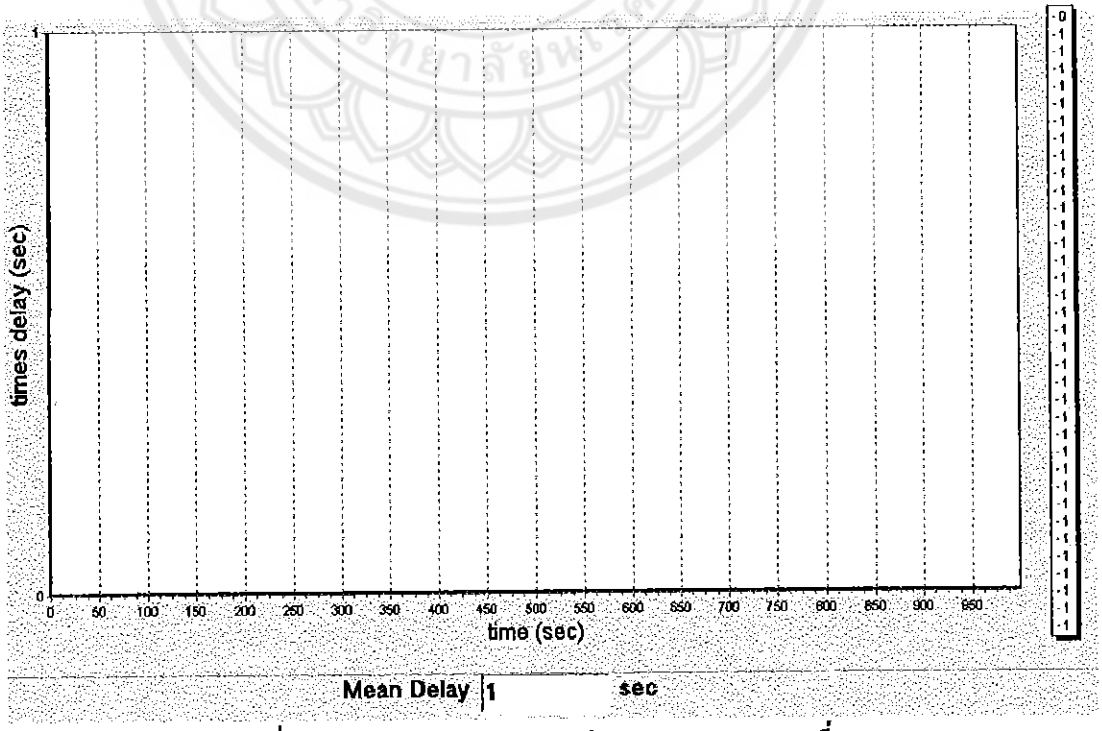
4.3 ผลการทดลอง

สร้างการทำงานตามข้อ4.2.1 โดยกำหนดคุณสมบัติต่างๆดังตารางต่อไปนี้

ตารางที่ 4.1 คุณสมบัติการทดสอบโปรแกรมแบบที่ 1

	name	interarrival pdf	interarrival argument (second)	packet size probability density function	packet size argument (bit)	mean delay
Generator	Generator	Constant	1	Constant	1000	-
	name	process model	icon name	service rate (bit/second)	Queue size (packet)	
Queue	Queue	Fifo	Queue	1000	10000	-
	name					
Transmitter	Transmitter	-	-	-	-	-
Duration	1000 (second)		Quantity	-		1

ผลการทำงานของโปรแกรม



รูปที่ 4.19 กราฟแสดงผลการรันโปรแกรมตามตารางที่ 4.1

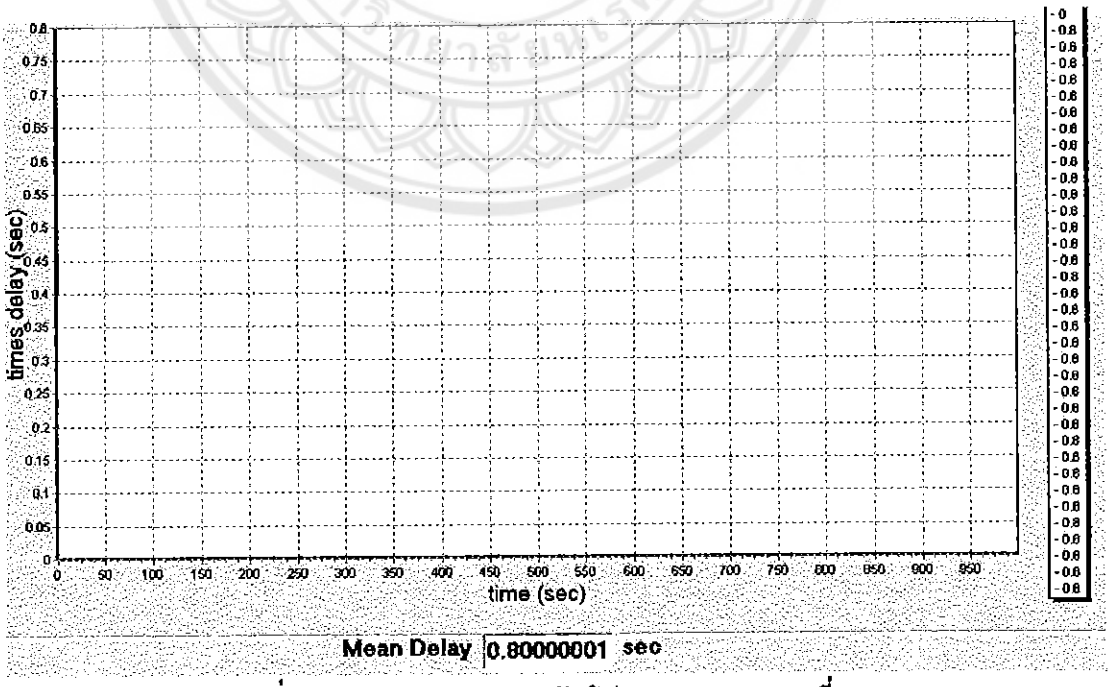
วิเคราะห์ผลการทดสอบ

จากคุณสมบัติตามตารางที่ 4.1 ผลคำนวณจากสมการที่ 4.2.2 หา Time Delay = 1 วินาที จากการผลการทำงานของโปรแกรม Time Delay = 1 วินาที เหมือนกัน

ตารางที่ 4.2 คุณสมบัติการทดสอบโปรแกรมแบบที่ 2

	name	interarrival pdf	interarrival argument (second)	packet size probability density function	packet size argument (bit)	mean delay
Generator	Generator	Constant	1	Constant	1000	-
	name	process model	icon name	service rate (bit/second)	Queue size (packet)	
Queue	Queue	Fifo	Queue	1200	10000	-
	name					
Transmitter	Transmitter	-	-	-	-	-
Duration	1000 (second)		Quantity	-		0.8

ผลการทำงานของโปรแกรม



รูปที่ 4.20 กราฟแสดงผลการรันโปรแกรมตามตารางที่ 4.2

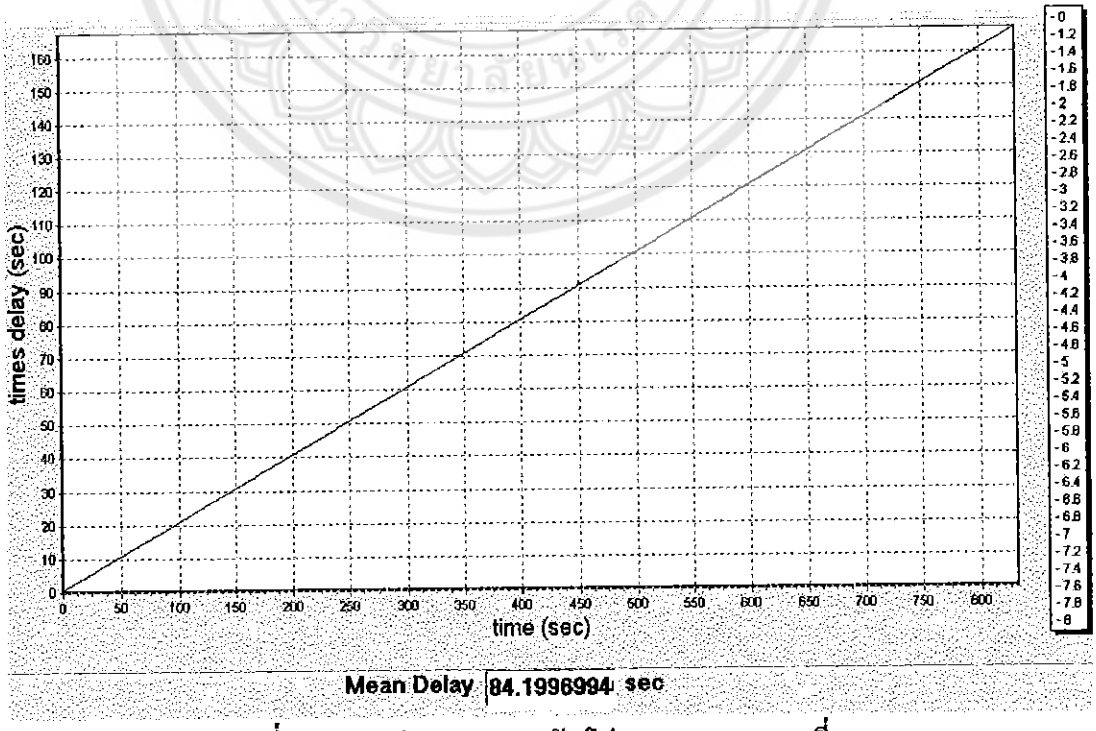
วิเคราะห์ผลการทดสอบ

จากคุณสมบัติตามตารางที่ 4.2 ผลคำนวณสมการที่ 4.2.2 หา Time Delay = 0.8 วินาที จากการผลการทำงานของโปรแกรม Time Delay = 0.8 วินาที เหมือนกัน

ตารางที่ 4.3 คุณสมบัติการทดสอบโปรแกรมแบบที่ 3

	name	interarrival pdf	interarrival argument (second)	packet size probability density function	packet size argument (bit)	mean delay
Generator	Generator	Constant	1	Constant	1000	-
	name	process model	icon name	service rate (bit/second)	Queue size (packet)	
Queue	Queue	Fifo	Queue	800	10000	-
	name					
Transmitter	Transmitter	-	-	-	-	-
Duration	1000 (second)		Quantity	-		

ผลการทำงานของโปรแกรม



รูปที่ 4.21 กราฟแสดงผลการรันโปรแกรมตามตารางที่ 4.3

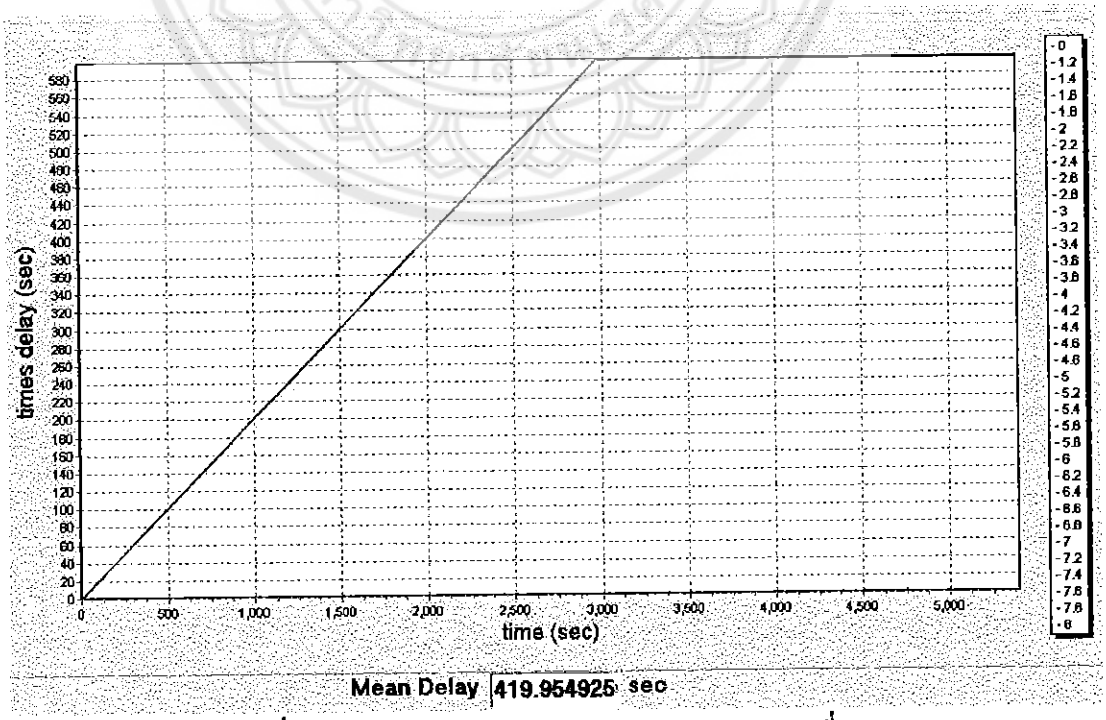
วิเคราะห์ผลการทดสอบ

จากคุณสมบัติตามตารางที่ 4.3 ผลวิเคราะห์หา Time Delay จะมีแนวโน้มที่จะเพิ่มขึ้นเรื่อย จากการผลการทำงานของโปรแกรม Time Delay มีแนวโน้มเพิ่มขึ้นเหมือนกัน

ตารางที่ 4.4 คุณสมบัติการทดสอบโปรแกรมแบบที่ 4

	name	interarrival pdf	interarrival argument (second)	packet size probability density function	packet size argument (bit)	mean delay
Generator	Generator	Constant	1	Constant	1000	-
	name	process model	icon name	service rate (bit/second)	Queue size (packet)	
Queue	Queue	Fifo	Queue	800	500	-
	name					
Transmitter	Transmitter	-	-	-	-	-
Duration	6000 (second)		Quantity	-		

ผลการทำงานของโปรแกรม



รูปที่ 4.22 กราฟแสดงผลการรันโปรแกรมตามตารางที่ 4.4

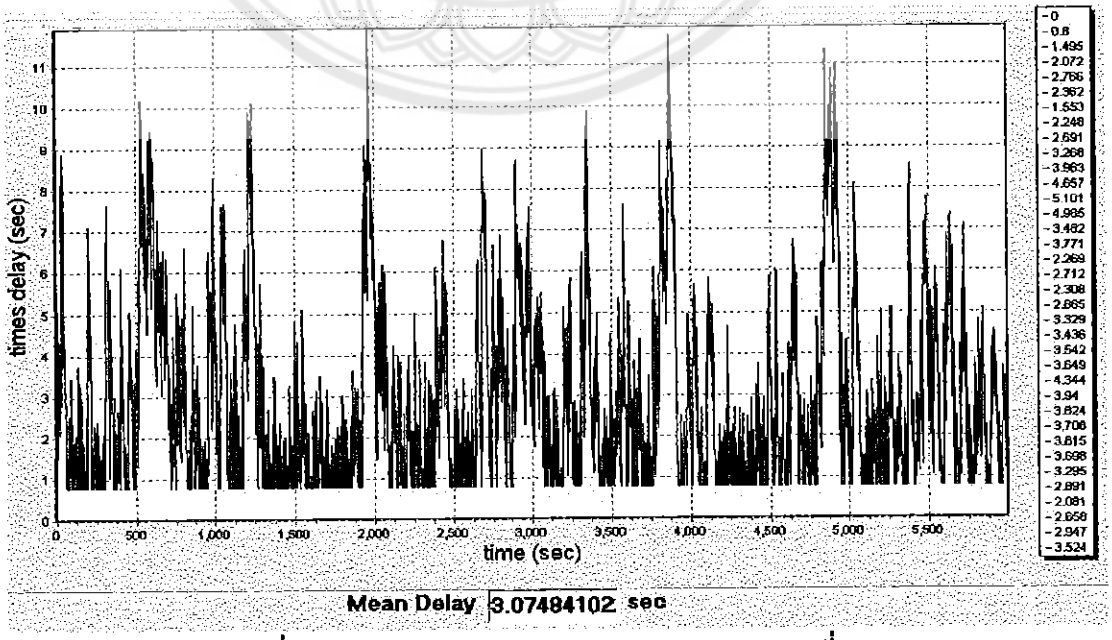
วิเคราะห์ผลการทดสอบ

จากคุณสมบัติตามตารางที่ 4.4 ผลวิเคราะห์หา Time Delay จะมีแนวโน้มที่จะเพิ่มขึ้นเรื่อยจนกระทั่งที่จุดหนึ่ง Time Delay เริ่มคงที่ซึ่งเป็นจุดที่คิว ไม่สามารถรับแพ็คเกจเพิ่มขึ้นได้ จึงทำให้ Time Delay คงที่ จากผลการทำงานของโปรแกรม Time Delay มีแนวโน้มเพิ่มขึ้นช่วงหนึ่งและหลังจากนั้นจะคงที่

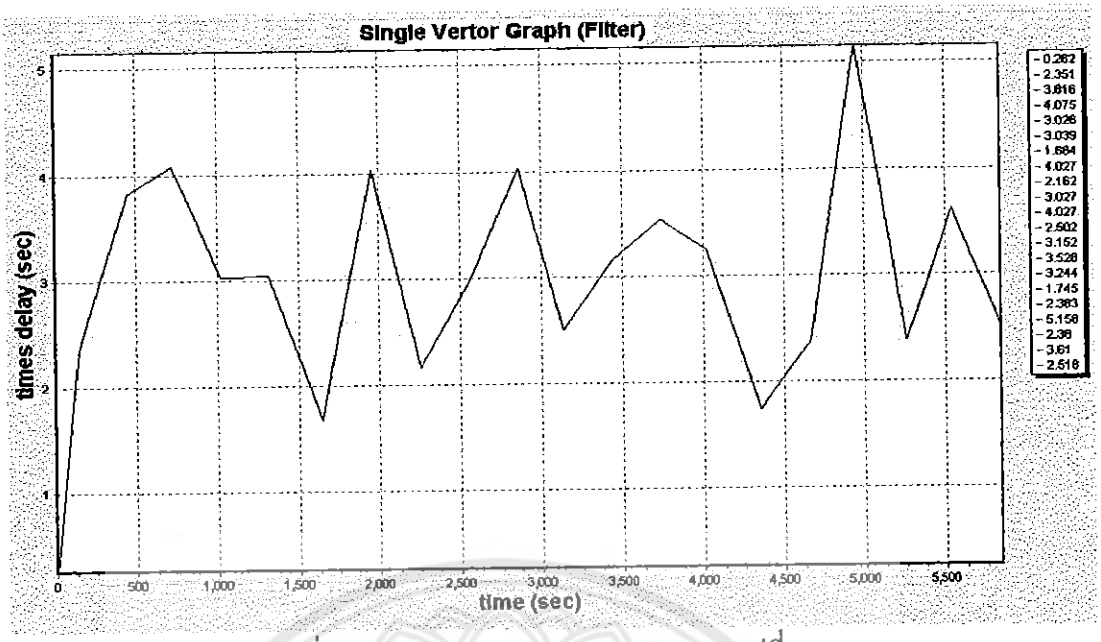
ตารางที่ 4.5 คุณสมบัติการทดสอบโปรแกรมแบบที่ 5

	name	interarrival pdf	interarrival argument (second)	packet size probability density function	packet size argument (bit)	mean delay
Generator	Generator	exponential	1	Constant	1000	-
	name	process model	icon name	service rate (bit/second)	Queue size (packet)	
Queue	Queue	Fifo	Queue	1200	10000	-
	name					
Transmitter	Transmitter	-	-	-	-	-
Duration	6000 (second)		Quantity	-		5(second)

ผลการทำงานของโปรแกรม



รูปที่ 4.23 กราฟแสดงผลการรันโปรแกรมตามตารางที่ 4.5



รูปที่ 4.24 กราฟแสดงผลการFilter ของกราฟที่ 4.23

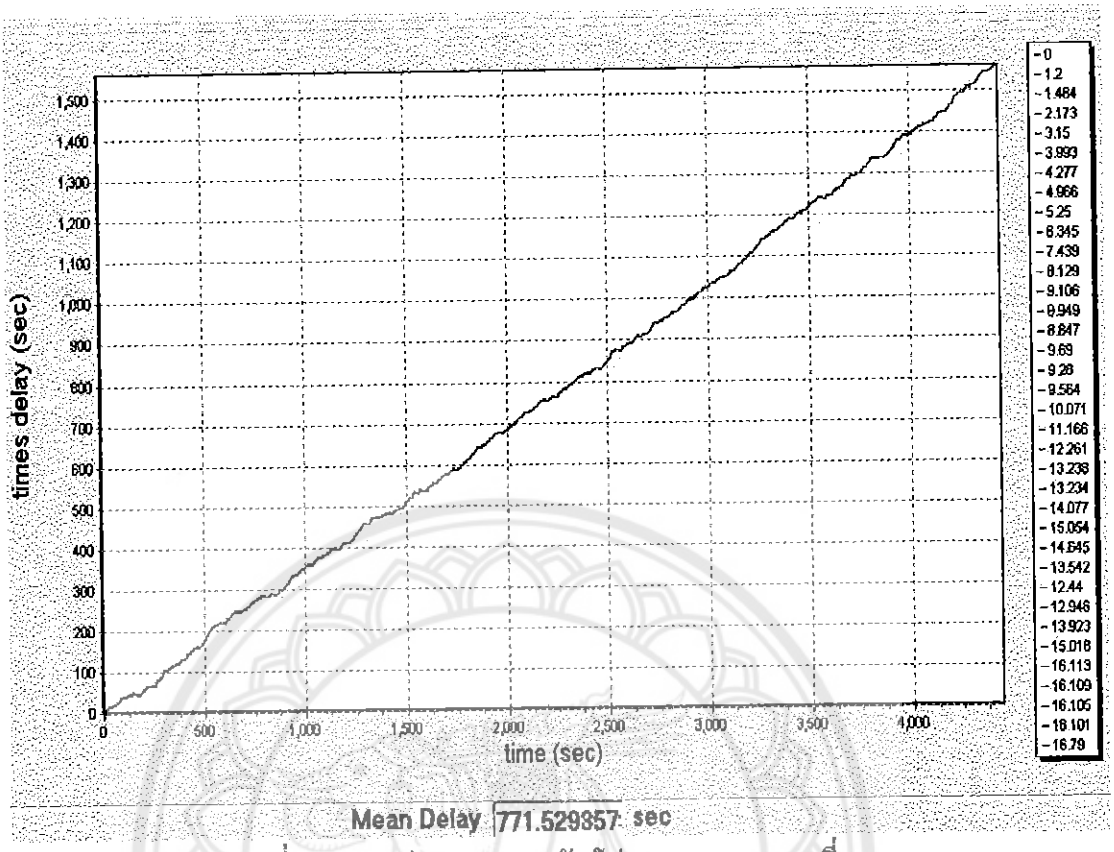
วิเคราะห์ผลการทดสอบ

จากคุณสมบัติตามตารางที่ 4.5 ผลวิเคราะห์หา Time Delay จะมีแนวโน้มที่เปลี่ยนไปเรื่อย เนื่องจากระยะเวลาที่แพ็คเกจเกิดขึ้นมีระยะห่างระหว่างแพ็คเกจเป็นการสุ่มค่าภายใต้เงื่อนไขของฟังก์ชัน Exponential จากสมการที่4.2.1 หา mean delay = 5 วินาที จากการผลการทำงานของโปรแกรม Time Delay มีแนวโน้มที่แกว่งอยู่ในช่วง 2-4 วินาที

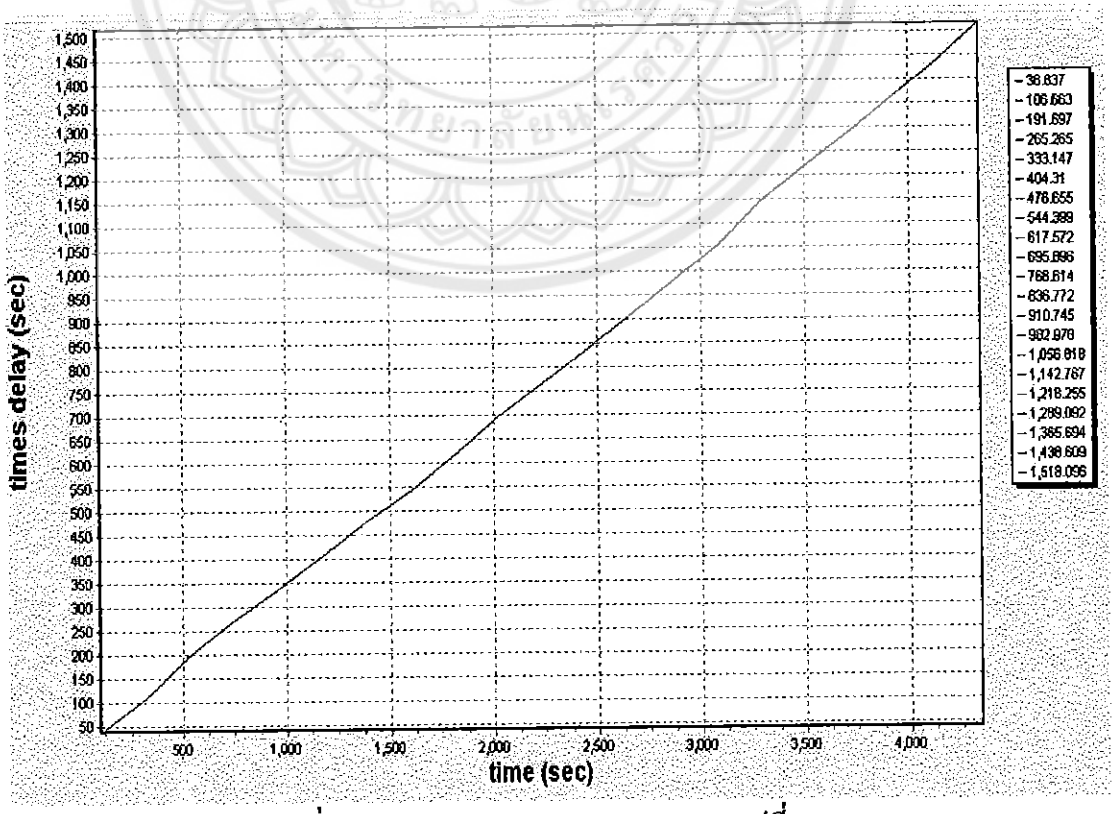
ตารางที่ 4.6 คุณสมบัติการทดสอบโปรแกรมแบบที่ 6

	name	interarrival pdf	interarrival argument (second)	packet size probability density function	packet size argument (bit)	mean delay
Generator	Generator	exponential	1	Constant	1000	-
	name	process model	icon name	service rate (bit/second)	Queue size (packet)	
Queue	Queue	Fifo	Queue	800	10000	-
	name					
Transmitter	Transmitter	-	-	-	-	-
Duration	6000 (second)		Quantity	-		

ผลการทำงานของโปรแกรม



รูปที่ 4.25 กราฟแสดงผลการรันโปรแกรมตามตารางที่ 4.6



รูปที่ 4.26 กราฟแสดงผลการFilter ของกราฟที่ 4.25

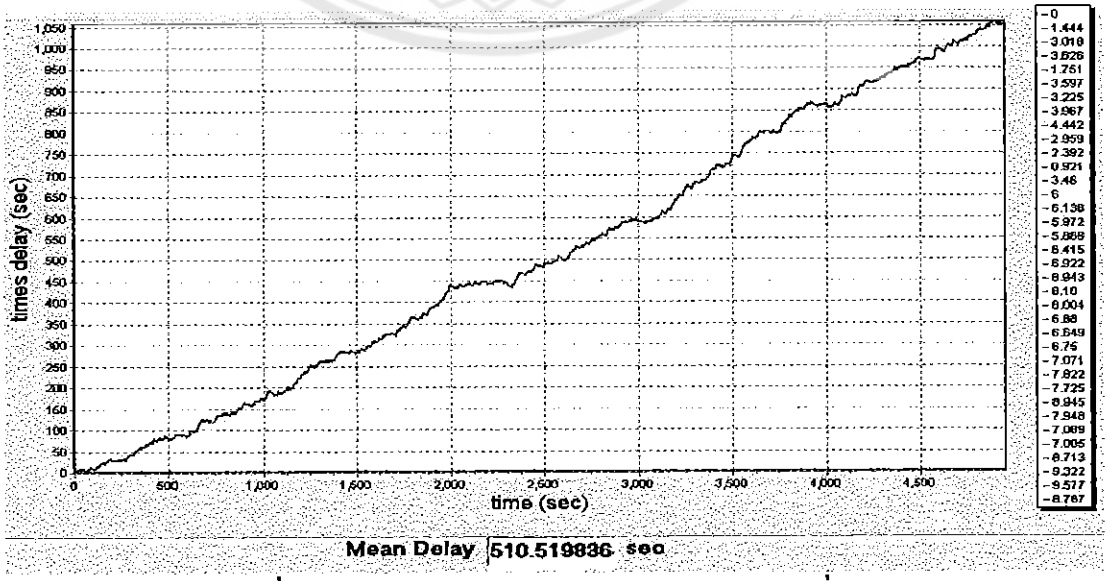
วิเคราะห์ผลการทดสอบ

จากคุณสมบัติตามตารางที่ 4.6 ผลวิเคราะห์หา Time Delay จะมีแนวโน้มที่เพิ่มขึ้นเรื่อยๆ เนื่องจากระยะเวลาที่แพ็คเกจเกิดขึ้นมีระยะห่างระหว่างแพ็คเกจเป็นการสุ่มค่าภายใต้เงื่อนไขของฟังก์ชัน Exponential แต่ความยาวคงที่ จากการผลการทำงานของโปรแกรม Time Delay มีแนวโน้มเพิ่มขึ้นเรื่อยๆ

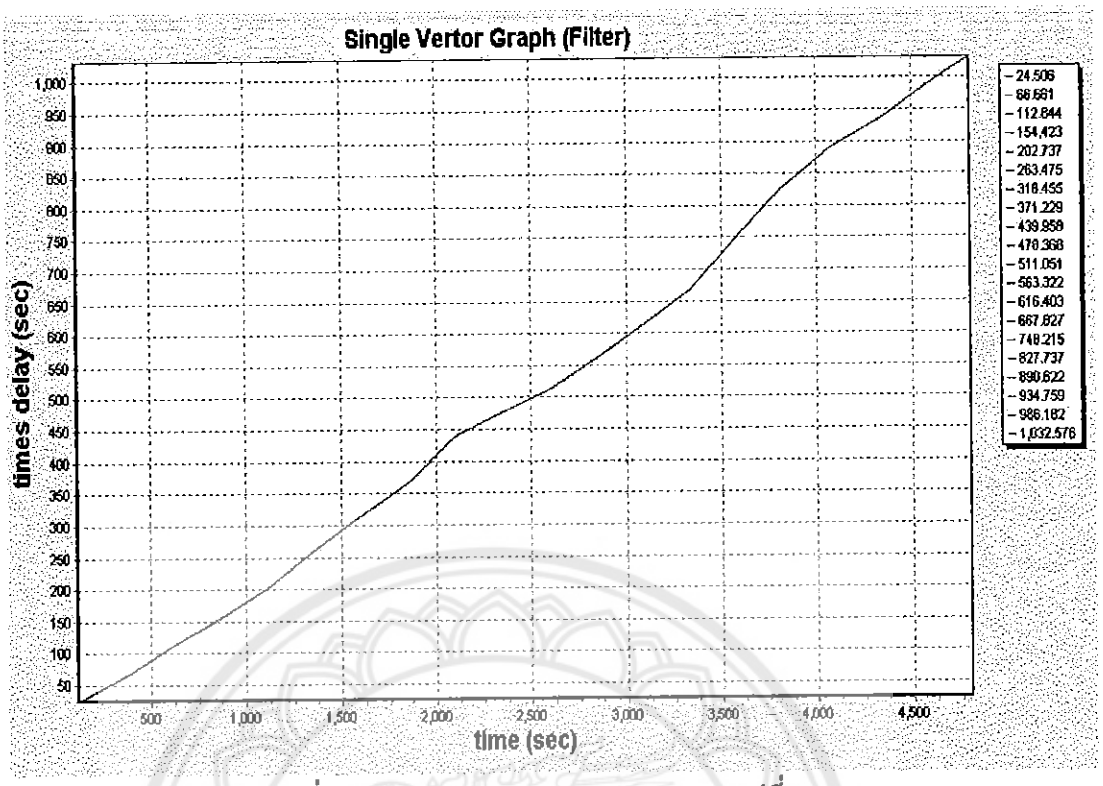
ตารางที่ 4.7 คุณสมบัติการทดสอบโปรแกรมแบบที่ 7

	name	interarrival pdf	interarrival argument (second)	packet size probability density function	packet size argument (bit)	mean delay
Generator	Generator	exponential	1	exponential	1000	-
	name	process model	icon name	service rate (bit/second)	Queue size (packet)	
Queue	Queue	Fifo	Queue	800	10000	-
	name					
Transmitter	Transmitter	-	-	-	-	-
Duration	6000 (second)		Quantity	-		

ผลการทำงานของโปรแกรม



รูปที่ 4.27 กราฟแสดงผลการรันโปรแกรมตามตารางที่ 4.7



รูปที่ 4.28 กราฟแสดงผลการFilter ของกราฟที่ 4.27

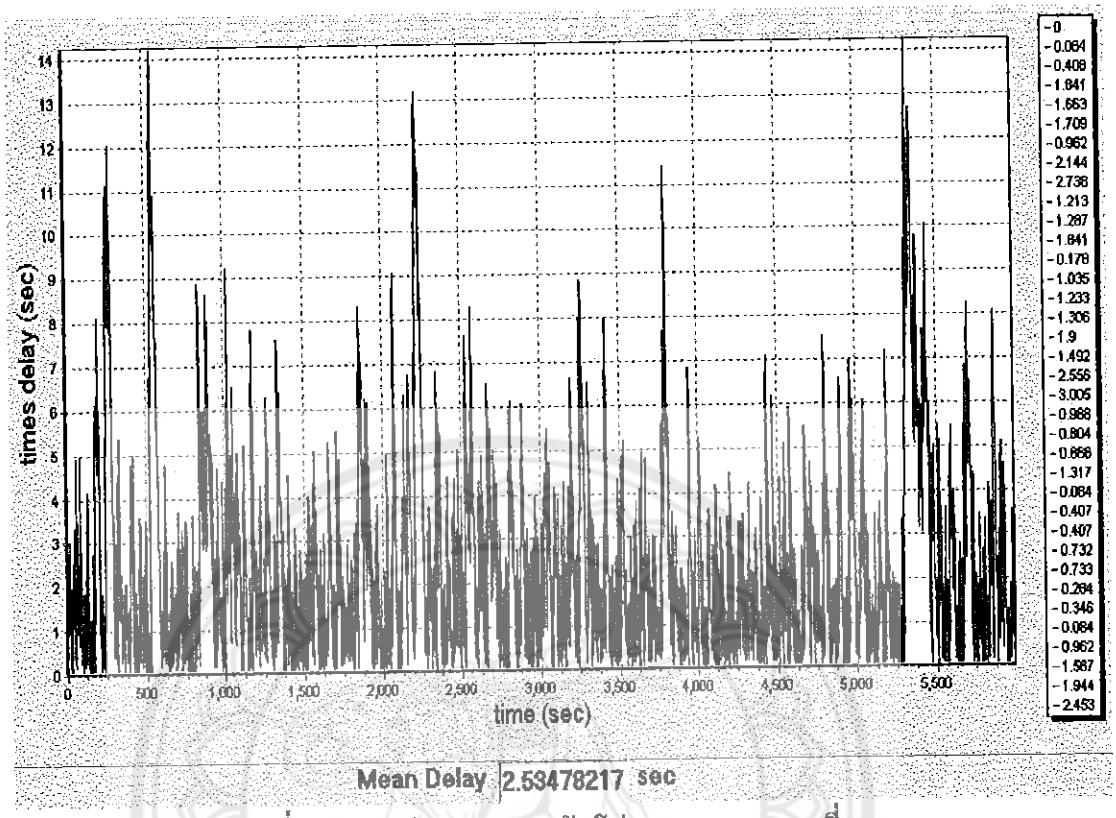
วิเคราะห์ผลการทดสอบ

จากคุณสมบัติตามตารางที่ 4.7 ผลวิเคราะห์หา Time Delay จะเห็นได้ว่า **Time Delay มีค่าเพิ่มสูงเรื่อยๆ**

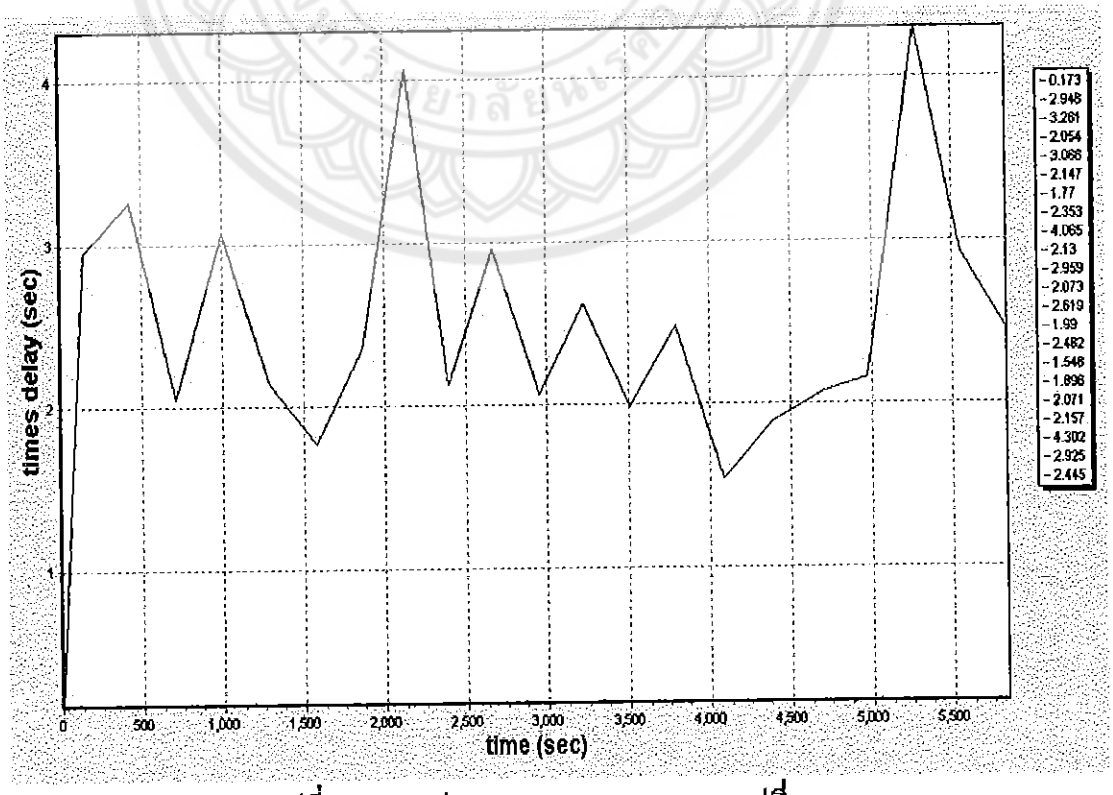
ตารางที่ 4.8 คุณสมบัติการทดสอบโปรแกรมแบบที่ 8

	name	interarrival pdf	interarrival argument (second)	packet size probability density function	packet size argument (bit)	mean delay
Generator	Generator	exponential	1	exponential	1000	-
	name	process model	icon name	service rate (bit/second)	Queue size (packet)	
Queue	Queue	Fifo	Queue	1200	10000	-
	name					
Transmitter	Transmitter	-	-	-	-	-
Duration	6000 (second)		Quantity	-		5(second)

ผลการทำงานของโปรแกรม



รูปที่ 4.29 กราฟแสดงผลการรันโปรแกรมตามตารางที่ 4.8



รูปที่ 4.30 กราฟแสดงผลการFilter ของกราฟที่ 4.29

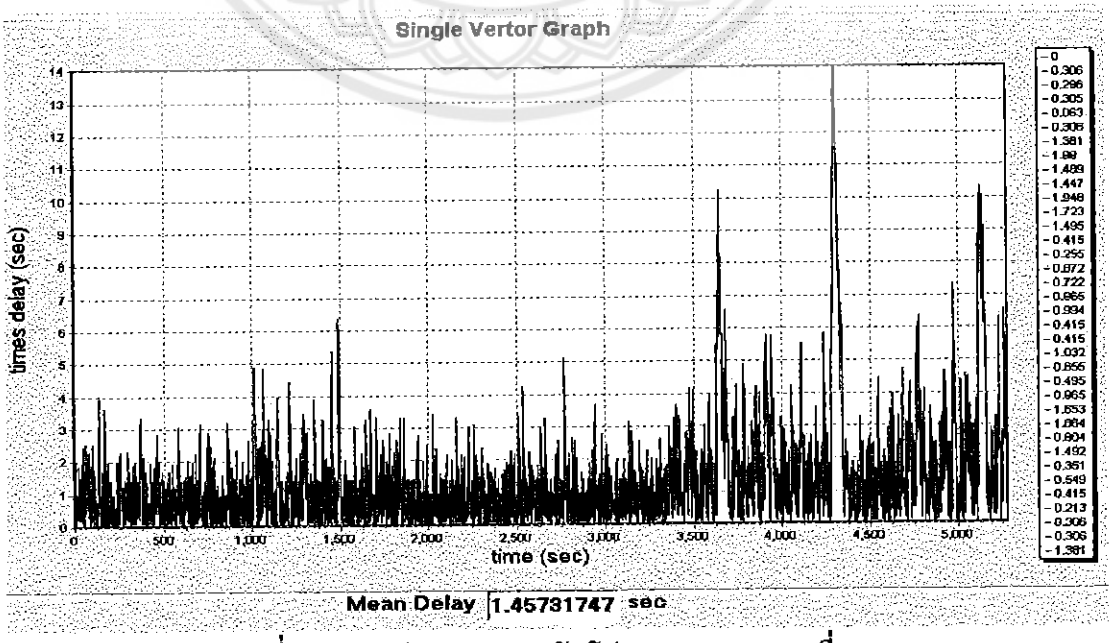
วิเคราะห์ผลการทดสอบ

จากคุณสมบัติตามตารางที่ 4.8 ผลวิเคราะห์หา Time Delay ในช่วงแรก Time Delay มีค่าอยู่ในช่วง 1-14 วินาที จากกราฟผลการ Fitter จะเห็นได้ว่าค่า mean delay ที่ได้จะประมาณ 2-4 วินาที เมื่อเปรียบเทียบ จากการคำนวณตามสมการที่ 4.2.1ได้ mean delay =5 วินาที

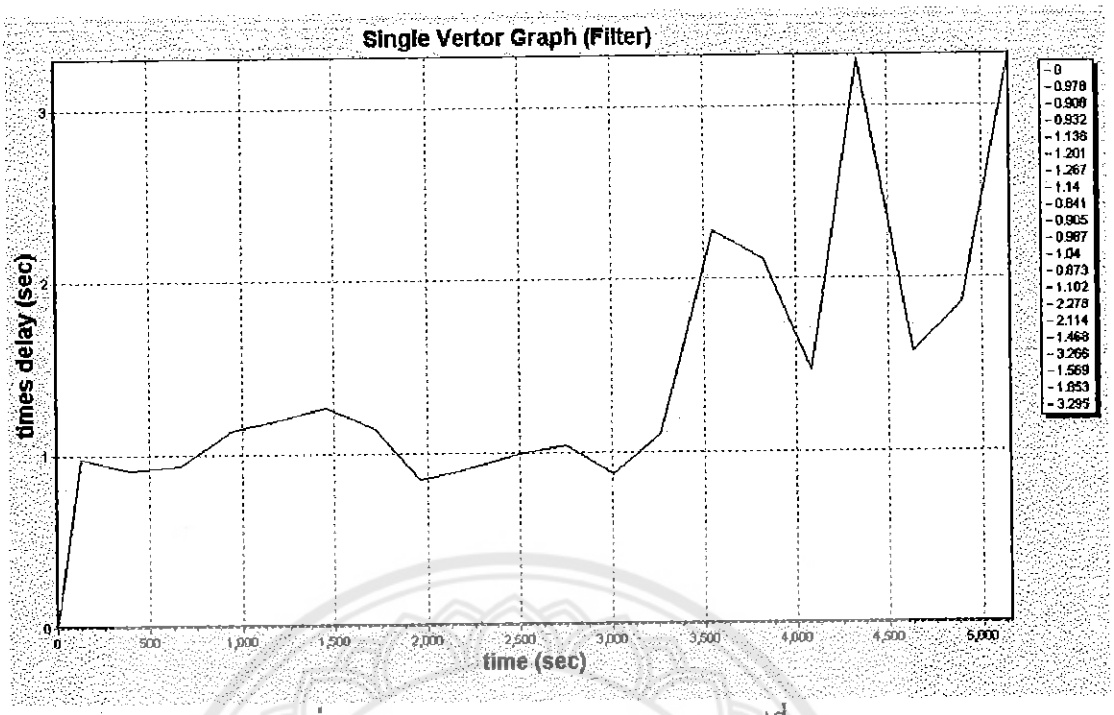
ตารางที่ 4.9 คุณสมบัติการทดสอบโปรแกรมแบบที่ 9

	name	interarrival pdf	interarrival argument (second)	packet size probability density function	packet size argument (bit)	mean delay
Generator	Generator	exponential	1	exponential	1000	-
	name	process model	icon name	service rate (bit/second)	Queue size (packet)	
Queue	Queue	Fifo	Queue	1500	10000	-
	name					
Transmitter	Transmitter	-	-	-	-	-
Duration	6000 (second)		Quantity	-		2(second)

ผลการทำงานของโปรแกรม



รูปที่ 4.31 กราฟแสดงผลการรันโปรแกรมตามตารางที่ 4.9



รูปที่ 4.32 กราฟแสดงผลการFilter ของกราฟที่ 4.31

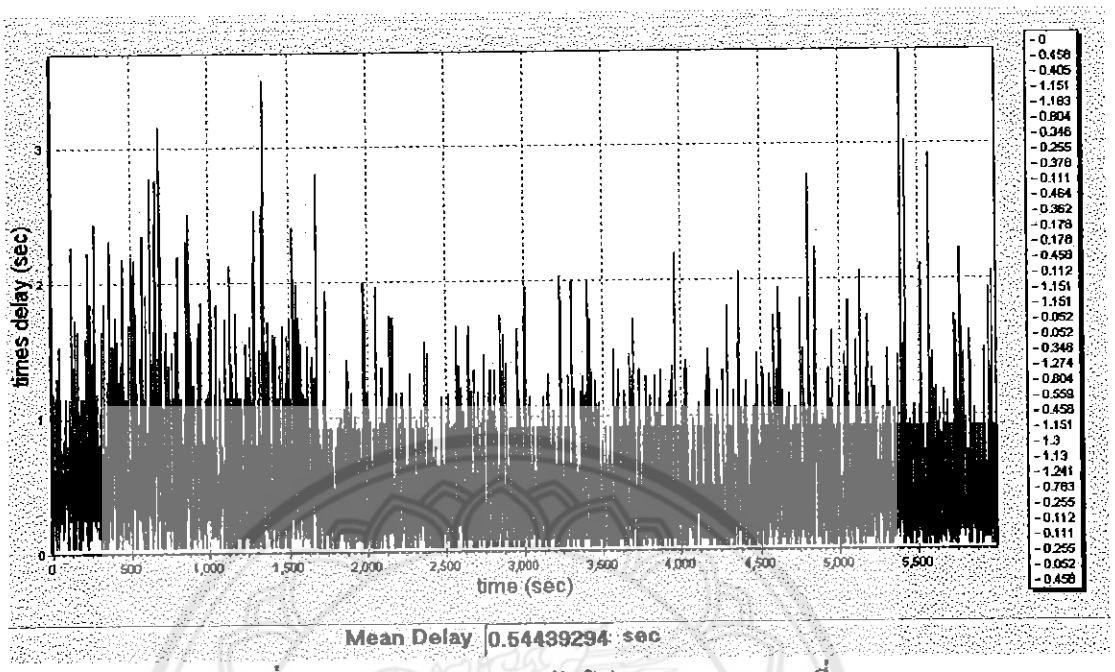
วิเคราะห์ผลการทดสอบ

จากคุณสมบัติตามตารางที่ 4.9 ผลวิเคราะห์หา Time Delay ในช่วงแรก Time Delay มีค่าอยู่ในช่วง 1-14 วินาที จากกราฟผลการ Fitter จะเห็นได้ว่าค่า mean delay ที่ได้จะประมาณ 1-3 วินาที เมื่อเปรียบเทียบ จากการคำนวณตามสมการที่ 4.2.1 ได้ mean delay =2 วินาที

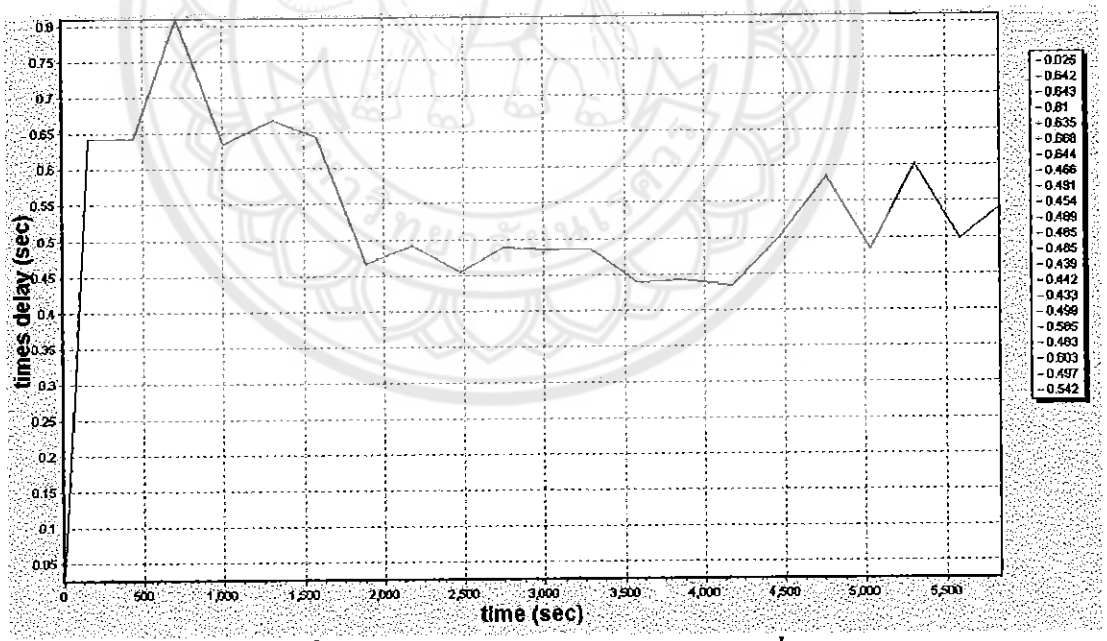
ตารางที่ 4.10 คุณสมบัติการทดสอบโปรแกรมแบบที่ 10

	name	interarrival pdf	interarrival argument (second)	packet size probability density function	packet size argument (bit)	mean delay
Generator	Generator	exponential	1	exponential	1000	-
	name	process model	icon name	service rate (bit/second)	Queue size (packet)	
Queue	Queue	Fifo	Queue	1800	10000	-
	name					
Transmitter	Transmitter	-	-	-	-	-
Duration	6000 (second)		Quantity	-		1.25(second)

ผลการทำงานของโปรแกรม



รูปที่ 4.33 กราฟแสดงผลการรันโปรแกรมตามตารางที่ 4.10



รูปที่ 4.34 กราฟแสดงผลการFilter ของกราฟที่ 4.33

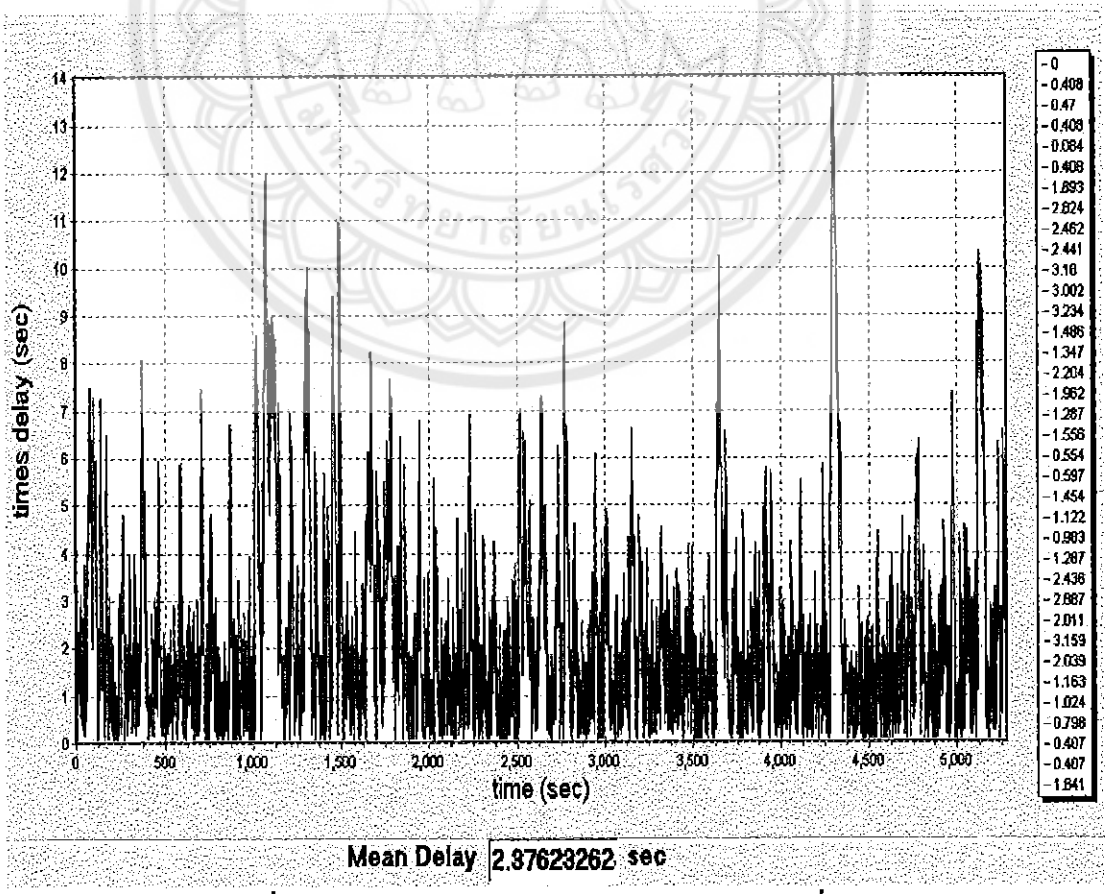
วิเคราะห์ผลการทดสอบ

จากคุณสมบัติตามตารางที่ 4.10 ผลวิเคราะห์หา Time Delay ในช่วงแรก Time Delay มีค่าอยู่ในช่วง 1-3.5 วินาที จากกราฟผลการ Fitter จะเห็นได้ว่าค่าที่ได้จะประมาณ 0.4-0.8วินาที เมื่อเปรียบเทียบ จากการคำนวณตามสมการที่ 4.2.1ได้ mean delay =1.25 วินาที

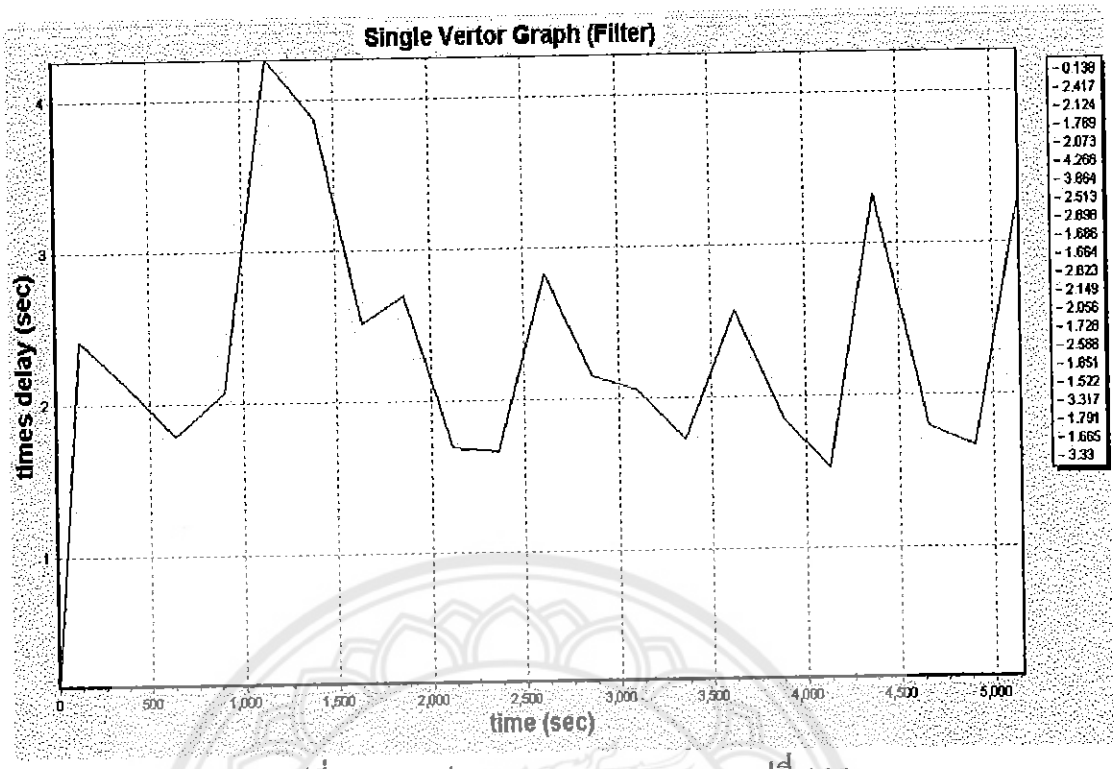
ตารางที่ 4.11 คุณสมบัติการทดสอบโปรแกรมแบบที่ 11

	name	interarrival pdf	interarrival argument (second)	packet size probability density function	packet size argument (bit)	mean delay
Generator	Generator	exponential	1	exponential	1000	-
	name	process model	icon name	service rate (bit/second)	Queue size (packet)	
Queue	Queue	Fifo	Queue	1200	1000	-
	name					
Transmitter	Transmitter	-	-	-	-	-
Duration	-		Quantity	6000 (packet)		5(second)

ผลการทำงานของโปรแกรม



รูปที่ 4.35 กราฟแสดงผลการรันโปรแกรมตามตารางที่ 4.11



รูปที่ 4.36 กราฟแสดงผลการFilter ของกราฟที่ 4.35

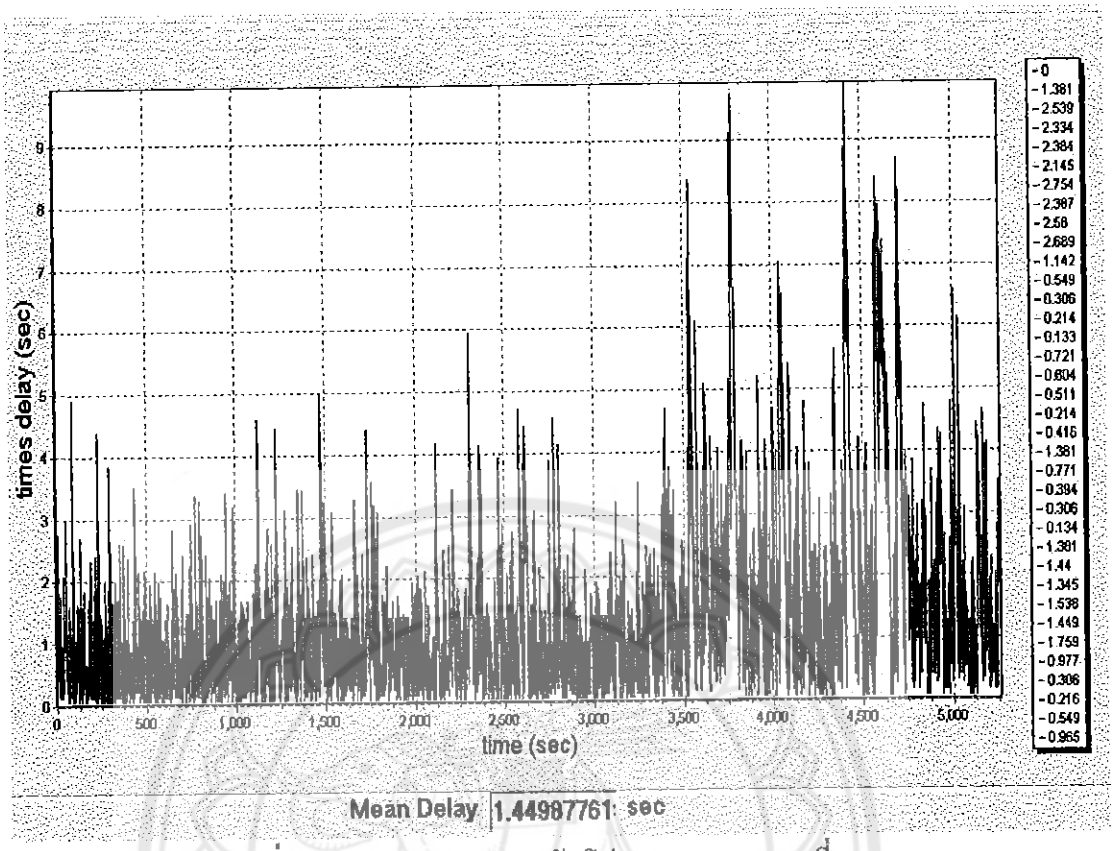
วิเคราะห์ผลการทดสอบ

จากคุณสมบัติตามตารางที่ 4.11 ผลวิเคราะห์หา Time Delay ในช่วงแรก Time Delay มีค่าอยู่ในช่วง 1-13 วินาที จากกราฟผลการ Fitter จะเห็นได้ว่าค่าที่ได้จะประมาณ 1-4 วินาที เมื่อเปรียบเทียบ จาการคำนวณตามสมการที่4.2.1 ได้ mean delay =5 วินาที มีลักษณะคล้ายคลึงกับผลการทำงานจาก ตารางที่ 8

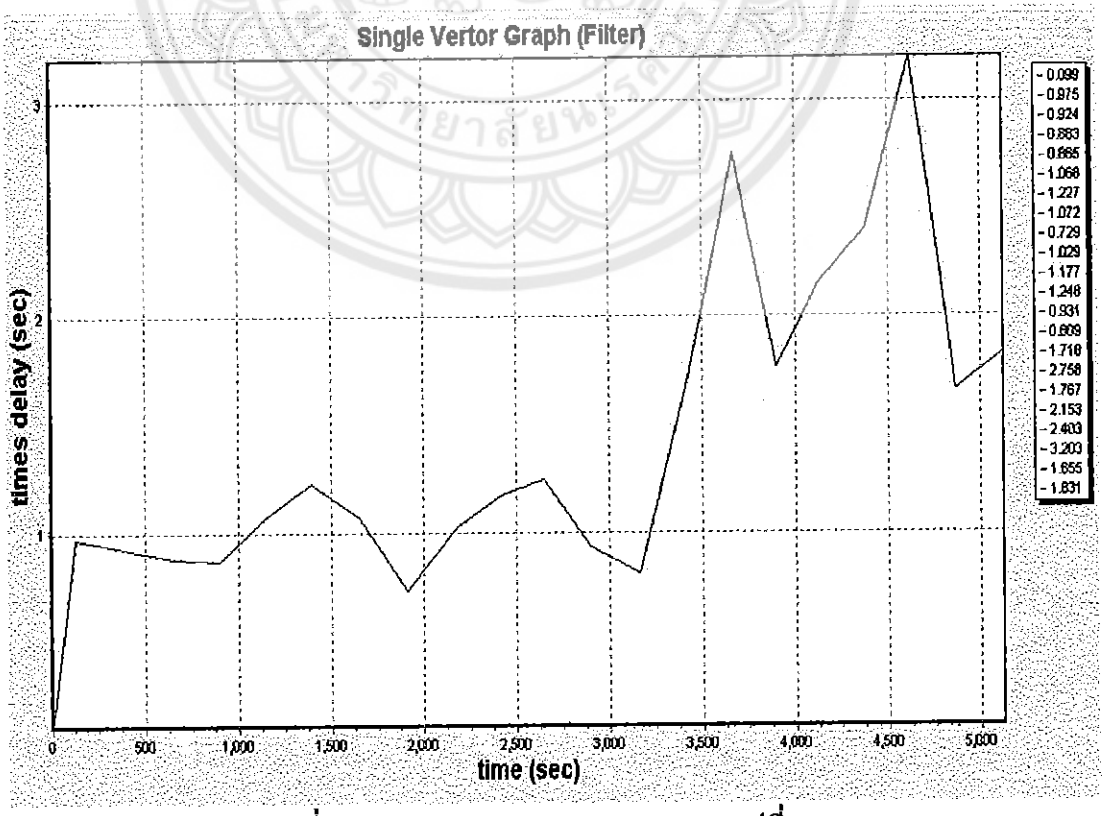
ตารางที่ 4.12 คุณสมบัติการทดสอบโปรแกรมแบบที่ 12

	name	interarrival pdf	interarrival arge	pk size pdf	pk size args (bit)	Mean delay
Generator	Generator	exponential	1	exponential	1000 (b)	-
	name	process model	icon name	service rate (bit/sec)	Queue size (packet)	
Queue	Queue	Fifo	Queue	1500(b/s)	1000 (b)	-
	name					
Transmitter	Transmitter	-	-	-	-	-
Duration	-		Quantity	6000(packet)		2(s)

ผลการทำงานของโปรแกรม



รูปที่ 4.37 กราฟแสดงผลการรันโปรแกรมตามตารางที่ 4.12



รูปที่ 4.38 กราฟแสดงผลการFilter ของกราฟที่ 4.37

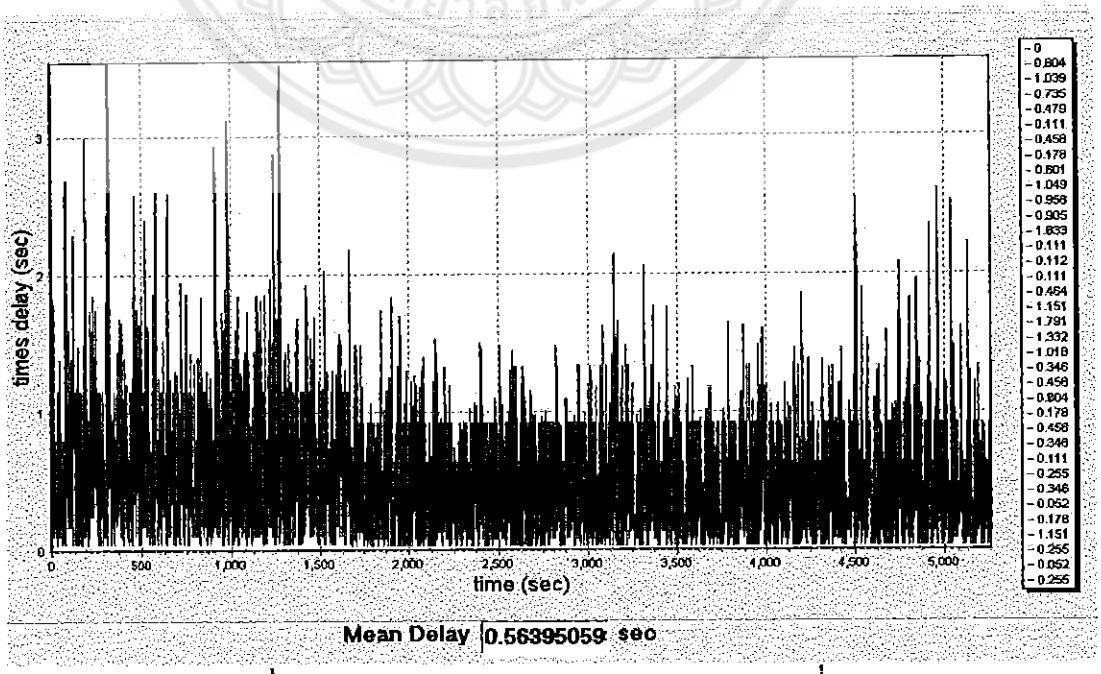
วิเคราะห์ผลการทดสอบ

จากคุณสมบัติตามตารางที่ 4.12 ผลวิเคราะห์หา Time Delay ในช่วงแรก Time Delay มีค่าอยู่ในช่วง 1-10 วินาที จากกราฟผลการ Fitter จะเห็นได้ว่าค่าที่ได้จะประมาณ 2-3 วินาทีเมื่อเปรียบเทียบ จากการคำนวณตามสมการที่4.2.1 ได้ mean delay =2 วินาที มีลักษณะคล้ายคลึงกับผลการทำงานจาก ตารางที่ 9

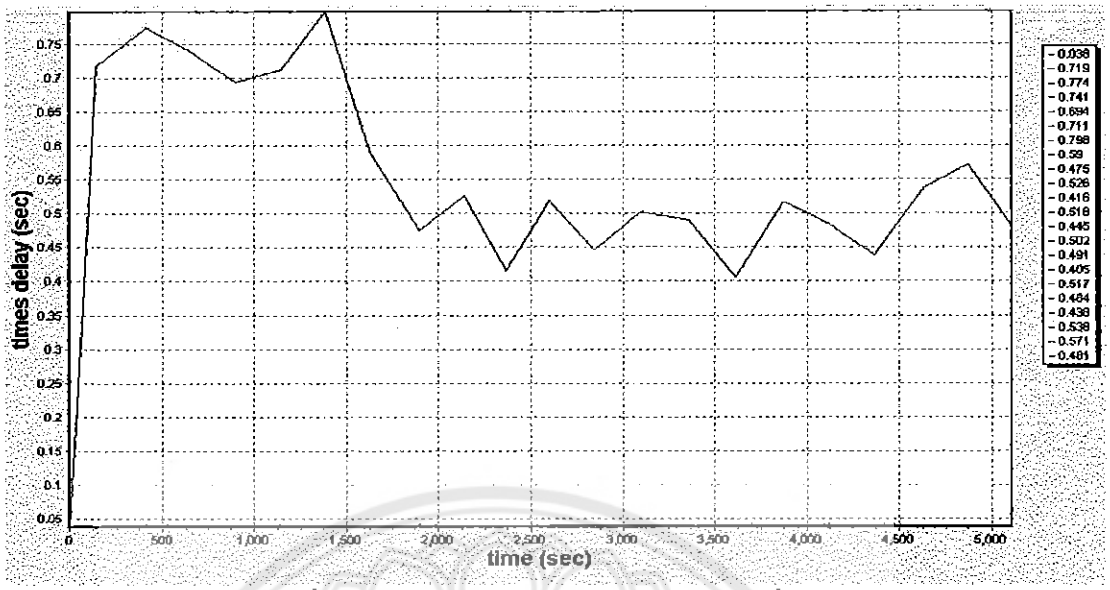
ตารางที่ 4.13 คุณสมบัติการทดสอบโปรแกรมแบบที่ 13

	name	interarrival pdf	interarrival arge	pk size pdf	pk size args (bit)	Mean delay
Generator	Generator	exponential	1	exponential	1000 (b)	-
	name	process model	icon name	service rate (bit/sec)	Queue size (packet)	
Queue	Queue	Fifo	Queue	1800(b/s)	1000 (b)	-
	name					
Transmitter	Transmitter	-	-	-	-	-
Duration	-		Quantity	6000 (packet)		1.25(s)

ผลการทำงานของโปรแกรม



รูปที่ 4.39 กราฟแสดงผลการรันโปรแกรมตามตารางที่ 4.13



รูปที่ 4.40 กราฟแสดงผลการFilter ของกราฟที่ 4.39

วิเคราะห์ผลการทดสอบ

จากคุณสมบัติตามตารางที่ 4.13 ผลวิเคราะห์หา Time Delay ในช่วงแรก Time Delay มีค่าอยู่ในช่วง 1-3 วินาที จากกราฟผลการ Fitter จะเห็นได้ว่าค่าที่ได้จะประมาณ 0.4-0.8 วินาที เมื่อเปรียบเทียบ จากการคำนวณตามสมการที่4.2.1 ได้ mean delay =1.25 วินาที มีลักษณะคล้ายคลึงกับผลการทำงานจาก ตารางที่10

บทที่5
บทสรุป

5.1 สรุปผลการทดสอบ

โปรแกรมสามารถสร้างแพ็คเกจ(Packet) และจำลองการทำงานระบบ M/M/1 Queue System และหาค่าของ Time Delay ได้ แต่เมื่อดูจากแนวโน้มของกราฟที่ได้ ไปเปรียบเทียบกับค่าที่คำนวณได้ จะยังมีค่าความคาดเคลื่อนอยู่

5.2 ปัญหาในการทดสอบและแนวทางแก้ไข

ปัญหา : ผลการทดสอบจะแสดงออกมาเป็นกราฟซึ่งมีความถี่สูงมาก

สาเหตุ : เนื่องจากว่าแพ็คเกจที่เกิดขึ้นมีจำนวนมากและมีค่าที่แตกต่างกันออกไป

ผล : ทำให้พิจารณาผลที่ได้ยากลำบากมาก

วิธีแก้ไข : นำเอาค่า Time Delay ที่ได้นำมาหาค่าเฉลี่ยเป็นช่วงๆ

ปัญหา : การกำหนดคุณสมบัติของโปรแกรมยังมีข้อจำกัดอยู่ไม่สามารถกำหนดค่าที่สูงมากได้

สาเหตุ : คุณสมบัติการรับค่าของตัวแปรมีข้อจำกัดเพียงค่าที่ตัวแปรถูกกำหนดให้เป็น เช่น ถ้ากำหนดตัวแปรให้เป็นอินทีเจอร์ ตัวแปรจะไม่สามารถรับค่าที่เกิน 32,767 หน่วยได้

ผล : ไม่มาสามารถรับค่าตัวแปรที่มีค่าสูงมากๆ ได้

วิธีแก้ไข : เพิ่มประสิทธิภาพในการทำงานของแต่ละรอบการทำงานของโปรแกรม โดยให้ 1 รอบ สามารถส่งข้อมูลได้มากกว่า 1 บิต

ปัญหา : โปรแกรมไม่สามารถทำงานในการทำงานระดับสูงๆ ได้ เช่น เมื่อเรากำหนดเวลาจำลองการทำงานเท่ากับ 2,000 วินาที ความยาวแพ็คเกจเท่ากับ 9,000 บิต Service rate เท่ากับ 9,600 บิต/วินาที คิวจะส่งข้อมูลขนาด 1 บิต ด้วยเวลา 0.000104 วินาที ซึ่งเราใช้เวลาในการเพิ่มเวลาในการจำลองระบบ เราจะคำนวณรอบการทำงานได้จาก 2,000/0.000104 ซึ่งจะได้ผลเท่ากับ 19,230,769 รอบ เมื่อทำการทดสอบโปรแกรม ผลปรากฏว่าโปรแกรมจะหยุดการทำงานที่ตำแหน่งก่อนการสิ้นสุดการทำงาน

ข้อสันนิษฐาน : มีรอบในการทำงานมากเกินไป ทำให้ตัวแปรบางค่าในโปรแกรมไม่สามารถรับค่าได้ หรือ พื้นที่หน่วยความจำเต็ม

ผล : โปรแกรมไม่สามารถแสดงผลรันได้

วิธีแก้ : หาวิธีลดการทำงานของโปรแกรมลง เพิ่มประสิทธิภาพของรอบในการทำงาน

5.3 ปัญหาการทำงานของโปรแกรมและแนวทางแก้ไข

ปัญหา : การเปรียบเทียบค่ามีความคาดเคลื่อน

สาเหตุ : การหารเลขก่อนที่จะนำไปทดสอบมีความละเอียดสูง

ผล : ผลการเปรียบเทียบมีความผิดพลาด

วิธีแก้ไข : ทำการกำหนดค่าความละเอียดของค่าที่นำไปทดสอบ

ปัญหา : โปรแกรมไม่สามารถวนลูปการทำงานในจำนวนรอบสูงๆ ได้

ข้อสันนิษฐาน : จำนวนรอบของการวนลูปมากเกินกว่าที่ตัวแปรรับได้

ผล : ทำให้โปรแกรมหยุดการทำงานที่ตำแหน่งนั้น

วิธีแก้ไข : เพิ่มความสามารถในการรับค่าของตัวแปรให้สูงขึ้น

5.4 แนวทางในการพัฒนาโปรแกรมต่อไป

1. ทำการวิเคราะห์ผลการรันโปรแกรม ให้มีความแน่นอนมากขึ้น โดยการ Transformation
2. ขยายประสิทธิภาพในการจำลองการทำงาน ให้เป็นระบบเครือข่ายแบบอื่นๆ
3. สามารถสร้างแพ็คเกจที่มีคุณสมบัติอื่นๆ ได้
4. สามารถบันทึกการทำงานของโปรแกรมได้
5. สามารถจำลองการทำงานของแต่ละเครือข่าย ให้เชื่อมต่อเครือข่ายอื่นได้
6. สามารถนำผลการรันของโปรแกรม ไปวิเคราะห์ในโปรแกรม MATHLAB ได้
7. ในการสร้าง Generator ให้มากกว่า 1 ตัว อาจทำได้โดยการออกแบบโครงสร้างข้อมูลจากที่เป็นอาร์เรย์ 2 มิติ ให้เพิ่มเป็นอาร์เรย์ 3 มิติ โดยมิติที่เพิ่มเข้ามาให้เป็นการบอกถึง Generator ตัวใดเป็นตัวสร้างแพ็คเกจ เช่น intPacket[n][m][1] หมายถึง แพ็คเกจที่เกิดจาก Generator ตัวที่ n ลำดับ m คุณสมบัติที่ 1 คือ เวลาในการเกิดของแพ็คเกจ เป็นต้น

เอกสารอ้างอิง

[1] ยุทธนา ลีลาศวัฒนกุล. **คู่มือการเขียนโปรแกรมและใช้งาน visual C++ 6.0 ฉบับโปรแกรมเมอร์.** กรุงเทพฯ : อินโฟเพรส, 2544.

[2] นุกูล กระจาย. **การเขียนโปรแกรมแบบวิชวลด้วย C++Builder 5.** กรุงเทพฯ : สุวีริยาสาสน์, 2544.

[3] นุกูล กระจาย. **การเขียนโปรแกรมในดอสและวินโดวส์ด้วยบอร์แลนด์ C++ 5.0.** กรุงเทพฯ: ซีเอ็ดยูเคชัน, 2544.

[4] OPNET. **OPNET Tutorial Manual.**

ภาคผนวก ก

พจนานุกรม

Build	คือ สร้างวัตถุ
Clear	คือ ลบวัตถุทิ้ง ลบวัตถุออก
Constant	คือ ค่าคงที่
Delete	คือ ลบทิ้ง ลบออก
Duration	คือ ระยะเวลาในการจำลองการทำงาน
Exponential	คือ ฟังก์ชันทางคณิตศาสตร์
FIFO (first in first out)	คือ ลักษณะการทำงานแบบเข้าก่อน ออกก่อน เข้าทีหลัง ออกทีหลัง
Generator	คือ ส่วนอุปกรณ์ที่ทำหน้าที่ เป็นตัวกำเนิดหรือสร้างแพ็คเกจภายในโนด
Graphic users interface	คือ ส่วนที่ติดต่อระหว่างผู้ใช้กับโปรแกรม
Icon name	คือ ลักษณะการประมวลผลภายในของวัตถุ เช่น ถ้ากำหนดเป็นคิว จะมีการทำงานภายในตามลักษณะแถวคอย
interarrival pdf	คือ ระยะห่างของการเกิดแพ็คเกจนั้น
interarrival args	คือ ค่าเฉลี่ยของระยะห่างของการเกิดแพ็คเกจนั้น
Link	คือส่วนที่เชื่อมโยงส่วนต่างๆ เข้าด้วยกัน
Mean delay	คือ เวลาเฉลี่ยในการประมวลผลที่แพ็คเกจต้องเสียเวลาจากต้นทางไปยังปลายทาง
Module	คือ ส่วนย่อยภายใน Node
Node	คือ วัตถุที่มีการส่งข้อมูลออกมา เช่น เครื่องคอมพิวเตอร์ PC ,ตู้เอทีเอ็ม เป็นต้น
Packet	คือ กลุ่มของข้อมูลข่าวสาร หรือส่วนของกลุ่มข้อมูลข่าวสารที่ทำการส่งผ่านระหว่างจุดในเครือข่าย
pk size args (packet size argument)	คือ ขนาดของความยาวแพ็คเกจโดยเฉลี่ย
pk size pdf (packet size probability density functon)	คือ การกำหนดให้ขนาดของแพ็คเกจมีค่าขึ้น อยู่กับฟังก์ชันต่างๆ เช่น ถ้ากำหนดให้เป็นค่าคงที่ ขนาดของแพ็คเกจก็จะคงที่ตลอด ถ้ากำหนดให้เป็นฟังก์ชันเอ็กโพเนนต์เชียล ความยาวของ แพ็คเกจจะเป็นไปตามลักษณะของฟังก์ชันเอ็กโพเนนต์เชียล

Processor	คือ ส่วนที่ทำหน้าที่ประมวลผลภายในโนด
Process model	คือ ลักษณะการประมวลผลภายในของวัตถุ เช่น ถ้ากำหนด เป็น FIFO ลักษณะการทำงานแบบเข้าก่อน ออกก่อน เข้าทีหลัง ออกทีหลัง
Promoted	คือ ยังไม่กำหนดค่าในขณะนั้น รอกำหนดค่าก่อนการ Simulation
Property	คือ คุณสมบัติ
Quantity	คือ การจำลองการทำงาน โดยกำหนดจำนวนแพ็คเกจที่ส่งได้สำเร็จ
Queue	คือ การเรียงลำดับข้อมูล เช่น การต่อแถวรอการใช้การบริการ
Queue Process	คือ ส่วนอุปกรณ์ที่ทำหน้าที่จัดการการต่อแถวรอ ภายในโนด
Queue size	คือ ความสามารถในการรองรับจำนวนแพ็คเกจของคิว
Reciever	คือ ส่วนที่ทำหน้าที่ในการรับข้อมูลจากภายนอกเข้าสู่โนด
Service rate	คือ อัตราการส่งข้อมูลของคิว
Simulation	คือ การจำลองการทำงาน
Single vector graph	คือ กราฟแสดงผล Time Delay ทั้งหมดของแพ็คเกจ
Throughput	คือ ประสิทธิภาพในการทำงานของเครือข่าย
Time Delay	คือ เวลาที่ใช้ในการประมวลผล
Transmitter	คือ ส่วนอุปกรณ์ที่ทำหน้าที่ส่ง Packet ออกนอกโนด

ภาคผนวก ข

Source Code Program

ส่วน simulation

```
#include <vcl.h>
#include <math.h>
#pragma hdrstop

#include "Simulation.h"
#include "Node.h"
#include "Graph1.h"
//---------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm18 *Form18;
float floatInter,floatService;
float floatTimes,floatTimes1,floatTimes3;
float floatTimes4,floatTimes5,floatTimes6,floatTimes7;
float floatTimesQn;
float intPacket[10001][8],intPacketQ[10001][8],intPacketS[10001][8];
float floatTrueput[10001];
float floatExp1,floatExp2,floatExp3,floatExp4,floatExp5;
float floatExp6,floatExp7,floatExp8,floatExp9,floatExp10;
float a1,a2;
int NumPacket,NumSink,NumTrueput,floatTimes2,q,intI,u,Num,t2;
int NumCount,a3;
int intExp1,intExp2,intExp3,intExp4,intExp5;
//---------------------------------------------------------------------
__fastcall TForm18::TForm18(TComponent* Owner)
    : TForm(Owner)
{
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm18::Button1Click(TObject *Sender)
{
  Edit1->Text=FloatToStr(NumPacket);
  Edit6->Text=FloatToStr(intPacketQ[1][6]);
  Edit7->Text=FloatToStr(NumSink);
  Edit8->Text=FloatToStr(floatTrueput[1]);
}
//---------------------------------------------------------------------------
void __fastcall TForm18::Button2Click(TObject *Sender)
{
 NumTrueput=NumTrueput+1;
 Series1->Clear();
 Series2->Clear();
 Series3->Clear();
 for(int l=0; l<10000; l++)
 {
  intPacketQ[l][1]=0;
  intPacketQ[l][2]=0;
  intPacketQ[l][3]=0;
  intPacketQ[l][4]=0;
  intPacketQ[l][5]=0;
  intPacketQ[l][6]=0;
 }
 for(int m=0; m<10000; m++)
 {
  intPacket[m][1]=0;
  intPacket[m][2]=0;
  intPacket[m][3]=0;
  intPacket[m][4]=0;
  intPacket[m][5]=0;
```

```
 intPacket[m][6]=0;
 intPacketS[m][1]=0;
 intPacketS[m][2]=0;
 intPacketS[m][3]=0;
 intPacketS[m][4]=0;
 intPacketS[m][5]=0;
 intPacketS[m][6]=0;
}
u=0;
if(RadioButton1->Checked==true)                    //แบบที่1 กำหนดระยะเวลาการ
{                                                  // simulation
 floatInter=StrToFloat(Form18->Edit2->Text)*10000;  //กำหนดระยะห่างของ Packet
 floatService=StrToFloat(Form18->Edit3->Text);      //กำหนดอัตราการส่ง Packet ของ
                                                    Queue
 floatTimes=StrToFloat(Form18->Edit4->Text)*10000; //กำหนดระยะเวลา Simulation
 floatTimes2=(10000/floatService);                  //กำหนดเวลาในการส่งข้อมูล 1 bit
 q=0;
 t2=0;
 NumPacket=0;
 floatTimes1=0;
 if((Form1->intPropertyNode[1]==1)&&( Form1->floatProperty[1][11][4]>=0))
 {                                                  //ส่วนสร้าง Packet
  Series1->Clear();
  if(Form1->floatProperty[1][11][1]==1)             //*ถ้าระยะห่าง Packet คงที่*
  {
   Series1->Clear();
   intPacket[0][1]=random(10000);                   //สุ่มหาเวลาที่เกิด Packet แรก
   t2=0;
   while(intPacket[t2][1]<floatTimes)               //หาเวลาที่เกิด Packet แต่ละ Packet
   {                                                //โดยนำเวลาในการส่งข้อมูล 1 bit
    t2=t2+1;                                        //มาบวกกับเวลาที่เกิด Packet ก่อนที่เกิด
    intPacket[t2][1]=intPacket[t2-1][1]+floatInter; //ขึ้นก่อน ก็จะได้เวลาที่ Packet ตัวต่อไป
```

```
    }
   }
   else if(Form1->floatProperty[1][11][1]==2)        //*ถ้าระยะห่าง Packet สอดคล้องสมการ
   {                                                 //exponential*
    floatExp1=random(10);                            //สุ่มหาค่าช่วง 0-10
    floatExp2=floatExp1/10;                          //ทำให้ค่าลดลงอยู่ในช่วง 0-1
    floatExp3=1-floatExp2;                           //นำค่าที่ได้ลบออกจาก 1
    intExp2=(-1*(floatInter))*logl(floatExp3);       //นำค่าที่ได้แทนในสมการเพื่อได้ระยะ
เวลา
    intPacket[0][1]=intExp2;                         //ของ Packet แรกที่เกิดขึ้น
    t2=0;                                            //กำหนดเริ่มนับจำนวน Packet ที่เกิดขึ้น
    while(intPacket[t2][1]<floatTimes)
    {
     floatExp1=random(10);                           //สุ่มหาค่าช่วง 0-10
     if((floatExp1>0)&&(floatExp1<10))
     {
      t2=t2+1;                                       //เพิ่มจำนวน Packet เกิดขึ้น
      floatExp2=floatExp1/10;                        //ทำให้ค่าลดลงอยู่ในช่วง 0-1
      floatExp3=1-floatExp2;                         //นำค่าที่ได้ลบออกจาก 1
      intExp2=intPacket[t2-1][1]+((-1*(floatInter))*logl(floatExp3));
      intPacket[t2][1]=intExp2;                      //นำค่าที่ได้แทนในสมการเพื่อได้ระยะห่าง
     }                                               //ของ Packet นำค่าไปบวกกับเวลาที่ Packet
    }                                                //เกิดก่อนหน้านี้จะได้เวลาที่ Packet เกิดขึ้น
   }
   if(Form1->floatProperty[1][11][3]==1)             //*ถ้าความยาว Packet คงที่*
   {
    a3=0;                                            //ตั้งค่านับจำนวน Packet =0
    while(a3<=t2)                                    //วน Loop สร้าง Packet
    {
     a3=a3+1;
     intPacket[a3][3]=11;                            //กำหนดจุดเริ่มต้นของ Packet
     intPacket[a3][4]=11;                            //กำหนดจุดปัจจุบันของ Packet
```

```
  intPacket[a3][5]=21;                    //กำหนดจุดเปลายทางของ Packet
  intPacket[a3][6]=Form1->floatProperty[1][11][4]; //กำหนดความยาวของ Packet มีค่าคงที่
  NumPacket=NumPacket+1;                  //เท่ากับค่า interarrival arge และเพิ่มจำนวน
 }                                        //Packet
}
else if(Form1->floatProperty[1][11][3]==2) //*ถ้าความยาวของ Packet สอดคล้องสมการ
{                                         //exponential*
 floatExp4=random(10);                    //สุ่มหาค่าช่วง 0-10
 floatExp5=floatExp4/10;                  //ทำให้ค่าลดลงอยู่ในช่วง 0-1
 floatExp6=1-floatExp5;                   //นำค่าที่ได้ลบออกจาก 1
 intExp1=(-1*(Form1->floatProperty[1][11][4]))*logl(floatExp6); //นำค่าที่ได้แทนในสมการ
 intPacket[0][6]=intExp1;                 //เพื่อได้ความยาว Packet แรก
 a3=0;
 while(a3<=t2)                            //วน loop สร้าง Packet จนครบจำนวน
 {
  floatExp4=random(10);
  if((floatExp4>0)&&(floatExp4<10))
  {
   a3=a3+1;
   intPacket[a3][3]=11;
   intPacket[a3][4]=11;
   intPacket[a3][5]=21;

   floatExp5=floatExp4/10;
   floatExp6=1-floatExp5;
   intExp1=(-Form1->floatProperty[1][11][4])*logl(floatExp6);
   intPacket[a3][6]=intExp1;

   NumPacket=NumPacket+1;
  }
 }
}
```

```
    intPacket[10000][1]=0;                    //กำหนดค่า Packet ที่อนันต์เป็น 0
    intPacket[10000][2]=0;
    intPacket[10000][3]=0;
    intPacket[10000][4]=0;
    intPacket[10000][5]=0;
    intPacket[10000][6]=0;
   }
   for(int i=0; i<=1000; i++)
   {
    Series1->AddXY(i,intPacket[i][1]/10000,"",clTeeColor);
   }
   floatTimesQn=intPacket[1][1]+floatTimes2;      //กำหนดค่าเวลาแรกที่ Queue จะส่ง
Packet
   floatTimes1=intPacket[1][1]-floatTimes2;       //กำหนดค่าเวลาเริ่มต้น simulation
   NumSink=1;
   Series2->Clear();
   Series3->Clear();
    ProgressBar1->Max=floatTimes;
   while(floatTimes1<=floatTimes)                 //เริ่มการ simulation
   {
    floatTimes1=floatTimes1+floatTimes2;          //เพิ่มค่าเวลาการ simulation
    ProgressBar1->Position=floatTimes1;
    if(floatTimes1>intPacket[1][1])               //ถ้าเวลาการ simulation มากกว่าเวลาที่
Packet
    {                                             //ส่งออกมา จะลดค่าเวลาการ simulation ลง
     floatTimes2=floatTimes1-intPacket[1][1];     //ให้เท่ากับเวลาที่ Packet ส่งออกมา
     floatTimes1=intPacket[1][1];
    }
    else                                          //ถ้าเวลาการ simulation น้อยกว่าหรือเท่ากับ
    {                                             //เวลาที่ Packet ส่งออกมา จะเพิ่มค่าเวลาการ
     floatTimes2=(1/floatService)*10000;          //simulation โดยบวกด้วยอัตราการส่ง Packet
    }                                             //ของ Queue ต่อ 1 bit
```

```
if(floatTimes1==floatTimesQn)              //ถ้าเวลาการ simulation เท่ากับเวลาการส่ง
{                                          // Packet ของ Queue
 if(intPacketQ[1][6]>0)                    //ถ้าตำแหน่งแรกของ Queue มีPacket อยู่จะลด
 {                                         //ค่าความยาว Packet นั้นลง 1 bit และเพิ่มค่า
  intPacketQ[1][6]=intPacketQ[1][6]-1;     //Packet ที่ Transmitter 1 bit
  if((intPacketQ[1][6]==0)&&(intPacketQ[2][6]==0))  //ถ้า Queue ส่ง Packet แรกใน
  {                                                 // Queue ออกไปหมดแล้วก็จะบันทึก
   intPacketS[NumSink][1]=intPacketQ[1][1];         //เวลาลงในตำแหน่งที่บันทึกเวลา
ขณะ
   intPacketS[NumSink][2]=floatTimes1;              //นั้น เป็นการสิ้นสุดการส่ง Packet
นั้น
   NumSink=NumSink+1;                               //ออกจาก Queue
  }
  if((intPacketQ[1][6]==0)&&(intPacketQ[2][6]>0))  //ถ้า Queue ส่ง Packet แรกใน
  {                                                 // Queue ออกไปหมดแล้ว แต่ในลำดับ
   intPacketS[NumSink][1]=intPacketQ[1][1];         //ต่อไปยังมี Packet อยู่จะบันทึกเวลา
   intPacketS[NumSink][2]=floatTimes1;              //ขณะนั้น เป็นการสิ้นสุดการส่ง Packet
   NumSink=NumSink+1;                               //นั้นออกจาก Queue แล้วเลื่อน Packet
                                                    //ต่อไปเข้ามาแทนที่เรื่อยๆ นับจำนวน
                                                    // Packet ที่ส่งสำเร็จ
   for(int i=1; i<=Form1->intQueueSize+1; i++)
   {
    intPacketQ[i][1]=intPacketQ[i+1][1];
    intPacketQ[i][2]=intPacketQ[i+1][2];
    intPacketQ[i][3]=intPacketQ[i+1][3];
    intPacketQ[i][4]=intPacketQ[i+1][4];
    intPacketQ[i][5]=intPacketQ[i+1][5];
    intPacketQ[i][6]=intPacketQ[i+1][6];
   }
   intPacketQ[Form1->intQueueSize][1]=0;
   intPacketQ[Form1->intQueueSize][2]=0;
   intPacketQ[Form1->intQueueSize][3]=0;
```

```
    intPacketQ[Form1->intQueueSize][4]=0;
    intPacketQ[Form1->intQueueSize][5]=0;
    intPacketQ[Form1->intQueueSize][6]=0;
   }
  }
  floatTimesQn=floatTimesQn+floatTimes2;          //เพิ่มเวลาที่ Queue ส่ง Packet ครั้งต่อไป
 }
 intI=1;
 u=1;
 if(floatTimes1==intPacket[1][1])                  //ถ้าเวลาการ simulation เท่ากับ เวลาที่
 {                                                 //Generator ส่ง Packet ออกมา
  q=q+1;                                           //นับจำนวน Packet ที่เกิดขึ้น
  int intI=1;
  while(intI<Form1->intQueueSize)                  //ส่ง Packet จาก Generator ไปที่ Queue โดย
  {                                                //ตรวจดูว่า Queue ที่ตำแหน่งไหนว่างอยู่
   if(intPacketQ[u][6]==0)                         //ก็จะส่ง Packet ไปที่ตำแหน่งนั้น
   {                                               //โดยตรวจดูเรียงจากน้อยไปหามาก
    intPacketQ[u][1]=intPacket[1][1];
    intPacketQ[u][2]=floatTimes1;
    intPacketQ[u][3]=intPacket[1][3];
    intPacketQ[u][4]=15;
    intPacketQ[u][5]=intPacket[1][5];
    intPacketQ[u][6]=intPacket[1][6];
    intI=Form1->intQueueSize+1;
    u=u+1;
   }
   else
   {
    intI=intI+1;
    u=u+1;
   }
  }
```

```
    for(int i=1; i<9999; i++)                    //เมื่อส่ง Packet ไปแล้วก็จะเลื่อน Packet ต่อ
ไป
    {                                            //มาแทนที่
     intPacket[i][1]=intPacket[i+1][1];
     intPacket[i][2]=intPacket[i+1][2];
     intPacket[i][3]=intPacket[i+1][3];
     intPacket[i][4]=intPacket[i+1][4];
     intPacket[i][5]=intPacket[i+1][5];
     intPacket[i][6]=intPacket[i+1][6];
    }
   }
  }
  a1=NumSink;
  a2=NumPacket;
  floatTrueput[NumTrueput]=a1/a2;
  for(int i=1; i<=NumTrueput; i++)
  {
   Series2->AddXY(i,floatTrueput[i],"",clTeeColor);
  }
  for(int i=1; i<=NumSink-1; i++)
  {
   Series3->AddXY(intPacketS[i][1]/10000,(intPacketS[i][2]-
intPacketS[i][1])/10000,"",clTeeColor);
  }
 }
 if(RadioButton2->Checked==true)                 //แบบที่2 Simulatiom แบบนับจำนวนPacket
 {
  ProgressBar1->Position=0;                      //ลักษณะการสร้าง Packet จะคล้อยกับแบบที่
1
  floatInter=StrToFloat(Form18->Edit2->Text)*10000; //ต่างกันที่จะสร้าง Packet เพียง 1,000
  floatService=StrToFloat(Form18->Edit3->Text);     // Packet แรก และเมื่อ simulation ไป
  NumCount=StrToInt(Form18->Edit5->Text);           //เรื่อยๆ จนเหลือ 5 Packet สุดท้าย ก็จะ
```

```
floatTimes2=(10000/floatService);                    //สร้าง Packet ใหม่ต่อจาก Packet เดิม
q=0;                                                 //อีก 995 Packet ทำเช่นนี้ไปเรื่อยๆจน
t2=0;                                                //ส่ง Packet ได้ตามจำนวนที่กำหนด
NumPacket=0;
floatTimes1=0;
if((Form1->intPropertyNode[1]==1)&&( Form1->floatProperty[1][11][4]>=0)&&(Form1-
>floatProperty[1][11][3]==1)&&(Form1->floatProperty[1][11][1]==1))
{
 Series1->Clear();
 intPacket[0][1]=random(10000);
 while(t2<=1000)
 {
  t2=t2+1;
  intPacket[t2][1]=intPacket[t2-1][1]+floatInter;
  intPacket[t2][3]=11;
  intPacket[t2][4]=11;
  intPacket[t2][5]=21;
  intPacket[t2-1][6]=Form1->floatProperty[1][11][4];
  Series1->AddXY(t2,intPacket[t2][1]/10000,"",clTeeColor);
 }
  intPacket[10000][1]=0;
  intPacket[10000][2]=0;
  intPacket[10000][3]=0;
  intPacket[10000][4]=0;
  intPacket[10000][5]=0;
  intPacket[10000][6]=0;
}
if((Form1->intPropertyNode[1]==1)&&( Form1->floatProperty[1][11][4]>=0)&&(Form1-
>floatProperty[1][11][3]==2)&&(Form1->floatProperty[1][11][1]==2))
{
 Series1->Clear();
 floatExp1=random(10);
```

```
floatExp2=floatExp1/10;
floatExp3=1-floatExp2;
intExp2=(-1*(floatInter))*logl(floatExp3);
intPacket[0][1]=intExp2;

floatExp4=random(10);
floatExp5=floatExp4/10;
floatExp6=1-floatExp5;
intExp1=(-1*(Form1->floatProperty[1][11][4]))*logl(floatExp6);
intPacket[0][6]=intExp1;
while(t2<=10000)
{
 floatExp1=random(10);
 floatExp4=random(10);
 if((floatExp1>0)&&(floatExp4>0)&&(floatExp1<10)&&(floatExp4<10))
 {
  t2=t2+1;
  floatExp2=floatExp1/10;
  floatExp3=1-floatExp2;
  intExp2=intPacket[t2-1][1]+((-1*(floatInter))*logl(floatExp3));
  intPacket[t2][1]=intExp2;

  floatExp5=floatExp4/10;
  floatExp6=1-floatExp5;
  intExp1=(-Form1->floatProperty[1][11][4])*logl(floatExp6);
  intPacket[t2][6]=intExp1;
 }
}
intPacket[10000][1]=0;
intPacket[10000][2]=0;
intPacket[10000][3]=0;
intPacket[10000][4]=0;
```

```
intPacket[10000][5]=0;
intPacket[10000][6]=0;
}
if((Form1->intPropertyNode[1]==1)&&( Form1->floatProperty[1][11][4]>=0)&&(Form1->floatProperty[1][11][3]==2)&&(Form1->floatProperty[1][11][1]==1))
{
Series1->Clear();
intPacket[0][1]=random(10000);
floatExp4=random(10);
floatExp5=floatExp4/10;
floatExp6=1-floatExp5;
intExp1=(-1*(Form1->floatProperty[1][11][4]))*logl(floatExp6);
intPacket[0][6]=intExp1;
while(t2<=10000)
{
floatExp4=random(10);
if((floatExp4>0)&&(floatExp4<10))
{
t2=t2+1;
intPacket[t2][1]=intPacket[t2-1][1]+floatInter;

floatExp5=floatExp4/10;
floatExp6=1-floatExp5;
intExp1=(-Form1->floatProperty[1][11][4])*logl(floatExp6);
intPacket[t2][6]=intExp1;
}
}
intPacket[10000][1]=0;
intPacket[10000][2]=0;
intPacket[10000][3]=0;
intPacket[10000][4]=0;
intPacket[10000][5]=0;
```

```
 intPacket[10000][6]=0;
 }
 if((Form1->intPropertyNode[1]==1)&&( Form1->floatProperty[1][11][4]>=0)&&(Form1-
>floatProperty[1][11][3]==1)&&(Form1->floatProperty[1][11][1]==2))
 {
 Series1->Clear();
 floatExp1=random(10);
 floatExp2=floatExp1/10;
 floatExp3=1-floatExp2;
 intExp2=(-1*(floatInter))*logl(floatExp3);
 intPacket[0][1]=intExp2;
 intPacket[0][6]=Form1->floatProperty[1][11][4];
 while(t2<=10000)
 {
  floatExp1=random(10);

  if((floatExp1>0)&&(floatExp1<10))
  {
   t2=t2+1;
   floatExp2=floatExp1/10;
   floatExp3=1-floatExp2;
   intExp2=intPacket[t2-1][1]+((-1*(floatInter))*logl(floatExp3));
   intPacket[t2][1]=intExp2;
   intPacket[t2][6]=Form1->floatProperty[1][11][4];
  }
 }
 intPacket[10000][1]=0;
 intPacket[10000][2]=0;
 intPacket[10000][3]=0;
 intPacket[10000][4]=0;
 intPacket[10000][5]=0;
 intPacket[10000][6]=0;
```

```
}
for(int i=0; i<=1000; i++)
{
 Series1->AddXY(i,intPacket[i][1]/10000,"",clTeeColor);
}
floatTimesQn=intPacket[1][1]+floatTimes2;
floatTimes1=intPacket[1][1]-floatTimes2;
NumSink=1;
Series2->Clear();
Series3->Clear();
ProgressBar1->Max=NumCount;
while(NumSink<=NumCount)                    //เริ่มการ simulation โดยทำขณะที่จำนวน
{                                           // Packet ที่ส่งสำเร็จน้อยกว่าหรือเท่ากับ
 floatTimes1=floatTimes1+floatTimes2;       //จำนวนที่กำหนด
 if(floatTimes1>intPacket[1][1])
 {
  floatTimes2=floatTimes1-intPacket[1][1];
  floatTimes1=intPacket[1][1];
 }
 else
 {
  floatTimes2=(1/floatService)*10000;
 }
 if(floatTimes1==floatTimesQn)
 {
  if(intPacketQ[1][6]>0)
  {
   intPacketQ[1][6]=intPacketQ[1][6]-1;

   if((intPacketQ[1][6]==0)&&(intPacketQ[2][6]==0))
   {
    intPacketS[NumSink][1]=intPacketQ[1][1];
```

```
    intPacketS[NumSink][2]=floatTimes1;
    NumSink=NumSink+1;
   }
   if((intPacketQ[1][6]==0)&&(intPacketQ[2][6]>0))
   {
    intPacketS[NumSink][1]=intPacketQ[1][1];
    intPacketS[NumSink][2]=floatTimes1;
    NumSink=NumSink+1;
    for(int i=1; i<=Form1->intQueueSize+1; i++)
    {
     intPacketQ[i][1]=intPacketQ[i+1][1];
     intPacketQ[i][2]=intPacketQ[i+1][2];
     intPacketQ[i][3]=intPacketQ[i+1][3];
     intPacketQ[i][4]=intPacketQ[i+1][4];
     intPacketQ[i][5]=intPacketQ[i+1][5];
     intPacketQ[i][6]=intPacketQ[i+1][6];
    }
    intPacketQ[Form1->intQueueSize+1][1]=0;
    intPacketQ[Form1->intQueueSize+1][2]=0;
    intPacketQ[Form1->intQueueSize+1][3]=0;
    intPacketQ[Form1->intQueueSize+1][4]=0;
    intPacketQ[Form1->intQueueSize+1][5]=0;
    intPacketQ[Form1->intQueueSize+1][6]=0;
   }
   ProgressBar1->Position=NumSink;
  }
  floatTimesQn=floatTimesQn+floatTimes2;
 }
 intI=1;
 u=1;
 if(floatTimes1==intPacket[1][1])
 {
```

```
NumPacket=NumPacket+1;
q=q+1;
if(intPacket[5][6]==0)                    //เมื่อจำนวน Packet เหลือ 5 Packet เริ่มสร้าง
{                                         // Packet ใหม่
  if((Form1->intPropertyNode[1]==1)&&( Form1->floatProperty[1][11][4]>=0)&&(Form1->floatProperty[1][11][3]==1)&&(Form1->floatProperty[1][11][1]==1))
  {
   intPacket[0][1]=random(10000);
   t2=5;
   while(t2<=1000)
   {
    t2=t2+1;
    intPacket[t2][1]=intPacket[t2-1][1]+floatInter;

    intPacket[t2][3]=11;
    intPacket[t2][4]=11;
    intPacket[t2][5]=21;
    intPacket[t2-1][6]=Form1->floatProperty[1][11][4];
   }
   intPacket[10000][1]=0;
   intPacket[10000][2]=0;
   intPacket[10000][3]=0;
   intPacket[10000][4]=0;
   intPacket[10000][5]=0;
   intPacket[10000][6]=0;
  }
  if((Form1->intPropertyNode[1]==1)&&( Form1->floatProperty[1][11][4]>=0)&&(Form1->floatProperty[1][11][3]==2)&&(Form1->floatProperty[1][11][1]==2))
  {
   floatExp1=random(10);
```

```
floatExp2=floatExp1/10;
floatExp3=1-floatExp2;
intExp2=(-1*(floatInter))*logl(floatExp3);
intPacket[0][1]=intExp2;

floatExp4=random(10);
floatExp5=floatExp4/10;
floatExp6=1-floatExp5;
intExp1=(-1*(Form1->floatProperty[1][11][4]))*logl(floatExp6);
intPacket[0][6]=intExp1;
t2=5;
while(t2<1000)
{
 floatExp1=random(10);
 floatExp4=random(10);
 if((floatExp1>0)&&(floatExp4>0)&&(floatExp1<10)&&(floatExp4<10))
 {
  t2=t2+1;
  floatExp2=floatExp1/10;
  floatExp3=1-floatExp2;
  intExp2=intPacket[t2-1][1]+((-1*(floatInter))*logl(floatExp3));
  intPacket[t2][1]=intExp2;
  floatExp5=floatExp4/10;
  floatExp6=1-floatExp5;
  intExp1=(-Form1->floatProperty[1][11][4])*logl(floatExp6);
  intPacket[t2][6]=intExp1;
 }
}
intPacket[10000][1]=0;
intPacket[10000][2]=0;
intPacket[10000][3]=0;
intPacket[10000][4]=0;
```

```
   intPacket[10000][5]=0;
   intPacket[10000][6]=0;
  }
}
int intI=1;
while(intI<Form1->intQueueSize)
{
 if(intPacketQ[u][6]==0)
 {
  intPacketQ[u][1]=intPacket[1][1];
  intPacketQ[u][2]=floatTimes1;
  intPacketQ[u][3]=intPacket[1][3];
  intPacketQ[u][4]=15;
  intPacketQ[u][5]=intPacket[1][5];
  intPacketQ[u][6]=intPacket[1][6];
  intI=Form1->intQueueSize+1;
    u=u+1;
 }
 else
 {
  intI=intI+1;
  u=u+1;
 }
}
for(int i=1; i<9999; i++)
{
 intPacket[i][1]=intPacket[i+1][1];
 intPacket[i][2]=intPacket[i+1][2];
 intPacket[i][3]=intPacket[i+1][3];
 intPacket[i][4]=intPacket[i+1][4];
 intPacket[i][5]=intPacket[i+1][5];
 intPacket[i][6]=intPacket[i+1][6];
```

```
      }
     }
    }
    a1=NumSink;
    a2=NumPacket;
    floatTrueput[NumTrueput]=a1/a2;
    for(int i=1; i<=NumTrueput; i++)
    {
     Series2->AddXY(i,floatTrueput[i],"",clTeeColor);
    }
    for(int i=1; i<=NumSink-1; i++)
    {
     Series3->AddXY(intPacketS[i][1]/10000,(intPacketS[i][2]-
intPacketS[i][1])/10000,"",clTeeColor);
    }
  }
}
//---------------------------------------------------------------------------
void __fastcall TForm18::RadioButton1Click(TObject *Sender)
{
 Edit4->Enabled=true;
 Label1->Enabled=true;
 Edit5->Enabled=false;
 Label2->Enabled=false;
}
//---------------------------------------------------------------------------
void __fastcall TForm18::RadioButton2Click(TObject *Sender)
{
 Edit5->Enabled=true;
 Label2->Enabled=true;
 Edit4->Enabled=false;
 Label1->Enabled=false;
```

```
}
//-----------------------------------------------------------------
```

ประวัติผู้ทำโครงงาน

ชื่อ นางสาวพัชนก คำก้อน

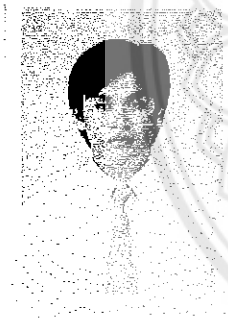
ภูมิลำเนา จังหวัดอุดรธานี

ประวัติการศึกษา - ปีการศึกษา 2536 จบชั้นประถมศึกษาปีที่ 6 โรงเรียนอนุบาลอุดรธานี จ.อุดรธานี

- ปีการศึกษา 2542 จบชั้นมัธยมศึกษาปีที่ 6 โรงเรียนกุดจับประชาสรรค์ จ.อุดรธานี

- ปัจจุบัน ปีการศึกษา 2546 กำลังศึกษา คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยนเรศวร จ.พิษณุโลก

E-mail hacha15@hotmail.com



ชื่อ นายอัฐราวุฒิ เดชผล

ภูมิลำเนา จังหวัดอุบลราชธานี

ประวัติการศึกษา - ปีการศึกษา 2534 จบชั้นประถมศึกษาปีที่ 6 โรงเรียนอุบลวิทยาคม จ.อุบลราชธานี

- ปีการศึกษา 2540 จบชั้นมัธยมศึกษาปีที่ 6 โรงเรียนเบ็ญจะมะมหาราช จ.อุบลราชธานี

- ปัจจุบัน ปีการศึกษา 2546 กำลังศึกษา คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยนเรศวร จ.พิษณุโลก

E-mail Auttarawut@hotmail.com