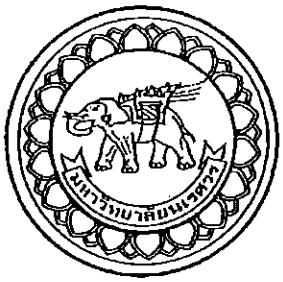


ไลบรารีสำหรับจำลองเน็ตเวิร์คโพรโตคอลสแตก

A Library For Simulating Network Protocol Stack

นายกิตติศักดิ์ ตุนาโป่ง รหัส 47370085

นายณัฐเสฏฐ์ วัชรานุรักษ์ รหัส 47370127

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2550

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	ไลบรารีสำหรับจำลองเน็ตเวิร์คโพรโตคอลสแตก		
ผู้ดำเนินโครงงาน	นายกิตติศักดิ์	ตุนาโป่ง	รหัส 47370085
	นายณัฐเสฏฐ์	วัชรานุรักษ์	รหัส 47370127
อาจารย์ที่ปรึกษา	ดร.สุรเดช จิตประไพกุลศาล		
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2550		

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ
(ดร.สุรเดช จิตประไพกุลศาล)

.....กรรมการ
(อาจารย์ แสงชัย มังกรทอง)

.....กรรมการ
(ดร.ชัยรัตน์ พินทอง)

หัวข้อโครงงาน	ไลบรารีสำหรับจำลองเน็ตเวิร์คโพรโตคอลสแตก		
ผู้ดำเนินโครงงาน	นายกิตติศักดิ์	ตุนาโป่ง	รหัส 47370085
	นายณัฐเสฏฐ์	วัชรานุรักษ์	รหัส 47370127
อาจารย์ที่ปรึกษา	ดร.สุรเดช จิตประไพกุลศาล		
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2550		

.....

บทคัดย่อ

โครงงานนี้ได้ศึกษาและพัฒนาไลบรารี สำหรับจำลองการทำงานของเน็ตเวิร์คโพรโตคอลสแตกแบบ ทีซีพี/ไอพี โมเดล เพื่อใช้เป็นโครงร่างในการจำลองที่แสดงให้เห็นถึงการทำงานอย่างเป็นลำดับขั้นของเน็ตเวิร์ค โครงงานนี้พัฒนาโดยใช้ Interface ในภาษาจาวา เพื่อให้ผู้ใช้สามารถนำไลบรารีนำไปใช้ได้และพัฒนาไลบรารีเพิ่มเติมได้ โดยไม่จำเป็นต้องไปแก้ไขโปรแกรมเก่า เพราะโครงร่างภายในมีความเป็นอิสระต่อกัน ทั้งนี้ในโครงการนี้จะนำเสนอเพียงการจำลองของเลเยอร์ 3 และ 4 จากการทดลองโปรแกรมสามารถจำลองเน็ตเวิร์คได้ทั้ง 2 เลเยอร์ ทั้งส่งและรับข้อความ อนึ่งตัวส่งและตัวรับสามารถทำงานได้ 1) บนเครื่องเดียวกัน 2) คนละเครื่องกันแต่ระบบปฏิบัติการเดียวกัน หรือ 3) คนละเครื่องกันและต่างระบบปฏิบัติการกัน ระบบปฏิบัติการที่ทดสอบได้แก่ Microsoft Windows XP และ Ubuntu Linux

Project Title	A Library For Simulating Network Protocol Stack.		
Name	Mr.Kittisak	Tunapong	ID. 47370085
	Mr.Nuttasate	Watcharanurak	ID. 47370127
Project Advisor	Dr.Suradet	Jitprapaikulsarn	
Major	Computer Engineering.		
Department	Electrical and Computer Engineering.		
Academic Year	2007		

.....

Abstract

This project developed a library for simulating TCP/IP Network protocol stack. We implement our library using interface in Java so that the users can either use the library as-is or develop their own library without changing any existing programs. In this project, we only demonstrate the simulation for layers 3 and 4 for both sending and receiving messages. Both sender and receiver programs can run on i) the same computer, ii) different computers with same operating system, or iii) different computers with different operating systems. The operating systems that we tests our programs are Windows XP and Ubuntu Linux.

กิตติกรรมประกาศ

โครงงานวิศวกรรมคอมพิวเตอร์สำเร็จได้ด้วยดี ก็เนื่องจากความอนุเคราะห์จากท่านอาจารย์ที่ปรึกษา ดร.สุรเดช จิตประไพกุลศาล ที่กรุณาให้คำปรึกษา แนะนำวิธีการในการทำงานตลอดถึงการตรวจสอบการทำงานพร้อมทั้งเสนอแนวทางการแก้ไขตลอดระยะเวลาการทำโครงงานสุดท้ายต้องขอขอบพระคุณอาจารย์ทุกท่านและเพื่อนๆ ทุกคนที่ยังไม่ได้เอ่ยนามที่คอยสนับสนุนในการทำโครงงานครั้งนี้

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ในการเรียนวิชา Principle of Network System Programming ได้มีการเขียนโปรแกรมจำลองการทำงานของระบบเครือข่าย แต่ยังขาด Library ที่ช่วยในการจำลอง Network Protocol Stack จึงมีความคิดที่จะสร้าง Library เพื่อช่วยในการจำลองให้ง่ายขึ้น และช่วยในการทำความเข้าใจเกี่ยวกับระบบเครือข่าย

1.2 วัตถุประสงค์โครงงาน

1. เพื่อสร้าง Library สำหรับจำลอง Network Protocol Stack
2. เพื่อให้ผู้ใช้สามารถนำไลบรารีไปพัฒนาเพิ่มเติมได้

1.3 ขอบเขตของโครงงาน

1. สามารถสร้างโครงร่างให้ผู้อื่นสามารถใช้งานได้
2. จำลองการทำงานที่เลเยอร์ 4 โดยใช้ TCP/IP Protocol เพื่อเป็นตัวอย่างได้
3. จำลองการทำงานที่เลเยอร์ 3 โดยใช้ IPv4 Protocol เพื่อเป็นตัวอย่างได้

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาความรู้ทั่วไปเกี่ยวกับ TCP model
2. ศึกษาการทำงานของ TCP/IP Protocol
3. ศึกษาการทำงานของ IPv4 Protocol
4. สร้าง Library ให้ได้ตามวัตถุประสงค์
5. ทดสอบความถูกต้องของ Library
6. จัดทำเอกสารและคู่มือการใช้

1.5 แผนการดำเนินงาน

กิจกรรม	ปี 2550					ปี 2551			
	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.
1. ศึกษาความรู้ทั่วไปเกี่ยวกับ TCP model									
2. ศึกษาการทำงานของ TCP/IP Protocol									
3. ศึกษาการทำงานของ IPv4 Protocol									
4. สร้าง Library ให้ได้ตามวัตถุประสงค์									
5. ทดสอบความถูกต้องของ Library									
6. จัดทำเอกสารและคู่มือการใช้									

1.6 ผลที่คาดว่าจะได้รับ

1. เพื่อช่วยในการศึกษาเกี่ยวกับ Network Protocol Stack ได้ง่ายขึ้น
2. เพื่อใช้ในการจำลอง Network Protocol Stack ได้ง่ายขึ้น
3. ใช้เป็นตัวอย่างในการจำลอง Network Protocol Stack

1.7 รายละเอียดงบประมาณของโครงงาน

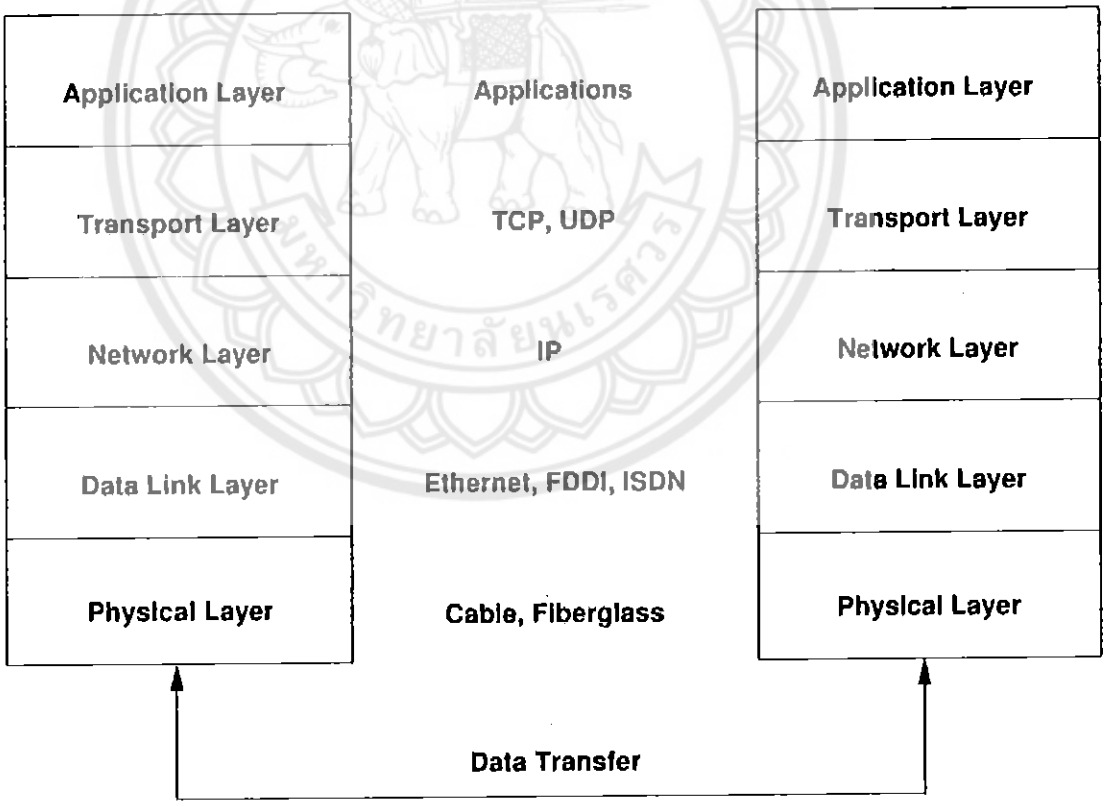
1. หนังสือประกอบการทำโครงงาน	1,000 บาท	
2. ค่าเอกสาร	500 บาท	
3. อื่นๆ	500 บาท	
รวมเป็นเงินทั้งสิ้น	2000 บาท	(สองพันบาทถ้วน)

บทที่ 2
หลักการและทฤษฎี

ในบทนี้จะกล่าวถึงทฤษฎีและหลักการที่ใช้ในการทำงานของโปรแกรม โดยจะกล่าวตั้งแต่ทีซีพีโมเดล (TCP/IP Model) และการเขียนโปรแกรมภาษาจาวา (Java Language) โดยใช้อินเทอร์-เฟซ (Interface) เพื่อเป็นพื้นฐานในการสร้างไลบรารีสำหรับจำลองเน็ตเวิร์คโพรโตคอลสแตก

2.1 โครงสร้างแบบ ทีซีพี/ไอพี (TCP/IP model)

เป็นมาตรฐานที่ทำให้คอมพิวเตอร์ภายในระบบเครือข่ายอินเทอร์เน็ต สามารถเชื่อมต่อเข้าหากัน และติดต่อสื่อสารแลกเปลี่ยนข้อมูลกันได้ เป็นมาตรฐานที่ว่าด้วยการกำหนดวิธีการติดต่อสื่อสารระหว่างคอมพิวเตอร์ โดยใช้แนวคิดของการแบ่งลำดับชั้นโพรโตคอล



รูปที่ 2.1 TCP/IP Model

2.1.1. Physical Layer และ Data Link Layer

หน้าที่ของชั้นนี้สำหรับการส่งข้อมูล เนื่องจากตัวแบบ ทีซีพี/ไอพี ไม่ได้กำหนดมาตรฐานในขั้นตอนนี้อย่างมากนัก กำหนดไว้เพียงว่าให้สามารถส่งข้อมูลสู่เครือข่ายได้เท่านั้น ทำให้ไม่สามารถระบุเนื้อหาหน้าที่ที่ชัดเจนได้ ดังนั้นจึงอาจจะยกระบบของ โครงสร้างแบบ OSI ทั้งสองชั้นแรกมาซึ่งได้แก่การจัดเตรียมข้อมูลเพื่อให้เหมาะสมก่อนที่จะส่งไปตามสายส่งไปยังที่หมายปลายทาง ซึ่งได้แก่การจัดเตรียม Packet Header การควบคุมระบบฮาร์ดแวร์และซอฟต์แวร์ที่จะใช้ในการจัดส่ง เช่น การเชื่อมต่อกับ Network card และการใช้งาน Device Driver หน้าที่สำหรับการรับข้อมูลคือ คอยรับกรอบของข้อมูลที่ได้รับ นำข้อมูลส่วนหัวออกมา และจัดเตรียมข้อมูลเพื่อส่งต่อไปยังชั้นเครือข่าย

2.1.2. Network Layer

ชั้นอินเตอร์เน็ต หรือ Internet Layer มีหน้าที่ส่งข้อมูลจากจุดเริ่มต้นไปยังปลายทาง โดยหาเส้นทางที่ข้อมูลจะใช้เดินทางผ่านเครือข่ายหนึ่งไปยังอีกเครือข่ายหนึ่งจนกระทั่งถึงปลายทาง

2.1.3. Transport Layer

ชั้นขนส่ง หรือ Transport layer เป็นชั้นที่มีหน้าที่ควบคุมการส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์สองเครื่องที่ติดต่อกัน และจัดส่งข้อมูลไปยัง Application ที่ต้องการข้อมูล โปรโตคอลที่นิยมใช้ในชั้นนี้ได้แก่ TCP, UDP, RTP

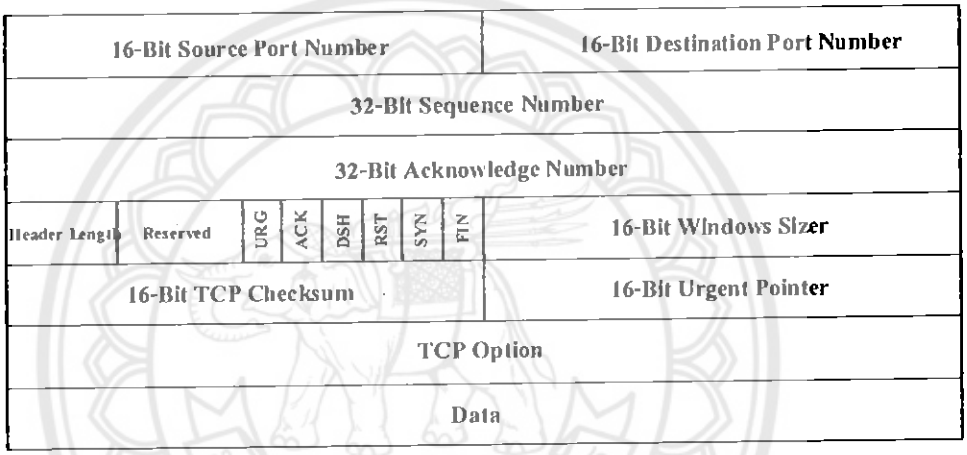
2.1.4. Application Layer

ชั้นการประยุกต์ใช้งาน หรือ Application layer จะครอบคลุมบริการที่เกี่ยวข้องกับการรักษาความปลอดภัย การเข้ารหัส การเชื่อมต่อระหว่างโปรแกรมประยุกต์ และเป็นชั้นที่โปรแกรมประยุกต์ใช้งานโดยตรง โดยโปรโตคอลที่อยู่บนชั้นนี้จะถูกออกแบบให้เหมาะสำหรับประเภทของโปรแกรมประยุกต์เฉพาะทาง เช่น โปรแกรม E-Mail ใช้โปรโตคอล SMTP สำหรับส่ง E-mail ใช้โปรโตคอล POP3 สำหรับรับและเรียกดู E-Mail, ส่วนโปรแกรม Web Browser ใช้โปรโตคอล HTTP สำหรับเรียกดู Web Page เป็นต้น

2.2 โครงสร้างของ TCP Header และ IPv4 Header

การส่งข้อมูลผ่านในแต่ละเลเยอร์ จะทำการประกอบข้อมูลที่ได้รับมา กับข้อมูลส่วนควบคุมซึ่งถูกนำมาไว้ในส่วนหัวของข้อมูลเรียกว่า Header ภายใน Header จะบรรจุข้อมูลที่สำคัญของโปรโตคอลที่ทำการ Encapsulate เมื่อผู้รับได้รับข้อมูล ก็จะเกิดกระบวนการทำงานย้อนกลับคือโปรโตคอลเดียวกัน ทางฝั่งผู้รับก็จะได้รับข้อมูลส่วนที่เป็น Header ก่อนและนำไปประมวลและทราบว่าข้อมูลที่ตามมามีลักษณะอย่างไร ซึ่งกระบวนการย้อนกลับนี้เรียกว่า Demultiplexing

2.2.1 TCP Header Format



รูปที่ 2.2 TCP Header Format

ข้อมูลในฟิลด์ของเฮดเดอร์ TCP มีความหมายดังต่อไปนี้

Source Port Number: หมายเลขพอร์ตต้นทางที่ส่งดาต้าแกรมนี้

Destination Port Number: หมายเลขพอร์ตปลายทางที่จะเป็นผู้รับดาต้าแกรม

Sequence Number: ฟิลด์ที่ระบุหมายเลขลำดับอ้างอิงในการสื่อสารข้อมูลแต่ละครั้ง เพื่อใช้ในการแยกว่าเป็นข้อมูลของชุดใด และนำมาจัดลำดับได้ถูกต้อง

Acknowledgment Number: ทำหน้าที่เช่นเดียวกับ Sequence Number แต่จะใช้ในการตอบรับ

Header Length: โดยปกติความยาวของเฮดเดอร์ TCP จะมีความยาว 20 ไบต์ แต่อาจจะมากกว่านั้น ถ้ามีข้อมูลในฟิลด์ Option แต่ต้องไม่เกิน 60 ไบต์

Flag: เป็นข้อมูลในระดับบิตที่ใช้เป็นตัวบอกคุณสมบัติของ TCP Segment ที่กำลังส่งอยู่นั้น และใช้เป็นตัวควบคุมจังหวะ การรับส่งข้อมูลด้วย ซึ่ง Flag ทั้งหมดมีอยู่ 6 บิต แต่ละบิตมีชื่อและมีความหมายดังนี้

- URG ใช้บอกความหมายว่าเป็นข้อมูลด่วน และมีข้อมูลพิเศษมาด้วย (อยู่ใน Urgent Pointer)
- ACK แสดงว่าข้อมูลในฟิลด์ Acknowledge Number นำมาใช้งานได้
- DSH เพื่อแจ้งให้ผู้รับทราบว่า ควรจะส่งข้อมูล Segment นี้ไปยังโพรเซสที่กำลังรออยู่ทันที
- RST ใช้ในกรณีที่เกิดการสับสนด้วยเหตุผลต่างๆ เช่น โฮสต์มีปัญหา ให้เริ่มต้นสื่อสารใหม่
- SYN ใช้ในการเริ่มต้นขอติดต่อกับปลายทาง
- FIN ใช้ส่งเพื่อแจ้งให้ปลายทางทราบว่ายุติการติดต่อ

Window Size: เป็นขนาดของการรับ - ส่งข้อมูลในแต่ละครั้ง

Checksum: ฟิลด์ที่ใช้ในการตรวจสอบความถูกต้องของข้อมูลใน TCP

Urgent Pointer: ข้อมูลเพิ่มเติมซึ่งจะอยู่ใน TCP Header เมื่อมีการตั้งค่า Option บางอย่างที่ต้องการข้อมูลเพิ่มเติมซึ่งไม่มีใน TCP Header เช่น MSS, Strict Route

2.2.1 IPv4 Header Format

Version	Header Length	Type Of Service	Total Length	
Identification			Flag	Fragment Offset
TTL		Protocol	Checksum	
Source Address				
Destination Address				
Options				Padding

รูปที่ 2.3 IPv4 Header Format

ข้อมูลในฟิลด์ของเฮดเดอร์ IPv4 มีความหมายดังต่อไปนี้

Version (4 บิต): ข้อมูล 4 บิตแรกที่บอกหมายเลขเวอร์ชันของ Protocol IP ที่ใช้ ซึ่งในปัจจุบันคือ เวอร์ชัน 4 (IPv4)

Internet Header Length หรือ IHL (4 บิต): ตัวเลขที่บอกความยาวของเฮดเดอร์ โดยทั่วไปถ้าไม่มีส่วน option จะมีค่าเป็น 5 หรือ 20 bytes

Type of Service หรือ TOS (8 บิต): ข้อมูลในฟิลด์นี้แต่ละบิตจะเป็น Flag บอกลำดับความสำคัญที่ใช้เป็นข้อมูลสำหรับ Router ในการตัดสินใจเลือกการจัดเส้นทางให้กับ Packet แต่ละ Packet (ในปัจจุบันไม่ได้มีการนำไปใช้งานแล้ว)

Total Length (16 บิต): ความยาวทั้งหมดของ Packet มีหน่วยเป็นจำนวนไบต์ ขนาดของฟิลด์นี้มีขนาด 16 บิต ดังนั้นความยาวสูงสุดของ Packet ที่เป็นไปได้ คือ 65536 bytes

Identification (16 บิต): เป็นหมายเลขของ Datagram ในกรณีที่มีการแยก Datagram ออกเป็น Packet หลาย Packet เมื่อข้อมูลส่งถึงปลายทางจะนำ Packet ที่มี Identification เดียวกันมารวมกัน

Flags (3 บิต): ใช้ในกรณีที่มีการแยก Datagram ออกเป็น Packet หลาย Packet

Fragment action offset (13 บิต): เป็นค่าบอกตำแหน่งของ Packet ใน Datagram ที่มีการแยกส่วนเพื่อให้สามารถนำ Packet กลับมาเรียงต่อกันได้อย่างถูกต้อง

Time to live หรือ TTL (8 บิต): บอกระยะเวลานานที่สุดที่ Packet จะสามารถอยู่ในโครงข่าย เพื่อป้องกันไม่ให้เกิดการส่งข้อมูลโดยไม่สิ้นสุด โดยเมื่อข้อมูลถูกส่งไป 1 Hop จะเป็นการลดค่า TTL ลง 1 เมื่อค่าของ TTL เป็น 0 และข้อมูลยังไม่ถึงปลายทาง ข้อมูลนั้น

จะถูกยกเลิก และ Router สุดท้ายจะส่งข้อมูล ICMP แจ้งกลับมายังต้นทางว่าเกิด Time out ในระหว่างการส่งข้อมูล

Protocol (8 บิต): ระบุ Protocol ของชั้นที่อยู่สูงกว่า เช่น TCP, UDP หรือ ICMP

Header checksum (16 บิต): ใช้ในการตรวจวัดความผิดพลาดของข้อมูลในเฮดเดอร์

Source IP address (32 บิต): เลขที่อยู่ไอพีของเครื่องคอมพิวเตอร์ต้นทาง

Destination IP address (32 บิต): เลขที่อยู่ไอพีของเครื่องคอมพิวเตอร์ปลายทาง

Data: ข้อมูลจาก Protocol ชั้นที่อยู่สูงกว่า ซึ่งมีความยาวไม่คงที่

2.3 การเขียนโปรแกรมโดยใช้อินเทอร์เฟซ (Interface)

ภาษาจาวามีรูปแบบและวิธีการเขียนแบบหนึ่ง คือการใช้งานอินเทอร์เฟซ (Interface) ร่วมกับอิมพลีเมนต์ (Implement) โดยที่เราจะจัดกลุ่มคลาส (Class) ที่มีความสามารถเหมือนกันโดยใช้อินเทอร์เฟซ ในอินเทอร์เฟซมีการประกาศว่าคลาสที่อยู่ในกลุ่มเดียวกันมีเมท็อด (Method) ใดบ้าง โดยจะประกาศเพียงชื่อเมท็อดเท่านั้น ไม่มีการอธิบายวิธีการทำงานในอินเทอร์เฟซ คลาสที่เข้ามาอยู่ในกลุ่มต้องอธิบายวิธีการทำงานของเมท็อดที่อยู่ในอินเทอร์เฟซโดยใช้อิมพลีเมนต์

รูปแบบสำหรับการสร้างอินเทอร์เฟซ

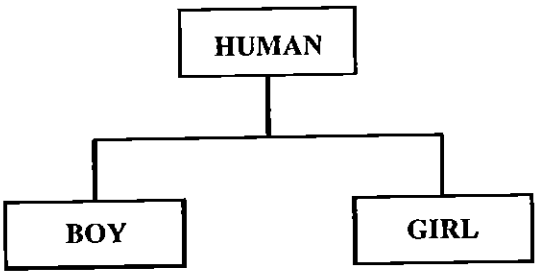
```
public interface Interface_name
{
        Data_Member
        abstract Method_Head
}
```

*หมายเหตุ : คำว่า public และ abstract สามารถละเว้นได้

รูปแบบสำหรับการสร้างคลาสที่ทำงานตาม Interface ที่กำหนด

```
Class Class_Name implement Interface_name
{
        Data_Member
        Method_Member
}
```

ตัวอย่างการใช้งานคลาสอินเทอร์เฟซ



รูปที่ 2.4 รูปตัวอย่างการใช้งานคลาสอินเทอร์เฟซ

จากรูป 2.4 จะทำการสร้างคลาสอินเทอร์เฟซขึ้นมา โดยใช้ชื่อคลาสว่า HUMAN และมีสมาชิกคลาส 2 คลาส คือ คลาส BOY และ คลาส GIRL ดังนี้

```
public interface HUMAN {
    public abstract int who();
}
```

```
public class BOY implements HUMAN{
    public int who(){
        System.out.println("This is a BOY.");
        return 0;
    }
}
```

```
public class GIRL implements HUMAN{
    public int who(){
        System.out.println("This is a GIRL.");
        return 0;
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        HUMAN B = new BOY();
        B.who();

        HUMAN G = new GIRL();
        G.who();
    }
}
```

รูปที่ 2.5 รูปตัวอย่างการเขียนโปรแกรมแบบอินเทอร์เฟซ

จากผลรันจะแสดงให้เห็นว่า สามารถเรียกใช้งานคลาส BOY และคลาส GIRL ได้

```
compile:
run:
This is a BOY.
This is a GIRL.
BUILD SUCCESSFUL (total time: 0 seconds)
```

รูปที่ 2.6 รูปผลการรันโปรแกรม

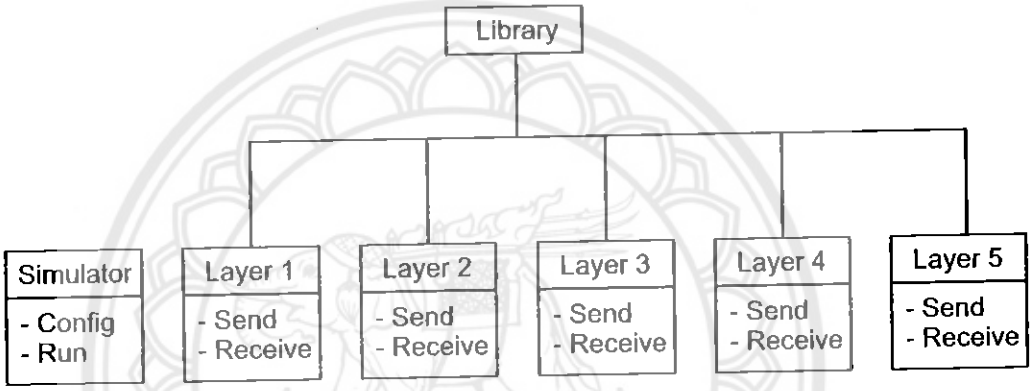
บทที่ 3

วิธีการดำเนินงาน

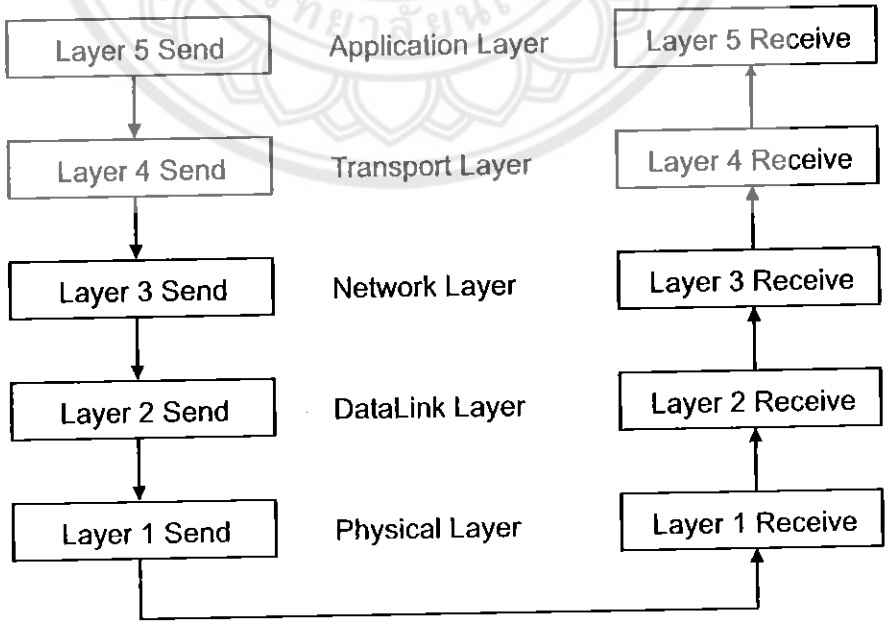
3.1 การออกแบบโครงสร้างของโปรแกรม

โปรแกรมนี้ประกอบไปด้วยส่วนหลักๆ คือ

- Simulator ซึ่งเป็นตัวตั้งค่าการใช้งาน และตัวควบคุมการทำงาน
- Layer 1 – Layer 5 ซึ่งในแต่ละเลเยอร์จะมีตัวส่งและตัวรับ เป็นตัวดำเนินงาน

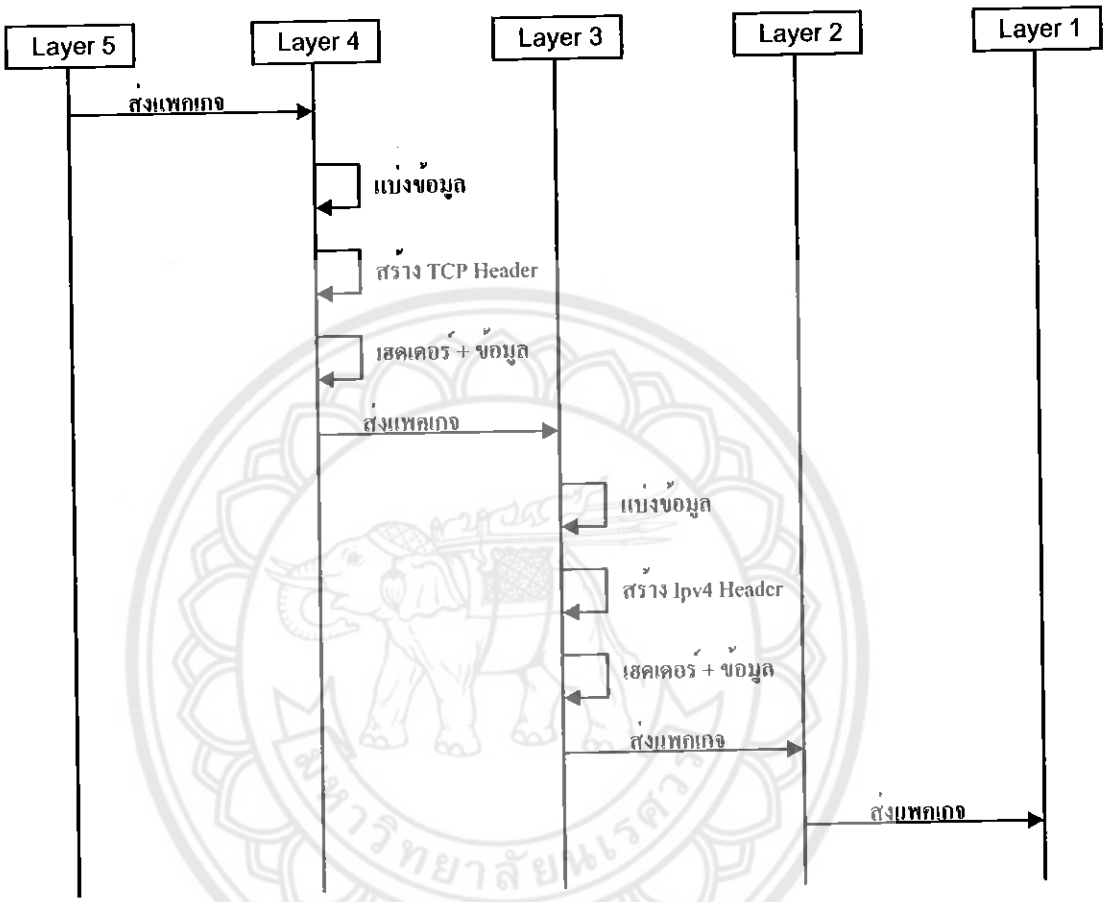


รูปที่ 3.1 โครงสร้างของโปรแกรม



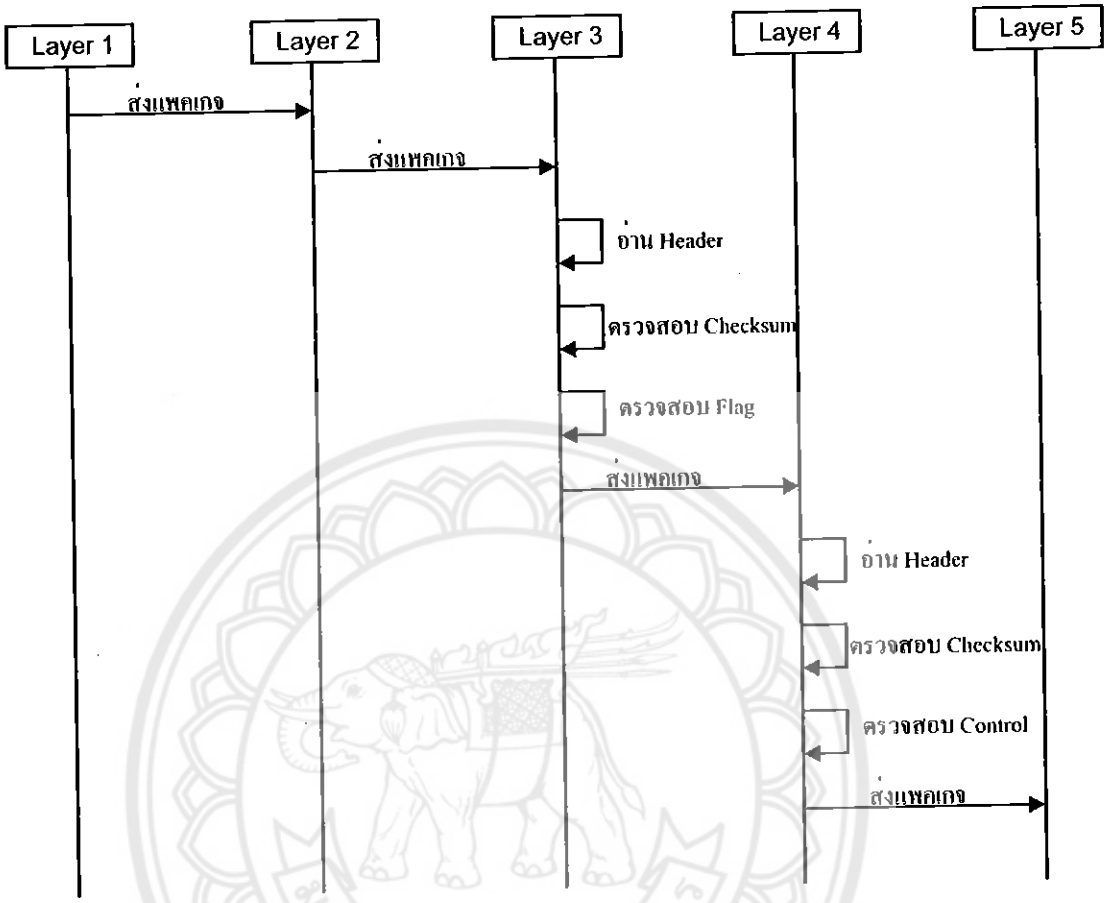
รูปที่ 3.2 การทำงานและการเชื่อมโยงของโปรแกรม

3.2 Sequence Diagram Sender



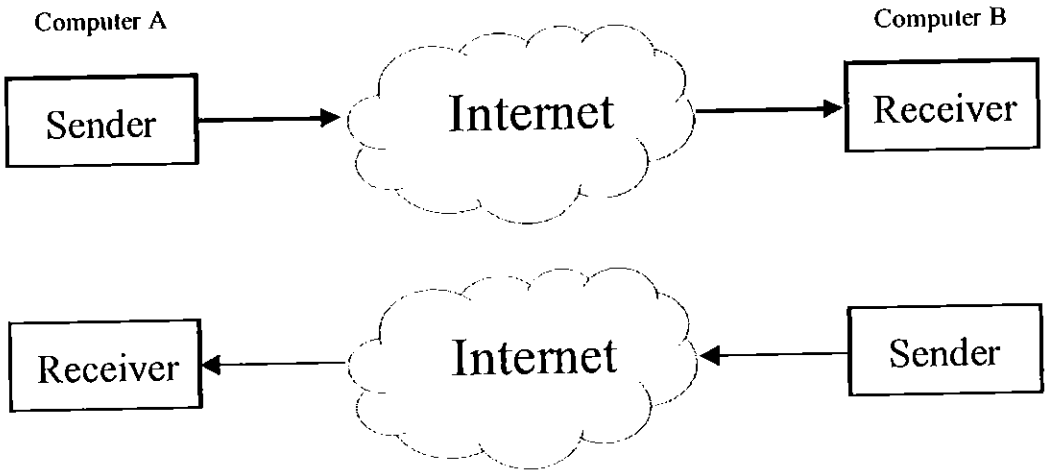
รูปที่ 3.3 Sequence Diagram Sender

3.3 Sequence Diagram Receiver



รูปที่ 3.4 Sequence Diagram Receiver

3.4 ความสัมพันธ์ระหว่าง Sender และ Receiver



รูปที่ 3.5 ความสัมพันธ์ระหว่าง Sender และ Receiver

3.5 การสร้างโครงร่างเพื่อใช้ในการจำลอง

3.5.1 สร้างคลาส OSIsimulator

คลาส OSIsimulator เป็นคลาสที่ใช้กำหนดค่าที่ต้องการในแต่ละเลเยอร์ เพื่อเตรียมที่จะทำการจำลอง และควบคุมการทำงานของโปรแกรม ซึ่งมีอยู่ 2 เมท็อด (Method) คือ

1) เมท็อด Config(...) ทำหน้าที่รับคลาสสมาชิกของแต่ละเลเยอร์เพื่อกำหนดค่า
2) เมท็อด Run() ทำหน้าที่ควบคุมการจำลอง โดยที่สามารถเลือกได้ว่าจะเป็น
   - ผู้ส่งข้อความ (Sender) หรือ
   - ผู้รับข้อความ (Receiver)

```
public class OSIsimulator {
    /** Creates a new instance of OSIsimulator */

    public OSIsimulator() {
        // TODO code application logic here
    }

    public int config(...) {
        // TODO code application logic here
        return 0;
    }

    public int run() {
        // TODO code application logic here
        return 0;
    }
```

รูปที่ 3.6 รูปแบบของคลาส OSIsimulator

```
public static void main(String[] args) {
    OSIsimulator sim = new OSIsimulator();
    I_Layer5 L5 = new L5_HTTP();
    I_Layer4 L4 = new L4_TCP();
    I_Layer3 L3 = new L3_IPv4();
    I_Layer2 L2 = new L2_Ethernet();
    I_Layer1 L1 = new L1_RS232();
    sim.config(L1, L2, L3, L4, L5);
    sim.run();
}
```

รูปที่ 3.7 ตัวอย่างการเรียกใช้คลาส OSIsimulator

3.5.2 สร้างอินเทอร์เฟซ (Interface) และคลาสที่ทำตามอินเทอร์เฟซ

อินเทอร์เฟส (Interface) เป็นตัวที่ใช้ในการจัดกลุ่มเลเยอร์ มีทั้งหมด 5 เลเยอร์ ได้แก่ I_Layer1, I_Layer2, I_Layer3, I_Layer4, I_Layer5

```
public interface I_Layer2 {
    public abstract int NextLayer(...);
    public abstract int Send(...);
    public abstract int Receive(...);
}
```

รูปที่ 3.8 รูปแบบของคลาสอินเทอร์เฟซ (Interface)

คลาสที่ทำตามอินเทอร์เฟส เป็นคลาสที่สร้างขึ้นมาเพื่อใช้ในการจำลองการทำงานของเน็ตเวิร์คโพรโตคอลสแตกในแต่ละเลเยอร์

```
public class L2_Ethernet implements I_Layer2{
    /**
     * Creates a new instance of L2_Ethernet
     */
    public L2_Ethernet() {
    }

    public int NextLayer(...) {
        //TODO code application logic here
        return 0;
    }

    public int Send(...) {
        //TODO code application logic here
        return 0;
    }

    public int Receive(...) {
        //TODO code application logic here
        return 0;
    }
}
```

รูปที่ 3.9 รูปแบบของคลาสสมาชิก

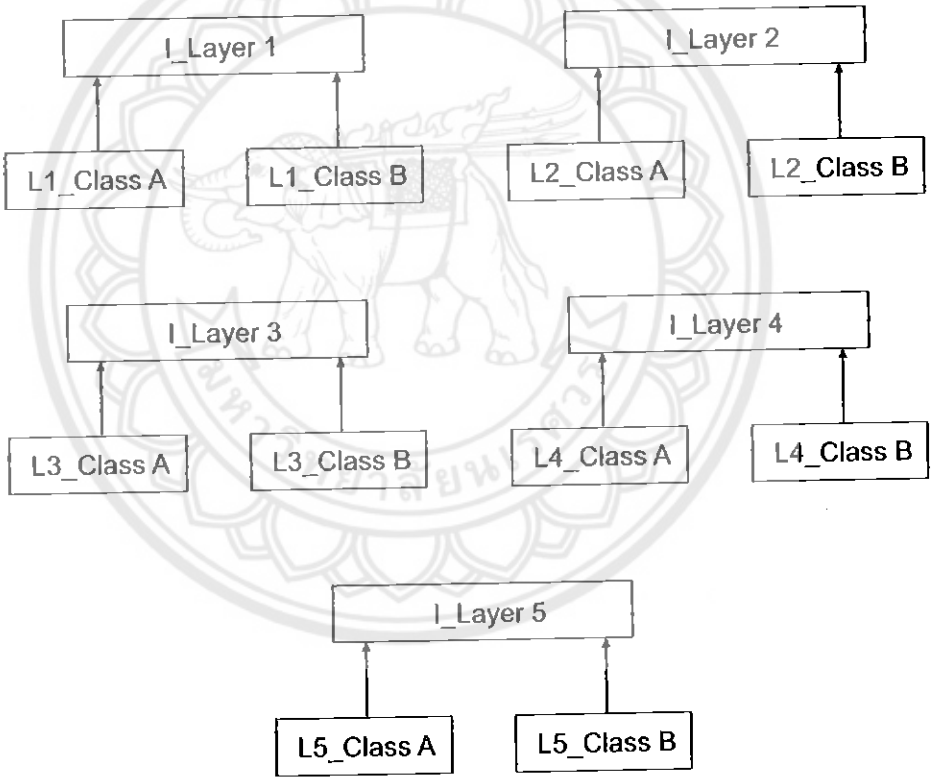
ซึ่งมีอยู่ 3 เมท็อด (Method) คือ

1) เมท็อด NextLayer(...) ทำหน้าที่รับค่าที่ได้คอนฟิกจากคลาส OSIsimulator มาแจ้งให้รู้ว่าในแต่ละเลเยอร์เลือกใช้สมาชิกตัวใด
2) เมท็อด Send(...) ทำหน้าที่เป็นผู้ส่งข้อความ
3) เมท็อด Receive(...)ทำหน้าที่เป็นผู้รับข้อความ

3.6 การสร้างตัวอย่างเพื่อใช้ในการทดสอบ

3.6.1 การสร้างคลาสเพื่อทดสอบโครงร่าง

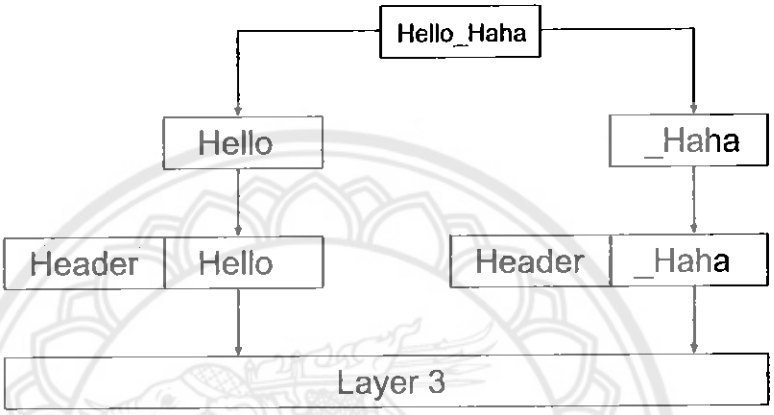
1. สร้างคลาสสมาชิก L1_classA, L1_classB ใน I_Layer1
2. สร้างคลาสสมาชิก L2_classA, L2_classB ใน I_Layer2
3. สร้างคลาสสมาชิก L3_classA, L3_classB ใน I_Layer3
4. สร้างคลาสสมาชิก L4_classA, L4_classB ใน I_Layer4
5. สร้างคลาสสมาชิก L5_classA, L5_classB ใน I_Layer5



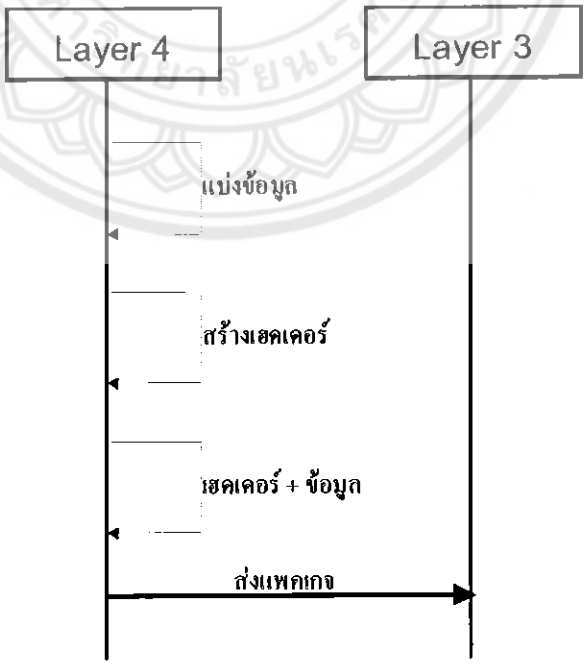
รูปที่ 3.10 ตัวอย่างของคลาสสมาชิก

3.6.2 การจำลองการส่งที่เลเยอร์ 4 (TCP/IP Protocol)

1. รับข้อมูลที่ต้องการส่ง มาทำการแบ่งข้อมูลตามขนาดที่กำหนด
2. ทำการสร้างเฮดเดอร์ของ TCP/IP Protocol
3. นำเฮดเดอร์ที่สร้างขึ้นมาต่อด้วยข้อมูลที่แบ่งแล้ว
4. ทำการส่งดาต้าแกรมไปที่เลเยอร์ถัดไป



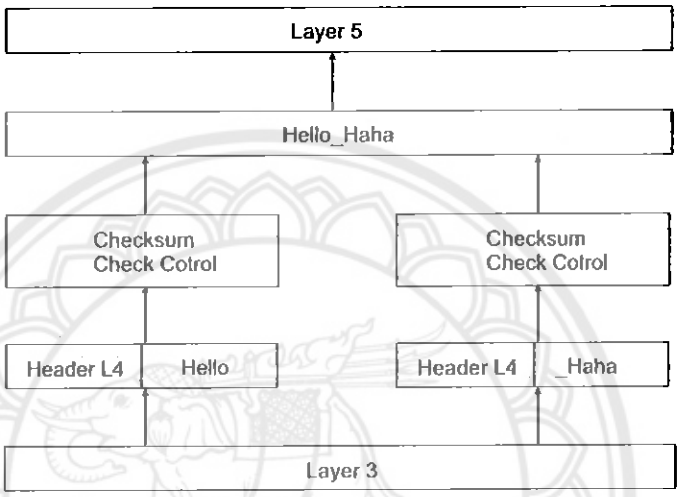
รูปที่ 3.11 รูปตัวอย่างการส่ง MSG จากเลเยอร์ 4 ไปเลเยอร์ 3



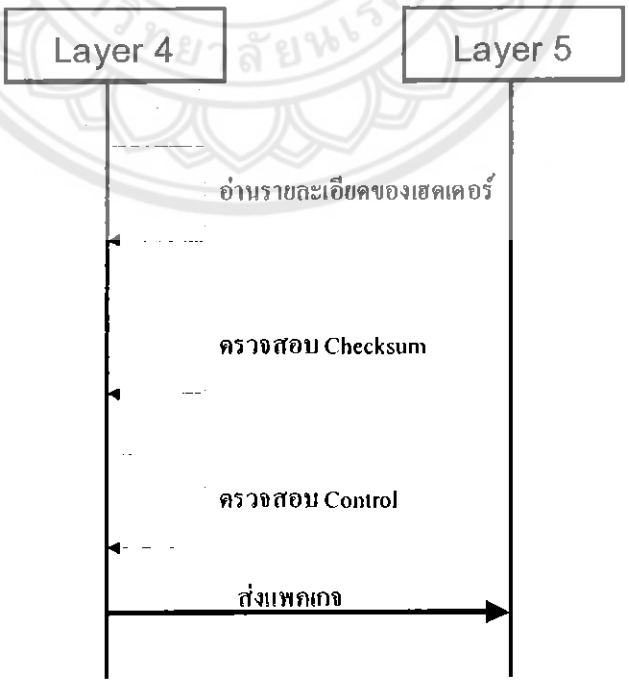
รูปที่ 3.12 Sequence Diagram การจำลองการส่งที่ Layer 4

3.6.3 การจำลองการรับที่เลเยอร์ 4 (TCP/IP Protocol)

1. นำข้อมูลที่รับมาทำการอ่านรายละเอียดของเฮดเดอร์
2. ตรวจสอบ Checksum ว่ารับข้อมูลมาถูกต้องหรือไม่
3. ตรวจสอบคอนโทรล (Control) ว่ามีข้อมูลครบหรือยัง ถ้ายังไม่ครบก็ทำการรอรับข้อมูลถัดไป
4. เมื่อรับข้อมูลครบทำการส่งไปที่เลเยอร์ถัดไป



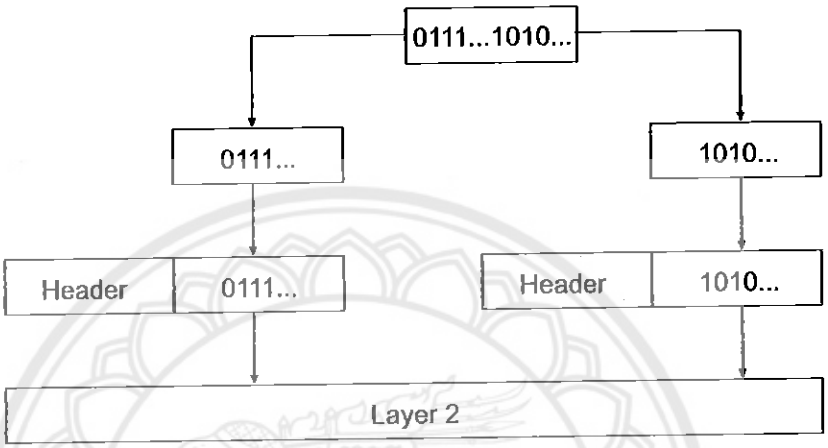
รูปที่ 3.13 รูปตัวอย่างการรับ MSG ที่เลเยอร์ 4



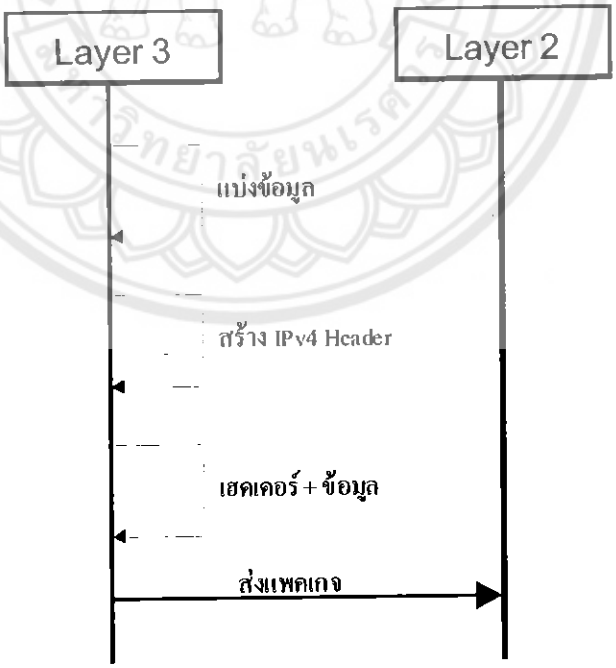
รูป 3.14 Sequence Diagram การจำลองการรับที่ Layer 4

3.6.4 การจำลองการส่งที่เลเยอร์ 3 (IPv4 Protocol)

1. รับข้อมูลที่ต้องการส่งมาทำการแบ่งข้อมูลตามขนาดที่กำหนด
2. ทำการสร้างเฮดเดอร์ของ IPv4 Protocol
3. นำเฮดเดอร์ที่สร้างขึ้นมาต่อด้วยข้อมูลที่แบ่งแล้ว
4. ทำการส่งแพคเกจไปที่เลเยอร์ถัดไป



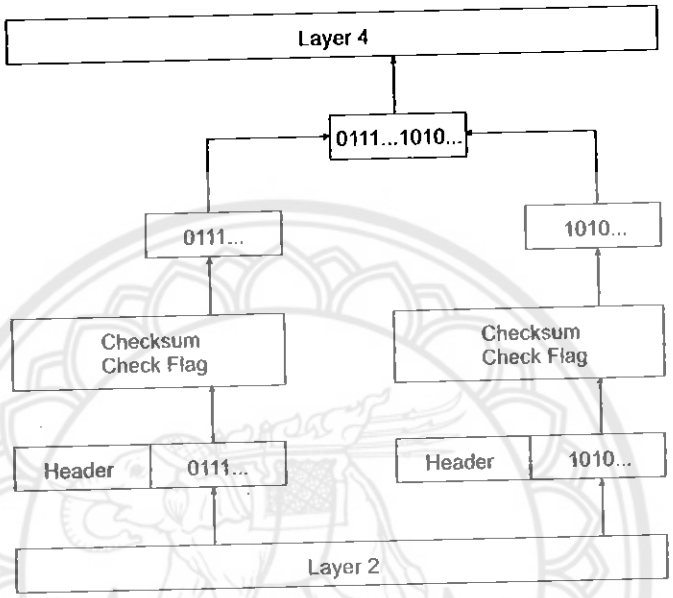
รูปที่ 3.15 รูปตัวอย่างการส่ง MSG จากเลเยอร์ 3 ไปเลเยอร์ 2



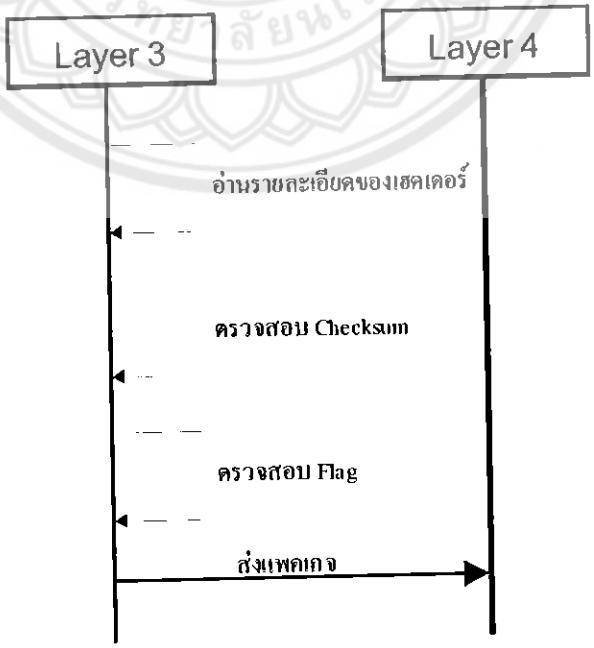
รูปที่ 3.16 Sequence Diagram การจำลองการส่งที่ Layer 3

3.6.5 การจำลองการรับที่เลเยอร์ 3 (IPv4 Protocol)

1. นำข้อมูลที่รับมาทำการอ่านรายละเอียดของเฮดเดอร์
2. ตรวจสอบ Checksum ว่ารับข้อมูลมาถูกต้องหรือไม่
3. ตรวจสอบแฟลก (Flag) ว่ามีข้อมูลอีกหรือไม่ ถ้ายังมีก็ทำการรอรับข้อมูลถัดไป
4. เมื่อรับข้อมูลครบทำการส่งไปที่เลเยอร์ถัดไป



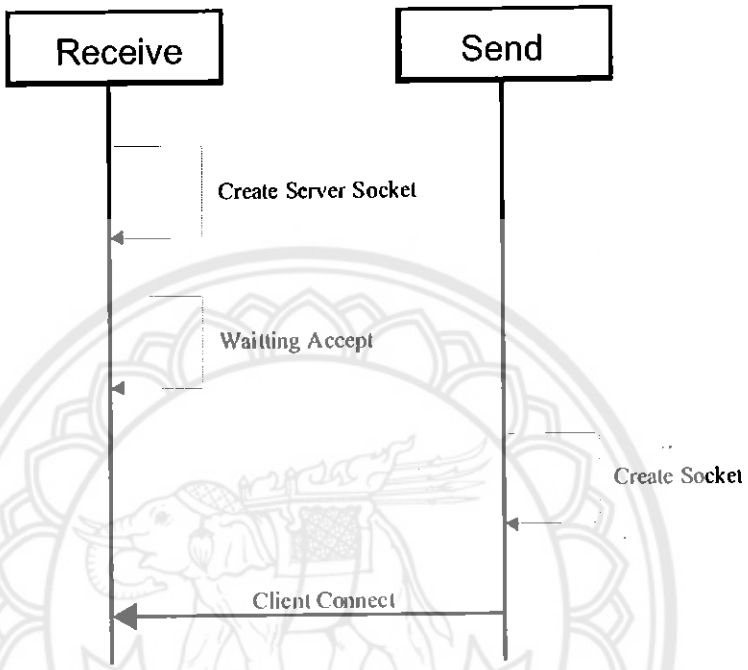
รูปที่ 3.17 รูปตัวอย่างการรับ MSG ที่เลเยอร์ 3



รูป 3.18 Sequence Diagram การจำลองการรับที่ Layer 3

3.6.6 การจำลองการส่งและการรับที่เลเยอร์ 1

ในการจำลองที่เลเยอร์นี้การส่งและการรับจะใช้ Socket เป็นตัวดำเนินการ เพื่อจำลองการส่งและการรับผ่านเน็ตเวิร์ค



รูป 3.19 การทำงานของ Socket ที่ Layer 1

บทที่ 4

ผลการทดลอง

ในบทนี้เราจะกล่าวถึงแผนการทดลอง ตามด้วยการทดสอบคลาส ทดสอบโครงร่าง และทดสอบการใช้งานโปรแกรมบนระบบปฏิบัติการต่างๆ

4.1 แผนการทดลอง

ตารางที่ 4.1 ตารางแผนการทดลอง

ลำดับ	รายการ
1	การทดสอบคลาส ● ทดสอบการส่งที่ Layer 4 คลาส L4_TCP ● ทดสอบการรับที่ Layer 4 คลาส L4_TCP ● ทดสอบการส่งที่ Layer 3 คลาส L3_IPv4 ● ทดสอบการรับที่ Layer 3 คลาส L3_IPv4
2	การทดสอบโครงร่าง ● ทดสอบการเรียกสมาชิก Class A ● ทดสอบการเรียกสมาชิก Class B ● ทดสอบการเรียกสมาชิก Class A และ Class B
3	การทดสอบการใช้งานโปรแกรมบนระบบปฏิบัติการ ● ทดสอบบนระบบปฏิบัติการ Windows XP ● ทดสอบบนระบบปฏิบัติการ Linux Ubuntu ● ทดสอบระหว่าง Windows XP และ Linux Ubuntu

4.2 การทดสอบคลาส

4.2.1 ทดสอบการส่งที่ Layer 4 คลาส L4_TCP

ทำการทดลองคลาสตัวอย่างการจำลอง TCP Protocol ที่ฝั่งผู้ส่งโดยใช้ข้อความว่า "Hello_Test" จะได้แพคเกจที่จำลอง 2 แพคเกจดังนี้

```
#### Header Detail ####
source port : 0001111110010000 (8080)
destination port : 0001111110010000 (8080)
sequence number : 00000000000000000000000000000001 (1)
ACK : 00000000000000000000000000000000
header length : 0101 (5)
reserved : 000000
control : 000010
windows size : 0000000000000101 (5)
checksum : 0011011010011011
urgent : 0000000000000000
option :

#### MSG Detail ####
Character MSG : Hello
```

รูปที่ 4.1 แพคเกจที่ 1

```
#### Header Detail ####
source port : 0001111110010000 (8080)
destination port : 0001111110010000 (8080)
sequence number : 00000000000000000000000000000010 (2)
ACK : 00000000000000000000000000000000
header length : 0101 (5)
reserved : 000000
control : 000011
windows size : 0000000000000101 (5)
checksum : 0010000110100011
urgent : 0000000000000000
option :

#### MSG Detail ####
Character MSG : _Test
```

รูปที่ 4.2 แพคเกจที่ 2

4.2.2 ทดสอบการรับที่ Layer 4 คลาส L4_TCP

ทำการทดลองคลาสตัวอย่างการจำลอง TCP Protocol ที่ฝั่งผู้รับ จะได้แพคเกจที่รับมาดังนี้


```
########################## L4 Receive ##########################
OS : Windows XP
source port : 0001111110010000 (8080)
destination port : 0001111110010000 (8080)
sequence number : 00000000000000000000000000000001 (1)
ACK : 00000000000000000000000000000000
header length : 0101 (5)
reserved : 000000
control : 000010
windows size : 0000000000000101 (5)
checksum : 0011011010011011
urgent : 0000000000000000
option :
L4 Receive message : Hello
Receive total message : Hello
L4 Checksum successful !!!
```

รูปที่ 4.3 รับแพคเกจที่ 1

```
########################## L4 Receive ##########################
OS : Windows XP
source port : 0001111110010000 (8080)
destination port : 0001111110010000 (8080)
sequence number : 00000000000000000000000000000010 (2)
ACK : 00000000000000000000000000000000
header length : 0101 (5)
reserved : 000000
control : 000011
windows size : 0000000000000101 (5)
checksum : 0010000110100011
urgent : 0000000000000000
option :
L4 Receive message : _Test
Receive total message : Hello_Test
L4 Checksum successful !!!
```

รูปที่ 4.4 รับแพคเกจที่ 2

4.2.3 ทดสอบการส่งที่ Layer 3 คลาส L3_IPv4

ทำการทดลองคลาสตัวอย่างการจำลอง IPv4 Protocol ที่ฝั่งผู้ส่งโดยใช้ข้อมูลจาก Layer 4 คลาส L4_TCP จะได้เฟรมที่จำลอง ดังนี้

```
########################## L3 Send ############################
Version : 0100 (4)
Header Length : 0101 (5)
DS : 00000000
TotalLength : 0000000100100000
Identification : 0000000000000001 (1)
Flag : 001
Fragmentation : 0000010000000
TTL : 00000000
Protocal : 00000110
Checksum : 1111010010010011
SourceIP : 00001010000000000000000010101100 (10.0.0.172)
DestinationIP : 00001010000000000000000011110000 (10.0.0.240)
Option :
Layer 3 MSG Send : 000111111001000000011111100100000000000000(
```

รูปที่ 4.5 รูปการส่งครั้งที่ 1 แพคเกจที่ 1

```
########################## L3 Send ############################
Version : 0100 (4)
Header Length : 0101 (5)
DS : 00000000
TotalLength : 0000000011101000
Identification : 0000000000000001 (1)
Flag : 010
Fragmentation : 0000001001000
TTL : 00000000
Protocal : 00000110
Checksum : 0000100110111111
SourceIP : 00001010000000000000000010101100 (10.0.0.172)
DestinationIP : 00001010000000000000000011110000 (10.0.0.240)
Option :
Layer 3 MSG Send : 000111111001000000011111100100000000000000
```

รูปที่ 4.6 รูปการส่งครั้งที่ 1 แพคเกจที่ 2

```
########################## L3 Send ##############################
Version : 0100 (4)
Header Length : 0101 (5)
DS : 00000000
TotalLength : 0000000100100000
Identification : 0000000000000010 (2)
Flag : 001
Fragmentation : 0000010000000
TTL : 00000000
Protocal : 00000110
Checksum : 1111010010010000
SourceIP : 00001010000000000000000010101100 (10.0.0.172)
DestinationIP : 00001010000000000000000011110000 (10.0.0.240)
Option :
Layer 3 MSG Send : 000111111001000000011111100100000000000000C
```

รูปที่ 4.7 รูปการส่งครั้งที่ 2 แพคเกจที่ 1

```
########################## L3 Send ##############################
Version : 0100 (4)
Header Length : 0101 (5)
DS : 00000000
TotalLength : 0000000011101000
Identification : 0000000000000010 (2)
Flag : 010
Fragmentation : 0000001001000
TTL : 00000000
Protocal : 00000110
Checksum : 0000100111000000
SourceIP : 00001010000000000000000010101100 (10.0.0.172)
DestinationIP : 00001010000000000000000011110000 (10.0.0.240)
Option :
Layer 3 MSG Send : 000111111001000000011111100100000000000000
```

รูปที่ 4.8 รูปการส่งครั้งที่ 2 แพคเกจที่ 2

4.2.4 ทดสอบการรับที่ Layer 3 คลาส L3_IPv4

ทำการทดลองคลาสตัวอย่างการจำลอง IPv4 Protocol ที่ฝั่งผู้รับ จะได้เฟรมที่รับมาดังนี้

```
########################## L3 Receive ##########################
Version : 0100 (4)
Header Length : 0101 (5)
DS : 00000000
TotalLength : 0000000100100000
Identification : 0000000000000001 (1)
Flag : 001
Fragmentation : 0000010000000
TTL : 00000000
Protocol : 00000110
Checksum : 1111010010010011
SourceIP : 00001010000000000000000011110000 (10.0.0.240)
DestinationIP : 00001010000000000000000010101100 (10.0.0.172)
Option :
Layer 3 MSG Receive : 00011111100100000001111111001000000000000(
Length of Layer 3 Receive : 128
L3 Checksum successful!!!
```

รูปที่ 4.9 รูปการรับครั้งที่ 1 แพคเกจที่ 1

```
########################## L3 Receive ########################
Version : 0100 (4)
Header Length : 0101 (5)
DS : 00000000
TotalLength : 0000000011101000
Identification : 0000000000000001 (1)
Flag : 010
Fragmentation : 0000001001000
TTL : 00000000
Protocol : 00000110
Checksum : 0000100110111111
SourceIP : 00001010000000000000000011110000 (10.0.0.240)
DestinationIP : 00001010000000000000000010101100 (10.0.0.172)
Option :
Layer 3 MSG Receive : 00110111010011011000000000000000000010010000
Length of Layer 3 Receive : 72
L3 Checksum successful!!!
```

รูปที่ 4.10 รูปการรับครั้งที่ 2 แพคเกจที่ 1

```
########################## L3 Receive ##########################
Version : 0100 (4)
Header Length : 0101 (5)
DS : 00000000
TotalLength : 0000000100100000
Identification : 0000000000000010 (2)
Flag : 001
Fragmentation : 0000010000000
TTL : 00000000
Protocol : 00000110
Checksum : 1111010010010000
SourceIP : 00001010000000000000000011110000 (10.0.0.240)
DestinationIP : 00001010000000000000000010101100 (10.0.0.172)
Option :
Layer 3 MSG Receive : 0001111111001000000011111100100000000000000
Length of Layer 3 Receive : 128
L3 Checksum successful!!!
```

รูปที่ 4.11 รูปการรับครั้งที่ 1 แพคเกจที่ 2

```
########################## L3 Receive ##########################
Version : 0100 (4)
Header Length : 0101 (5)
DS : 00000000
TotalLength : 0000000011101000
Identification : 0000000000000010 (2)
Flag : 010
Fragmentation : 0000001001000
TTL : 00000000
Protocol : 00000110
Checksum : 0000100111000000
SourceIP : 00001010000000000000000011110000 (10.0.0.240)
DestinationIP : 00001010000000000000000010101100 (10.0.0.172)
Option :
Layer 3 MSG Receive : 00100001101000110000000000000000000010111110
Length of Layer 3 Receive : 72
L3 Checksum successful!!!
```

รูปที่ 4.12 รูปการรับครั้งที่ 2 แพคเกจที่ 2

4.3 การทดสอบโครงร่าง

4.3.1 ทดสอบเรียกสมาชิกคลาส A

ทำการทดลองเรียกคลาส A ของทุกเลเยอร์ โดยวนลูปภายในเครื่องแล้วแสดง ชื่อคลาส, สถานะการทำงาน และ ข้อความ

```
public static void main(String[] args) {
    // TODO code application logic here
    OSIsimulator sim = new OSIsimulator();

    I_Layer5 L5 = new L5_classA();
    I_Layer4 L4 = new L4_classA();
    I_Layer3 L3 = new L3_classA();
    I_Layer2 L2 = new L2_classA();
    I_Layer1 L1 = new L1_classA();

    sim.config(L1, L2, L3, L4, L5);
    sim.run();
}
```

รูปที่ 4.13 ตัวอย่างการเรียกใช้สมาชิกคลาส A

จะได้ผลการทดลองดังนี้

run:

L5_classA Send : Hello Test.

L4_classA Send : Hello Test.

L3_classA Send : Hello Test.

L2_classA Send : Hello Test.

L1_classA Send : Hello Test.

L1_classA Receive : Hello Test.

L2_classA Receive : Hello Test.

L3_classA Receive : Hello Test.

L4_classA Receive : Hello Test.

L5_classA Receive : Hello Test.

4.3.2 ทดสอบเรียกสมาชิกคลาส B

ทำการทดลองเรียกคลาส B ของทุกเลเยอร์ โดยวนลูปภายในเครื่องแล้วแสดง ชื่อคลาส, สถานะการทำงาน และ ข้อความ

```
public static void main(String[] args) {
    // TODO code application logic here
    OSIsimulator sim = new OSIsimulator();

    I_Layer5 L5 = new L5_classB();
    I_Layer4 L4 = new L4_classB();
    I_Layer3 L3 = new L3_classB();
    I_Layer2 L2 = new L2_classB();
    I_Layer1 L1 = new L1_classB();

    sim.config(L1, L2, L3, L4, L5);
    sim.run();
}
```

รูปที่ 4.14 ตัวอย่างการเรียกใช้สมาชิกคลาส B

จะได้ผลการทดลองดังนี้

```
run:
L5_classB Send : Hello Test.
L4_classB Send : Hello Test.
L3_classB Send : Hello Test.
L2_classB Send : Hello Test.
L1_classB Send : Hello Test.
L1_classB Receive : Hello Test.
L2_classB Receive : Hello Test.
L3_classB Receive : Hello Test.
L4_classB Receive : Hello Test.
L5_classB Receive : Hello Test.
```

4.3.3 ทดสอบเรียกสมาชิกคลาส A และคลาส B

ทำการทดลองเรียกคลาส A และคลาส B สลับกัน โดยวนลูปภายในเครื่องแล้วแสดง ชื่อคลาส, สถานะการทำงาน และ ข้อความ

```
public static void main(String[] args) {
    // TODO code application logic here
    OSIsimulator sim = new OSIsimulator();

    I_Layer5 L5 = new L5_classA();
    I_Layer4 L4 = new L4_classB();
    I_Layer3 L3 = new L3_classA();
    I_Layer2 L2 = new L2_classB();
    I_Layer1 L1 = new L1_classA();

    sim.config(L1, L2, L3, L4, L5);
    sim.run();
}
```

รูปที่ 4.15 ตัวอย่างการเรียกใช้สมาชิกคลาส A และ B สลับกัน

จะได้ผลการทดลองดังนี้

run:

L5_classA Send : Hello Test.

L4_classB Send : Hello Test.

L3_classA Send : Hello Test.

L2_classB Send : Hello Test.

L1_classA Send : Hello Test.

L1_classA Receive : Hello Test.

L2_classB Receive : Hello Test.

L3_classA Receive : Hello Test.

L4_classB Receive : Hello Test.

L5_classA Receive : Hello Test.

4.4 การทดสอบการใช้งานโปรแกรมบนระบบปฏิบัติการ

ตารางที่ 4.2 ตารางการทดสอบการใช้งานโปรแกรมบนระบบปฏิบัติการ

ลำดับ	รายการ	ผลการทดสอบ
1	ทดสอบบนระบบปฏิบัติการ Windows XP ทั้ง 2 เครื่อง	ใช้งานได้
2	ทดสอบบนระบบปฏิบัติการ Linux Ubuntu ทั้ง 2 เครื่อง	ใช้งานได้
3	ทดสอบระหว่างระบบปฏิบัติการ Windows XP และ Linux Ubuntu	ใช้งานได้

4.5 สรุปผลการทดลอง

ตารางที่ 4.3 ตารางสรุปผลการทดลอง

ลำดับ	รายการ	ผลการทดลอง
1	การทดสอบคลาส	
	• ทดสอบการส่งที่ Layer 4 คลาส L4_TCP	ผ่าน
	• ทดสอบการรับที่ Layer 4 คลาส L4_TCP	ผ่าน
	• ทดสอบการส่งที่ Layer 3 คลาส L3_IPv4	ผ่าน
	• ทดสอบการรับที่ Layer 3 คลาส L3_IPv4	ผ่าน
2	การทดสอบโครงร่าง	
	• ทดสอบการเรียกสมาชิก Class A	ผ่าน
	• ทดสอบการเรียกสมาชิก Class B	ผ่าน
	• ทดสอบการเรียกสมาชิก Class A และ Class B	ผ่าน
3	**การทดสอบการใช้งานโปรแกรมบนระบบปฏิบัติการ**	
	• ทดสอบบนระบบปฏิบัติการ Windows XP	ผ่าน
	• ทดสอบบนระบบปฏิบัติการ Linux Ubuntu	ผ่าน
	• ทดสอบระหว่าง Windows XP และ Linux Ubuntu	ผ่าน

บทที่ 5
สรุปผล

โครงงานนี้ได้ศึกษาและพัฒนาไรบลารี สำหรับจำลองการทำงานของเน็ตเวิร์คโพรโตคอลสแตก (Network Protocol Stack) แบบ ทีซีพี/ไอพี (TCP/IP Model) โดยโครงงานนี้ใช้ Interface ในภาษาจาวา เพื่อให้ผู้ใช้สามารถสร้าง Class ที่จำลองการทำงานของเน็ตเวิร์คได้ โดยอิมพลีเมนต์ (Implements) อิทเทอร์เฟซ (interface) ของเลเยอร์ (Layer) ที่ต้องการจะจำลอง จึงทำให้ผู้ใช้เลือกจำลองเลเยอร์ไหนก็ได้

ดังนั้นกลุ่มผู้พัฒนาได้ใช้ภาษาจาวา (Java Language) ในการพัฒนา โดยใช้โปรแกรม NetBeans 5.5.1 เป็นเครื่องมือในการพัฒนาโปรแกรม

5.1 ผลการทดลอง

ในการศึกษานี้เราได้จำลองการทำงานของ 2 เลเยอร์ คือ เลเยอร์ 3 และ เลเยอร์ 4 (Layer 3 and Layer 4) ฝั่งส่ง (Sender) โปรแกรมสามารถรับแมจเสจและทำการแบ่งแมจเสจตามขนาดที่กำหนดแล้วสร้างเฮดเดอร์ เพื่อใช้ในการส่งไปยังเลเยอร์ถัดไป ในส่วนของเลเยอร์ 3 และ เลเยอร์ 4 ฝั่งรับ (Receiver) ก็สามารถรับแพคเกจ (Packet) แล้วทำการอ่านรายละเอียดของเฮดเดอร์และอ่าน message ได้

ในการทดสอบโครงร่าง โปรแกรมสามารถเรียกใช้คลาสสำหรับการจำลองการทำงานของเน็ตเวิร์คที่ผู้ใช้นำมาเพิ่มเติมในไลบรารี โดยการนำคลาสที่ต้องการจำลองมาอิมพลีเมนต์ในเลเยอร์ต่างๆที่ต้องการได้

การทดลองรันโปรแกรมโดยการนำโปรแกรมมาทดสอบทำการส่งและรับแพคเกจบนระบบปฏิบัติการ Windows XP และ ระบบปฏิบัติการ Linux Ubuntu ซึ่งแบ่งได้เป็น 3 กรณี ดังนี้

1. สามารถทำงานบนระบบปฏิบัติการ Windows XP ทั้ง 2 เครื่อง ได้
2. สามารถทำงานบนระบบปฏิบัติการ Linux Ubuntu ทั้ง 2 เครื่อง ได้
3. สามารถทำงานระหว่างระบบปฏิบัติการ Windows XP และ Linux Ubuntu ได้

5.2 ปัญหาและอุปสรรค

1. ขาดความชำนาญในการเขียนโปรแกรมด้วยภาษาจาวา (Java Language) จึงทำให้เกิดความล่าช้าในการพัฒนาโปรแกรม
2. การนำ Socket มาใช้ในการจำลองอาจจะเกิดปัญหา Connection Reset เนื่องจากสภาพแวดล้อมของเน็ตเวิร์ค, ประสิทธิภาพของเครื่องคอมพิวเตอร์ที่ใช้ หรือระบบเครือข่ายของระบบปฏิบัติการนั้นๆ
3. การเรียก IP ของตัวเองขึ้นมา ต้องเรียกใช้ชุดคำสั่งที่สนับสนุนระบบปฏิบัติการนั้นๆด้วย เนื่องจากบางชุดคำสั่งไม่สามารถทำงานได้หลายระบบปฏิบัติการ
4. ขาดความชำนาญในการเขียนโปรแกรมแบบ Graphic User Interface (GUI)

5.3 ข้อเสนอแนะ

1. ผู้ใช้ต้องมีความรู้พื้นฐานในการเขียนโปรแกรมด้วยภาษาจาวา (Java Language) และความรู้ในการ Implements Class ในภาษาจาวาพอสมควร
2. ในการใช้งาน Socket ควรจะทำความเข้าใจในการทำงานของ Socket ในภาษาจาวา

5.4 สรุป

จากผลการทดลองไลบรารีสำหรับจำลองเน็ตเวิร์กโพรโตคอลสแตก (Network Protocol Stack) แบบ TCP/IP Model โดยใช้หลักการอิมพลีเมนต์ (Implements) ในภาษาจาวา (Java Language) ทำให้ผู้ใช้สามารถนำซอสโค้ด (Source Code) ที่ผู้ใช้จำลองขึ้น มาใช้ในโครงร่างได้ และโปรแกรมสามารถจำลองการทำงานของเน็ตเวิร์กได้ทั้งหมด 5 เลเยอร์ โดยโปรแกรมสามารถทำงานได้บนระบบปฏิบัติการไมโครซอฟวินโดว์ เอ็กพี (Microsoft Windows XP) และระบบปฏิบัติการลีนุกซ์ อุบุนตู (Linux Ubuntu) ได้ จากการทดลองแสดงให้เห็นว่าโปรแกรมสามารถทำงานได้ตามจุดประสงค์

ภาคผนวก ก

คู่มือการใช้งาน

การใช้งานโครงร่างสำหรับจำลอง Network Protocol Stack

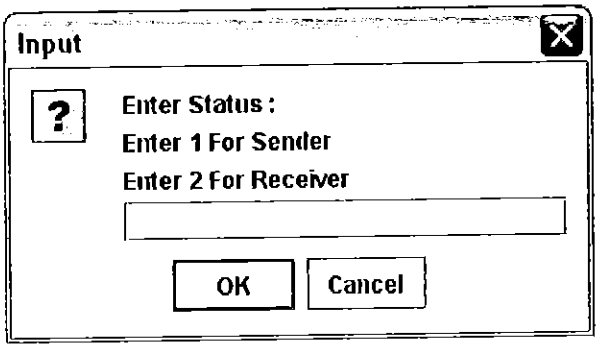
```
public static void main(String[] args) {

    OSIsimulator sim = new OSIsimulator();

    I_Layer5 L5 = new L5_HTTP();
    I_Layer4 L4 = new L4_TCP();
    I_Layer3 L3 = new L3_IPv4();
    I_Layer2 L2 = new L2_Ethernet();
    I_Layer1 L1 = new L1_RS232();

    sim.config(L1, L2, L3, L4, L5);
    sim.run();
}
```

1

รูปที่ ก.1 การใช้งานโครงร่าง

หมายเลข 1 เป็นส่วนที่ระบุว่า ต้องการเรียกคลาสใดขึ้นมาจำลองการทำงานในแต่ละเลเยอร์ โดยสามารถเปลี่ยนเป็นคลาสที่ผู้ใช้จำลองขึ้นมาแทนที่ได้

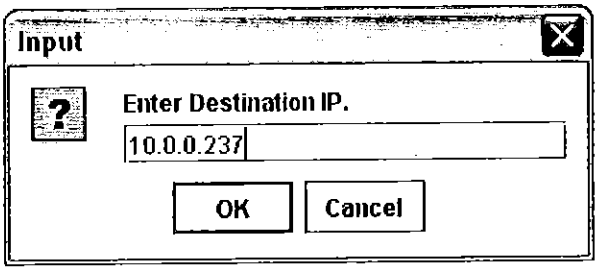
หลังจากเลือกคลาสที่ต้องการจำลองแล้ว เมื่อรันโปรแกรมจะปรากฏหน้าต่างขึ้นมาให้เลือกว่า จะเป็นผู้ส่ง พิมพ์ 1 หรือผู้รับ พิมพ์ 2 ดังรูป



รูปที่ ก.2 หน้าต่างสำหรับเลือกการทำงาน

กรณีเป็นผู้ส่ง

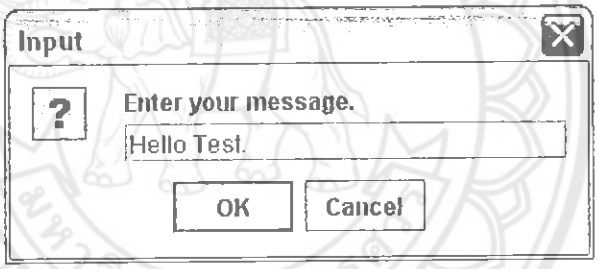
กรณีเลือกเป็นผู้ส่ง (พิมพ์ 1) จะมีหน้าต่างขึ้นมาให้ใส่ IP Address ปลายทางที่ผู้ใช้ต้องการจะติดต่อ



รูปที่ ก.3 หน้าต่างรับ IP Address ปลายทาง

เมื่อกดปุ่ม 

จะปรากฏหน้าต่างเพื่อกรอกข้อความดังรูป



รูปที่ ก.4 หน้าต่างกรอกข้อความ

หลังจากกดปุ่ม 

โปรแกรมก็จะส่งข้อมูลเพื่อเริ่มจำลองการทำงาน

กรณีเป็นผู้รับ

กรณีเลือกเป็นผู้รับ (พิมพ์ 2) โปรแกรมก็จะรอรับข้อมูลจากฝั่งผู้ส่ง ดังรูป

```
run:
########################## Ll Receive ##########################
```

รูปที่ ก.5 รูปการเริ่มทำงานในฝั่งผู้รับ

รูปแบบการสร้าง Class ที่ใช้สำหรับจำลองใน Layer 1

```
                1
public class Class_Name implements I_Layer1{
    I_Layer1 L1;
    I_Layer2 L2;

    public int NextLayer(I_Layer1 param1, I_Layer2 param2) {
        L1 = param1;
        L2 = param2;
        return 0;
    }

    public int Send(String msg, String ip) {
    2   //TODO code application logic here
        this.Receive(msg, ip);
        return 0;
    }

    public int Receive(String msg, String ip) {
    3   //TODO code application logic here
        L2.Receive(msg, ip);
        return 0;
    }
}
```

รูปที่ ก.6 รูปการสร้าง Class ที่ใช้สำหรับจำลองใน Layer 1

หมายเลข 1 Class_Name คือ ชื่อคลาสที่ผู้ใช้สร้างขึ้นมาเพื่อจำลองการทำงานที่ Layer 1

หมายเลข 2 เป็นส่วนที่ใช้จำลองการทำงานในฝั่งส่งของ Layer 1

หมายเลข 3 เป็นส่วนที่ใช้จำลองการทำงานในฝั่งรับของ Layer 1

รูปแบบการสร้าง Class ที่ใช้สำหรับจำลองใน Layer 2

```
1
public class Class_Name implements I_Layer2{
    I_Layer1 L1;
    I_Layer3 L3;

    public int NextLayer(I_Layer1 param1, I_Layer3 param2) {
        L1 = param1;
        L3 = param2;
        return 0;
    }

    public int Send(String msg, String ip) {
2       //TODO code application logic here
        L1.Send(msg, ip);
        return 0;
    }

    public int Receive(String msg, String ip) {
3       //TODO code application logic here
        L3.Receive(msg, ip);
        return 0;
    }
}
```

รูปที่ ก.7 รูปการสร้าง Class ที่ใช้สำหรับจำลองใน Layer 2

หมายเลข 1 Class_Name คือ ชื่อคลาสที่ผู้ใช้สร้างขึ้นมาเพื่อจำลองการทำงานที่ Layer 2

หมายเลข 2 เป็นส่วนที่ใช้จำลองการทำงานในฝั่งส่งของ Layer 2

หมายเลข 3 เป็นส่วนที่ใช้จำลองการทำงานในฝั่งรับของ Layer 2

รูปแบบการสร้าง Class ที่ใช้สำหรับจำลองใน Layer 3

```
1
public class Class_Name implements I_Layer3{
    I_Layer2 L2;
    I_Layer4 L4;
    I_Layer1 L1;

    public int NextLayer(I_Layer2 param1, I_Layer4 param2, I_Layer1 param3) {
        L2 = param1;
        L4 = param2;
        L1 = param3;
        return 0;
    }

    public int Send(String msg, String ip) {
2       //TODO code application logic here
        L2.Send(msg, ip);
        return 0;
    }

    public int Receive(String msg, String ip) {
3       //TODO code application logic here
        L4.Receive(msg, ip);
        return 0;
    }
}
```

รูปที่ ก.8 รูปการสร้าง Class ที่ใช้สำหรับจำลองใน Layer 3

หมายเลข 1 Class_Name **คือ** ชื่อคลาสที่ผู้ใช้สร้างขึ้นมาเพื่อจำลองการทำงานที่ Layer 3

หมายเลข 2 **เป็นส่วนที่ใช้จำลองการทำงานในฝั่งส่งของ** Layer 3

หมายเลข 3 **เป็นส่วนที่ใช้จำลองการทำงานในฝั่งรับของ** Layer 3

รูปแบบการสร้าง Class ที่ใช้สำหรับจำลองใน Layer 4

```
1
public class Class_Name implements I_Layer4{
    I_Layer3 L3;
    I_Layer5 L5;
    I_Layer1 L1;

    public int NextLayer(I_Layer3 param1, I_Layer5 param2, I_Layer1 param3) {
        L3 = param1;
        L5 = param2;
        L1 = param3;
        return 0;
    }

    public int Send(String msg, String ip) {
      2 //TODO code application logic here
        L3.Send(msg, ip);
        return 0;
    }

    public int Receive(String msg, String ip){
      3 //TODO code application logic here
        L5.Receive(msg, ip);
        return 0;
    }
}
```

รูปที่ ก.9 รูปการสร้าง Class ที่ใช้สำหรับจำลองใน Layer 4

หมายเลข 1 Class_Name คือ ชื่อคลาสที่ผู้ใช้สร้างขึ้นมาเพื่อจำลองการทำงานที่ Layer 4

หมายเลข 2 เป็นส่วนที่ใช้จำลองการทำงานในฝั่งส่งของ Layer 4

หมายเลข 3 เป็นส่วนที่ใช้จำลองการทำงานในฝั่งรับของ Layer 4

รูปแบบการสร้าง Class ที่ใช้สำหรับจำลองใน Layer 5

```
1
public class Class_Name implements I_Layer5{
    I_Layer4 L4;

    public int NextLayer(I_Layer4 param) {
        L4 = param;
        return 0;
    }

    public int Send(String msg, String ip) {
2       //TODO code application logic here
        L4.Send(msg, ip);
        return 0;
    }

    public int Receive(String msg, String ip) {
3       //TODO code application logic here
        return 0;
    }
}
```

รูปที่ ก.10 รูปการสร้าง Class ที่ใช้สำหรับจำลองใน Layer 5

หมายเลข 1 Class_Name คือ ชื่อคลาสที่ผู้ใช้สร้างขึ้นมาเพื่อจำลองการทำงานที่ Layer 5

หมายเลข 2 เป็นส่วนที่ใช้จำลองการทำงานในฝั่งส่งของ Layer 5

หมายเลข 3 เป็นส่วนที่ใช้จำลองการทำงานในฝั่งรับของ Layer 5

เอกสารอ้างอิง

[1] Behrouz A. Forouzan. **TCP/IP Protocol Suit.** Third Edition. New York : McGraw-Hill. Inc. 2006.

[2] รุ่งโรจน์ โพนคำ. **กะเทาะเปลือกจาวา.** กรุงเทพมหานคร. บริษัท จิรวัฒน์ เอ็กซ์เพรส จำกัด. พ.ศ. 2543

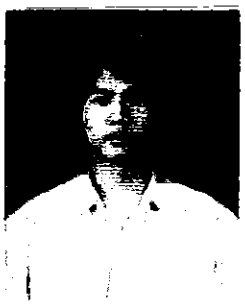
[3] วรเศรษฐ สุวรรณิก และ ทศพล ธนะทิพานนท์. **เขียนโปรแกรม Java เบื้องต้น.** กรุงเทพมหานคร. บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน). พ.ศ. 2549

[4] ดร.วีรศักดิ์ ซึงถาวร. **Java Programming Volume I (JavaSE 5.0).** กรุงเทพมหานคร. บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน). พ.ศ. 2549

[5] วรเศรษฐ สุวรรณิก. **Java GUI using NetBeans.** กรุงเทพมหานคร. โรงพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย. พ.ศ. 2551

[6] C.Thomas Wu. **An Introduction to Object-Oriented Programming with Java.** Fourth Edition. New York : McGraw-Hill. Inc. 2006.

ประวัติผู้เขียนโครงงาน



ชื่อ กิตติศักดิ์ ตุนาโป่ง

ภูมิลำเนา 266/3 หมู่ 12 ตำบลหัวรอ อำเภอเมือง
จังหวัดพิษณุโลก 65000

ประวัติการศึกษา

- จบ ปวช. จากวิทยาลัยเทคนิคพิษณุโลก
- ปัจจุบันกำลังศึกษาอยู่ระดับปริญญาตรีชั้นปีที่ 4
  สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

E-mail: tunapong@hotmail.com

ชื่อ นายณัฐเสฏฐ์ วัชรานุรักษ์

ภูมิลำเนา 35 บ้านฟ่อน หมู่ 2 ตำบลชมพู อำเภอเมือง
จังหวัดลำปาง 52100

ประวัติการศึกษา

- จบมัธยมศึกษาจากโรงเรียนบุญวาทย์วิทยาลัย จังหวัดลำปาง
- ปัจจุบันกำลังศึกษาอยู่ระดับปริญญาตรีชั้นปีที่ 4
  สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

E-mail: nujunior@gmail.com