ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์)

ปริญญา

วิทยาการคอมพิวเตอร์	วิทยาการคอมพิวเตอร์
สาขา	**ภาควิชา**

เรื่อง การเปรียบเทียบประสิทธิภาพการทำงานโพรโทคอล TCP และ SCTP บนระบบการส่งข้อความด่วน (Instant Messaging)

Efficiency Comparison of TCP Protocol and SCTP Protocol on Instant Messaging System

นามผู้วิจัย นายวโรดม ลิมปะพันธุ์

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(ผู้ช่วยศาสตราจารย์สุขุมาล กิติสิน, Ph.D.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม

(ผู้ช่วยศาสตราจารย์ชวลิต ศรีสถาพรพัฒน์, Ph.D.)

หัวหน้าภาควิชา

(ผู้ช่วยศาสตราจารย์ศิริกร จันทร์นวล)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

วันที่ **เดือน** **พ.ศ.**

วิทยานิพนธ์

เรื่อง

การเปรียบเทียบประสิทธิภาพการทำงานโพรโทคอล TCP และ SCTP บนระบบการส่งข้อความด่วน (Instant Messaging)

Efficiency Comparison of TCP Protocol and SCTP Protocol on Instant Messaging System

โดย

นายวโรดม ลิมปะพันธุ์

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์)
พ.ศ. 2555

วโรดม ลิมปะพันธุ์ 2555: การเปรียบเทียบประสิทธิภาพการทำงานโพรโทคอล TCP และ SCTP บนระบบการส่งข้อความด่วน (Instant Messaging) ปริญญาวิทยาศาสตร์มหาบัณฑิต (วิทยาการคอมพิวเตอร์) สาขาวิทยาการคอมพิวเตอร์ ภาควิชาวิทยาการคอมพิวเตอร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: ผู้ช่วยศาสตราจารย์สุขุมาล กิติสิน, Ph.D. 89 หน้า

วัตถุประสงค์ของงานวิจัยนี้คือการปรับใช้โพรโทคอล SCTP ในการใช้งานร่วมกับระบบการส่งข้อความด่วน (Instant Messaging) และทำการทดสอบประสิทธิภาพเปรียบเทียบกับโพรโทคอล TCP ในสภาพการทำงานและสภาพเครือข่ายที่แตกต่างกัน เพื่อเป็นแนวทางในการปรับใช้โพรโทคอล SCTP กับระบบการส่งข้อความด่วนและระบบอื่นๆได้อย่างมีประสิทธิภาพ

ผลการทดสอบประสิทธิภาพในการวิจัยนี้แสดงให้เห็นว่าโพรโทคอล SCTP สามารถปรับปรุงประสิทธิภาพการทำงานของระบบการส่งข้อความด่วนได้ และเมื่อวิเคราะห์ข้อมูลที่ได้จากการทดสอบจะเห็นได้ว่าการใช้งานโพรโทคอล SCTP มีประสิทธิภาพที่ดีกับข้อมูลที่มีขนาดใหญ่กว่า 1 Kbytes ในระบบที่ไม่มีปัจจัยภายนอกรบกวน และเมื่อการจราจรบนเครือข่ายมีความหนาแน่นหรือมีการสูญหายของข้อมูลเกิดขึ้นการใช้คุณลักษณะของโพรโทคอล SCTP จะเพิ่มประสิทธิภาพการทำงานเมื่อเปรียบเทียบกับโพรโทคอล TCP แต่ด้วยข้อจำกัดจากกระบวนการทำงานของโพรโทคอล SCTP เองในการทำงานร่วมกับกลุ่มข้อมูลซึ่งมีขนาดเล็กกว่า 1 Kbyte ในงานวิจัยพบว่าการใช้งานโพรโทคอล TCP ยังคงสามารถใช้งานได้อย่างมีประสิทธิภาพ ดังนั้นสามารถสรุปได้ว่าโพรโทคอล SCTP สามารถทำงานได้มีประสิทธิภาพดีในสภาวะ ที่สามารถใช้งานคุณสมบัติที่ดีกว่าของโพรโทคอล SCTP เองได้แก่ การทำงานกับข้อมูลขนาดใหญ่, ระบบที่มีการสูญหายของข้อมูล และประเภทของข้อมูลแบบบีบอัด

_____ _____ ___ / ___ / ___

ลายมือชื่อนิสิต ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Varodom Limpabandhu 2012: Efficiency Comparison of TCP Protocol and SCTP Protocol on Instant Messaging System. Master of Science (Computer Science), Major Field: Computer Science, Department of Computer Science.
Thesis Advisor: Assistant Professor Sukumal Kitisin, Ph.D. 89 pages.

The objective of this research to implement an instant messaging program with SCTP protocol to compare the efficiency of TCP protocol and SCTP protocol in different operating conditions and environments. As a prototype to demonstrate deploying SCTP protocol for instant messaging and other systems effectively.

Performance of test results show that SCTP protocol can improve the performance of instant messaging program as well. The results show that SCTP can provides better performance, but there are some restrictions of SCTP protocol when working with very small data (1 Kbytes) that protocol TCP can provide better performance. But all totally it can work effectively in condition that enable main feature of SCTP protocol including large data, packet loss environment and compress data.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

ในการทำงานวิจัยวิทยานิพนธ์ฉบับนี้ได้รับข้อแนะนำและคำปรึกษา เกี่ยวกับทฤษฎี ความรู้และการออกแบบ แนะแนวทางแก้ไขปัญหาและตรวจแก้ข้อบกพร่องต่างๆ ในงานวิจัยตลอดจาก ผู้ช่วยศาสตราจารย์ สุขุมาล กิติสิน ประธานกรรมการที่ปรึกษา และผู้ช่วยศาสตราจารย์ ชวลิต ศรีสถาพรพัฒน์ กรรมการที่ปรึกษาร่วม ที่ให้ความช่วยเหลืออย่างดียิ่งในทุกด้าน รวมถึงขั้นตอนในการดำเนินงาน ให้คำแนะนำ และข้อเสนอแนะในการทำวิทยานิพนธ์นี้

ขอกราบขอบพระคุณ บิดาและมารดาที่คอยเลี้ยงดูมาเป็นอย่างดีโดยตลอดและให้การสนับสนุนในศึกษาต่อปริญญาโทจนจบการศึกษา และขอขอบคุณทุกคนในครอบครัวที่ได้ให้การสนับสนุน และความช่วยเหลืออย่างที่สุดมาโดยตลอดจนสำเร็จการศึกษา ขอขอบพระคุณคณาจารย์ทุกท่านที่ให้การอบรม สั่งสอนวิชาความรู้จนสำเร็จการศึกษา

ขอขอบคุณเพื่อนรุ่นที่ 6 และรุ่นพี่, รุ่นน้องทุกคนที่ช่วยเหลือ ให้คำแนะนำอย่างดีมาตลอด รวมถึงทุกสิ่งที่ทำให้วิทยานิพนธ์นี้สำเร็จลุล่วงไปได้ด้วยดี และขอขอบคุณ โครงการปริญญาโทภาคที่ได้สนับสนุนด้านอุปกรณ์ และสถานที่ตลอดระยะเวลาในการทำวิทยานิพนธ์

วโรดม ลิมปะพันธุ์

มีนาคม 2555

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

คำอธิบายสัญลักษณ์และคำย่อ

AP	=	Access Point
DHCP	=	Dynamic Host Configuration Protocol
FTP	=	File Transfer Protocol
HA	=	Home Agent
HTTP	=	Hypertext Transfer Protocol
ICMP	=	Internet Control Message Protocol
IM	=	Instant Messenger
IP	=	Internet Protocol
IPv4	=	Internet Protocol version 4
IPv6	=	Internet Protocol version 6
LAN	=	Local Area Network
MANETs	=	Mobile Adhoc Networks
SMTP	=	Simple Mail Transfer Protocol
STCP	=	Stream Control Transmission Protocol
TCP	=	Transmission Control Protocol
UDP	=	User Datagram Protocol
WLAN	=	Wireless Local Area Network

การเปรียบเทียบประสิทธิภาพการทำงานโพรโทคอล TCP และ SCTP บนระบบการส่งข้อความด่วน (Instant Messaging)

Efficiency Comparison of TCP Protocol and SCTP Protocol on Instant Messaging System

คำนำ

ในปัจจุบันมีการใช้งานระบบการส่งข้อความด่วน Instant Messaging อย่างแพร่หลายเป็นจำนวนมากบนเครือข่ายการสื่อสารทุกชนิดและมีแอพลิเคชันทำงานภายใต้วัตถุประสงค์หลากหลายบนอุปกรณ์ที่แตกต่างกันเช่น เครื่องคอมพิวเตอร์ตั้งโต๊ะ, คอมพิวเตอร์พกพา, แท็ปเล็ท, อุปกรณ์พกพา, สมาร์ทโฟน รวมถึงระบบ Embedded System ที่ผนวกรวมกับระบบควบคุมเครื่องใช้ไฟฟ้า โดยมาตรฐานการส่งผ่านข้อมูลของแอปพลิเคชันที่ใช้กันอย่างแพร่หลายนิยมทำงานบนระบบ TCP โพรโทคอลเป็นหลัก

การส่งผ่านข้อมูลบนโพรโทคอลระบบ TCP ของระบบการส่งข้อความด่วน Instant Messaging แม้จะสามารถรับประกันความถูกต้องและความแน่นอนในการส่งก็ตาม แต่ในปัจจุบันรูปแบบการใช้งานเปลี่ยนแปลงไปอย่างรวดเร็ว และในปัจจุบันการสื่อสารข้อมูลจะประกอบไปด้วยข้อมูลมัลติมีเดียซึ่งมีขนาดใหญ่เป็นจำนวนมาก และสื่อกลางที่ใช้เชื่อมต่ออุปกรณ์หลายรูปแบบก็หลากหลายตามไปด้วย ไม่ว่าจะเป็นระบบแลน, Wireless, Cellular, 3G, ระบบดาวเทียมสื่อสารข้อมูล, Bluetooth, Cable Modem, xDSL หรือการส่งผ่านด้วยสื่ออื่นๆ ดังนั้นระบบการส่งข้อความด่วนในปัจจุบันและอนาคตจะต้องทำงานบนสื่อกลางที่มีคุณสมบัติแตกต่างกันทั้งความเร็ว ขนาดช่องทาง(Bandwidth) ตลอดจนการสูญหายของข้อมูลที่แตกต่างกันอีกด้วย โพรโทคอล TCP อาจไม่ใช่ทางเลือกที่ดีที่สุดในทุกสภาพแวดล้อมการทำงาน

โพรโทคอล SCTP ได้ถูกพัฒนาขึ้นมาเพื่อใช้งานทดแทนตามข้อจำกัดของโพรโทคอล TCP เช่นมีความทนทานต่อการโจมตี มีความทนทานต่อการสูญหายของข้อมูล ความสามารถในการเปลี่ยนเส้นทางการสื่อสารข้อมูล ความสามารถแก้ไขการสูญเสียการเชื่อมต่อ และการส่งผ่าน

ข้อมูลผ่านช่องทางจำนวนมาก (Multi Stream) เป็นต้น ข้อดีของการนำเอา SCTP มาใช้งานทดแทนโพรโทคอล TCP บนระบบการส่งข้อความด่วน Instant Messaging ถือเป็นการการปรับปรุงข้อด้อยหรือข้อจำกัดของการทำงานแบบเดิมบนระบบ TCP โพรโทคอลโดยอาศัยคุณสมบัติที่ดีของ SCTP ไม่ว่าจะเป็นการสื่อสารแบบหลายช่องทางหรือการทำ 4-way Handshake ซึ่งจะทำให้สามารถส่งข้อมูลได้มีเสถียรภาพทนทานต่อการโจมตี คุณสมบัติ Multi Stream ซึ่งจะทำให้มีความเร็วรับส่งข้อมูลสูงขึ้นตลอดจนมีความปลอดภัยจากการสูญหายจากคุณสมบัติ Multi Home

อย่างไรก็ตามการนำโพรโทคอล SCTP มาใช้งานกับระบบการส่งข้อความด่วน Instant Messaging อาจทำให้มีความยุ่งยากและความล่าช้าในการทำงานซึ่งเกิดจากลักษณะการทำงานบางประการของตัวโพรโทคอลเองดังนั้นจึงมีแนวความคิดที่จะทำการวิจัยทดสอบวัดผลเปรียบเทียบการทำงานการส่งผ่านข้อมูลระหว่างโพรโทคอล TCP และโพรโทคอล SCTP ด้วยขนาดข้อมูล ประเภทข้อมูลและสภาพแวดล้อมระบบที่แตกต่างกันเพื่อหาแนวโน้มประสิทธิภาพการทำงานที่เปลี่ยนแปลงตามสภาพแวดล้อมเครือข่าย

การทำงานวิจัยนี้คำนึงถึงสภาพการใช้งานบนอุปกรณ์และชนิดเครือข่ายที่หลากหลายดังที่กล่าวไปแล้วดังนั้นจึงมีการจำกัดขนาดช่องทาง (Bandwidth) ให้เหมาะสมเนื่องจากการสื่อสารบางชนิดจะไม่ใช้ช่องทางที่กว้างเพื่อประหยัดทรัพยากร เช่นระบบ Embeded System หรือระบบที่มีการสื่อสารสองทางไม่เท่าเทียมกัน (Asyncrous) เช่น ADSL ซึ่งช่องทางการส่งข้อมูล (Uplink) ถูกจำกัดไว้เพียง 1 Mbits เท่านั้น

ผลที่คาดว่าจะได้รับเมื่อทำงานวิจัยชิ้นนี้คือสามารถเปรียบเทียบประสิทธิภาพการทำงานการส่งผ่านข้อมูลในรูปแบบและขนาดซึ่ง ต่างกันระหว่างโพรโทคอล TCP และโพรโทคอล SCTP บนระบบการส่งข้อความด่วน Instant Messaging เพื่อให้สามารถปรับใช้โพรโทคอลที่เหมาะสมในการทำงานบนสภาพแวดล้อมระบบที่แตกต่างกันได้

วัตถุประสงค์

1. เพื่อปรับใช้ SCTP ในการใช้งานระบบการส่งข้อความด่วน Instant Messaging
2. เปรียบเทียบประสิทธิภาพของการทำงานด้วยระยะเวลาในการส่งชุดข้อมูลระหว่างโพรโทคอล TCP และ โพรโทคอล SCTP
3. เปรียบเทียบประสิทธิภาพโพรโทคอล TCP และ SCTP บนสภาพแวดล้อมบนเครือข่ายขนาดข้อมูล และการสูญหายของข้อมูลต่างกัน

ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถปรับปรุงประสิทธิภาพการทำงานของระบบการส่งข้อความด่วน Instant Messaging โดยอาศัยโพรโทคอล SCTP
2. สามารถเป็นแนวทางในการปรับใช้โพรโทคอล SCTP ร่วมกับแอพลิเคชันอย่างมีประสิทธิภาพ
3. สามารถเป็นแนวคิดเพื่อใช้งานและการพัฒนาต่อไปได้

ขอบเขตและข้อจำกัด

1. ในปัจจุบันแอพลิเคชันต่างๆ ได้พยายามพัฒนาโพรโทคอลของตัวเองขึ้นมาโดยมีคุณสมบัติและความสามารถเฉพาะตัวซึ่งไม่เป็นไปตามมาตรฐานทั่วไป งานวิจัยนี้จึงจำกัดขอบเขตอยู่กับมาตรฐาน TCP และ SCTP ปกติเท่านั้น
2. การทำการวิจัยและทดสอบประสิทธิภาพการทำงานครั้งนี้มีพื้นฐานอยู่บนระบบเครือข่ายเสมือนซึ่งได้มีการควบคุมสภาพแวดล้อมให้เป็นตามที่กำหนดซึ่งผลที่ได้อาจต่างจากสภาพการใช้งานจริง

การตรวจเอกสาร

ทฤษฎีพื้นฐานเกี่ยวกับงานวิจัย

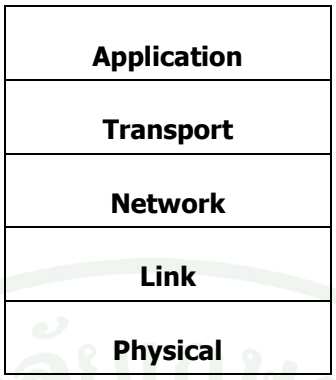
เนื่องจากงานวิจัยนี้เป็นการปรับปรุงการทำงานระบบ Instant Messenger ด้วย SCTP โพรโทคอล ดังนั้นทฤษฎีพื้นฐานจะประกอบด้วย โพรโทคอล TCP, SCTP และระบบ Instant Messenger

1. โพรโทคอล TCP/IP

โพรโทคอล TCP/IP เป็นชุดของโพรโทคอลที่มีการพัฒนามาตั้งแต่ปี 1960 โดยมีวัตถุประสงค์ให้สามารถใช้ในการสื่อสารจากต้นทางข้ามเน็ตเวิร์กไปยังปลายทางได้ และสามารถหาเส้นทางที่จะส่งข้อมูลไปได้เองโดยอัตโนมัติ ถึงแม้ว่าในในระหว่างทางอาจจะผ่านเครือข่ายที่มีปัญหา โพรโทคอลก็ยังคงหาเส้นทางส่งผ่านข้อมูลไปให้ถึงปลายทางจนได้ ในระยะเริ่มต้นโพรโทคอลนี้ใช้ในวงแคบๆเฉพาะราชการและสถานศึกษาของอเมริกา จนในช่วงปี 90 จึงมีการนำมาใช้ในทางธุรกิจ และเป็นจุดเริ่มต้นของอินเทอร์เน็ทในปัจจุบัน

1.1 การแบ่งชั้น (Layering)

TCP/IP เป็นชุดของโพรโทคอลที่ประกอบด้วยโพรโทคอลย่อยหลายตัวโดยแต่ละตัวก็จะทำหน้าที่ในแต่ละชั้นหรือเลเยอร์(Layer) ซึ่งรับผิดชอบและแปลความหมายของข้อมูลในแต่ละระดับของการสื่อสารซึ่งในภาพรวมแล้ว TCP/IP แบ่งออกเป็น 5 เลเยอร์ดังรูป



ภาพที่ 1 TCP/IP Layer

โดยหน้าที่ความรับผิดชอบแต่ละเลเยอร์มีดังนี้

- Link Layer ในเลเยอร์นี้จะเป็นดีไวซ์ไดรเวอร์ที่ทำงานอยู่บนระบบปฏิบัติการแต่ละระบบทำหน้าที่รับผิดชอบในการรับส่งข้อมูลตั้งแต่ระดับกายภาพ, สัญญาณไฟฟ้า จนถึงการแปลความจากระดับจนเป็นข้อมูลทางคอมพิวเตอร์, โพรโทคอลระดับนี้ เช่น Ethernet และ SLIP (Serial Line Internet Protocol)

- Network Layer รับผิดชอบในการรับ-ส่งข้อมูลในเน็ตเวิร์ก ส่งต่อข้อมูลไปจนถึงจุดหมายปลายทางโพรโทคอลระดับนี้ ได้แก่ IP, ICMP, IGMP

- Transport Layer รับผิดชอบในการส่งข้อมูลระหว่างเครื่องหนึ่ง (Host) ไปยังอีกโฮสต์หนึ่ง และจะส่งข้อมูลไปให้ Application Layer นำไปต่อใช้งานต่อ มีโพรโทคอลที่จัดอยู่ในเลเยอร์นี้คือ TCP และ UDP ซึ่งมีลักษณะในการรับส่งที่แตกต่างกันออกไป

- Application Layer เป็นเลเยอร์ที่แอพพลิเคชันเรียกใช้โพรโทคอลระดับล่างๆลงเพื่อวัตถุประสงค์แตกต่างกัน เช่น

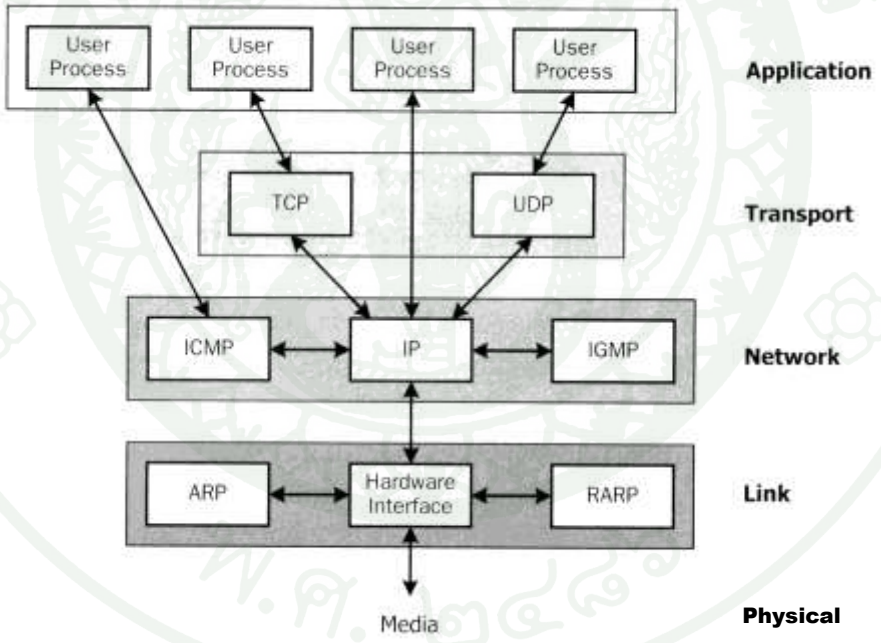
FTP (File Transfer Protocol)	ใช้สำหรับรับส่งแฟ้มข้อมูล
SMTP (Simple Mail Transfer Protocol)	ใช้รับส่งจดหมายอิเล็กทรอนิกส์
Telnet	ใช้สำหรับควบคุมเครื่องระยะไกล
HTTP (Hypertext Transfer Protocol)	เป็นโพรโทคอลที่ใช้รับส่งข้อมูลเว็บเพจระหว่างบราวเซอร์ และเซิร์ฟเวอร์
POP (Post Office Protocol)	ใช้สำหรับดาวน์โหลดอีเมล์จากเมล์เซิร์ฟเวอร์มาไว้ที่เครื่องไคลเอนต์ (PC)

1.2 การแบ่งชั้น TCP/IP (TCP/IP Layering)

ในชุดของโพรโทคอล TCP/IP ประกอบด้วยโพรโทคอลหลายตัวทำงานร่วมกันในเลเยอร์ต่างๆ และมีหน้าที่แตกต่างกันออกไป ภาพที่ 2.18 แสดงให้เห็นถึงโพรโทคอลในแต่ละเลเยอร์ที่รวมกันเป็นชุดของ TCP/IP ซึ่งตัวหลักก็คือ

- TCP อยู่ในทรานสปอร์ตเลเยอร์ ทำหน้าที่จัดการและควบคุมการรับส่งข้อมูลให้มีเสถียรภาพและเชื่อถือได้

- UDP อยู่ในทรานสปอร์ตเลเยอร์ ทำหน้าที่จัดการและควบคุมการรับส่งข้อมูลเช่นเดียวกันแต่ไม่มีกลไกการรับส่งที่มีเสถียรภาพและเชื่อถือได้ โดยปล่อยหน้าที่นี้ให้กับแอพพลิเคชั่นเลเยอร์เป็นผู้ทำหน้าที่แทน



ภาพที่ 2 เลเยอร์ของโพรโทคอลต่างๆในชุด TCP/IP Suite

- IP อยู่ในเน็ตเวิร์กเลเยอร์ เป็นโพรโทคอลหลักในการสื่อสารข้อมูล ซึ่งกลไกสำคัญที่ ทำให้ข้อมูลสามารถเคลื่อนที่ไปยังปลายทางได้ก็คือโพรโทคอล IP

- ICMP (Internet Control Message Protocol) อยู่ในเน็ตเวิร์กเลเยอร์ทำหน้าที่เสริมให้การทำงานของ IP ให้สมบูรณ์ โดยจะเป็นโพรโทคอลที่คอยส่งข่าวสารและแจ้ง ความผิดพลาด

ให้แก่ IP แต่ในบางโอกาสแอพพลิเคชันเลเยอร์ก็เรียกใช้ ICMP โดยตรงเพื่อใช้ประโยชน์จากความสามารถของ ICMP ด้วย

- IGMP (Internet Group Management Protocol) อยู่ในเน็ตเวิร์กเลเยอร์ ทำหน้าที่ในกาส่ง UDP ดาต้าแกรมไปยังกลุ่มของโฮสต์ หรือโฮสต์หลายๆตัวพร้อมกัน

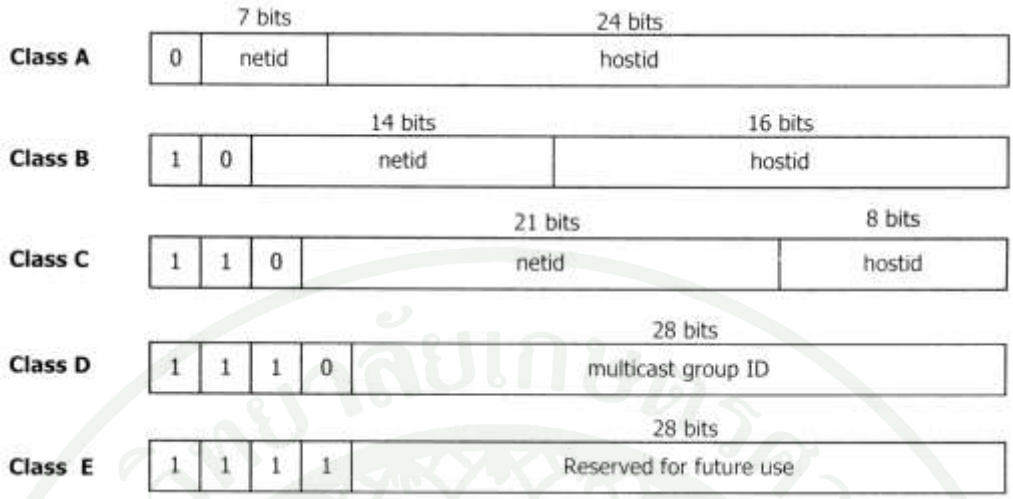
- ARP (Address Reservation Protocol) อยู่ในลิงค์เลเยอร์ทำหน้าที่เปลี่ยนระหว่างแอด-เดรสที่ใช้โดย IP ให้เป็นแอดเดรสของ Network Interface ให้เป็นแอดเดรสที่ใช้โดย IP

1.3 ที่อยู่บนระบบเครือข่าย (Internet Address)

ทุกอินเทอร์เฟซที่ต่ออยู่บนอินเทอร์เน็ทจะต้องมีหมายเลขประจำตัวเพื่อใช้ในการสื่อสารข้อมูลเรียกว่า Internet Address หรือเรียกย่อๆว่า IP Address โดยค่า IP Address นี้จะเป็นหมายเลขจำนวน 32 บิต แต่แทนที่จะกำหนดให้เลขทั้ง 32 บิตนั้นถูกนับต่อเนื่องกันไป ตั้งแต่ 0 – 2^{32} ก็ใช้วิธีการแบ่งเลขหมายดังกล่าวออกเป็นกลุ่มของเลขขนาด 8 บิตจำนวน 4 ชุด และแต่ละชุดด้วยจุด ตัวอย่างเช่น 192.168.13.204

นอกจากนี้ IP Address นั้นยังถูกแบ่งออกเป็น 2 ส่วนคือ ส่วนที่เป็นแอดเดรสของเครือข่าย (Network ID) และส่วนที่เป็นแอดเดรสของโฮสต์ (Host ID) ซึ่งข้อมูลในส่วนนี้จะถูกใช้สำหรบค้นหาเส้นทางของ IP ในการที่จะขนส่งจากข้อมูลจากต้นทางให้ถึงปลายทางอย่างถูกต้อง

เพื่อเป็นการกำหนดขนาดของเน็ตเวิร์กสำหรับ IP Address ต่างดังนั้นจึงมีการจัด IP Address ในแต่ละช่วงออกเป็นคลาส (class) ต่างๆกันจาก A ถึง E เพื่อจะได้ทำการจัดสรร IP Address ได้อย่างเหมาะสมกับขนาดของเน็ตเวิร์ก



ภาพที่ 3 การกำหนดแอดเดรสสำหรับคลาสต่างๆ

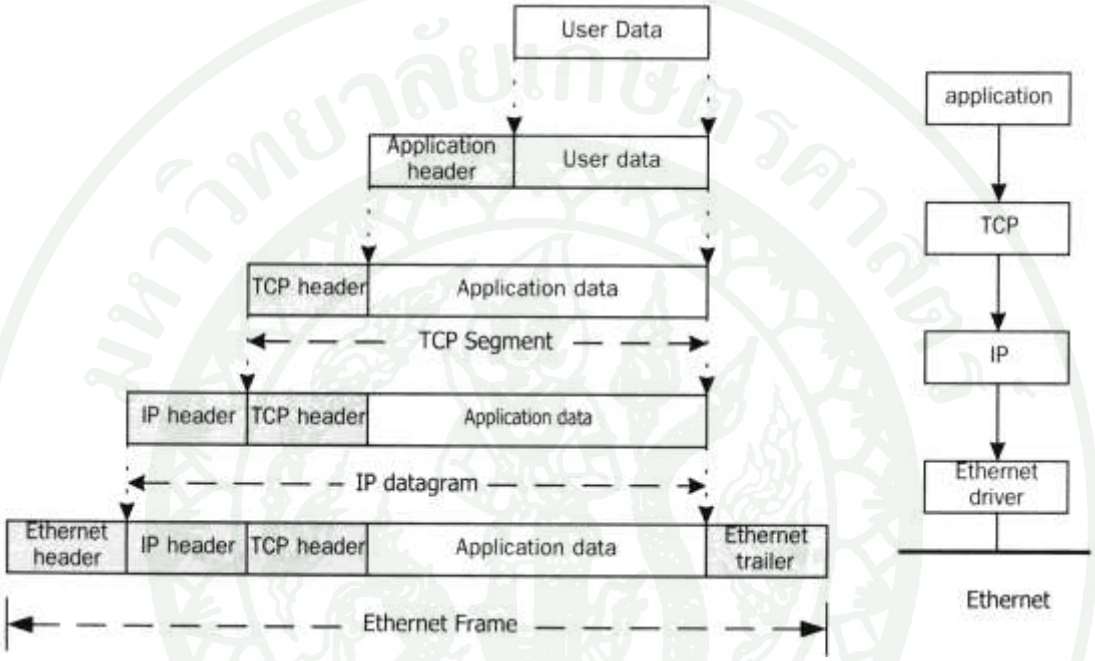
ตารางที่ 1 แสดงช่วง IP Address แต่ละคลาส

Class	Range
A	0.0.0.0 – 127.255.255.255
B	128.0.0.0 – 191.255.255.255
C	192.0.0.0 – 223.255.255.255
D	224.0.0.0 – 239.255.255.255
E	240.0.0.0 – 255.255.255.255

1.4 การประกอบข้อมูล (Encapsulation)

การ Encapsulate คือ การนำข้อมูลที่ต้องการส่งมาประกอบรวมกับข้อมูลที่เป็นส่วนควบคุมของโพรโทคอล โดยข้อมูลส่วนที่เป็นควบคุมนั้นจะถูกนำมาไว้ในส่วนหัวของข้อมูล เรียกว่า เฮดเดอร์ (Header) ซึ่งในการรับข้อมูลจะได้รับเฮดเดอร์จากนั้นก็นำเฮดเดอร์ไปแปลและทราบว่าข้อมูลที่ตามมานั้นมีลักษณะอย่างไรเพื่อจัดการได้ถูกต้อง ภายในเฮดเดอร์ของโพรโทคอลส่วนใหญ่จะประกอบด้วยข้อมูลหลักที่สำคัญของโพรโทคอลที่ทำการ Encapsulate คือ แอดเดรสต้นทาง,แอดเดรสปลายทาง, ความยาวข้อมูล, รหัสตรวจสอบความผิดพลาดข้อมูลซึ่งสิ่งที่จะต้อง

เน้นให้ใช้ก็คือ จะมีข้อมูลสำคัญเฉพาะโพรโทคอลที่ทำการ Encapsulate มาเท่านั้น ตัวอย่างเช่น การ Encapsulate ของ Ethernet ก็จะมีการระบุ Ethernet address ลงในเฮดเดอร์เท่านั้น จะไม่มีการบรรจุ IP address ลงมาใน Ethernet Header ด้วยแต่อย่างใดเพราะในเลเยอร์ของ Ethernet จะไม่รู้จัก IP Address หรือรหัสควบคุมใดๆของ TCP/IP



ภาพที่ 4 การ Encapsulate ข้อมูลผ่านชั้นของโพรโทคอลแต่ละระดับ

ในการรับส่งข้อมูลนั้น ข้อมูลที่จะรับส่งบนเน็ตเวิร์กจะประกอบด้วย 2 ส่วน คือข้อมูลจริงและ header ของโพรโทคอลตามลำดับดังนี้

ลำดับที่ 1 . TCP

ลำดับที่ 2. IP

ลำดับที่ 3 . Ethernet

ฝ่ายรับข้อมูลก็จะต้อง Decapsulate ตามลำดับเช่นกัน ส่วนการ Encapsulate ในแต่ละระดับก็จะมีข้อมูลที่อยู่ภายในแตกต่างกันออกไป ข้อมูลที่ทำการ Encapsulate เรียบร้อยแล้วจาก TCP ส่งไปยัง IP เรียกว่า TCP segment ในระดับ IP ก็จะถือว่า TCP Segment เป็นข้อมูลทั้งชุด เมื่อไปรวมกับ IP Header ส่งไปยังเลเยอร์ DataLink จะเรียกว่า IP Datagram ในระดับ DataLink เมื่อ

นำ IP Datagram มาใส่ Ethernet header ข้อมูลทั้งหมดจะถูกเรียกว่า Ethernet Frame จะเห็นได้ว่าทุกแพ็กเกตของข้อมูลจะต้องเสียส่วนหนึ่งไปเป็นเฮดเดอร์เสมอ

1.5 หมายเลขพอร์ต (Port Number)

ข้อมูลทั้งหมดทุกเซ็กเมนต์จะต้องผ่านเลเยอร์เยอร์บนสุดของ TCP/IP นั่นคือแอพพลิเคชั่นเลเยอร์เพราะ แอพพลิเคชั่นเลเยอร์จะครอบคลุมการสื่อสารทั้งหมด ดังนั้นหากมีหลายแอพพลิเคชั่นต้องการรับข้อมูลส่งข้อมูลผ่าน TCP/IP แต่ละแอพลิเคชั่นจะสามารถแยกแยะโดยอาศัยพอร์ต (port) ในโพรโทคอล TCP/IP ได้ถูกออกแบบให้มีหมายเลขพอร์ตอยู่ในเฮดเดอร์ เพื่อระบุว่าข้อมูลเซ็กเมนต์นี้เป็นของแอพพลิเคชั่นอะไร ในโฮสต์นั้นแอพพลิเคชั่นแต่ละตัวที่ให้บริการอยู่ในเครื่องต่างจะมีหมายเลขพอร์ตประจำตัว เพื่อจะสามารถเลือกนำข้อมูลมาใช้เป็นของแอพพลิเคชั่นตนเอง หมายเลขพอร์ตนี้เรารู้จักกันดีแล้วใช้เป็นมาตรฐาน ได้แก่ พอร์ต 20, 21 เป็นของ FTP , พอร์ต 23 Telnet , พอร์ต 25 SMTP , พอร์ต 30 HTTP เป็นต้น

โดยทั่วไปหมายเลขพอร์ตจะมีความสำคัญกับฝั่งของเซิร์ฟเวอร์เท่านั้น เนื่องจากแอพพลิเคชั่นฝั่งเซิร์ฟเวอร์จะต้องคอยรอรับการรีเควสท์ (Request) หรือขอรับบริการจากไคลเอนต์ที่พอร์ตเดิมเสมอ ส่วนในฝั่งไคลเอนต์เองหมายเลขพอร์ตไม่จำเป็นต้องเป็นหมายเลขที่ตายตัวและคงที่ เพราะหมายเลขพอร์ตจะเป็นการสุ่มขึ้นมาใช้ชั่วคราว (Ephtmeral Port) และจะมีการเรียกใช้พอร์ตใหม่ทุกครั้งที่มีการรับส่งข้อมูลเซสชั่น (session)

1.6 พอร์ตสงวน (Reserved Port)

ในระบบปฏิบัติการ Unix มีการสงวนพอร์ต บางส่วนไว้ให้สำหรับโพรเซสที่มีสิทธิพิเศษของ Super user เท่านั้นที่สามารถใช้พอร์ตในช่วง 1 – 1023 ได้ แต่สำหรับ Windows ไม่ได้สงวนแต่อย่างใด ในบางครั้งคำว่า Unix Reserved Port ก็หมายถึงพอร์ต 1 – 1023

ทั้ง TCP และ UDP ต่างก็ใช้งานพอร์ตในลักษณะเดียวกันคือ ใช้ระบุแอพพลิเคชั่น หมายเลขพอร์ตที่สามารถระบุได้จะเป็นข้อมูล 16 บิต นั่นหมายความว่าเราสามารถมีพอร์ตที่สามารถใช้งานได้ทั้งสิ้น 65535 พอร์ต ในแต่ละโพรโทคอล ดังนั้นจำนวนพอร์ตทั้งหมดนี้สามารถ

ใช้งานในเครื่องเราได้เมื่อใช้โพรโทคอล TCP/IP คือ 128 K นั่นเอง โดยเป็น TCP 64 K และของ UDP อีก 64 K

1.7 ปัญหาความน่าเชื่อถือ (Unreliable)

ด้วยตัวโพรโทคอล IP (UDP) เองมีกลไกลในการหาเส้นทางเดินของข้อมูล (routing) ไปยังปลายทางโดยจะพยายามใช้กลไกลเหล่านั้นในการส่งข้อมูลไปถึงปลายทางให้จงได้ แต่ไม่มีการตรวจสอบว่าปลายทางได้รับข้อมูลหรือไม่ กลไกลการแจ้งกับมีเพียงในกรณีที่มีปัญหาเกิดขึ้นระหว่างการเดินทาง เช่น เราเตอร์มีปัญหา, หาเส้นทางไม่ได้ ก็จะใช้ ICMP ส่งข้อความกลับมาให้ผู้ส่งทราบ ซึ่งอาจจะถึงผู้ส่งหรือไม่ก็ได้ ดังนั้นหากส่งด้วย IP แล้วไม่มีการตอบกลับมาก็อาจจะเป็นไปได้ว่าข้อมูลถึงปลายทางอย่างสมบูรณ์ หรือข้อมูลไปไม่ถึงแต่ส่ง error message กลับมาไม่ได้

1.8 การสูญเสียการเชื่อมต่อ (Connectionless)

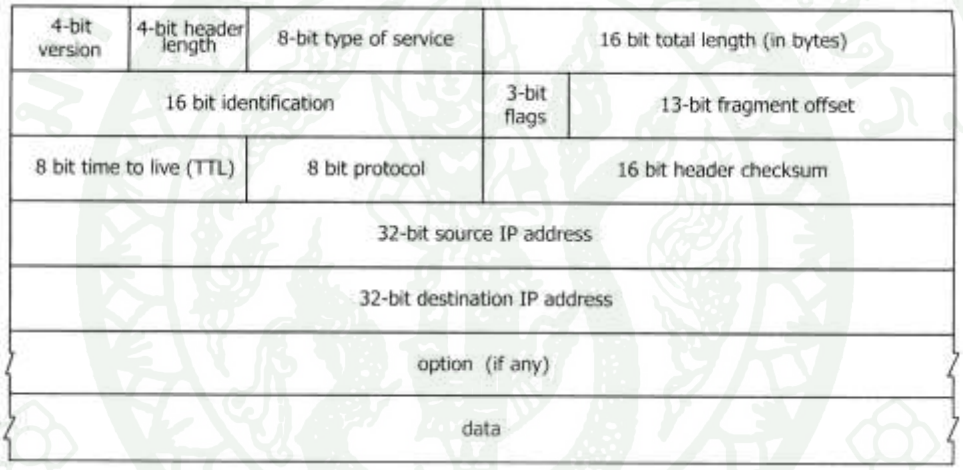
หมายถึง IP จะไม่มีสถานะเหมือนการเชื่อมต่อกันระหว่างต้นทางกับปลายทาง โดยทั่วไปการที่โฮสต์สองโฮสต์จะมีการเชื่อมต่อกันได้นั้น อันดับแรกสุดก็จะต้องเชื่อมต่อกันให้ถึงกันก่อน หลังจากนั้นก็จะสามารถรับส่งข้อมูลได้ และเราก็สามารถรับส่งข้อมูลกันได้ตลอดตราบใดที่ยังมีการเชื่อมต่ออยู่ สำหรับโพรโทคอลระดับ IP หมายถึงการเชื่อมต่อแบบลอจิคัลโดยมีการเชื่อมต่อแบบ connectionless นั่นคือหากรับส่งข้อมูลผ่าน IP จะต้องทำการเชื่อมต่อใหม่ทุกครั้งที่จะรับส่งข้อมูล 1 IP ดาต้าแกรม ในระดับ IP จะรู้จักข้อมูลเพียงแต่ในดาต้าแกรมเดียวเท่านั้น จะไม่รู้จักดาต้าแกรมอื่นที่ส่งก่อนหน้าหรือตามมา และจะไม่รู้ว่าดาต้าแกรมอื่นมีความสัมพันธ์กันหรือไม่

บางกรณีที่ IP มีการสื่อสารอย่างต่อเนื่องและมีความสำพันธ์กันระหว่างดาต้าแกรม ก็คือกรณีของ Fragmentation ที่มีการกระจายข้อมูลออกเป็นหลายแพ็กเก็ตและเมื่อแพ็กเก็ตส่งถึงปลายทางแล้วจะต้องนำมารวมกันอีกครั้งหนึ่งตามลำดับ ซึ่งแสดงว่า IP น่าจะมีความสำคัญกันระหว่างดาต้าแกรม แท้ที่จริงแล้วความสัมพันธ์ดังกล่าวมิใช่ความสัมพันธ์กันระหว่างดาต้าแกรมแต่อย่างใด เป็นความสัมพันธ์ภายในดาต้าแกรมเดียวกันแต่ถูกกระจายออกจากกันและนำกลับมารวมกันใหม่ให้เป็นดาต้าแกรมเดียวเหมือนเดิมในระดับ application มิใช่การนำข้อมูลดาต้าแกรมอื่นมาเชื่อมโยงรวมกัน ข้อมูลหนึ่งดาต้าแกรมไม่จำเป็นต้องส่งเพียงครั้งเดียว หากความยาวเกินกว่า

ที่จะส่งได้ในแต่ละครั้งนั้น ดาต้าแกรมที่สามารถจะแบ่งย่อยเพื่อให้ขนาดของแพ็กเก็ตมีขนาดเหมาะสมได้

1.9 IP Header

ขนาดของ IP Header โดยปกติจะมีขนาด 20 ไบต์ยกเว้นในกรณีที่มีการเพิ่มเติมออปชั่นบน IP Header จากภาพจะแสดงเป็นท่อนละ 32 บิต ก็คือ 4 ไบต์ โดยการส่งข้อมูลจะเรียงลำดับให้ไบต์แรกก่อนตามลำดับ สำหรับข้อมูลแต่ละส่วนในเฮดเดอร์มีความหมายดังนี้



ภาพที่ 5 IP Header

บิต 0-3 เวอร์ชั่นของ TCP/IP ปัจจุบันเป็นเวอร์ชั่น 4

บิต 4-7 Header Length ความยาวของเฮดเดอร์ โดยทั่วไปถ้าไม่มีออปชั่นค่าในส่วนนี้จะเป็น 5 หมายความว่าความยาวของข้อมูลมีขนาด 5*32 บิต หรือเท่ากับ 20 ไบต์

บิต 8 – 15 Type of Service (TOS) ปัจจุบันไม่ได้ใช้งานแล้ว ถูกออกแบบมาเพื่อเก็บค่าของ Minimize delay, Maximize Throughput, Maximize reliability, Minimize Monetary cost เพื่อใช้เป็นตัวบ่งชี้ให้เราต์เตอร์ตัดสินใจในการเลือกเราต์ข้อมูลดาต้าแกรม ปัจจุบันไม่มีการนำส่วนนี้ไปใช้งาน

บิต 16 – 31 Total length เป็นฟิลด์ที่บอกจำนวนไบต์ทั้งหมดของ IP Datagram ด้วยขนาด 16 บิตของฟิลด์นี้แสดงว่าขนาดความยาวข้อมูลของ IP ดาต้าแกรมจะมีขนาดได้สูงสุด 65535

ไบต์ ขึ้นอยู่กับขนาดของ MTU โดยทั่วไปแล้วถึงแม้ว่าเราส่งข้อมูลใน 1 ดาต้าแกรมได้สูงถึง 65535 ไบต์

บิต 32 – 47 Identification เป็นหมายเลขของดาต้าแกรมที่ส่งในกรณีที่มีการกระจายของดาต้าแกรม

บิต 48 – 50 แฟลก ใช้ในการการแฟรกเมนต์ของดาต้าแกรม

บิต 51 – 63 Fragment offset กลับมาต่อเรียงกันในตำแหน่งของข้อมูลที่ถูกต้อง

บิต 64 – 71 Time To Live (TTL) เป็นฟิลด์ที่กำหนดจำนวนครั้งสูงสุดที่ดาต้าแกรมนี้จะถูกเราต์ (route) เพื่อป้องกันมิให้ดาต้าแกรมถูกเราต์ไปโดยไม่มีที่สิ้นสุด ด้วยคุณสมบัติของ IP ที่จะถูกเราต์ไปเรื่อยๆจนกว่าจะถึงปลายทาง TTL จะเป็นตัวบอกจำนวนสูงสุดของการเราต์หากว่าจำนวนครั้งของดาต้าแกรมถูกเราต์ไปเท่ากับ TTL แล้วยังไม่ถึงปลายทางก็ให้ทำการดร็อปดาต้าแกรมนี้ทิ้ง แล้วแจ้งกลับมายังต้นทางว่า Time out

บิต 72 – 79 Protocol อย่างที่กล่าวไปแล้วข้างต้นว่ามีโพรโทคอลย่อยหลายตัวที่อาศัย IP ในการขนส่งข้อมูลดังนั้นเพื่อระบุว่าข้อมูลที่ IP กำลังส่งอยู่นี้เป็นโพรโทคอลอะไรก็ต้องระบุไว้ในฟิลด์นี้ เช่น TCP , UDP , ICMP เป็นต้น

บิต 80 –95 Header checker เป็นส่วนตรวจสอบความถูกต้องของข้อมูลในเฮดเดอร์เพื่อป้องการผิดพลาดในการส่งข้อมูลซึ่งจะป้องกันข้อมูลในเฮดเดอร์

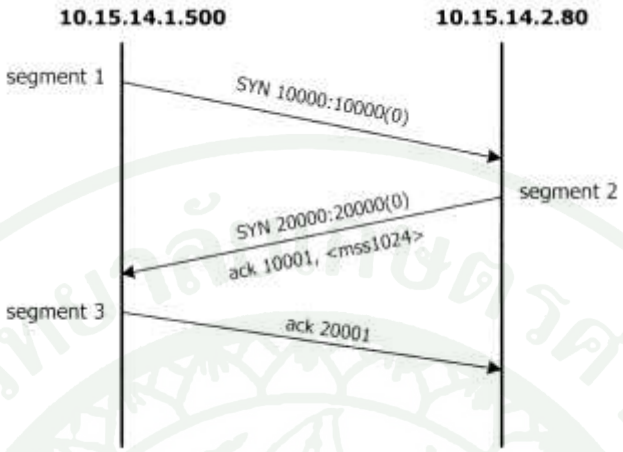
บิต 96 – 127 Source IP Address คือ IP Address ของผู้ส่งข้อมูลดาต้าแกรม

บิต 128 –163 Destination IP Address คือ IP Address ของปลายทางผู้รับของข้อมูลดาต้าแกรม

1.10 เออาร์พี - ARP (Address Resolution Protocol)

ARP (Address Resolution Protocol) เป็นกระบวนการเปลี่ยนค่าระหว่าง IP ไปเป็น Ethernet Address (หรือที่เรารู้จักกันในชื่อ MAC Address คือแอดเดรสทางฮาร์ดแวร์ของอุปกรณ์ เช่นการ์ด LAN) IP Address นั้นเป็นเพียงลอจิคัลแอดเดรสที่ผู้ใช้กำหนดขึ้นมาให้กับโฮสต์ตัวใดตัวหนึ่ง การตั้งค่า IP Address ทำโดยผู้ใช้ตั้งค่า IP ซึ่งเป็นสภาวะลอจิคัลที่ไม่ตายตัว ต่างกับในระดับอีเธอร์เน็ต ซึ่งอุปกรณ์ทุกชนิดจะมีหมายเลขประจำตัว (Ethernet Address) MAC address ซึ่งเป็นเลขที่ระบุติดตัวอยู่กับอุปกรณ์นั้นๆเป็นเลขขนาด 48 บิต (6 ไบต์) จะสามารถมีค่าแตกต่างกันได้ 2^{48} (281,474,976,710,656) ค่า และอุปกรณ์ Ethernet ทุกชนิดไม่มี Ethernet Address ที่ซ้ำกัน

1.11 การเชื่อมต่อ (Connection Establishment และ Termination)



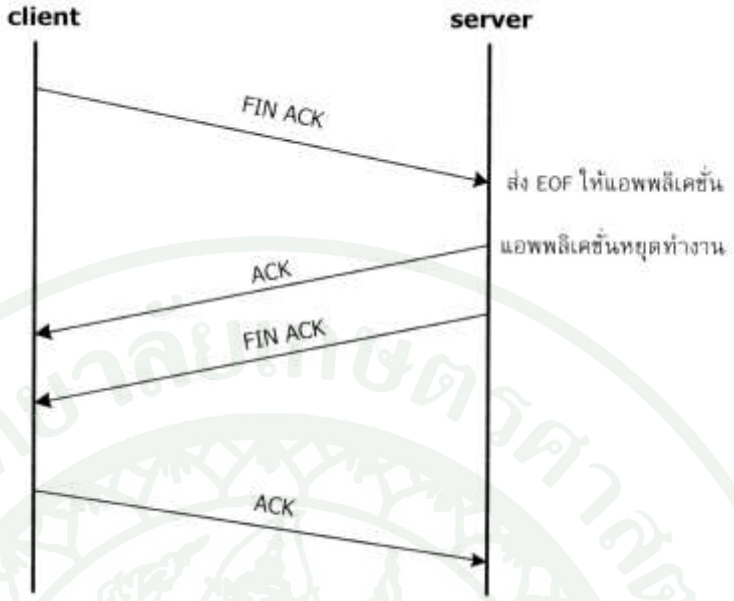
ภาพที่ 6 Connection Establishment

ก่อนที่ TCP จะสามารถรับส่งข้อมูลได้จะต้องมีการสถาปนา (Establishment) ซึ่งโพรโทคอล TCP ได้กำหนดขั้นตอนในการเริ่มต้นสร้าง Connection ไว้ดังนี้

1. เครื่องไคลเอนต์จะทำการส่งเซกเมนต์ โดยเปิด SYN Flag ระบุหมายเลขพอร์ตที่ต้องการติดต่อบนเซิร์ฟเวอร์และระบุหมายเลขลำดับของข้อมูล (ISN – Initial Sequence Number)
2. เครื่องเซิร์ฟเวอร์เมื่อได้รับข้อมูลเซกเมนต์จากข้อ 1 ก็จะตอบกลับด้วยการเพิ่มค่า ISN ที่รับขึ้นอีก 1 พร้อมทั้งระบุหมายเลขลำดับ (ISN) ของตนเอง และเปิด SYN กับ ACK Flag
3. ไคลเอนต์เมื่อได้รับการตอบกลับจากเซิร์ฟเวอร์ตามข้อ 2 ก็จะทำการตอบรับกลับไปโดยการเพิ่มค่า ISN ของเซิร์ฟเวอร์ขึ้นอีก 1 และเปิด ACK Flag

หลังจากการรับส่งข้อมูลได้ยุติลง จะต้องทำขั้นตอนการยุติการรับส่งข้อมูล การสิ้นสุดการรับ-ส่ง ข้อมูลมีอยู่ 4 ขั้นตอนคือ

1. ไคลเอนต์ทำการส่ง ISN พร้อมกับ FIN ACK Flag ไปยังเซิร์ฟเวอร์
2. เซิร์ฟเวอร์ทำการตอบรับ ISN และบวกค่า ISN อีก 1 พร้อมกับ ACK Flak
3. เซิร์ฟเวอร์ทำการส่ง ISN พร้อมกับ FIN ACK Flag ไปยังไคลเอนต์
4. ไคลเอนต์ตอบรับการยุติการสื่อสารด้วย ISN+ 1 พร้อมกับ ACK Flag



ภาพที่ 7 Connection termination

1.12 ปัญหา Head of Line (HOL)

เนื่องจาก TCP ได้มีการกำหนดช่องทางในการส่งข้อมูลภายในเพียงช่องทางเดียวในแต่ละการเชื่อมต่อ ซึ่ง TCP จะจัดเรียงข้อมูลที่ได้รับจาก server ตามหมายเลขลำดับข้อมูล โดยถ้าหากข้อมูลส่วนใดส่วนหนึ่งขาดหายไป TCP จะไม่ส่งข้อมูลส่วนถัดไปให้กับ Application layer แต่จะทำการร้องขอข้อมูลในส่วนที่ขาดไปมาให้ได้ก่อน จึงจะสามารถดำเนินการส่งข้อมูลให้กับ Application layer ได้ เมื่อเป็นเช่นนี้แล้ว เมื่อข้อมูลในส่วนใดของการเชื่อมต่อหายไป TCP จะไม่สามารถดำเนินการต่อไปได้เลย ถ้าไม่ได้รับข้อมูลในส่วนนั้นมาก่อนอย่างถูกต้อง และเนื่องจาก TCP มีช่องทางเพียงช่องทางเดียวในการเชื่อมต่อ จึงทำให้ทั้ง server และ client ไม่สามารถดำเนินการรับส่งข้อมูลในส่วนอื่นระหว่างกันได้อีกเลย ปัญหานี้เรียกว่า HOL (Head Of line problem)

2. SCTP (Stream Control Transmission Protocol)

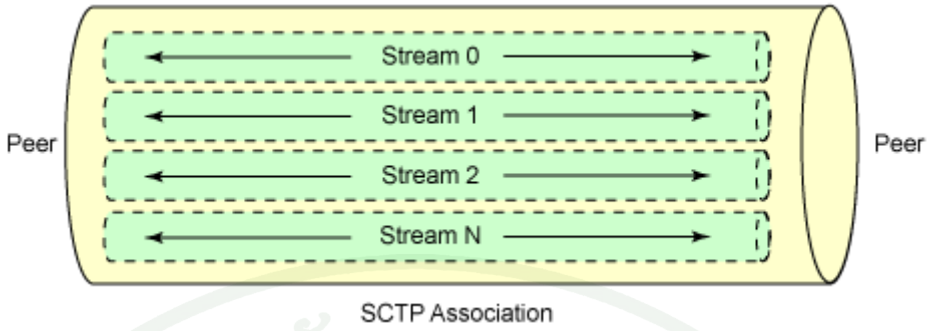
SCTP เป็น IP Transport protocol ในรูปแบบใหม่ที่ทำงานเทียบเท่ากับโพรโทคอลที่มีอยู่เดิมนั่นคือโพรโทคอล UDP และโพรโทคอล TCP ซึ่งโพรโทคอล SCTP สามารถทำงานได้ในหลายระดับชั้นเลเยอร์รวมทั้งมีคุณสมบัติการนำส่งข้อมูลไปยังแอพลิเคชันอื่น นอกจากนี้ SCTP ได้รับการรับรองโดย IETF ให้เป็นมาตรฐานที่ใช้ได้สำหรับการตรวจหาข้อผิดพลาดของขั้นตอนการรับส่งข้อมูล และยังคงเปิดรับการปรับปรุงในอนาคตข้างหน้าให้ทันสมัยขึ้นตาม IETF และมาตรฐาน RFC (ข้อมูลปี 2542) เช่นเดียวกับการทำงานของโพรโทคอล TCP ตัวโพรโทคอล SCTP มีคุณสมบัติการนำส่งข้อมูลที่เชื่อถือได้เช่นเดียวกัน ข้อมูลดังกล่าวจะสามารถนำส่งข้ามผ่านเครือข่ายโดยมีกระบวนการการแก้ไขข้อผิดพลาดและกระบวนการเรียงตามลำดับตามขั้นตอนอย่างถูกต้อง และเช่นเดียวกับโพรโทคอล TCP ตัวโพรโทคอล SCTP ก็ถือว่าเป็น session-oriented mechanism ซึ่งหมายความว่ากระบวนการการเชื่อมต่อจะถูกสร้างขึ้นระหว่างจุดเชื่อมต่อของแต่ละจุดหมายก่อน และการเชื่อมต่อนี้จะถูกรักษาไว้จนกระทั่งการนำส่งข้อมูลสำเร็จครบถ้วน การทำงานของโพรโทคอล SCTP มีการจัดการข้อมูลให้เรียงลำดับโดยอาศัยเนื้อที่สำหรับพักรอข้อมูล เมื่อเปรียบเทียบกับโพรโทคอล TCP แล้วโพรโทคอล TCP จะต้องได้รับข้อมูลเรียงตามลำดับซึ่งต่างจากการทำงานของโพรโทคอล SCTP ที่อาศัยโครงสร้างที่ได้กำหนดขึ้นตามรูปแบบของโพรโทคอล SCTP และมีที่พักรอข้อมูลขนาดใหญ่ดังนั้นจึงสามารถรับข้อมูลหลายเส้นทางได้โดยไม่จำเป็นต้องเรียงลำดับข้อมูลในการส่ง

โพรโทคอล SCTP ถูกพัฒนาขึ้นมาเพื่อรองรับการส่งข้อมูลที่อาจไม่เหมาะที่จะส่งด้วยโพรโทคอล TCP โดย IETF (Internet Engineering Task Force) และ SIGTRAN (Signaling Transport) เป็นกลุ่มผู้พัฒนาโพรโทคอลใหม่และได้นำเสนอโพรโทคอลนี้ออกสู่สาธารณะชนเมื่อเดือนตุลาคม พ.ศ.2543 โดยเผยแพร่ใน RFC 2960 ซึ่งโพรโทคอล SCTP มีลักษณะการทำงานที่คล้ายคลึงกับโพรโทคอล TCP แต่ได้เพิ่มคุณสมบัติหลายประการเพื่อแก้ปัญหาต่างๆ ที่พบได้ทั่วไปในโพรโทคอล TCP ได้แก่

2.1 Multi-Streaming

โพรโทคอล SCTP สามารถทำงานขนส่งข้อมูลหลายเส้นทางพร้อมกัน การทำงานของโพรโทคอล SCTP จะยอมให้ข้อมูลถูกแยกเป็นหลายส่วน และแยกส่งข้อมูลหลายเส้นทาง เป็นคุณสมบัติการนำส่งแบบไม่พึ่งพาอาศัยซึ่งกันและกันทำงานแยกออกจากกัน ดังนั้นหากมีข้อมูลหายไปในแต่ละเส้นทางที่นำส่งข้อมูล จะเกิดผลกระทบในเส้นทางนั้นเพียงเส้นทางเดียว ไม่ส่งผลกระทบต่อเส้นทางอื่นที่นำส่งข้อมูล ซึ่งแตกต่างจากโพรโทคอล TCP ที่ทำงานโดยอาศัยการนำส่งข้อมูลเพียงเส้นทางเดียว ซึ่งมีข้อดีที่สามารถมั่นใจว่าในเส้นทางนำส่งข้อมูลนั้นข้อมูลมีการเรียงลำดับแล้ว ขณะที่มีการนำส่งข้อมูลหรือมีการบันทึกข้อมูล ถ้ามีข้อมูลหายหรือการเรียงลำดับข้อมูลผิดพลาดเกิดขึ้น อาจจะก่อให้เกิดภาวะล่าช้าและเป็นภาระเพิ่มขึ้นในเครือข่าย เมื่อเกิดสภาวะดังกล่าวโพรโทคอล TCP จะต้องมีการนำส่งข้อมูลที่ล่าช้า และขณะเดียวกันต้องทำการจัดลำดับข้อมูลใหม่ให้ถูกต้อง

จากปัญหาที่พบใน TCP คือมีช่องทางในการสื่อสารเพียงช่องทางเดียว SCTP จึงพัฒนาให้สามารถส่งข้อมูลไปได้พร้อมกันได้ โดยเปรียบเสมือนกับมีหลายช่องทาง (stream) การสื่อสารในการสถาปนาการเชื่อมต่อเพียงครั้งเดียว โดยจะกำหนดค่าเฉพาะให้กับข้อมูลในแต่ละช่องทาง เพื่อให้ข้อมูลสามารถส่งออกไปได้พร้อมกัน เมื่อ client ได้รับข้อมูลในแต่ละช่องทางมาแล้ว client จะดำเนินการจัดเรียงและตรวจสอบข้อมูลในแต่ละช่องทางที่ได้รับ ถ้ามีการสูญหายของข้อมูลในช่องทางใด client จะร้องขอข้อมูลเฉพาะของช่องทางนั้นเท่านั้น โดยที่ไม่มีผลกระทบกับข้อมูลในช่องทางอื่น ทำให้ช่องทางอื่นยังสามารถดำเนินการรับข้อมูลต่อไปได้ ไม่จำเป็นต้องหยุดรอทั้งหมด การสร้างช่องทางการสื่อสารของ SCTP สามารถสร้างได้สูงสุด 64K ช่องทางทั้งทางด้านส่งออกและ 64K ช่องทางในด้านรับข้อมูลทำให้การเชื่อมต่อจะสามารถมีช่องทางการสื่อสารได้ถึง 128K ช่องทาง



ภาพที่ 8 Multi-Streaming

2.2 Multi-Homing

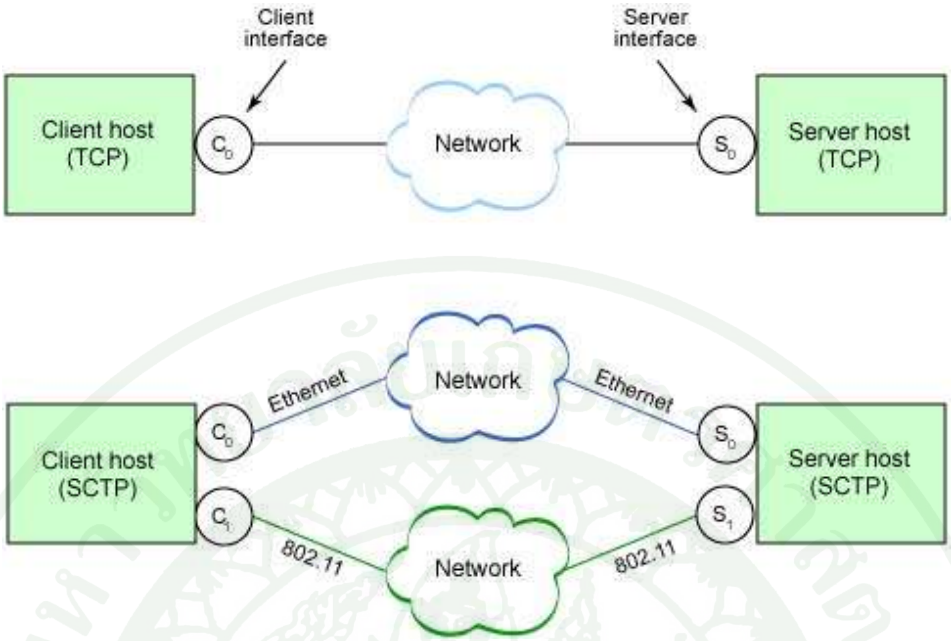
องค์ประกอบหลักอีกอย่างหนึ่งของโพรโทคอล SCTP คือการมีเป้าหมายได้หลายเป้าหมาย หรือกล่าวได้ว่าความสามารถของโพรโทคอล SCTP สามารถจะส่งไปได้หลายเป้าหมายพร้อมกัน การที่มีเป้าหมายได้หลายแห่งจะมีประโยชน์มากเมื่อเครือข่ายล้มเหลว ข้อมูลที่ส่งยังน่าจะสามารถนำส่งได้อยู่ ขณะที่การส่งข้อมูลแบบดั้งเดิมจะเกิดอุปสรรค เมื่อเกิดเครือข่ายล่มหรือไม่สามารถหาเป้าหมายได้ ดังนั้นเมื่อเกิดการล่มของเครือข่ายจะทำให้การส่งข้อมูลเกิดเหตุขัดข้องชั่วคราวจนกว่าจะสามารถกำหนดเส้นทางขึ้นมาใหม่จากจุดที่ล้มเหลว การใช้ Multi-Homed ของโพรโทคอล SCTP ซึ่งมีคุณสมบัติ redundant LAN จะทำให้สามารถแจ้งเส้นทางเลือกไปยังการสื่อสารข้อมูลส่วนอื่นๆในระบบ ดังนั้นระบบเครือข่ายที่จะสามารถลดการพึ่งพาการส่งข้อมูลเพียงเส้นทางเดียวซึ่งมีความเสี่ยงต่อเกิดความล้มเหลวของการนำส่งข้อมูลไปยังเป้าหมายได้ การใช้คุณสมบัติเป้าหมายหลายแห่งของ Multi-homed ในโพรโทคอล SCTP จะทำการ routing หลายเส้นทางโดยอาศัยเทคนิค route-pinning หรือเทคนิคอื่นๆได้ตามความต้องการของแอพลิเคชัน

ในรูปแบบปัจจุบันของโพรโทคอล SCTP ไม่ได้ใช้รับส่งข้อมูลเป็นหลักแต่จะใช้ Multi- Homing ในลดความคับคั่งของเครือข่ายเป็นวัตถุประสงค์หลักในการทำงาน การทำงานโดยทั่วไป ปลายทางหนึ่งจะถูกเลือกเป็นเป้าหมายแรก และถูกใช้เป็นจุดหมายปลายทางของข้อมูลทั้งหมดในการนำส่งข้อมูลปกติ การนำข้อมูลมารวมกันอีกครั้งจะสามารถใช้ทางเลือกของเป้าหมายหลายแห่งหรือหลายเส้นทาง เพื่อแก้ไขความผิดพลาดในการการนำส่งข้อมูลถึงจุดหมาย หากการ

นำส่งข้อมูลไปยังเป้าหมายแรกเกิดการล้มเหลว จะส่งผลต่อการตัดสินใจที่จำนำส่งข้อมูลส่วนที่เหลือไปยังเส้นทางเลือกอื่น

ในการสนับสนุนการใช้งาน Multi-Homing ของโพรโทคอล SCTP นั้นจะมีการแลกเปลี่ยนรายชื่อปลายทางของเป้าหมายขณะที่มีการสถาปนาการเชื่อมต่อกันตั้งแต่เริ่มต้น เป้าหมายในแต่ละจุดจะสามารถรับข้อมูลได้จากทุกผู้ส่งที่มีความเกี่ยวข้องกับเป้าหมาย ในการใช้งานจริง การมีระบบควบคุมที่แน่นอนสามารถนำมาใช้ประโยชน์ได้

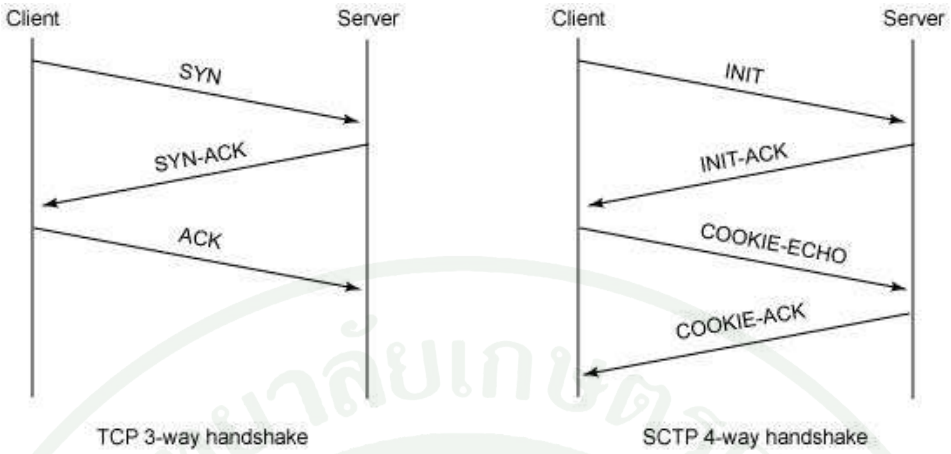
ในการทำงาน Multi-Homingเป็นคุณสมบัติที่ยอมให้มี server ได้มากกว่า 1 เครื่องในเครือข่าย เพื่อขจัดปัญหาเครือข่ายหยุดทำงาน โดยเมื่อ server ใดมีปัญหาไม่สามารถให้บริการได้ client ยังสามารถร้องขอข้อมูลจาก server อื่นได้ โดยไม่จำเป็นต้องเริ่มขอสถาปนาการเชื่อมต่อใหม่ ซึ่ง SCTP มีคุณสมบัติ Multi-Homing ที่สามารถกำหนด server ในเครือข่ายได้หลายเครื่อง จึงสามารถลดปัญหาเครือข่ายหยุดทำงานที่พบใน TCP ซึ่งมี server ได้เพียงเครื่องเดียวได้ ดังภาพที่ 9 เปรียบเทียบระหว่าง TCP ที่ไม่มีคุณสมบัติ Multi-Homing กับ SCTP ที่สามารถกำหนด server ได้หลายเครื่อง รูปส่วนบนเป็นลักษณะการเชื่อมต่อเครือข่ายของ TCP จะเห็นว่ามี server ให้บริการเครื่องเดียว ทำให้เมื่อระบบเครือข่ายหยุดทำงาน client ไม่สามารถขอเชื่อมต่อกับ server ได้เลย แต่ในเครือข่ายของ SCTP ในรูปส่วนล่าง จะเห็นว่าสามารถติดตั้งเครื่อง server ได้มากกว่าหนึ่งเครื่อง และยังสามารถแยก server แต่ละเครื่องให้อยู่คนละเครือข่ายอีกด้วย ซึ่งเมื่อเครือข่ายใดหยุดทำงาน client ก็ยังคงสามารถขอเชื่อมต่อได้จากเครือข่ายอื่น



ภาพที่ 9 Multi-Homing

2.3 4-way Handshake

รูปแบบการสถาปนาการเชื่อมต่อใน SCTP จะเป็นแบบ 4-way Handshake ดังภาพที่ 6 client -จะเริ่มขอสถาปนาการเชื่อมต่อโดยการส่ง INIT segment มื่อ server ได้รับ INIT segment แล้วจะตอบรับการขอสถาปนาโดยส่ง INIT-ACK segment กลับให้ client ตาม IP Address ที่ client ส่งมาและกลุ่มตัวเลขสุ่มที่ server สร้างขึ้นมาเรียกว่า COOKIE แต่ server จะยังไม่ทำการจองหน่วยความจำของระบบ ซึ่งจะต่างจาก TCP ที่จะจองหน่วยความจำของระบบตั้งแต่กระบวนการนี้ซึ่งทำให้เสี่ยงต่อกาถูกโจมตีด้วย SYN-flood ได้ เมื่อ client ได้รับ INIT-ACK segment และ COOKIE แล้ว client จะส่ง COOKIE ที่ได้ไปกับ COOKIE-ECHO segment เมื่อ server ได้รับ COOKIE-ECHO segment และตรวจสอบว่าเป็น COOKIE ที่ตนเองสร้างขึ้นมาจริง server ก็จะจองหน่วยความจำของระบบเพื่อใช้ในการรับ-ส่งข้อมูลและส่ง COOKIE-ACK segment กลับ และเริ่มการรับ-ส่งข้อมูลระหว่างกัน

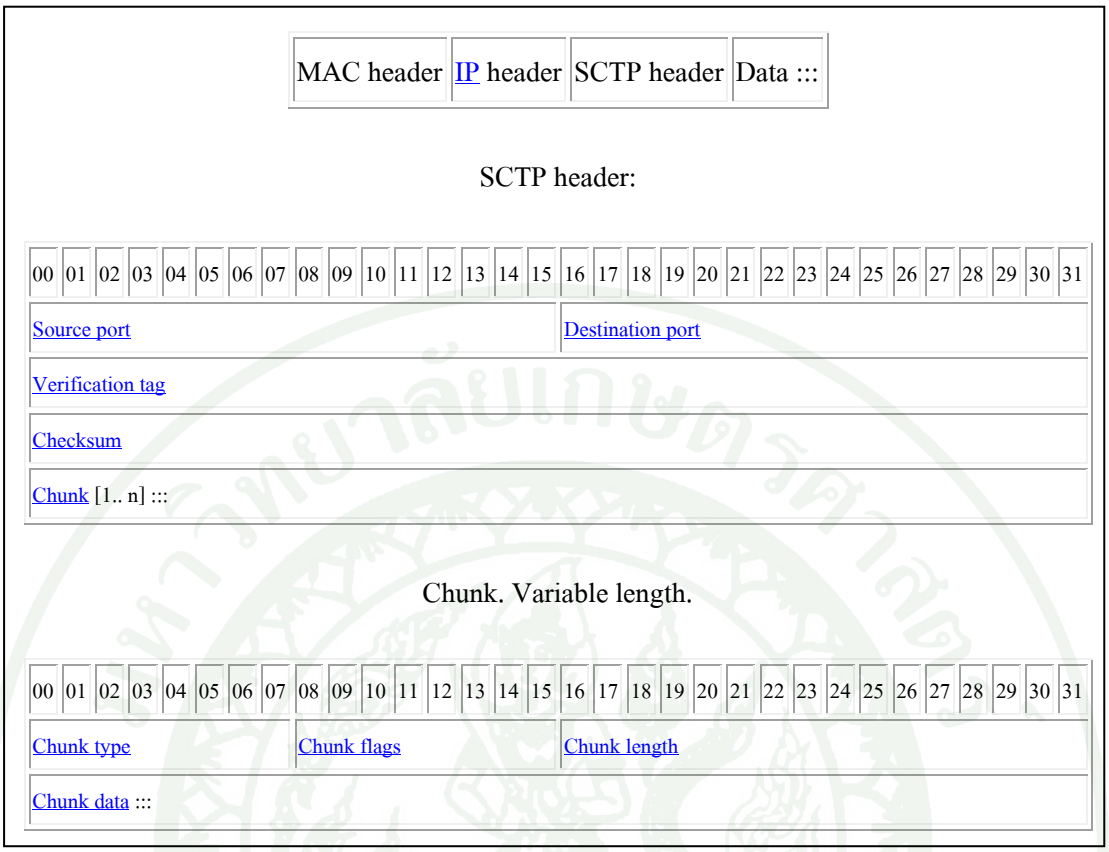


ภาพที่ 10 4-way Handshake

อย่างไรก็ตามการขอสถาปนาการเชื่อมต่อแบบ 4-way Handshake น่าจะทำให้ประสิทธิภาพลดลง แต่ในความเป็นจริงแล้ว ในการสถาปนาการเชื่อมต่อใน SCTP นั้น client สามารถส่งข้อมูลแนบไปกับ COOKIE-ECHO segment ได้ ซึ่งจะเท่ากับว่าเป็นกระบวนการขั้นตอนที่ 3 เทียบเท่ากับ TCP เดิม และเพิ่มความปลอดภัยเครือข่ายขึ้น

2.4 Receive Windows ขนาด 32-bit

ในการส่งข้อมูลของ server จำเป็นต้องทราบขนาดหน่วยความจำของ client ที่ยังสามารถรับได้ เพื่อ server จะไม่ทำการส่งข้อมูลมากไปกว่า client จะรับได้ ดังนั้น client จำเป็นต้องส่งขนาดหน่วยความจำที่ยังเหลือของตนเอง เรียกว่า receive windows ไปให้ server โดยส่งแนบไปกับ ACK segment ระบบเครือข่ายในยุคเริ่มแรกที่ใช้ TCP ในการส่งข้อมูล โดยที่ TCP packet กำหนดขนาด receive windows ไว้เพียง 16-bit ซึ่งในระบบเครือข่ายที่มีประสิทธิภาพมากขึ้นในปัจจุบัน การจำกัดค่า receive windows ที่ 16-bit ไม่สามารถทำงานได้อย่างเต็มประสิทธิภาพ SCTP พัฒนาให้สามารถประกาศขนาด receive windows ได้ถึง 32-bit เพื่อสามารถใช้ประสิทธิภาพของระบบเครือข่ายที่มีประสิทธิภาพสูงขึ้นได้อย่างคุ้มค่าและสามารถรองรับกับขนาดหน่วยความจำที่เครื่องในเครือข่ายในอนาคตจะมีได้



MAC header	IP header	SCTP header	Data :::

SCTP header:

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Source port																Destination port															
Verification tag																															
Checksum																															
Chunk [1.. n] :::																															

Chunk. Variable length.

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Chunk type								Chunk flags								Chunk length															
Chunk data :::																															

ภาพที่ 11 แสดงโครงสร้าง Packet ของโพรโทคอล SCTP

3. ระบบการส่งข้อความด่วน (Instant Messaging)

Instant Messaging (IM) คือระบบการส่งข้อความด่วน ระหว่างสองคน หรือกลุ่มคนในเน็ตเวิร์ก เดียวกัน เช่น การส่งข้อความผ่านทางอินเทอร์เน็ตการทำงานของระบบการส่งข้อความด่วนจำเป็นต้องใช้ไคลเอนท์ซอฟต์แวร์ โดยซอฟต์แวร์ทำการเชื่อมต่อระบบที่บริการระบบการส่งข้อความด่วน การส่งข้อความผ่านระบบการส่งข้อความด่วนในยุคแรก ตัวอักษรแต่ละตัวที่ทำการพิมพ์จะปรากฏทางหน้าจอของผู้ที่ส่งข้อความด้วยทันที ในขณะเดียวกัน การลบตัวอักษรแต่ละตัวจะลบข้อความทันที ซึ่งแตกต่างกับระบบระบบการส่งข้อความด่วนในปัจจุบัน โดยข้อมูลที่ปรากฏจะเกิดขึ้นหลังจากที่มีตกลงยอมรับส่งข้อความแล้ว ในปัจจุบันระบบการส่งข้อความด่วนที่ได้รับความนิยมได้แก่ MSN Messenger AOL Instant Messenger Yahoo! Messenger Google Talk .NET Messenger Service Jabber และ ICQ

Instant Messaging ได้ถูกพัฒนาขึ้นเพื่อใช้งานในระบบ PLATO ในยุค 1970 ในเวลาต่อมาช่วงยุค 1980 ได้มีการนำมาใช้ในวงการศึกษาและวิศวกร ในชื่อ UNIX talk และแพร่หลายในระบบอินเทอร์เน็ตอย่างรวดเร็ว ในช่วงต้นยุค 1990 ได้มีการพัฒนา ICQ มาใช้เป็นเมสเซนเจอร์ในเครื่องที่ไม่ใช่ UNIX เป็นครั้งแรก หลังจากนั้นได้มีการพัฒนาเมสเซนเจอร์ เป็นจำนวนมากโดยใช้โพรโทคอลของตัวเอง ทำให้มีผู้ใช้งานโปรแกรมเมสเซนเจอร์หลายตัวภายในเครื่องเดียวกัน หรือผู้ใช้งานสามารถเลือกได้ที่จะใช้โปรแกรมที่รองรับการทำงานหลายโพรโทคอล ได้แก่ Gaim หรือ Trillian ในปัจจุบัน นอกจากความสามารถในการส่งข้อความ เมสเซนเจอร์ต่างๆ ได้มีการรับรองการใช้งานการส่งไฟล์ การประชุมออนไลน์ และ VoIP

ตารางที่ 2 แสดงประเภทการทำงานระบบส่งข้อความด่วน (ข้อมูลจาก Wikipedia, 2010)

แอพลิเคชัน	จำนวนผู้ใช้งาน	โพรโทคอล
AIM	53ล้านใช้งาน/100ล้านลงทะเบียน	TCP
Camfrog	ไม่มีข้อมูล	ไม่มีข้อมูล
eBuddy	35ล้าน	TCP
Gizmo5	ไม่มีข้อมูล	ไม่มีข้อมูล
Gagu-Gadu	6ล้าน	ไม่มีข้อมูล
IBM Lotus sametime	40ล้าน	ไม่มีข้อมูล
iChat	ไม่มีข้อมูล	TCP
ICQ	50ล้าน	TCP
IMVU	1ล้าน	Jabber
XMPP	40ล้าน	ไม่มีข้อมูล
Mail.RU Agent	1ล้าน	ไม่มีข้อมูล
Meebo	1ล้าน	TCP

ตารางที่ 2 (ต่อ)

แอพลิเคชัน	จำนวนผู้ใช้งาน	โพรโทคอล
MXit	11ล้าน	TCP
PalTalk	3.3ล้าน	TCP
PSYC	1ล้าน	ไม่มีข้อมูล
Skype	19ล้านใช้งาน/309ล้านลงทะเบียน	ไม่มีข้อมูล
Tencent QQ	61.3ล้านใช้งาน 990ล้านลงทะเบียน	TCP
VZOchat	550,000	TCP
MSN	330ล้าน	TCP
Xfire	16ล้าน	TCP
Yahoo! Messanger	248ล้าน	TCP
Google Talk	ไม่มีข้อมูล	Jabber
Facebook	400ล้าน	TCP

3.1 การแบ่งการทำงานตามลักษณะการทำงานระบบส่งข้อความด่วน

ลักษณะการทำงานของแอพลิเคชันประเภท Instant Messaging ทำงานด้วยการส่งผ่านข้อมูลและเมื่อพิจารณาลักษณะวิธีการส่งผ่านข้อมูลจากต้นทางไปยังปลายทางจะพบว่าสามารถจำแนกรูปแบบการทำงานทั่วไปของระบบการส่งข้อความด่วนได้ 3 รูปแบบดังนี้

3.1.1 การทำงานด้วยรูปแบบ Client-Server

Client-Server เป็นระบบเครือข่ายที่มีประสิทธิภาพสูง และมีการใช้งานกันอย่างกว้างขวางมากกว่าระบบเครือข่ายแบบอื่นที่มีในปัจจุบัน ระบบ Client/Server สามารถสนับสนุนให้มีเครื่องลูกข่ายได้เป็นจำนวนมาก และสามารถเชื่อมต่อกับเครื่องคอมพิวเตอร์ได้หลายแพลตฟอร์ม ระบบนี้จะทำงานโดยมีเครื่อง Server ที่ให้บริการ เป็นศูนย์กลางอย่างน้อย 1 เครื่อง และมีการบริหารจัดการทรัพยากรต่างๆ จากส่วนกลาง และควบคุมการให้บริการทรัพยากรต่างๆ นอกจากนี้เครื่องลูกข่ายยังจะต้องมีความสามารถในการประมวลผล และมีพื้นที่สำหรับจัดเก็บข้อมูลท้องถิ่นเป็นของตนเองอีกด้วย ระบบเครือข่ายแบบ Cleint/Server เป็นระบบที่มีความยืดหยุ่นสูง สนับสนุนการทำงานแบบ Multiprocessor สามารถเพิ่มขยายขนาดของจำนวนผู้ใช้ได้ตามต้องการ นอกจากนี้ยังสามารถเพิ่มจำนวนเครื่อง Servers สำหรับให้บริการต่างๆ เพื่อช่วยกระจายภาระของระบบได้ ส่วนข้อเสียของระบบนี้ก็คือ มีความยุ่งยากในการติดตั้งมากกว่าระบบ Peer-to-Peer รวมทั้งต้องมีการบริหารจัดการระบบโดยเฉพาะอีกด้วย

การทำงานของระบบการส่งข้อความด่วนแบบ Client-Server จะอาศัย Server เป็นตัวกลางเก็บเส้นทางการเชื่อมต่อไว้ โดยเมื่อ client เริ่มใช้งานจะต้องมีการลงทะเบียนและ update สถานะไว้กับ server หลักโดย client ในระบบนี้จะไม่มีการติดต่อสื่อสารข้อมูลกันโดยตรง ข้อดีของระบบนี้คือ การปลอมตัวตนทำได้ยาก การ update สถานะของ client ให้ client อื่นรับทราบตลอดจนเถียรภาพการส่งผ่านข้อมูลสูง แต่อย่างไรก็ตามการส่งผ่านข้อมูลระบบนี้ อาจมีต้นทุนด้านระยะเวลาเนื่องจากข้อมูลต้องส่งผ่านตัวกลาง ดังนั้นอาจมีปัญหากับข้อมูลประเภท streaming media

3.1.2 การทำงานด้วยรูปแบบ Peer to Peer

Peer to Peer คือสถาปัตยกรรมเครือข่ายหรือคอมพิวเตอร์แบบกระจาย ที่แบ่งส่วนภาระงานหรือปริมาณงานระหว่าง peer โดยแต่ละ peer จะมีศักดิ์และอำนาจเท่ากัน peer ต่างๆ จะเชื่อมต่อกันเป็นเครือข่ายและจะแบ่งจากทรัพยากรของตน เช่น พลังประมวลผล พื้นที่ดิสก์ หรือช่องสัญญาณเครือข่าย ให้กับผู้เข้าร่วมในเครือข่าย โดยไม่จำเป็นต้องมีการจัดการจากส่วนกลาง

peer เป็นทั้งผู้จัดหาและผู้บริโภคทรัพยกร ต่างกับโมเดลแบบ Client Server ซึ่งเซิร์ฟเวอร์จะเป็นผู้จัดหา (ส่ง) และไคลเอนต์จะเป็นผู้บริโภค (รับ) เท่านั้น อีกนัยหนึ่งสามารถเรียก Peer to peer เป็นซอฟท์แวร์ที่ปฏิบัติงานในรูปแบบสถาปัตยกรรมเครือข่ายแบบกระจาย (distributed application architecture) ที่สามารถแบ่งหน้าในการคำนวน หรือ ประมวลผลงานหนึ่งๆ ออกเป็นส่วนๆในแต่ละpeer หรือ โหนด ที่ทำงานร่วมกันภายในเครือข่าย แต่ละpeer หรือ โหนด ภายในเครือข่ายเดียวกัน จะมีความเท่าเทียมกันในเสมือนหนึ่งในสมาชิกใน แอพพลิเคชั่นนั้นๆ คือ อาจจะทำหน้าที่เป็น ผู้ให้ ผู้จัดหา (Server) และ ขณะเดียวกัน peer นั้นๆอาจทำหน้าที่เป็น ผู้ร้องขอ (Client) ได้เช่นกัน ทุกๆ peer สามารถแบ่งส่วนทรัพยากรที่มีอยู่ เช่น กำลังของการประมวลผลข้อมูล พื้นที่ของการจัดเก็บข้อมูล หรือ bandwidth โดยจัดสรรพื้นที่ และ กำลังการประมวลผล หรือแม้แต่ แชร์แบนด์วิธ ของโหนดตนเองให้กับสมาชิกอื่นๆ ที่อยู่ในเครือข่ายเดียวกัน โดยไม่ต้องถูกควบคุมด้วย Server ศูนย์กลาง

ระบบ Peer to peer ส่วนใหญ่จะถูกประยุกต์ใช้ปรับปรุงในรูปแบบของ overlay network ที่ถูกสร้างขึ้นด้านบนชั้นของ Peer to peer คือ ชั้นของ Application Layer และอยู่บนชั้นบนของการเชื่อมต่อเครือข่ายทางกายภาพ คือ การเชื่อมต่อ ด้วยสายเคเบิ้ล หรือ สัญญาณจากแซทเทิ้ลไลท์ ที่เรียกรวมกันว่า physical layer โดยจะทำหน้า เปรียบได้กับ สารบัญ หรือ ตัวบ่งชี้ และ peer จะทำหน้าที่ค้นหา จากข้อมูลที่ได้รับจึงทำให้ แต่ละ peer มีอิสระจากการเชื่อมต่อทางกายภาพ อย่างไรก็ตามระบบ Peer to Peer สามารถถูกสร้างให้มีระเบียบได้ เรียกว่า Structured Peer to Peer Network โดยแต่ละpeer จะถูกจัดการให้ปฏิบัติตามเงื่อนไขเฉพาะเจาะจง ด้วยอัลกอริทึม ที่จะก่อให้เกิดโครงสร้างใหม่ ที่ถือกำเนิดอยู่ชั้นโอเวอร์เลย์ ที่โครงสร้างนั้นๆ จะมีความเป็นระเบียบเรียบร้อยตรงตามเงื่อนไขของอัลกอริทึม และ มีรูปร่าง Topologies และ คุณสมบัติที่เป็นไปตามเงื่อนไขนั้นๆเช่นเดียวกัน โดยปกติ ที่เห็นอยู่ทั่วไปโครงสร้าง Peer to Peer ส่วนใหญ่จะถูกสร้างบนพื้นฐานของระบบดัชนี distributed hash table-based (DHT) เช่น Chord system หรือ เครือข่าย Unstructured peer-to-peer ที่ไม่ได้จัดการให้เป็นรูปร่าง หรือ โครงสร้างเฉพาะเจาะจง ที่แต่ละโหนดในเครือข่ายจะไม่ถูกจัดการ โดยแต่ละ peer หรือ โหนด ถูกปล่อยให้เป็นอิสระในการเชื่อมโยงเครือข่าย

การทำงานของระบบการส่งข้อความด่วนโดยอาศัยการทำงานรูปแบบ Peer to Peer จะไม่อาศัย Server ตัวกลางใดในการเชื่อมต่อและส่งผ่านข้อมูล การรับรู้สถานะและเส้นทางของ Peer ต่างๆสามารถทำได้เนื่องจาก Peer ทุก Peer มีหน้าที่กระจาย package ระบุตัวตนและสถานะออกไปในวงเน็ทเวอร์กเป็นระยะ จุดเด่นของระบบนี้คือความรวดเร็วและความทันสมัยของข้อมูลที่ดีกว่าระบบ Client Server แต่มีข้อด้อยด้านความปลอดภัยของข้อมูล

3.1.3 การทำงานด้วยรูปแบบผสม (Hybrid)

เป็นการนำเอาจุดเด่นหรือข้อดีทั้งสองแบบมาใช้ร่วมกันโดยอาศัยการ update สถานะและเส้นทางจาก Server ศูนย์กลาง แต่การส่งข้อมูลระหว่าง Client จะทำกันโดยตรงดังนั้นจึงสามารถใช้ข้อดีของการจัดสรรทรัพยากรณ์และการรักษาความปลอดภัยรวมถึงความรวดเร็วในการส่งข้อมูล ลักษณะการทำงานในรูปแบบนี้ถูกใช้งานในหลายแอพลิเคชันในปัจจุบันตัวอย่างของระบบ hybrid โดนเด่นและเป็นที่นิยม คือ ระบบ file sharing Napster โดยตัวอย่างของรูปแบบ peer to peer Centralized ตัวอย่างอื่นๆ ได้แก่ Freenet, gnutella และเครือข่าย Kazaa ก็เป็นตัวอย่างของระบบในรูปแบบ hybrid เช่นกัน

4. บทความวิจัยที่เกี่ยวข้องกับงานวิจัย

ในศึกษาก่อนการทำงานวิจัยนี้ได้ทำการวิเคราะห์บทความที่เกี่ยวข้องหรือมีวิธีการทำการวิจัยใกล้เคียงกับการวิจัยที่ได้พัฒนาขึ้น เพื่อนำมาเป็นแนวทางในการวางแผนงานวิจัย ประกอบด้วย

4.1 A comparison of TCP and SCTP performance using the HTTP protocol

การวิจัยนี้มีแนวความคิดที่จะวัดผลการส่งข้อมูลแบบ HTTP บนโปรโตรคอล TCP และโพรโทคอล SCTP และคาดว่าจะสามารถเพิ่มประสิทธิภาพการทำงานของการส่งข้อมูลแบบ HTTP ด้วยคุณสมบัติ Multi-streaming ของโพรโทคอล SCTP ได้

วิธีการทำการวิจัย

กำหนดจำนวน packet สูญหาย 0% และ 10%
กำหนดการใช้งาน Multi-Streaming 1, 25 และ 50 เส้นทาง
กำหนดขนาดข้อมูล, การร้องขอ, ความถี่การร้องขอคงที่

ทำการวัดผล 2 รูปแบบ

การส่งแบบเส้นทางเดียวและการสูญหายของข้อมูล 0%, 10%
ส่งแบบหลายเส้นทางและการสูญหายของข้อมูล 0%, 10%
ทำการวัดผลด้วยปริมาณการส่งผ่านข้อมูล Troughput

คาดหวังผลการทดลองด้วยคุณสมบัติ Multi Stream ของโพรโทคอล SCTP

SCTP ควรมีประสิทธิภาพดีกว่า TCP เมื่อเพิ่มขนาดข้อมูล
SCTP ควรมีประสิทธิภาพดีกว่า TCP เมื่อมีการสูญหายของข้อมูล
SCTP ควรมีประสิทธิภาพดีกว่า TCP เมื่อมีการส่งหลายเส้นทาง

ตารางที่ 3 แสดงปริมาณการส่งผ่านข้อมูลด้วยโพรโทคอล TCP

ขนาด File	ไม่ Drop Packets	Drop 10^{th} Packets	ประสิทธิภาพต่าง
100 bytes	10.2	9.04	-11.4%
1 KB	33.2	21.0	-36.7%
10 KB	262.7	159.68	-39.2%
100 KB	2556.82	1945.28	-23.9%
1 MB	26055.88	10051.76	-61.4%
10 MB	155750.18	95392.34	-38.8%

ตารางที่ 4 แสดงปริมาณการส่งผ่านข้อมูลด้วยโพรโทคอล STCP แบบ 1 Stream

ขนาด File	ไม่ Drop Packets	Drop 10^{th} Packets	ประสิทธิภาพต่าง
100 bytes	10.2	2.44	-76.1%
1 KB	33.2	7.26	-78.1%
10 KB	262.6	27.0	-89.7%
100 KB	2554.1	786.96	-69.2%
1 MB	25855.6	5301.48	-79.5%
10 MB	35203.12	13936.1	-60.4%

ตารางที่ 5 แสดงปริมาณการส่งผ่านข้อมูลด้วยโพรโทคอล STCP แบบ 10 Stream

ขนาด File	ไม่ Drop Packets	Drop 10th Packets	ประสิทธิภาพต่าง
100 bytes	122.6	5.16	-95.8%
1 KB	399.06	13.38	-96.6%
10 KB	3120.52	162.58	-94.8%
100 KB	4273.96	1038.58	-75.7%
1 MB	31169.0	2185.44	-93.1%
10 MB	50844.4	15737.22	-69.0%

ตารางที่ 6 แสดงปริมาณการส่งผ่านข้อมูลด้วยโพรโทคอล STCP แบบ 100 Stream

ขนาด File	ไม่ Drop Packets	Drop 10th Packets	ประสิทธิภาพต่าง
100 bytes	944.36	181.38	-80.1%
1 KB	63.5	154.42	+43.2%
10 KB	1796.5	127.175	-92.3%
100 KB	16393.72	208.62	-98.7%
1 MB	38824.74	196.6	-99.5%
10 MB	47137.08	15474.63	-67.2%

ผลการทดสอบของการวิจัยนี้บ่งชัดว่าการทำงานของโพรโทคอล SCTP ไม่ให้ประสิทธิภาพที่ดีกว่าเมื่อเปรียบเทียบกับโพรโทคอล TCP ซึ่งในการวิเคราห์ให้ความคิดเห็นว่าประสิทธิภาพดังกล่าวน่าจะเกิดจากวิธีการทำงานของตัว Browser ซึ่งไม่ได้ออกแบบมาเพื่อรองรับการทำงานแบบ Multi Stream ของโพรโทคอล SCTP นอกจากนี้ยังสรุปว่าการใช้งานโพรโทคอล SCTP ที่ 10 ช่องทางจะเหมาะสมกับการทำงานแบบ HTTP ที่สุด

4.2 SCTP versus TCP Comparing the Performance of Transport Protocols for Web Traffic

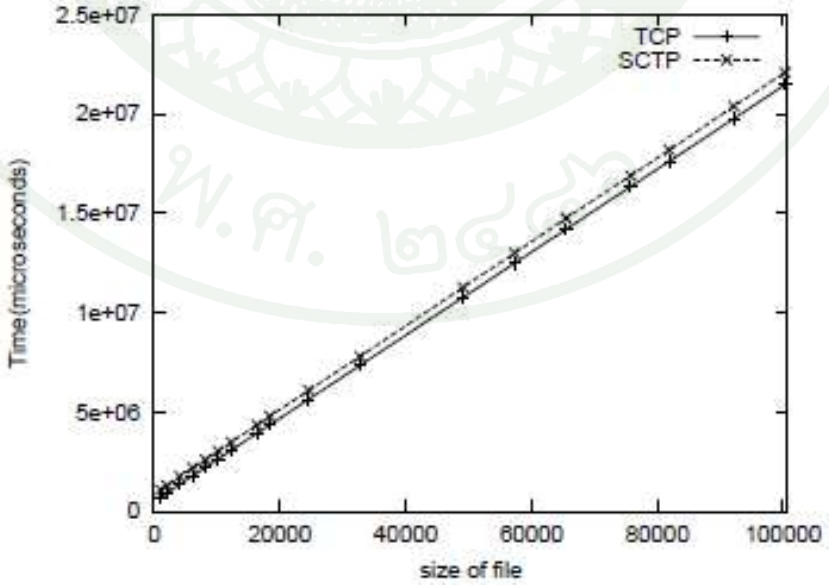
บทความการทำการวิจัยชิ้นนี้มีแนวคิดถึงการใช้งานโพรโทคอล SCTP ว่าการส่งข้อมูลแบบ Multi-Stream ควรจะมีความเร็วกว่าแบบเส้นทางเดียว และการสูญหายของข้อมูลมีผลต่อความเร็วในการรับส่งข้อมูล

วิธีการทำการวิจัย

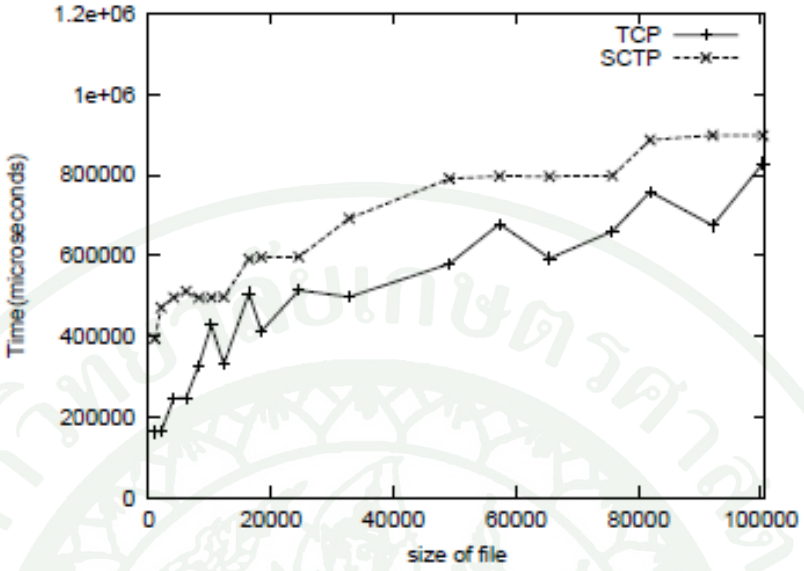
กำหนด RTT ที่ 80 ms
กำหนดค่าการสูญหายที่ 0 – 25%
กำหนด Bandwidth 40, 400 Kbps, 3 และ 10 Mbps

ทำการวัดผล 2 รูปแบบ

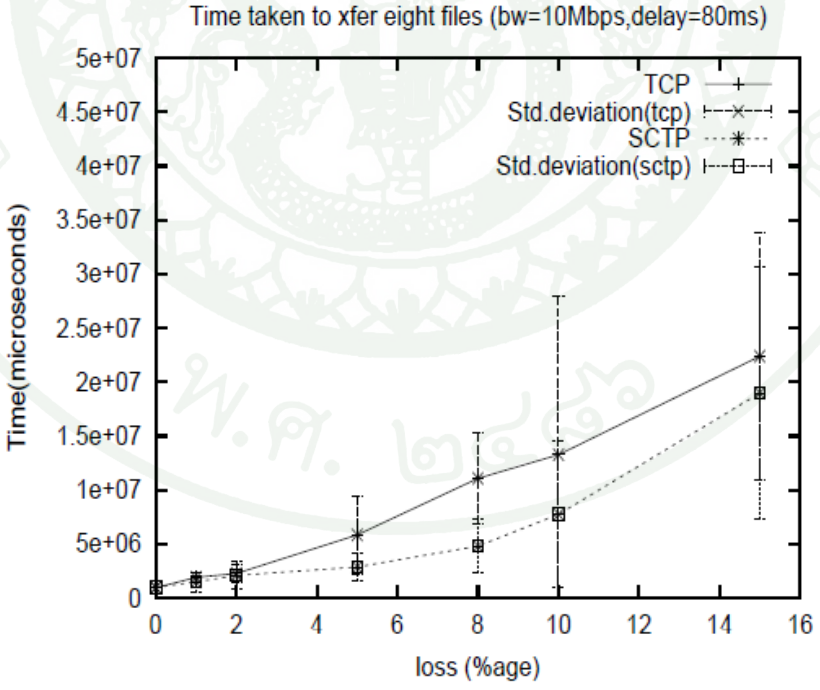
การส่งแบบไฟล์เดียวและไม่มีการสูญหายของข้อมูล
การส่งแบบหลายไฟล์และมีการสูญหายของข้อมูล



ภาพที่ 12 เปรียบเทียบการทำงานโพรโทคอล TCP และ SCTP จำกัด Bandwidth



ภาพที่ 13 เปรียบเทียบการทำงานโพรโทคอล TCP และ SCTP ไม่จำกัด Bandwidth



ภาพที่ 14 เปรียบเทียบการทำงานโพรโทคอล TCP และ SCTP มีข้อมูลสูญหาย

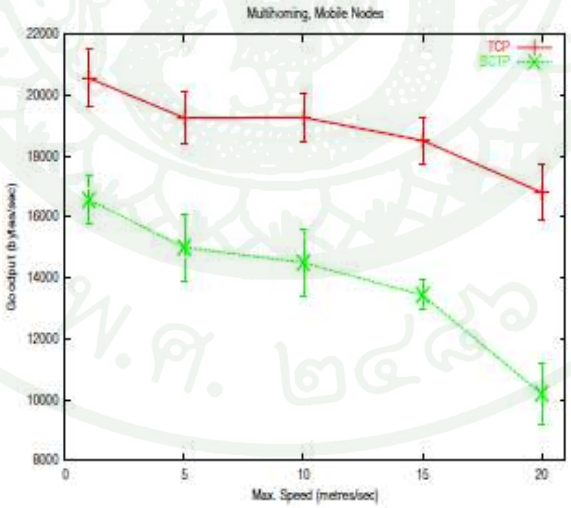
ผลการทดสอบการทำงานระหว่างโพรโทคอล TCP และ โพรโทคอล SCTP พบว่า

โพรโทคอลTCP เร็วกว่าเมื่อไม่มีการสูญหาย (SCTP ทำงาน user space)
โพรโทคอล SCTP ทำงานเร็วกว่า TCP เมื่อมีการสูญหายข้อมูลมากขึ้น

4.3 SCTP vs TCP : Performance Comparison in MANETs

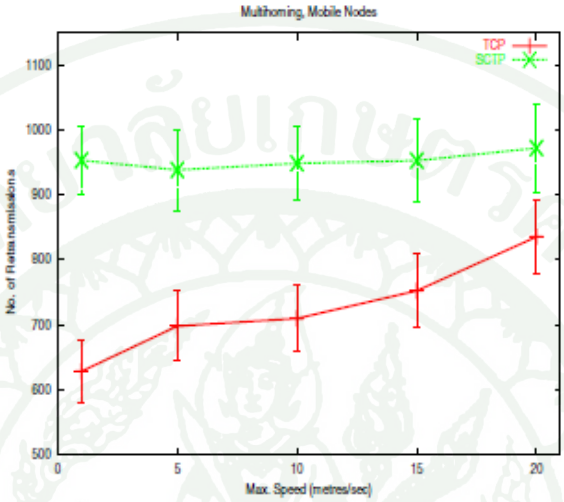
บทความนี้ทำการทดสอบตามแนวคิดว่า การส่งข้อมูลบนระบบ MANETs มีประสิทธิภาพเท่ากันเสมอไม่ว่าจะเกิดบนโพรโทคอลใด และการส่งผ่านข้อมูลบนตัวระบบ MANETs จะไม่มีกำหนดข้อจำกัดของ Bandwidth

ในส่วนแรกกำหนดไม่ใช้ Multihoming เพื่อลดการได้เปรียบจากการส่งผ่านข้อมูลหลายเส้นทางของโพรโทคอล SCTP พบว่าการทำงานด้วยช่องทางเพียงช่องทางเดียว การทำงานบนโพรโทคอล TCP ให้ผลการทดสอบที่ดีกว่า



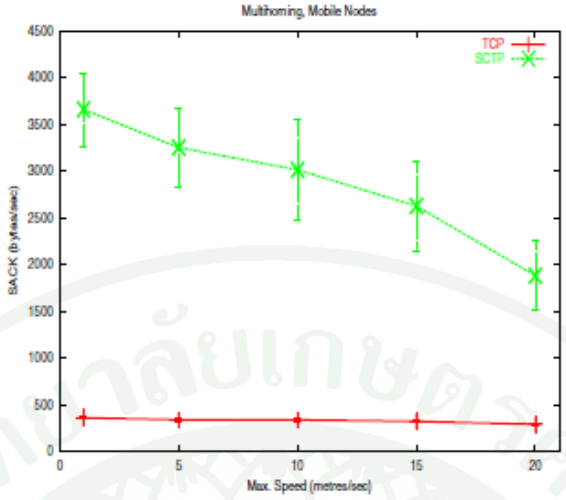
ภาพที่ 15 แสดงการทำงาน 1 ช่องทางบนระบบ MANETs

ต่อมาได้กำหนดให้ข้อมูลสูญหายและทำการส่งซ้ำและทำการเปรียบเทียบเวลาที่ใช้ทั้งหมดโดยไม่มีการลบ Cost ใดออกได้ผลการทดสอบว่า โพรโทคอล SCTP ทำให้สารส่งผ่านข้อมูลเร็วขึ้นกว่าโพรโทคอล TCP



ภาพที่ 16 เปรียบเทียบเวลาที่ใช้ส่งข้อมูลซ้ำด้วยโพรโทคอล TCP และ โพรโทคอล SCTP

ทำการทดสอบโดยกำหนดให้โพรโทคอล SCTP ใช้ SACK Bandwidth เพื่อลดข้อได้เปรียบของโพรโทคอล TCP ในการสถาปนาการเชื่อมต่อแบบ 3 ways hand shake ได้ผลการทดสอบตามที่คาดไว้ นั่นคือ การส่งผ่านข้อมูลด้วยโพรโทคอล SCTP สามารถทำงานได้รวดเร็วกว่าโพรโทคอล TCP



ภาพที่ 17 เปรียบเทียบการส่งข้อมูลด้วยโพรโทคอล TCP และ SCTP SACK Bandwidth

อุปกรณ์และวิธีการ

อุปกรณ์

ในส่วนของ Hardware ประกอบด้วย Server ซึ่งทำงานแบบสิ่งแวดล้อมเสมือน (Virtual Machine) โดยมี Server เสมือนจำนวนหนึ่ง (Guest OS) และอุปกรณ์เครือข่ายเสมือน (Virtual Switch) ทำงานอยู่บนระบบดังกล่าว และนอกจากนี้ยังมีการนำอุปกรณ์เครือข่าย (Switch Layer 3) มาใช้ร่วมทดสอบด้วย

Server (Dell R710) Specification

CPU Xeon® 4core 2.4GHz * 2

RAM 128 G

OS VMWARE ESXI 4.1

- Free BSD Linux (Guest OS)
- CentOS Linux (Guest OS)
- Windows 2003 (Guest OS)
- WindowsXP (Guest OS)
- VM Switch (Virtual Switch)

Switch L3 Manage (3COM 5500G) Specification

48 port 1000BASE-T

232.0 Gb/s switching capacity

8 hardware queues per port

IEEE 802.1p (CoS/QoS)

Remarking of packets based on priority:

- Type of Service (ToS)
- IEEE 802.1p CoS
- IP precedence

- Physical port
- Source/destination MAC address
- VLAN information
- Ethertype
- Source/destination IP address
- Source/destination TCP port
- Source/destination UDP port

Traffic redirection

Time-based Access Control Lists (ACLs)

Auto-prioritization

Weighted Round Robin (WRR), including WRR+SP

Strict Priority Queuing (SP)

DiffServ Code Point Expedited Forwarding (DSCP EF)

Application rate limiting and blocking on ingress

Port-based traffic shaping on egress

IEEE 802.3af Power over Ethernet standards-compliant

ในส่วนของซอฟต์แวร์ประกอบด้วยส่วนที่เป็น Source Code ซึ่งจะนำมาปรับปรุงใช้งานแอพลิเคชันที่ใช้ควบคุมและสร้างสภาวะเครือข่าย และระบบจัดเก็บข้อมูลการทำงาน

ตารางที่ 7 แสดงซอฟต์แวร์ที่ใช้ในการทำวิจัย

ซอฟต์แวร์	ประเภท	วัตถุประสงค์
Open Source SCTP (LKSCTP)	Open Source Module	ใช้ปรับปรุงระบบการทำงานระบบรับส่งข้อความด่วนให้สามารถใช้โพรโทคอล SCTP ในการทำงาน
Client Application (Delphi)	Open Source Application	ระบบรับข้อความด่วนบนระบบปฏิบัติการ Windows
Client Application (C compiler)	Open Source Application	ระบบรับข้อความด่วนบนระบบปฏิบัติการ Linux
Server Application (C compiler)	Open Source Application	ระบบรับส่งข้อความด่วนบนระบบปฏิบัติการ Linux
Instant Messenger (Open Source)	Open Source Application	ต้นแบบระบบรับข้อความด่วนบนระบบปฏิบัติการ Linux
Dummy Net	Network Generator	ระบบจำลองการสื่อสารข้อมูลบนเครือข่าย ระบบปฏิบัติการ FreeBSD หรือ Linux
Traffic Shapeing	Network Control	ระบบการจำกัดช่องทางการใช้งาน (Bandwidth) บนระบบปฏิบัติการ FreeBSD
Traffic Generator	Network Generater	ระบบจำลองการสื่อสารข้อมูลบนเครือข่าย ระบบปฏิบัติการ Windows
Squid	Log Keeper	ระบบจัดเก็บข้อมูลการจราจรบนเครือข่าย ระบบปฏิบัติการ Linux
Traffic Logger	Log Keeper	ระบบจัดเก็บข้อมูลการจราจรบนเครือข่ายโดยอาศัยข้อมูลจาก Switch Layer 3 บนระบบ 3COM Network

วิธีการ

ผู้วิจัยได้ดำเนินการทดลอง โดยพัฒนาโปรแกรม Open source Instant Messenger แก้ไขเพิ่มเติมด้วย LKSCTP (Open Source SCTP) เพื่อให้เปลี่ยนแปลงวิธีการส่งข้อมูลจาก TCP โพรโทคอลไปเป็น SCTP โปรโคคอลโดยใช้คุณสมบัติ Multi Streaming และทำการทดสอบวัดผลจากระยะเวลาในการส่งชุดข้อมูลเสร็จสิ้นสมบูรณ์โดยแบ่งวิธีการดำเนินงาน 2 ช่วง โดยในการทำการทดสอบส่วนแรกต้องการทำการทดสอบประสิทธิภาพโดยรวมระหว่างการทำงานโพรโทคอล TCP และโพรโทคอล SCTPในสภาวะแวดล้อมควบคุม และในการทำการทดสอบส่วนที่สองต้องการทำการวัดผลการส่งชุดข้อมูลระหว่างโพรโทคอล TCP และโพรโทคอล SCTP ด้วยโปรแกรม Instane Messanger ซึ่งทำงานในรูปแบบ Client Server, Peer to Peer และ Hybrid

ในการปรับปรุงให้โปรแกรม Instant Messanger สามารถทำงานด้วยโพรโทคอล SCTP ต้องทำการเพิ่มเติม Library ของ SCTP ให้กับ Source Code โปรแกรม Instant Messanger และปรับเปลี่ยนวิธีการเชื่อมต่อระหว่างโปรแกรมให้สามารถเลือกได้ระหว่างโพรโทคอล TCP และโพรโทคอล SCTP โดยส่วน code ที่มีความสำคัญประกอบด้วย

การสร้างการเชื่อมต่อระหว่าง application กับโพรโทคอลที่ใช้รับส่งข้อมูล หรือที่เรียกว่า Create SCTP TCP-Style Socket ด้วยชุดคำสั่ง

```
listenSock = socket( AF_INET, SOCK_STREAM, IPPROTO_SCTP );
```

การตอบรับการเชื่อมต่อระหว่าง application ด้วยโพรโทคอล SCTP ที่กำหนดผ่านทาง socket ที่สร้างไว้ หรือที่เรียกว่า Accept connections ด้วยชุดคำสั่ง

```
bzero( (void *)&servaddr, sizeof(servaddr) );
```

```
servaddr.sin_family = AF_INET;

servaddr.sin_addr.s_addr = htonl( INADDR_ANY );

servaddr.sin_port = htons(MY_PORT_NUM);
```

การสร้างช่องทางการสื่อสารข้อมูลของ SCTP โพรโทคอล โดยตัวโพรโทคอล SCTP เองสามารถสร้างช่องทาง (Stream) ได้ 131,070 ช่องทาง แบ่งเป็นช่องทางรับข้อมูลและช่องทางส่งข้อมูลอย่างละ 65,535 เส้นทาง สำหรับชุดคำสั่งที่นำมาแสดงจะสร้างช่องทาง 5 เส้นทาง ตามชุดคำสั่งดังนี้

```
/* Specify that a maximum of 5 streams will be available per socket */

memset( &initmsg, 0, sizeof(initmsg) );

initmsg.sinit_num_ostreams = 5;

initmsg.sinit_max_instreams = 5;

initmsg.sinit_max_attempts = 4;

ret = setsockopt( connSock, IPPROTO_SCTP,
SCTP_INITMSG,&initmsg,

sizeof(initmsg) );
```

การส่งข้อมูลเบื้องต้นในการทำงานผ่านช่องทางแรก (Stream 0) โดยในช่องทางนี้จะทำการส่งข้อมูลเวลาของ Server (local time stream) ด้วยชุดคำสั่ง

```
snprintf( buffer, MAX_BUFFER, "%s\n", ctime(&currentTime) );
```

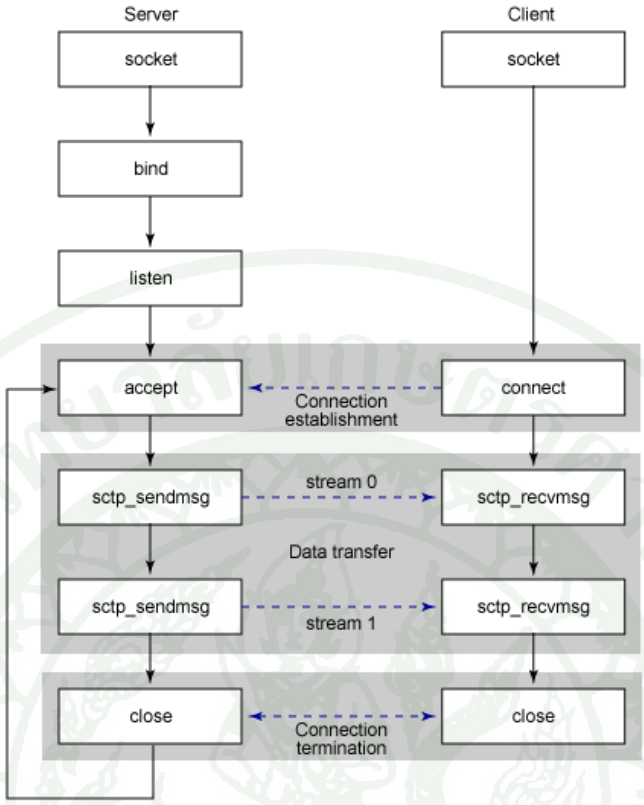
```
ret = sctp_sendmsg( connSock,(void *)buffer,(size_t)strlen(buffer),
NULL, 0, 0, 0, LOCALTIME_STREAM, 0, 0 );
```

การส่งข้อมูลเบื้องต้นในการทำงานผ่านช่องทางที่สอง (Stream 1) โดยในช่องทางนี้จะทำการส่งข้อมูล GMT ของชุดข้อมูล (GMT stream) ด้วยชุดคำสั่ง

```
snprintf( buffer, MAX_BUFFER, "%s\n",asctime( gmtime(
&currentTime ) ) );
ret = sctp_sendmsg( connSock,(void *)buffer, (size_t)strlen
(buffer),NULL, 0, 0, 0, GMT_STREAM, 0, 0 );
```

การตัดการเชื่อมต่อการส่งข้อมูลระหว่าง application ภายหลังจากการส่งขช้อมูลเสร็จสิ้นด้วยชุดคำสั่ง

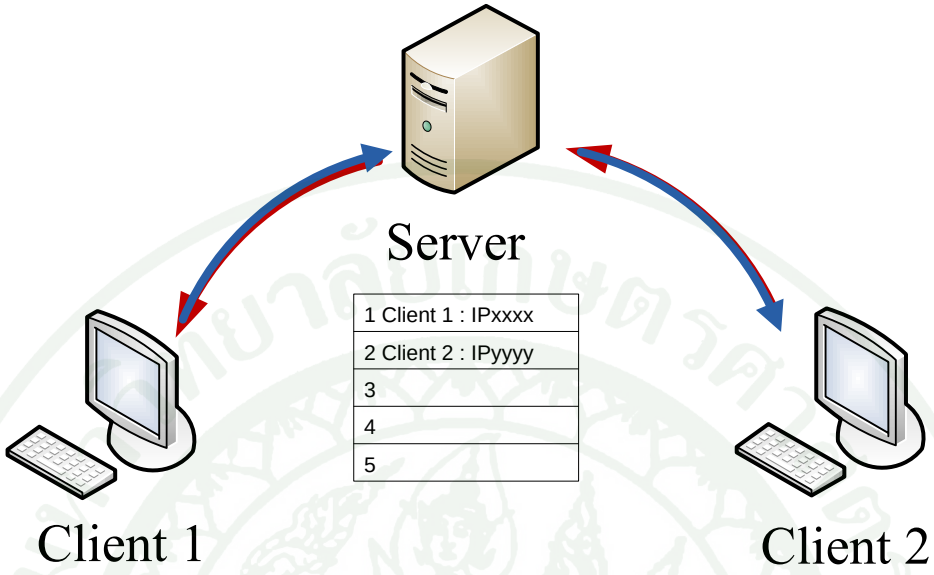
```
close( connSock );
```



ภาพที่ 18 แสดงโครงสร้างการทำงานของ Instant Messager ด้วย SCTP โพรโทคอล

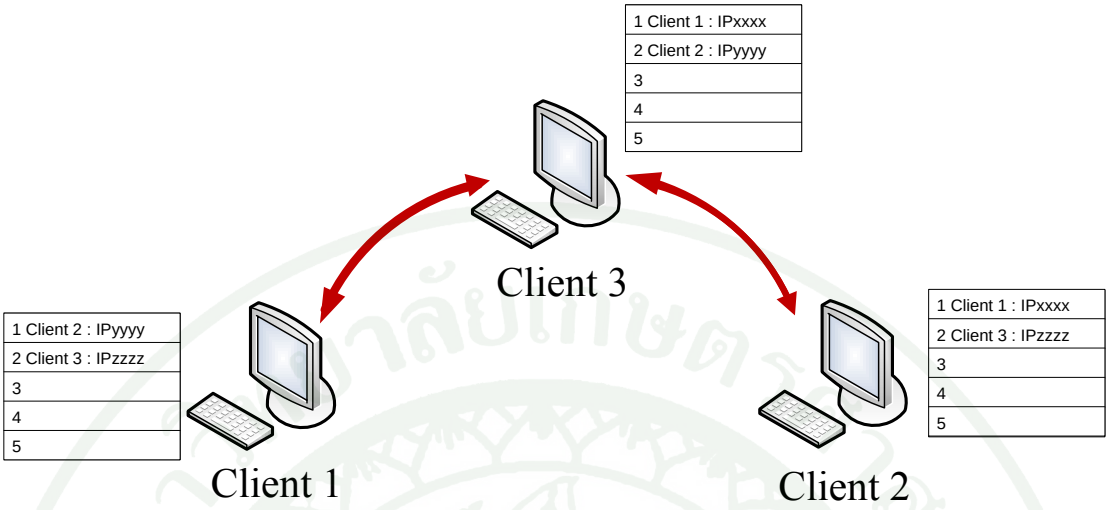
การปรับการทำงานของ Open Source Instant Messanger นอกจากการปรับเลือกใช้โพรโทคอล SCTP แล้ว การวิจัยนี้ยังปรับปรุงให้สามารถทำงานแบบ Client Server แบบ Peer to Peer และ Hybrid อีกด้วยเพื่อศึกษาประสิทธิภาพจากการทำงานในรูปแบบดังกล่าว

การปรับการทำงานของ Open Source Instant Messanger ให้สามารถทำงานแบบ Client Server ทำโดยกำหนดให้ตัว Instant Messanger 1 node ทำหน้าที่เป็น Server และให้ Instant Messanger node อื่นๆทำการลงทะเบียนที่อยู่ (Address) กับ node แรกด้วยโพรโทคอล UDP ดั้งเดิมของตัว Instant Messanger เอง เมื่อต้องการส่งข้อมูลระหว่าง node (Client ในระบบ Client Server) ผู้ส่งจะทำการสอบถามสถานะจาก node (Server) ด้วยโพรโทคอล UDP ดั้งเดิมเช่นกัน และเมื่อเริ่มทำการส่งข้อมูลจึงทำการเชื่อมต่อช่องทางไปยังผู้รับผ่านตัว Server ด้วยโพรโทคอล TCP และโพรโทคอล SCTP ต่อไป



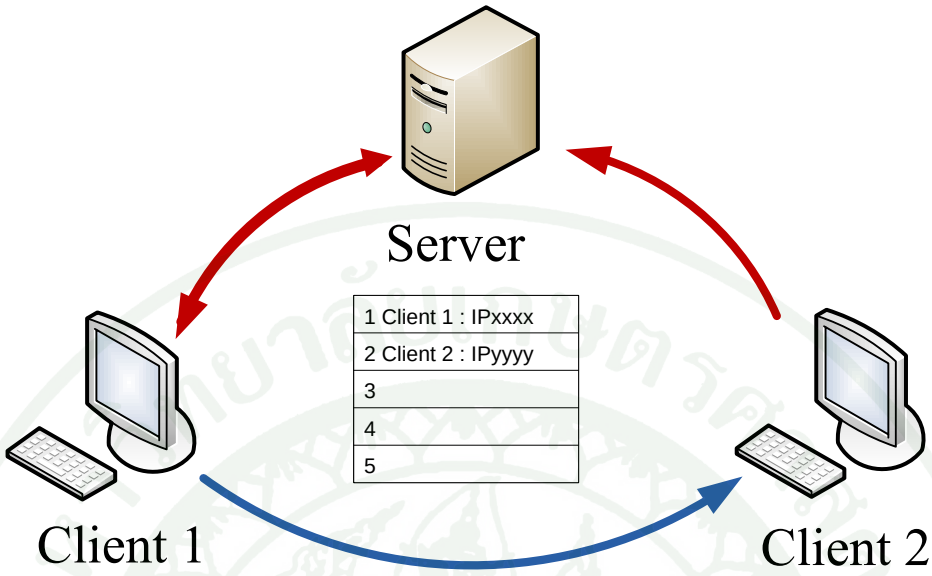
ภาพที่ 19 แสดงโครงสร้างการทำงาน Instant Messanger แบบ Client Server

การปรับการทำงานของ Open Source Instant Messanger ให้สามารถทำงานแบบ Peer to Peer ทำโดยกำหนดให้ตัว Instant Messanger ทุก node ทำหน้าที่เก็บสถานะและที่อยู่ (Address) และให้ Instant Messanger ทุก node ทำการลงทะเบียนที่อยู่ (Address) กับ node อื่นๆด้วยการ Broadcast ด้วยโพรโทคอล UDP เมื่อต้องการส่งข้อมูลระหว่าง node (Peer ในระบบ Peer to Peer) ผู้ส่งจะทำการตรวจสอบที่อยู่และสถานะที่เก็บไว้ และเมื่อเริ่มทำการส่งข้อมูลจึงทำการเชื่อมต่อช่องทางไปยัง Peer ผู้รับโดยตรงโดยอาศัยโพรโทคอล TCP หรือโพรโทคอล SCTP



ภาพที่ 20 แสดงโครงสร้างการทำงาน Instant Messanger แบบ Peer to Peer

การปรับการทำงานของ Open Source Instant Messanger ให้สามารถทำงานแบบ Hybrid ทำโดยกำหนดให้ตัว Instant Messanger 1 node ทำหน้าที่เป็น Server และให้ Instant Messanger node อื่นๆทำการลงทะเบียนที่อยู่ (Address) กับ node แรกด้วยโพรโทคอล UDP ดั้งเดิม ของตัว Instant Messanger เอง เมื่อต้องการส่งข้อมูลระหว่าง node (Client ในระบบ Client Server) ผู้ส่งจะทำการสอบถามสถานะจาก node (Server) ด้วยโพรโทคอล UDP ดั้งเดิมเช่นกัน และเมื่อเริ่มทำการส่งข้อมูลจึงทำการเชื่อมต่อช่องทางไปยังผู้รับโดยตรงโดยอาศัยโพรโทคอล TCP หรือโพรโทคอล SCTP



ภาพที่ 21 แสดงโครงสร้างการทำงาน Instant Messanger แบบ Hybrid

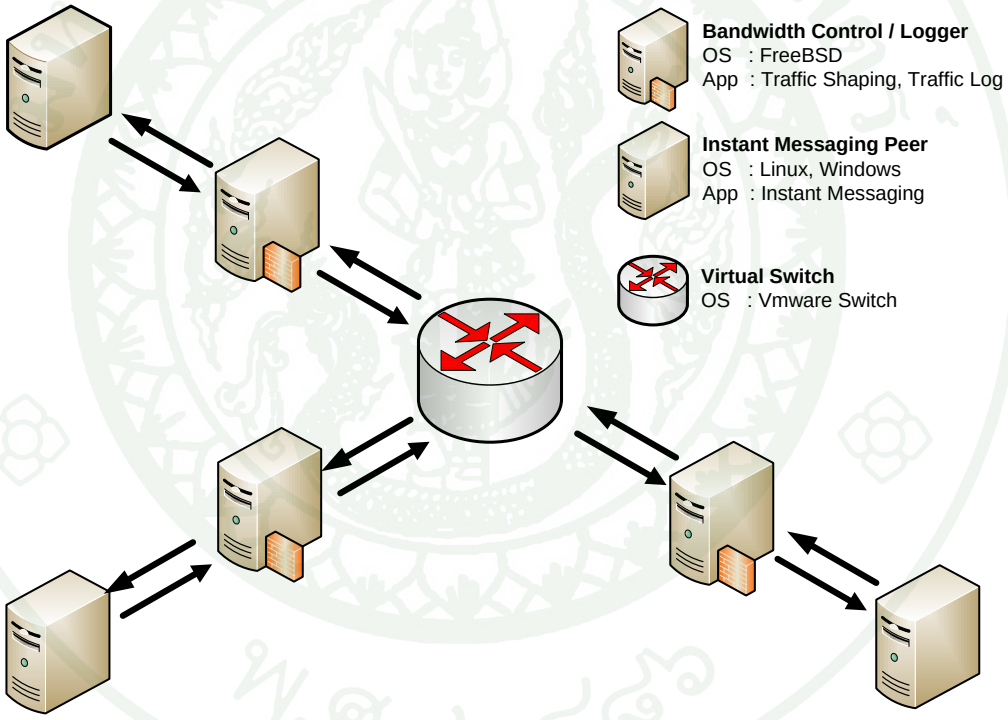
การวัดผลในเบื้องต้นต้องทำการปรับเทียบเวลาระหว่าง Client และ Server ให้ตรงกันด้วย Aplication SyncTime และบันทึกเวลา Time Stamp ของ Server และ Client ทั้งเวลาส่ง packet, เวลาที่ได้รับ packet ผลที่ได้สามารถชี้วัดได้ถึง ระยะเวลาที่ใช้ในการส่งข้อมูลทั้งหมด ตลอดจนระยะเวลาที่ข้อมูลใช้ในการเดินทาง (Cost)

Cost = TimeStamp(Packet Send) – Timestamp(Packet Receive)

การชี้วัดผลที่ใช้ในการทำการวิจัยนี้จะทำการชี้วัดโดยใช้ระยะเวลาที่ถูกใช้เพื่อส่งผ่านข้อมูลทั้งชุดไปยังจุดหมาย (Client หรือ Peer) โดยตัดภาระอันเกิดจากการเดินทางของข้อมูล (Cost) ออกไปเนื่องจาก ภาระดังกล่าวเมื่อทดสอบกับเครือข่ายหลายเครื่องภาระอันเกิดจากการเดินทางของข้อมูล (Cost) ผ่านเส้นทางต่างกันจะมีค่าไม่เท่ากัน อาจทำให้ผลการทดสอบผิดพลาดได้

Time = TimeStamp(Last Packet Receive) - TimeStamp(1st Packet Send) - Cost

ในการทำการทดสอบส่วนแรกจะทำการทดสอบประสิทธิภาพโดยรวมระหว่างการทำงานโพรโทคอล TCP และโพรโทคอล SCTP ในสภาพแวดล้อมที่เป็น Virtual Machine (VMware) โดยใช้ โปรแกรม Traffic Generator และ Dummy Net สร้างการจราจรบนเครือข่ายและ FreeBSD ซึ่งเป็น Transperent Proxy ร่วมกับปลักอิน Traffic Chapper ทำหน้าที่ควบคุม Bandwidth และเก็บการจราจรซึ่งมีการบันทึก Timestamp ไว้และนำมาวิเคราะห์ต่อไป



ภาพที่ 22 แสดงโครงสร้างการทำงานภายในระบบทดสอบที่ 1

1. ทำการทดสอบในระบบปิด ไม่มีปัจจัยภายนอกรบกวนโดยกำหนดให้ทำการทดสอบกับข้อมูลตัวอักษร (Text Files) และชุดข้อมูลแบบบีบอัด (Compress Files) บนระบบ Instant Messanger ที่ทำงานแบบ Peer to Peer และได้ทำการเชื่อมต่อกันไว้แล้วเพื่อหลีกเลี่ยงปัจจัยอันเกิดจากภาระการสถาปนาการเชื่อมต่อ

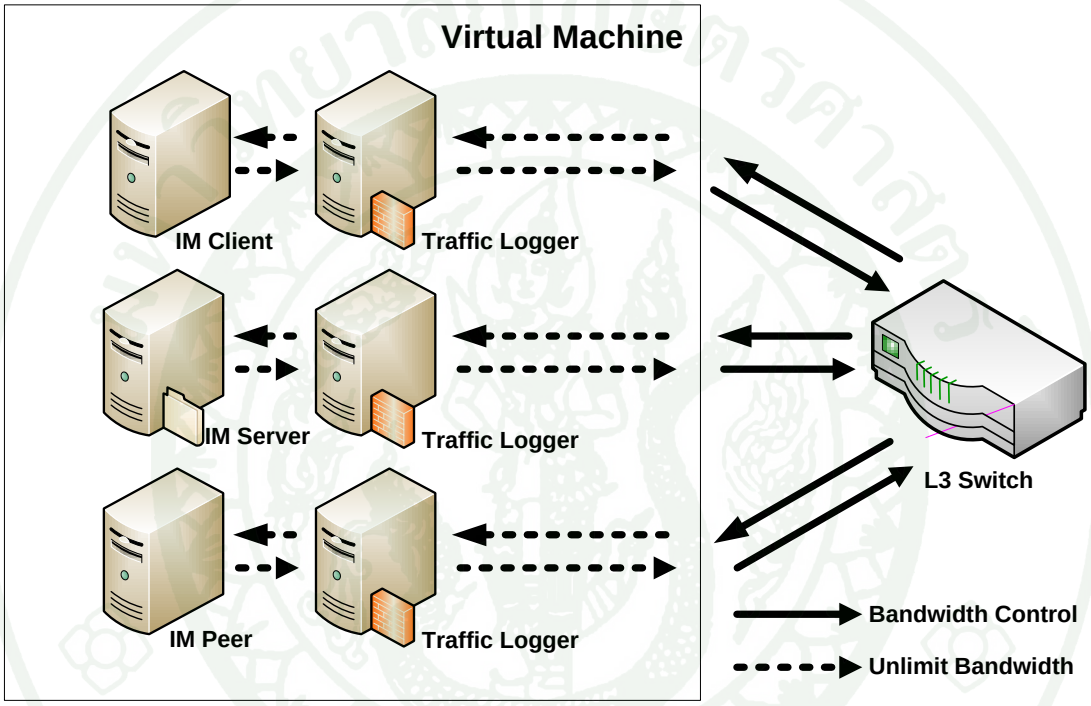
- กำหนดขนาด Bandwidth 1 Mbps
- ทดสอบกับ Text files ขนาด 1-64K byte
- ทดสอบกับ Random Compress files ขนาด 32-64K byte
- กำหนดให้ความหนาแน่นเครือข่าย 0%
- กำหนดข้อมูลสูญหายระดับ 0%

2. ทำการทดสอบในระบบเปิดมีปัจจัยภายนอกรบกวนโดยกำหนดให้ทำการทดสอบกับข้อมูลตัวอักษร (Text Files) และชุดข้อมูลแบบบีบอัด (Compress Files) บนระบบ Instant Messanger ที่ทำงานแบบ Peer to Peer และได้ทำการเชื่อมต่อกันไว้แล้วเพื่อหลีกเลี่ยงปัจจัยอันเกิดจากภาระการสถาปนาการเชื่อมต่อ โดยตัว Proxy ได้เพิ่มเติมการทำหน้าที่ Reject Packet แบบสุ่มเพื่อจำลองการสูญหายของข้อมูล

- กำหนดขนาด Bandwidth 1 Mbps
- ทดสอบกับ Text files ขนาด 1 - 64K byte
- ทดสอบกับ Random Compress files ขนาด 32 - 64K byte
- กำหนดให้ความหนาแน่นเครือข่าย 10 - 60%
- กำหนดข้อมูลสูญหายหลายระดับ 1 - 10%

ในการทำการทดสอบส่วนที่สองจะทำการทดสอบประสิทธิภาพโดยรวมระหว่างการทำงานโพรโทคอล TCP และโพรโทคอล SCTP ในสภาพแวดล้อมที่เป็น Virtual Machine (VMware) โดยใช้ โปรแกรม Traffic Generator และ Dummy Net สร้างการจราจรบนเครือข่ายและ

Free BSD เก็บการจราจรซึ่งมีการบันทึก Timestamp ไว้เพื่อนำมาวิเคราะห์ นอกจากนี้มีการนำเอา Switch Layer 3 Management มาใช้เพื่อปรับปรุงประสิทธิภาพในการควบคุม Bandwidth แทน Transperent Proxy ซึ่งมีการทำงานแบบ Store and Forward โดยตัว Proxy ได้เพิ่มเติมการทำหน้าที่ Reject Packet แบบสุ่มเพื่อจำลองการสูญหายของข้อมูล



ภาพที่ 23 แสดงโครงสร้างการทำงานภายในระบบทดสอบที่ 2

1. ทำการทดสอบในระบบปิด ไม่มีปัจจัยภายนอกรบกวนโดยกำหนดให้ทำการทดสอบกับชุดข้อมูลแบบบีบอัด (Compress Files) และเปรียบเทียบบนระบบ Instant Messanger ที่ทำงานแบบ Client Server, Peer to Peer และ Hybrid

- กำหนดขนาด Bandwidth 1 Mbps
- ทดสอบกับ Random Compress files ขนาด 128 - 512K bytes

- กำหนดให้ความหนาแน่นเครือข่าย 0%
- กำหนดข้อมูลสูญหายระดับ0%
- เปรียบเทียบร่วมกับโพรโตคอล TCP

2. ทำการทดสอบในระบบเปิด มีปัจจัยภายนอกรบกวนโดยกำหนดให้ทำการทดสอบกับชุดข้อมูลแบบบีบอัด (Compress Files) และเปรียบเทียบบนระบบ Instant Messanger ที่ทำงานแบบ Client Server, Peer to Peer และ Hybrid

- กำหนดขนาด Bandwidth 1 Mbps
- ทดสอบกับ Random Compress files ขนาด 128 - 512K bytes
- กำหนดให้ความหนาแน่นเครือข่าย 10 - 60%
- กำหนดข้อมูลสูญหายหลายระดับ 1 - 10%
- เปรียบเทียบร่วมกับโพรโตคอล TCP

ผลและวิจารณ์

ผล

ผลที่ได้จากการทำการวิจัยตามวิธีการซึ่งได้วางแผนไว้ได้ข้อมูลจำนวนมาก และได้นำมาวิเคราะห์ความเกี่ยวเนื่องระหว่างปัจจัยควบคุมต่างๆได้แก่ ประเภทข้อมูล, Bandwidth, ความหนาแน่นเครือข่าย และ การสูญหายข้อของมูล โดยนำมาวิเคราะห์เปรียบเทียบหาความสัมพันธ์ระหว่างความเร็วและเวลาที่ใช้ในการรับส่งข้อมูล

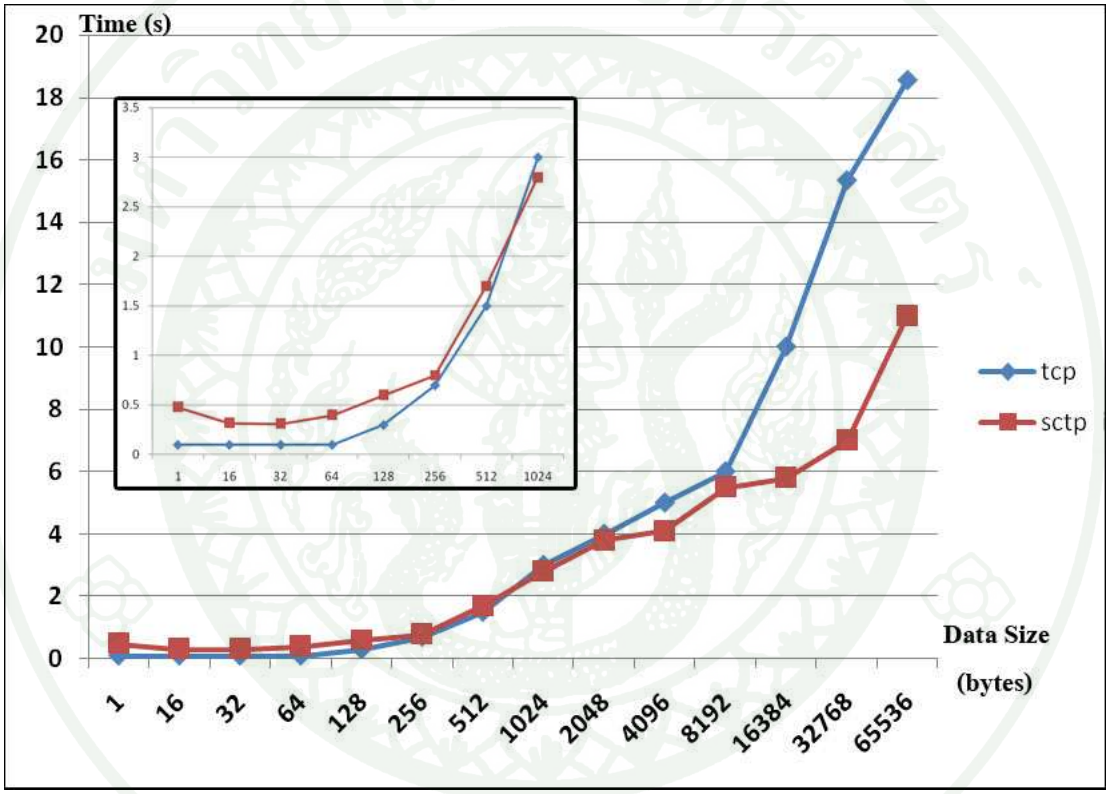
ในการทำการวิจัยได้ทำการทดสอบประสิทธิภาพโดยรวมระหว่างการทำงานโพรโทคอล TCP และโพรโทคอล SCTP ในสภาพแวดล้อมที่เป็น Virtual Machine (VMware) โดยใช้โปรแกรม Traffic Generator และ Dummy Net สร้างการจราจรบนเครือข่ายและ FreeBSD ซึ่งเป็น Transperent Proxy ร่วมกับปลักอิน Traffic Chapper ทำหน้าที่ควบคุม Bandwidth และเก็บการจราจรซึ่งมีการบันทึก Timestamp ไว้ทำซ้ำจำนวน 100 การทดสอบและนำมาหาค่าเฉลี่ยและทำการวิเคราะห์ข้อมูล

การทำการวิจัยนี้ในส่วนแรกได้ทำการวิจัยเปรียบเทียบการทำงานระหว่างโปรโคคอล SCTP และโปรโคคอล TCP ในระบบการส่งข้อความด่วน (Instant Messenging) โดยเปรียบเทียบจากประเภทข้อมูล, ขนาดข้อมูล และปัจจัยภายนอกซึ่งได้กำหนดขึ้นได้ผลการทดสอบดังนี้

เปรียบเทียบการส่งข้อมูลแบบไม่บีบอัดบนระบบปิด โดยกำหนดระบบเบื้องต้นดังนี้

ประเภทข้อมูล	:	ไม่บีบอัด (Text File)
Bandwidth	:	1Mbps
ความหนาแน่นเครือข่าย	:	0%

นำข้อมูลที่ได้ 100 การทดสอบเฉลี่ยและประมวลผลแผนภาพตามภาพที่ 24



ภาพที่ 24 เปรียบเทียบการส่งข้อมูลไม่บีบอัดบนระบบปิดในช่วง 1byte – 64 Kbytes

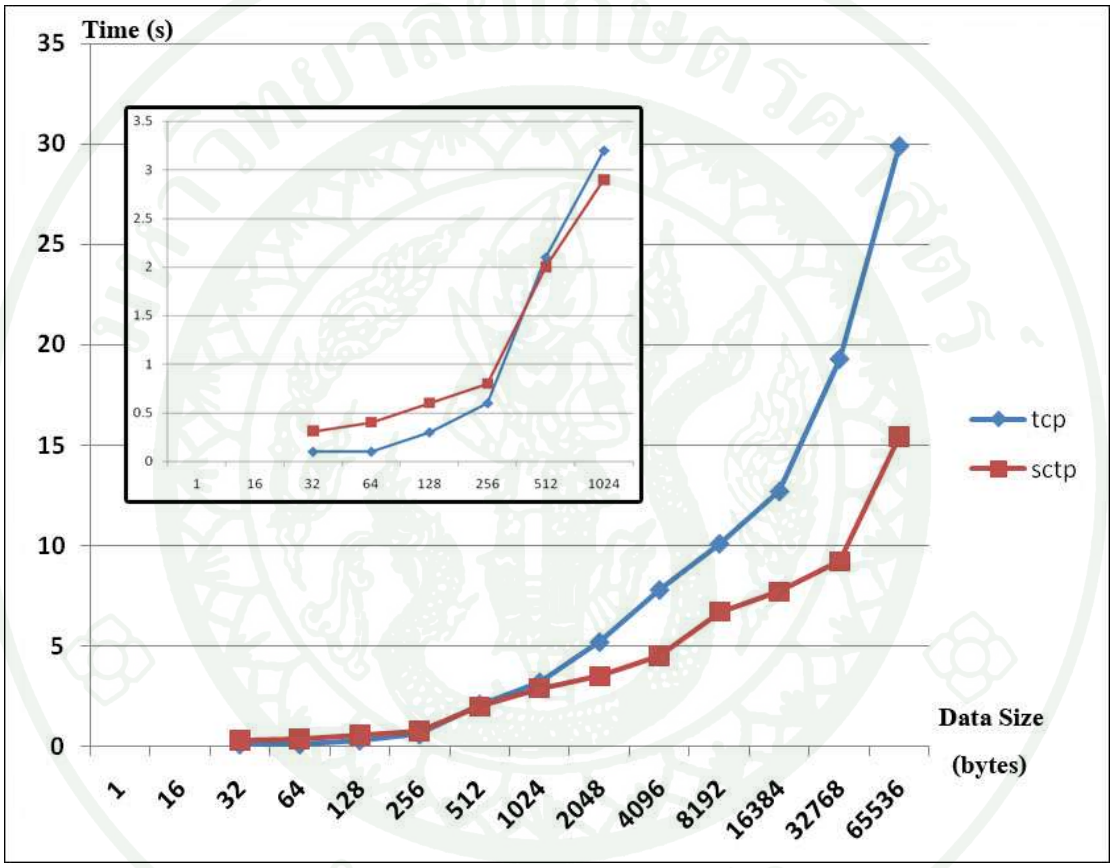
เปรียบเทียบการส่งข้อมูลแบบบีบอัดบนระบบปิด โดยกำหนดระบบเบื้องต้นดังนี้

ประเภทข้อมูล : บีบอัด (Jpeg File)

Bandwidth : 1Mbps

ความหนาแน่นเครือข่าย : 0%

นำข้อมูลที่ได้ 100 การทดสอบเฉลี่ยและประมวลผลแผนภาพตามภาพที่ 25

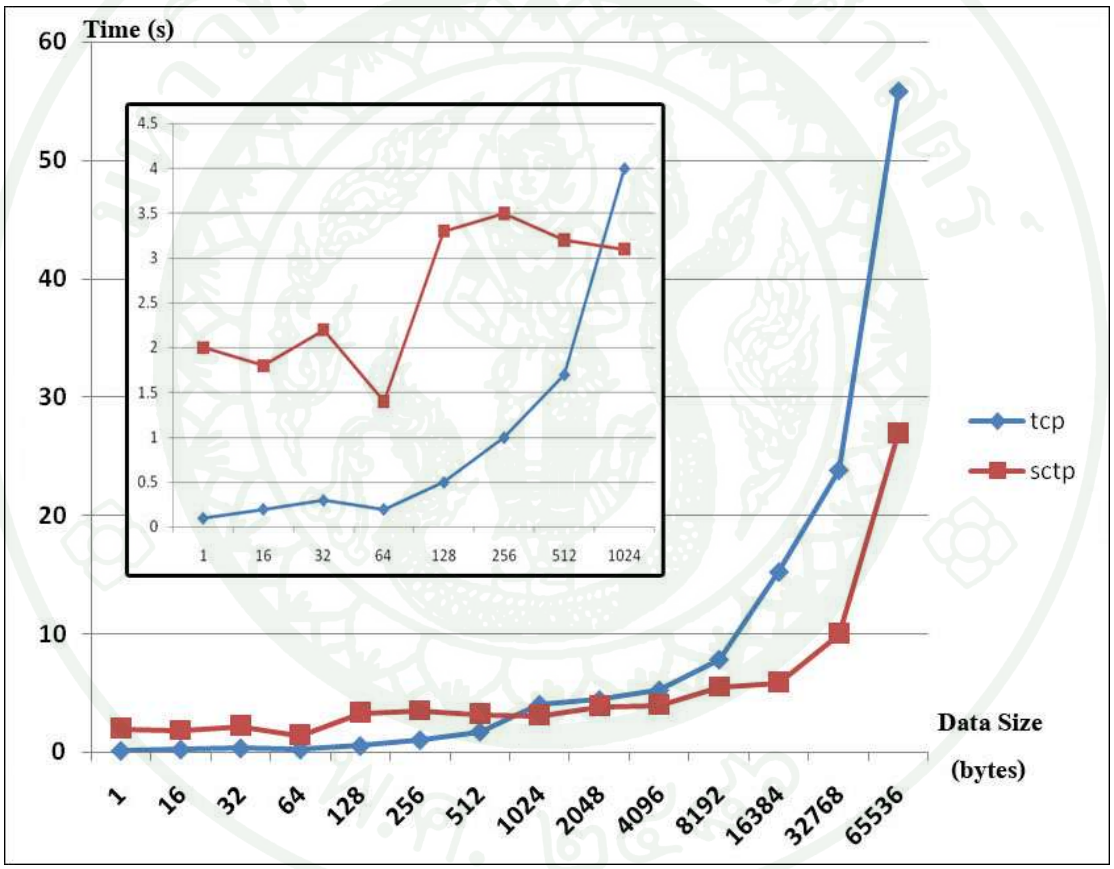


ภาพที่ 25 เปรียบเทียบการส่งข้อมูลบีบอัดบนระบบปิดในช่วง 32byte – 64Kbytes

เปรียบเทียบการส่งข้อมูลแบบไม่บีบอัดบนระบบเปิด โดยกำหนดระบบเบื้องต้นดังนี้

ประเภทข้อมูล	:	ไม่บีบอัด (Text File)
Bandwidth	:	1Mbps
ความหนาแน่นเครือข่าย	:	60%
การสูญหายข้อมูล	:	10%

นำข้อมูลที่ได้ 100 การทดสอบเฉลี่ยและประมวลผลแผนภาพตามภาพที่ 26

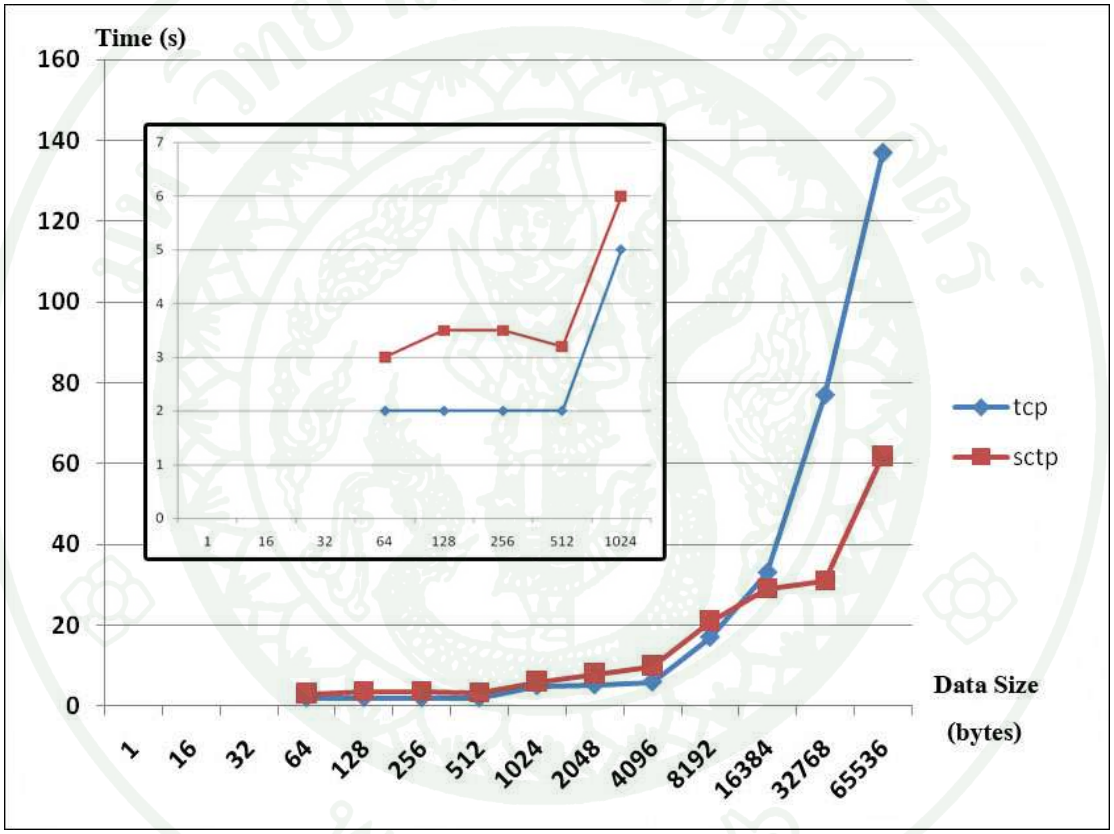


ภาพที่ 26 เปรียบเทียบการส่งข้อมูลไม่บีบอัดบนระบบเปิดในช่วง 1byte – 64Kbytes

เปรียบเทียบการส่งข้อมูลแบบบีบอัดบนระบบเปิด โดยกำหนดระบบเบื้องต้นดังนี้

ประเภทข้อมูล	:	ไม่บีบอัด (Jpeg File)
Bandwidth	:	1Mbps
ความหนาแน่นเครือข่าย	:	60%
การสูญหายข้อมูล	:	10%

นำข้อมูลที่ได้ 100 การทดสอบเฉลี่ยและประมวลผลแผนภาพตามภาพที่ 27



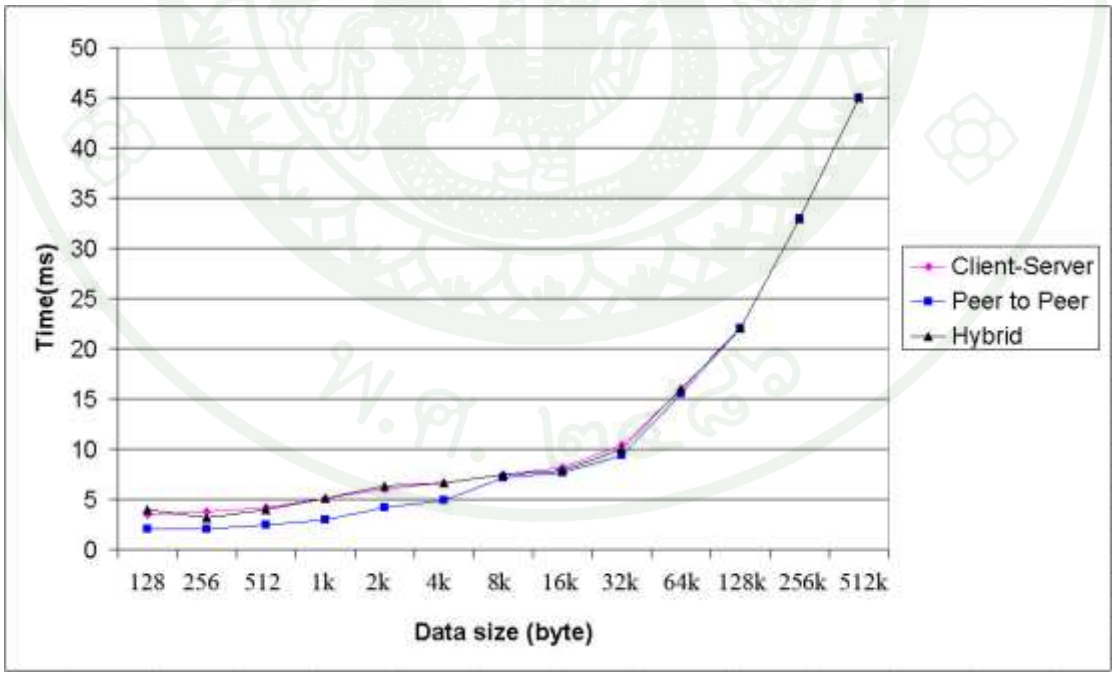
ภาพที่ 27 เปรียบเทียบการส่งข้อมูลบีบอัดบนระบบเปิดในช่วง 64byte – 64Kbytes

การทำการวิจัยในส่วนที่สองได้ทำการทดสอบหาประสิทธิภาพการทำงานของโพรโทคอล SCTP บนระบบการส่งข้อความด่วน การวิจัยเพิ่มเติมในส่วนนี้ต้องการชี้วัดประสิทธิภาพการทำงานของโพรโทคอล SCTPในระบบการส่งข้อความด่วนซึ่งทำงานบนระบบที่แตกต่างกันได้แก่ การทำงานแบบ Client – Server, การทำงานแบบ Peer to Peer และการทำงานแบบ Hybrid โดยเปรียบเทียบในระบบควบคุม และเปลี่ยนแปลงขนาดของข้อมูลพื่อวัดประสิทธิภาพ ได้ผลการวิจัยนำมาสรุปตามรูปแผนภาพดังนี้

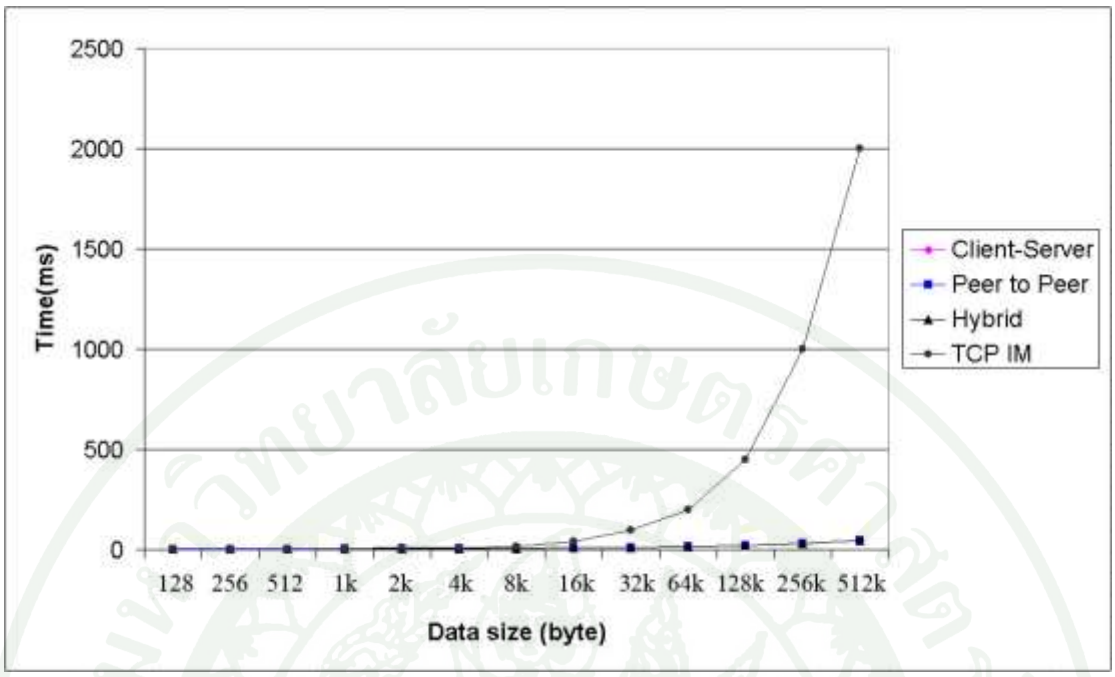
เปรียบเทียบการส่งข้อมูลแบบบีบอัดบนระบบเปิด โดยกำหนดระบบเบื้องต้นดังนี้

ประเภทข้อมูล	:	ข้อมูลตัวอย่างแบบผสม
Bandwidth	:	1Mbps
ความหนาแน่นเครือข่าย	:	0%
การสูญหายข้อมูล	:	0%

นำข้อมูลที่ได้ 100 การทดสอบเฉลี่ยและประมวลผลแผนภาพตามภาพที่ 28 และภาพที่ 29



ภาพที่ 28 เปรียบเทียบการส่งข้อมูลด้วยโปรแกรมส่งข้อความด่วนบนระบบปิด

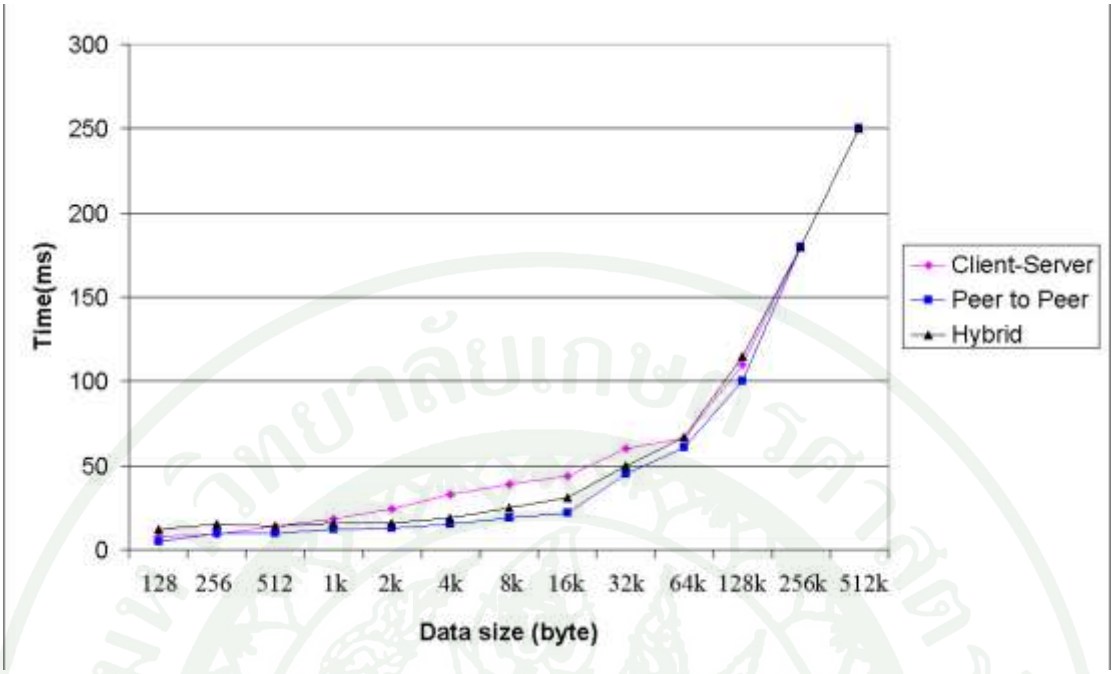


ภาพที่ 29 เปรียบเทียบการส่งข้อมูลด้วยโปรแกรมส่งข้อความด่วนและ TCP บนระบบปิด

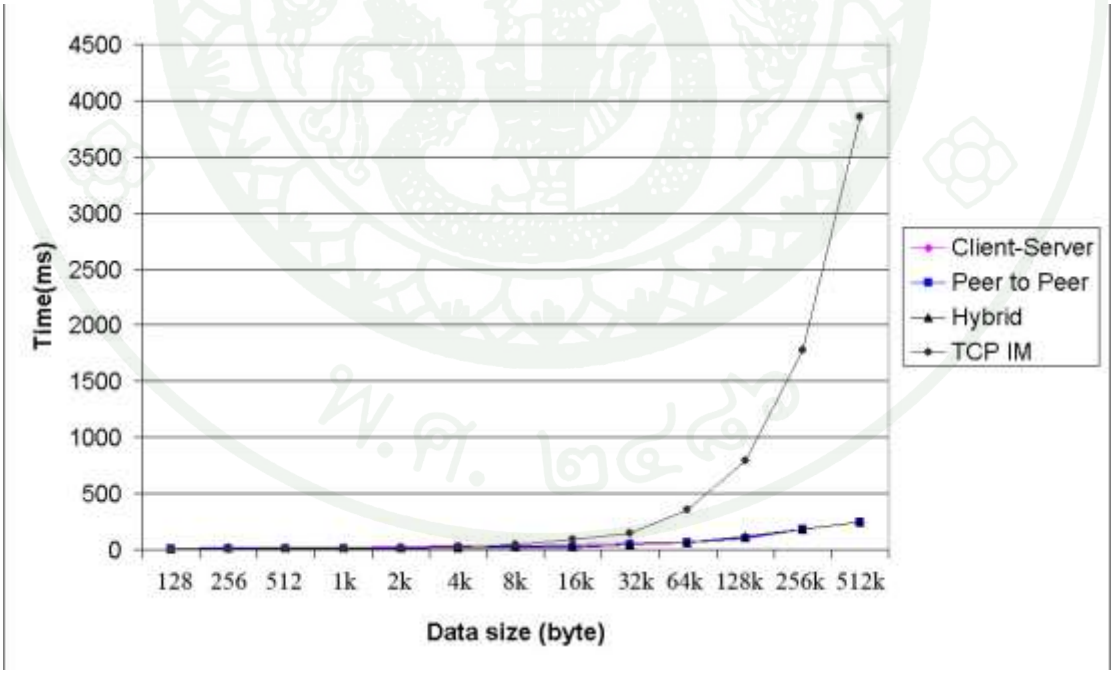
เปรียบเทียบการส่งข้อมูลแบบบีบอัดบนระบบเปิด โดยกำหนดระบบเบื้องต้นดังนี้

ประเภทข้อมูล	:	ข้อมูลตัวอย่างแบบผสม
Bandwidth	:	1Mbps
ความหนาแน่นเครือข่าย	:	60%
การสูญหายข้อมูล	:	10%

นำข้อมูลที่ได้ 100 การทดสอบเฉลี่ยและประมวลผลแผนภาพตามภาพที่ 30 และภาพที่ 31



ภาพที่ 30 เปรียบเทียบการส่งข้อมูลด้วยโปรแกรมส่งข้อความด่วนบนระบบเปิด



ภาพที่ 31 เปรียบเทียบการส่งข้อมูลด้วยโปรแกรมส่งข้อความด่วนและ TCP บนระบบเปิด

วิจารณ์

จากการวิจัยในส่วนแรกแสดงให้เห็นถึงประสิทธิภาพการทำงานของโพรโทคอล SCTP เมื่อเปรียบเทียบกับโพรโทคอล TCP โดยอาศัยการเปลี่ยนแปลงปัจจัยแวดล้อม และนำมาวิเคราะห์หาความสัมพันธ์ระหว่างขนาดของข้อมูลและระยะเวลาที่ใช้ในการส่งข้อมูลไปยังผู้รับด้วยโปรแกรมทดสอบการส่งข้อมูลผ่านระบบการส่งข้อความด่วน

ประสิทธิภาพอันเกิดจากการเปลี่ยนแปลงขนาดของข้อมูลโดยเฉพาะภาพส่วนต้นจากกราฟภาพที่ 24, ภาพที่ 25, ภาพที่ 26 และภาพที่ 27 แสดงให้เห็นว่า โพรโทคอล TCP ทำงานได้ดีกว่าเมื่อข้อมูลมีขนาดเล็ก หรือกล่าวคือเมื่อข้อมูลเล็กกว่า 1กิโลไบต์ (ข้อมูลไม่บีบอัด ไม่มีการสูญหายของข้อมูล) ถึง 16กิโลไบต์ (ข้อมูลบีบอัดและมีการสูญหายของข้อมูล) โดยโพรโทคอล SCTP จะทำงานได้ดีกว่าเมื่อขนาดของข้อมูลใหญ่ขึ้นและอัตราส่วนประสิทธิภาพสูงขึ้นตามขนาดของข้อมูลด้วย เห็นได้จากส่วนข้อมูลขนาดใหญ่ภายในกราฟภาพที่ 24, ภาพที่ 25, ภาพที่ 26 และภาพที่ 27 จากการวิเคราะห์การทำงานของโปรแกรมทดสอบและจากคุณสมบัติของโพรโทคอล SCTP เองสามารถบอกได้ถึงสาเหตุซึ่งทำให้โพรโทคอล SCTP มีประสิทธิภาพต่ำกว่าโพรโทคอล TCP คือภาระในการสถาปนาการเชื่อมต่อแบบ 4-way handshake ซึ่งมีขั้นตอนมากกว่าและการรับส่งข้อมูลแบบ streaming ซึ่งกำหนดขนาดของชุดข้อมูล ทำให้การส่งข้อมูลชุดต้นๆ ของโพรโทคอล SCTP ทำได้ช้ากว่าโพรโทคอล TCP และเมื่อข้อมูลมีขนาดใหญ่ขึ้นจึงจะสามารถใช้ประสิทธิภาพจากคุณสมบัติการทำงานของตัวโพรโทคอล SCTP ได้เต็มที่ ดังการวิจัยจะเห็นได้ว่าประสิทธิภาพการทำงานของโพรโทคอล SCTP จะสูงกว่าเป็นลำดับตามขนาดของข้อมูล

ประสิทธิภาพอันเกิดจากประเภทของข้อมูล จากกราฟภาพที่ 24 เปรียบเทียบกับกราฟภาพที่ 25 และ จากกราฟภาพที่ 26 เปรียบเทียบกับ กราฟภาพที่ 27 พบว่าประสิทธิภาพของการส่งข้อมูลแบบบีบอัดผ่านโพรโทคอล SCTP จะมีประสิทธิภาพลดลงเมื่อเปรียบเทียบกับข้อมูลแบบไม่บีบอัด แต่อย่างไรก็ตามเมื่อเปรียบเทียบกับการทำงานของโพรโทคอล TCP แล้วจะพบว่าประสิทธิภาพจะลดลงในอัตราส่วนที่น้อยกว่า

ประสิทธิภาพอันเกิดจากปริมาณความหนาแน่นเครือข่ายและการสูญหายของข้อมูล (Packet Lost) จากกราฟภาพที่ 24 เทียบกับกราฟภาพที่ 26 ซึ่งเป็นข้อมูลไม่บีบอัด และกราฟภาพที่ 25 เทียบกับกราฟภาพที่ 27 ซึ่งเป็นข้อมูลแบบบีบอัด พบว่าประสิทธิภาพการทำงานของโพรโทคอล

SCTP ดีกว่า การทำงานของโพรโทคอล TCP อย่างเห็นได้ชัดกับการทำงานร่วมกับข้อมูลทุกประเภท อันเกิดจากคุณสมบัติการ streaming ของตัวโพรโทคอล SCTP นั่นเองเนื่องจากไม่จำเป็นต้องรอการส่งซ้ำ ของชุดข้อมูลสูญหายซึ่งต่างจากโพรโทคอล TCP ที่จะหยุดรอข้อมูลชุดดังกล่าวก่อนหลังจากนั้นจึงเริ่มต้นทำงานต่อ

การวิจัยในส่วนที่สองแสดงให้เห็นถึงประสิทธิภาพการทำงานของโพรโทคอล SCTP บนระบบการส่งข้อความด่วนซึ่งมีรูปแบบการทำงานต่างกัน ได้แก่ การทำงานแบบ Client-Server, การทำงานแบบ Peer to Peer และการทำงานแบบ Hybrid ทำการเปรียบเทียบประสิทธิภาพระหว่างขนาดของข้อมูลและเวลาที่ใช้ในการส่งข้อมูล

ประสิทธิภาพอันเกิดจากโพรโทคอล จากกราฟภาพที่ 29 และภาพที่ 31 พบว่าประสิทธิภาพการทำงานของระบบการส่งข้อความด่วนด้วยโพรโทคอล SCTP มีประสิทธิภาพสูงกว่าโพรโทคอล TCP โดยรวมอย่างชัดเจน แม้ว่า จากกราฟภาพที่ 29 และภาพที่ 31 โพรโทคอล TCP จะแสดงให้เห็นว่ามีประสิทธิภาพที่ดีกับข้อมูลซึ่งมีขนาดเล็กกว่า 1024 byte

ประสิทธิภาพอันเกิดจากรูปแบบการทำงาน จากกราฟภาพที่ 28 และภาพที่ 30 พบว่าการทำงานของโพรโทคอล SCTP บนระบบการส่งข้อความด่วนทุกแบบเมื่อข้อมูลมีขนาดใหญ่จะมีประสิทธิภาพไกล้เคียงกัน โดยประสิทธิภาพที่สามารถวัดได้จะเป็นช่วงข้อมูลขนาดเล็ก โดยสามารถสรุปถึงความแตกต่างประสิทธิภาพการทำงานได้ว่า การทำงานของโพรโทคอล SCTP บนระบบการส่งข้อความด่วนแบบ Hybrid จะสูงกว่าเล็กน้อยและใกล้เคียงกับการทำงานในรูปแบบ Client Server แต่จะดีกว่าการทำงานแบบ Peer to Peer ในกลุ่มขนาดข้อมูลขนาดเล็กมาก แต่เมื่อข้อมูลมีขนาดใหญ่ขึ้นประสิทธิภาพการทำงานแบบ Peer to Peer มีประสิทธิภาพสูงที่สุด โดยสามารถวิเคราะห์ถึงผลการวัดประสิทธิภาพเมื่อเทียบกับลักษณะการทำงานได้

การทำงานของระบบการส่งข้อความด่วนแบบ Client Server และการทำงานแบบ Hybrid มีประสิทธิภาพใกล้เคียงกันในการทำงานกับข้อมูลขนาดเล็กที่สุดเนื่องจากจะมีการเชื่อมต่อระหว่าง Client และ Server ตั้งแต่เริ่มทำงานเช่นเดียวกันซึ่งแตกต่างจาก การทำงานแบบ Peer to Peer ซึ่งจะมีการสถาปนาการเชื่อมต่อเมื่อต้องการส่งข้อมูลจึงมีภาระมากกว่าและทำงานช้ากว่า และเมื่อขนาดของข้อมูลตัวอย่างมีขนาดใหญ่ขึ้น ระบบการส่งข้อความด่วนซึ่งมีการทำงานแบบ Peer to Peer จะมีประสิทธิภาพการทำงานสูงสุดอันเนื่องมาจากไม่มีภาระต้องส่งข้อมูลผ่าน Server เหมือนระบบการ

ส่งข้อความด่วนแบบ Client Server และ Hybrid สำหรับการเปรียบเทียบระหว่างระบบการส่งข้อความด่วนแบบ Client Server และ Hybrid ด้วยชุดข้อมูลขนาดใหญ่ขึ้นการทำงานแบบ Hybrid จะมีประสิทธิภาพดีกว่าเนื่องจากจะมีการเชื่อมต่อส่งข้อมูลโดยตรงระหว่าง Client ทำให้ลดภาระลงได้

เมื่อข้อมูลมีขนาดใหญ่ขึ้นการทำงานของระบบการส่งข้อความด่วนแบบ Client Server, Peer to Peer และ Hybrid มีประสิทธิภาพใกล้เคียงกันเนื่องจากใช้โพรโทคอล SCTP เช่นเดียวกัน ภาระที่แตกต่างกันไม่ว่าการสถาปนาการเชื่อมต่อ ภาระการส่งต่อผ่าน Server จะมีผลต่อประสิทธิภาพกับข้อมูลที่ส่งในช่วงแรกเท่านั้น เมื่อข้อมูลมีขนาดใหญ่ขึ้นภาระดังกล่าวจะมีอัตราส่วนต่อประสิทธิภาพน้อยลงตามลำดับ

สรุปและข้อเสนอแนะ

สรุป

วัตถุประสงค์ของงานวิจัยนี้คือการปรับใช้โพรโทคอล SCTP ในการใช้งานระบบการส่งข้อความด่วน และทำการทดสอบประสิทธิภาพเปรียบเทียบกับโพรโทคอล TCP ในสภาพการทำงานและสภาพแวดล้อมต่างๆกัน เพื่อเป็นแนวทางในการปรับใช้โพรโทคอล SCTP กับระบบการส่งข้อความด่วนและระบบอื่นๆอย่างมีประสิทธิภาพ

ผลการทดสอบประสิทธิภาพแสดงให้เห็นว่าโพรโทคอล SCTP สามารถปรับปรุงประสิทธิภาพการทำงานของระบบการส่งข้อความด่วนได้โดยรวม แต่หากแยกดูรายละเอียดจะเห็นได้ว่าโพรโทคอล SCTP จะสามารถทำงานได้อย่างมีประสิทธิภาพกับข้อมูลที่มีขนาดใหญ่มากว่า 1024 bytes และการทำงานของโพรโทคอล TCP จะสามารถทำงานได้ดีกับกลุ่มข้อมูลที่มีขนาดเล็กกว่า 1 Kbyte ดังนั้นจึงสามารถสรุปได้ว่าโพรโทคอล SCTP จะสามารถทำงานได้อย่างมีประสิทธิภาพในสภาวะที่สามารถใช้งานคุณสมบัติของโพรโทคอล SCTP ได้แก่ ข้อมูลขนาดใหญ่กว่า 1 Kbyte ระบบที่มีการสูญหายของข้อมูล และประเภทของข้อมูลแบบบีบอัด เป็นต้น

การเลือกใช้รูปแบบการทำงานของระบบการส่งข้อความด่วนร่วมกับโพรโทคอล SCTP อย่างมีประสิทธิภาพในกลุ่มข้อมูลขนาดกลางตั้งแต่ 512 bytes ถึง 256 Kbyte คือการทำงานแบบ Peer to Peer และในกลุ่มข้อมูลขนาดเล็กกว่า 512 bytes การทำงานแบบ Client Server ให้ผลการทำงานที่ดี

ข้อเสนอแนะ

แนวทางการพัฒนาและใช้งานโพรโทคอล SCTP ร่วมกับระบบการส่งข้อความด่วน สามารถใช้รูปแบบผสมผสานทั้งส่วนของโพรโทคอล SCTP และโพรโทคอล TCP เนื่องจากแต่ละโพรโทคอลมีคุณสมบัติต่างกันทำให้สามารถทำงานได้อย่างมีประสิทธิภาพสูงสุดในสถานการณ์ต่างกัน นอกจากนี้ยังสามารถเลือกใช้รูปแบบการทำงานแบบ Peer to Peer ในขณะที่มีการใช้งานโพรโทคอล SCTP ในระบบการส่งข้อความด่วนเพื่อให้ได้ประสิทธิภาพสูงสุด

จากผลการทดสอบที่ได้จากการวิจัยนี้สามารถประยุกต์ใช้กับระบบอื่น ซึ่งมีการส่งข้อมูลหลายรูปแบบผ่านเครือข่ายได้

การวิจัยทดสอบสามารถทำเพิ่มได้เช่นในระบบที่มีสภาวะเครือข่ายไม่มั่นคงและทดสอบการใช้งานคุณสมบัติอื่นของโพรโทคอล SCTP เช่น Multihoming

เอกสารและสิ่งอ้างอิง

A. Balk, M. Sigler, M. Gerla, and M.Y. Sanadidi. 2002. Investigation of MPEG-4 video streaming over SCTP, In Proc. IIIS World Multi-Conf. System. Cybern. Inform. (SCI), Orlando, Florida.

T. Ravier, R. Brennan, and T. Curran. 2001. Experimental studies of SCTP multihoming, First Joint IEI/IEE Symposium on Telecommunications Systems Research, Dublin, Ireland.

R. Brennan and T. Curran. 2001. SCTP Congestion Control: Initial Simulation Studies, Proc. 17th Int'l Teletraffic Congress, Elsevier Science.

A. Caro, K. Shah, J. Iyengar, P. Amer, R. Stewart. 2003. SCTP and TCP Variants Congestion Control Under Multiple Losses," submitted to ACM Computer Communication Review.

KJ. Grinnemo, T. Andersson, A. Brunstrom. 2005. Performance Benefits of Avoiding Head-of-Line Blocking in SCTP, Proceedings of the ICAS/ICNS.

P. Natarajan, JR. Iyengar, PD. Amer, R. Stewart. 2006. SCTP An innovative transport layer protocol for the web, Proceedings of the 15th international conference on World Wide Web.

R. Rajamani, S. Kumar, N. Gupta. 2002 SCTP versus TCP Comparing the Performance of Transport Protocols for Web Traffic, University of Wisconsin-Madison.

Henrik Osterdahl. 2005. A comparison of TCP and SCTP performance using the HTTP protocol, http://www.it.kth.se/courses/2G1305/Sample-papers/Henrik_Oesterdahl-sctp-2005 0525.pdf.

T. Ravier, R. Brennan, and T. Curran. 2001. Experimental studies of SCTP multihoming, First Joint IEI/IEE Symposium on Telecommunications Systems Research, Dublin, Ireland.

M. Molteni and M. Villari. 2003. Using SCTP with Partial Reliability for MPEG-4 Multimedia Streaming, Published in Proc. of BSDCon Europe.

Ahmed Abd El Al, Tarek Saadawi and Myung Lee. 2007. LS-SCTP: a bandwidth aggregation technique for stream control transmission protocol, Department of Electrical Engineering, City College and Graduate Center of City University of New York, New York.

Nicholas Feamster. 2000. Adaptive deleveryof real-time streaming video. Master'sthesis, Institute of Technology.

ภาคผนวก

ภาคผนวก ก

การเปรียบเทียบประสิทธิภาพการทำงานโพรโทคอล TCP และ SCTP บนระบบการส่งข้อความด่วน (Instant Messaging)

(JCSSE 2011), มีนาคม, 2554

การเปรียบเทียบประสิทธิภาพการทำงานโพรโทคอล TCP และ SCTP บนระบบการส่งข้อความด่วน

Compare efficiency of TCP protocol and SCTP protocol on Instant Messaging

วโรดม ลิมปะพันธุ์
ภาควิชาวิทยาการคอมพิวเตอร์,
มหาวิทยาลัยเกษตรศาสตร์
varodom@gmail.com

สุขุมาล กิติสิน
ภาควิชาวิทยาการคอมพิวเตอร์,
มหาวิทยาลัยเกษตรศาสตร์
fscismi@ku.ac.th

ชวลิต ศรีสถาพรพัฒน์
ภาควิชาวิทยาการคอมพิวเตอร์,
มหาวิทยาลัยเกษตรศาสตร์
fscicls@ku.ac.th

บทคัดย่อ

การทำงานของระบบการส่งข้อความด่วน (Instant Messaging) ซึ่งเป็นที่นิยมใช้งานในปัจจุบัน โดยส่วนใหญ่นิยมทำงานโดยอาศัยโพรโทคอล TCP เป็นส่วนมาก และข้อมูลหลักที่ถูกส่งผ่านระบบประกอบด้วยข้อมูลแบบอักษร (Text) และข้อมูลมัลติมีเดียหลากหลายรูปแบบซึ่งถูกบีบอัดแตกต่างกัน

การนำเอาโพรโทคอล SCTP ซึ่งพัฒนาขึ้นเพื่อทดแทน โพรโทคอล TCP มาประยุกต์ใช้กับระบบการส่งข้อความด่วน (Instant Messaging) ซึ่งมีข้อมูลหลายประเภท และหลายวัตถุประสงค์ เช่นข้อความ รูปภาพ ข้อมูลมัลติมีเดีย ข้อมูล Streaming โดยข้อมูลดังกล่าวจะถูกส่งผ่านเครือข่ายตัวกลาง หลากหลายรูปแบบซึ่งมีคุณสมบัติแตกต่างกันไป ดังนั้นประสิทธิภาพการทำงานนอกจากจะขึ้นอยู่กับคุณสมบัติของตัวโพรโทคอลเองแล้วยังขึ้นอยู่กับสิ่งแวดล้อมดังกล่าวอีกด้วย

การทดสอบประสิทธิภาพในการทำการวิจัยนี้แสดงให้เห็นว่าไม่มีโพรโทคอลใดทำงานดีที่สุดในทุกสภาพแวดล้อม นอกจากนี้ยังแสดงให้เห็นถึงแนวโน้มประสิทธิภาพเมื่อเปลี่ยนแปลงสิ่งแวดล้อมอีกด้วย ดังนั้นจึงเป็นแนวทางในการเลือกใช้โพรโทคอลให้เหมาะสมกับแอพลิเคชั่นและลักษณการใช้งาน

คำสำคัญ : โพรโทคอล SCTP, ทดสอบประสิทธิภาพ, ระบบส่งข้อความด่วน, Instant Messaging

1. บทนำ

ในปัจจุบันมีการใช้งานระบบการส่งข้อความด่วน Instant Messaging อย่างแพร่หลายบนเครือข่ายการสื่อสารหลากหลายประเภทโดยมาตรฐานการส่งผ่านข้อมูลของแอปพลิเคชันที่ใช้กันอย่างแพร่หลายนิยมใช้งาน TCP โพรโทคอลเป็นหลัก

การส่งผ่านข้อมูลบนโพรโทคอลระบบ TCP ของระบบการส่งข้อความด่วน Instant Messaging แม้จะสามารถรับประกันความถูกต้องและความแน่นอนในการส่งก็ตาม แต่ในปัจจุบันรูปแบบการใช้งานเปลี่ยนแปลงไปอย่างรวดเร็ว และในปัจจุบันการสื่อสารข้อมูลจะประกอบไปด้วยข้อมูลมัลติมีเดียซึ่งมีขนาดใหญ่เป็นจำนวนมาก

ข้อดีของการนำเอา SCTP มาใช้งานระบบการส่งข้อความด่วน Instant Messaging ถือเป็นการการปรับปรุงข้อด้อยของการทำงานแบบเดิมบนระบบ TCP โพรโทคอลโดยอาศัยคุณสมบัติที่ดีของ SCTP ไม่ว่าจะเป็นการสื่อสารแบบหลายช่องทาง (Multi Streaming) การส่งข้อมูลแบบ Multi-Homing หรือการทำ 4-way Handshake ซึ่งจะทำให้สามารถส่งข้อมูลได้มีเสถียรภาพและความเร็วสูงขึ้นตลอดจนมีความปลอดภัยดีขึ้นด้วย

อย่างไรก็ตามการนำโพรโทคอล SCTP มาใช้งานกับระบบการส่งข้อความด่วน Instant Messaging อาจทำให้มีความยุ่งยากและความล่าช้าในการใช้งานซึ่งเกิดจากลักษณะการทำงานของตัวโพรโทคอลเองดังนั้นจึงมีแนวความคิดทำการวิจัยทดสอบวัดผลเปรียบเทียบประสิทธิภาพการทำงานการส่งผ่านข้อมูลระหว่างโพร

โทคอล TCP และโพรโทคอล SCTP ด้วยลักษณะข้อมูลและสิ่งแวดล้อมที่แตกต่างกัน

ผลที่คาดว่าจะได้รับเมื่อทำงานวิจัยชิ้นนี้คือได้ผลการเปรียบเทียบประสิทธิภาพการทำงานในการส่งผ่านข้อมูลในรูปแบบ ขนาดและสภาพแวดล้อมซึ่งต่างกัน ระหว่างโพรโทคอล TCP และโพรโทคอล SCTP บนระบบการส่งข้อความด่วน Instant Messaging

2.ทฤษฎีและโพรโทคอลที่เกี่ยวข้อง

2.1 โพรโทคอล TCP/IP

ในการใช้โพรโทคอล TCP/IP มีวัตถุประสงค์เพื่อใช้ในการสื่อสารจากต้นทางข้ามเน็ตเวิร์กไปยังปลายทางได้ และสามารถหาเส้นทางที่จะส่งข้อมูลไปได้เองโดยอัตโนมัติ ถึงแม้ว่าในในระหว่างทางอาจจะผ่านเครือข่ายที่มีปัญหาโพรโทคอลก็ยังคงหาเส้นทางส่งผ่านข้อมูลไปให้ถึงปลายทางจนได้

โดยทั่วไปการที่โฮสต์จะมีการเชื่อมต่อกันได้นั้น จะต้องเชื่อมต่อกันให้ถึงกันก่อน หลังจากนั้นก็จะสามารถรับส่งข้อมูลได้ และจะสามารถรับส่งข้อมูลกันได้ตลอดตราบใดที่ยังมีการเชื่อมต่ออยู่ หากต้องการการส่งข้อมูลที่มีขนาดใหญ่ขึ้นจะมีการกระจายข้อมูออกเป็นหลายแพ็กเก็ตและเมื่อแพ็กเก็ตส่งถึงปลายทางแล้วจะต้องนำมารวมกันอีกครั้งหนึ่งตามลำดับแสดงให้เห็นได้ว่าข้อมูลหนึ่งดาต้าแกรมไม่จำเป็นต้องส่งเพียงครั้งเดียว หากความยาวเกินกว่าที่จะส่งได้ในแต่ละครั้งดาต้าแกรมที่สามารถจะแบ่งย่อยเพื่อให้ขนาดของแพ็กเก็ตมีขนาดเหมาะสมได้

ด้วยคุณสมบัติโพรโทคอล TCP ได้มีการกำหนดช่องทางในการส่งข้อมูลภายในเพียงช่องทางเดียวในแต่ละการเชื่อมต่อ ซึ่ง TCP จะจัดเรียงข้อมูลที่ได้รับจากผู้ส่ง ตามหมายเลขลำดับข้อมูล โดยถ้าหากข้อมูลส่วนใดส่วนหนึ่งขาดหายไป TCP จะไม่ส่งข้อมูลส่วนถัดไปให้กับ Application layer ปลายทางแต่จะทำการร้องขอข้อมูลในส่วนที่ขาดไปมาให้ได้ก่อน จึงจะสามารถดำเนินการส่งข้อมูลถัดไปให้กับ Application layer ฝั่งผู้รับได้ เมื่อเป็นเช่นนี้แล้ว เมื่อข้อมูลในส่วนใดของการเชื่อมต่อหายไป TCP จะไม่สามารถดำเนินการต่อไปได้เลยหากไม่ได้รับข้อมูลในส่วนนั้นมาก่อนอย่างถูกต้อง และเนื่องจาก TCP มีช่องทางเพียงช่องทางเดียวในการเชื่อมต่อ จึงทำให้ทั้งผู้ส่ง และผู้รับข้อมูลไม่สามารถดำเนินการรับส่งข้อมูลในส่วนอื่นระหว่างกันได้ ปัญหานี้เรียกว่า HOL (Head Of line problem)

2.2 SCTP (Stream Control Transmission Protocol)

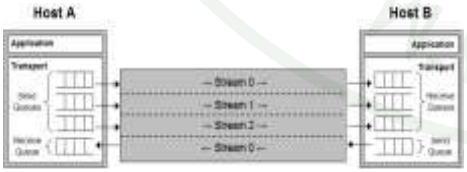
SCTP ถูกพัฒนาขึ้นมาเพื่อรองรับการส่งข้อมูลที่ไม่เหมาะสมที่จะส่งด้วย TCP โดย IETF (Internet Engineering Task Force) SIGTRAN (Signaling Transport) เป็นกลุ่มผู้พัฒนาโพรโทคอลใหม่และได้นำเสนอโพรโทคอลใหม่นี้ออกสู่สาธารณะชนเมื่อเดือนตุลาคม พ.ศ.2543 โดยเผยแพร่ใน RFC2960 ซึ่ง SCTP มีลักษณะการทำงานที่คล้ายคลึงกับ TCP แต่ได้เพิ่มคุณสมบัติหลายประการเพื่อแก้ปัญหาต่างๆ ที่พบใน TCP ได้แก่

Multi-Streaming ซึ่งจะมีลักษณะการทำงานขนส่งข้อมูลหลายเส้นทางโดยในการทำงานจะยอมให้ข้อมูลถูกแยกส่งข้อมูลหลายเส้นทาง ซึ่งจะมีคุณสมบัติการนำส่งที่ไม่พึ่งพาอาศัยซึ่งกันและกันและแยกการทำงานออกจากกัน ดังนั้นหากมีข้อมูลหายไปในแต่ละเส้นทางที่นำส่งข้อมูลจะเกิดผลกระทบในเส้นทางนั้นเพียงเส้นทางเดียว ไม่ส่งผลกระทบต่อเส้นทางอื่นที่นำส่งข้อมูลซึ่งแตกต่างจากTCP ซึ่งเป็นการนำส่งข้อมูลเพียงเส้นทางเดียว โดคาดเดาว่าข้อมูลซึ่งถูกนำส่งจะเรียงตามสำดับการส่งการคาดเดานี้ อาจจะก่อให้เกิดภาวะล่าช้าเพิ่มขึ้นถ้ามีข้อมูลหายหรือการเรียงลำดับข้อมูลผิดพลาดเกิดขึ้นในเครือข่าย

จากปัญหาที่พบในโพรโทคอล TCP ซึ่งมีช่องทางในการสื่อสารเพียงช่องทางเดียวโพรโทคอล

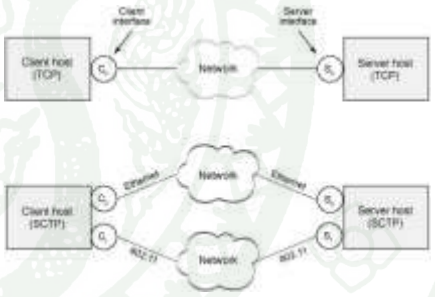
SCTP จึงพัฒนาให้สามารถส่งข้อมูลไปได้พร้อมกันหลายเส้นทาง โดยเปรียบเสมือนกับมีหลายช่องทาง (stream) การสื่อสาร โดยจะมีการสถาปนาการเชื่อมต่อเพียงครั้งเดียว โดยจะกำหนดค่าเฉพาะให้กับข้อมูลในแต่ละช่องทาง เพื่อให้ข้อมูลสามารถส่งออกไปได้พร้อมกัน เมื่อผู้รับได้รับข้อมูลในแต่ละช่องทางมาแล้ว เมื่อผู้รับจะดำเนินการจัดเรียงและตรวจสอบข้อมูลในแต่ละช่องทางที่ได้รับ ถ้ามีการสูญหายของข้อมูลในช่องทางใด เมื่อผู้รับจะร้องขอข้อมูลเฉพาะของช่องทางนั้นเท่านั้น โดยที่ไม่มีผลกระทบกับข้อมูลในช่องทางอื่น ทำให้ช่องทางอื่นยังสามารถดำเนินการรับข้อมูลต่อไปได้ ไม่จำเป็นต้องหยุดรอทั้งหมด ซึ่งเป็นการแก้ปัญหา HOL ที่พบในโพรโทคอล TCP

แสดงการส่งข้อมูลระหว่าง client กับ server ซึ่ง client สร้างช่องทางการสื่อสาร 3 ช่องทาง จะเห็นว่าถ้าหากข้อมูลลำดับที่ 3 ซึ่งถูกส่งด้วยช่องทางด้านบนสูญหายไป จะไม่กระทบกับการส่งข้อมูลในช่องทางอื่นเลย ช่องทางอื่นจะยังคงสามารถดำเนินการส่งข้อมูลระหว่างกันได้ มีเฉพาะช่องทางด้านบนเท่านั้นที่หยุดรอเพื่อให้ข้อมูลส่งมาในลำดับที่ถูกต้องโดยการสร้างช่องทางการสื่อสารของ SCTP สามารถสร้างได้สูงสุด 65535 ช่องทางทั้งทางด้านส่งออกและ 65535 ช่องทางในด้านรับข้อมูลทำให้การเชื่อมต่อจะสามารถมีช่องทางการสื่อสารได้ถึง 131070 ช่องทาง



ภาพที่ 1 Multi-Streaming

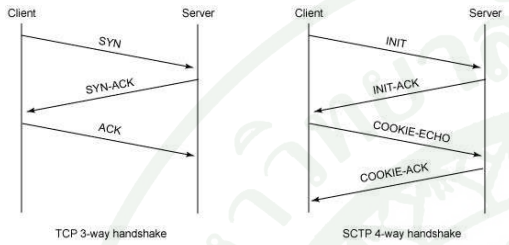
Multi-Homing คือในการส่งข้อมูลมีเป้าหมายหลายแห่ง คุณสมบัติของโพรโทคอล SCTP จะสามารถจะส่งข้อมูลไปได้หลายเป้าหมาย การที่มีเป้าหมายได้หลายแห่งจะมีประโยชน์สูงเมื่อเครือข่ายล้มเหลว เมื่อเกิดเครือข่ายล้มเหลวหรือ เมื่อเกิดการล่มของเครือข่ายจะทำให้การนำส่งข้อมูลหยุดชั่วคราวจนกระทั่ง IP routing protocols จะสามารถค้นหาเส้นทางใหม่ได้ การใช้ multi-homed ของ SCTP ซึ่งมี redundant LANs จะทำให้เกิดแรงกระตุ้นไปยัง ตัวเลือกอื่นๆในเครือข่ายซึ่งสามารถส่งข้อมูลทดแทนได้ ดังนั้นจึงสามารถลดการพึ่งพาเครือข่ายที่มีเสถียรภาพสูง กล่าวคือเป็นการทำงานของโพรโทคอลที่ยอมให้มีผู้ส่งข้อมูลชุดเดียวกันได้มากกว่า 1 เครื่องในเครือข่าย เพื่อขจัดปัญหาเครือข่ายหยุดทำงาน โดยเมื่อผู้ส่งหลักมีปัญหาไม่สามารถให้บริการได้ผู้รับยังสามารถร้องขอข้อมูลจากผู้ส่ง อื่นได้ โดยไม่จำเป็นต้องเริ่มขอสถาปนาการเชื่อมต่อใหม่ ซึ่งโพรโทคอล SCTP ซึ่งใช้คุณสมบัติ Multi-Homing ที่สามารถกำหนดผู้ส่ง (server) ในเครือข่ายได้หลายเครื่อง จึงสามารถลดปัญหาเครือข่ายหยุดทำงานที่พบใน TCP ซึ่งมี server ได้เพียงเครื่องเดียวได้



ภาพที่ 2 Multi-Homing

4-way Handshake เป็นรูปแบบการสถาปนาการเชื่อมต่อในโพรโทคอล SCTP จะเป็นแบบ 4-way Handshake โดยผู้รับจะเริ่มขอสถาปนาการเชื่อมต่อโดยการส่ง INIT segment เมื่อผู้ส่งได้รับ INIT segment จะตอบรับการขอสถาปนาโดยส่ง INIT-ACK segment กลับให้ผู้รับตาม IP Address ที่ผู้รับส่งมาโดนแนบกลุ่มตัวเลขสุ่มที่ server สร้างขึ้นมาเรียกว่า COOKIE แต่ผู้ส่ง (server) จะยังไม่ทำการจองหน่วยความจำของระบบ ซึ่งจะต่างจาก TCP ที่จะจองหน่วยความจำของระบบตั้งแต่กระบวนการนี้ ซึ่งทำให้เสี่ยงต่อการถูกโจมตีด้วย SYN-flood เมื่อผู้รับได้รับ INIT-ACK segment และ COOKIE แล้วจะส่ง COOKIE ที่ได้ไปกับ COOKIE-ECHO

segment เมื่อผู้ส่งได้รับ COOKIE-ECHO segment และตรวจสอบว่าเป็น COOKIE ที่ตนเองสร้างขึ้นมาจริง ผู้ส่งก็จะจองหน่วยความจำของระบบเพื่อใช้ในการรับ-ส่งข้อมูลและส่ง COOKIE-ACK segment กลับ และเริ่มทำการรับ-ส่งข้อมูล



ภาพที่ 3 Multi-Homing

จากรูปแบบการขอสถาปนาการเชื่อมต่อแบบ 4-way Handshake ซึ่งมีขั้นตอนมากขึ้นอาจทำให้ประสิทธิภาพลดลง แต่ในการทำงานจริงแล้วในการสถาปนาการเชื่อมต่อในโพรโทคอล SCTP ผู้รับสามารถส่งข้อมูลแนบไปกับ COOKIE-ECHO segment ได้ ซึ่งสมารถนับได้ว่าเป็น 3 ขั้นตอนเทียบเท่ากับโพรโทคอล TCP แต่เพิ่มความปลอดภัยเข้ามา

2.3 ระบบการส่งข้อความด่วน (Instant Messaging)

Instant Messaging ได้ถูกพัฒนาขึ้นเพื่อใช้งานในระบบ PLATO ในยุค 1970 ในเวลาต่อมาช่วงยุค 1980 ได้มีการนำมาใช้ในวงการศึกษาและวิศวกร ในชื่อ UNIX talk และแพร่หลายในระบบอินเทอร์เน็ตอย่างรวดเร็ว ในช่วงต้นยุค 1990 ได้มีการพัฒนา ICQ มาใช้เป็นเมสเซนเจอร์ในเครื่องที่ไม่ใช่ UNIX เป็นครั้งแรก หลังจากนั้นได้มีการพัฒนาเมสเซนเจอร์ เป็นจำนวนมากโดยใช้โพรโทคอลของตัวเอง ทำให้มีผู้ใช้งานโปรแกรมเมสเซนเจอร์หลายตัวภายในเครื่องเดียวกัน หรือผู้ใช้งานสามารถเลือกได้ที่จะใช้โปรแกรมที่รองรับการทำงานหลายโพรโทคอล ได้แก่ Gaim หรือ Trillian ในปัจจุบัน นอกจากความสามารถในการส่งข้อความบนระบบการส่งข้อความด่วนต่างๆ ได้รองรับการใช้งาน การส่งไฟล์ การประชุมออนไลน์ และ VoIP ลักษณะการทำงานของแอพลิเคชันประเภท Instant Messenger เมื่อพิจารณาลักษณะวิธีการส่งผ่านข้อมูลจากตนทางไปยังปลายทาง จะพบว่าสามารถจำแนกรูปแบบการทำงานได้ 3 รูปแบบ คือ

แบบClient-Server การทำงานแบบ Client-Server จะอาศัย Server เป็นตัวกลางเก็บเส้นทางการเชื่อมต่อไว้โดยเมื่อ client เริ่มใช้งานจะต้องมีการลงทะเบียนและ update สถานะไว้กับ server หลักโดย client ในระบบนี้จะไม่มีการติดต่อสื่อสารข้อมูลกันโดยตรง ข้อดีของระบบนี้คือ การปลอมตัวตนทำได้ยาก การ update สถานะของ client ให้ client อื่นรับทราบตลอดจนเถียรภาพการส่งผ่านข้อมูลสูง แต่อย่างไรก็ตาม การส่งผ่านข้อมูลระบบนี้ อาจมี cost ด้านระยะเวลาเนื่องจากข้อมูลต้องส่งผ่านตัวกลาง ดังนันอาจมีปัญหากับข้อมูลประแภท streaming media

แบบ Peer to Peer การทำงานแบบ Peer to Peer จะไม่อาศัย Server ตัวกลางใดในการเชื่อมต่อและส่งผ่านข้อมูล การรับรู้สถานะและเส้นทางของ Peer ต่างๆสามารถทำได้เนื่องจาก Peer ทุก Peer มีหน้าที่กระจาย package ระบุตัวตนและสถานะออกไปในวงเน็ทเวอร์กเป็นระยะ จุดเด่นของระบบนี้คือความรวดเร็วและความทันสมัยของข้อมูล

แบบผสม (Hybrid) เป็นการนำเอาจุดเด่นหรือข้อดีทั้งสองแบบมาใช้ร่วมกันโดยอาศัยการ update สถานะและเส้นทางจาก Server ศูนย์กลาง แต่การส่งข้อมูลระหว่าง Client ในลักษณะการ streaming จะทำกันโดยตรง ลักษณะการทำงานในรูปแบบนี้ถูกใช้งานในหลายแอพลิเคชันในปัจจุบัน

3 งานวิจัยที่เกี่ยวข้อง

งานวิจัย (Brennan and Curran, 2001) และงานวิจัย (Caro et al., 2003) ทำการทดลอง congestion

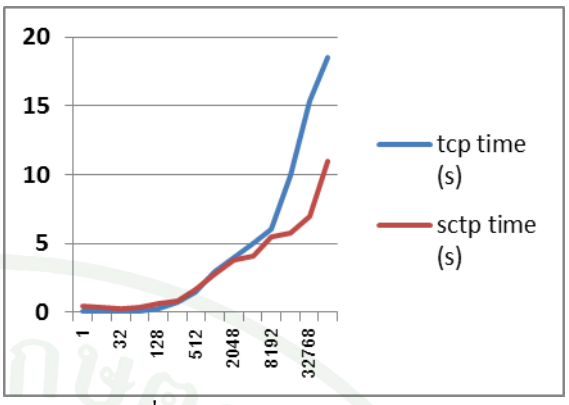
control ใน SCTP โดยทำการทดสอบทั้งในเครือข่ายที่มีความคับคั่งสูงและเครือข่ายที่มีการสูญหายของข้อมูลสูง โดยกำหนดให้ SCTP ทำงานเพียง stream เดียว และทำการปรับรูปแบบการทำงาน ให้ทำงานสอดคล้องกับการทำงานของ TCP ผลการทดลองสรุปว่า SCTP และ TCP มีประสิทธิภาพและมีการจัดการความคับคั่งที่ใกล้เคียงกัน

งานวิจัย (Grinnemo et al., 2005) ทดลองส่งข้อมูลแบบ ordered และ unordered ใน SCTP เพื่อเปรียบเทียบและทดสอบประสิทธิภาพการแก้ปัญหา HOL จากการทดลองผลสรุปว่า การส่งข้อมูลแบบ unordered สามารถส่งข้อมูลได้เร็วกว่าแบบ ordered ประมาณ 0% - 18%

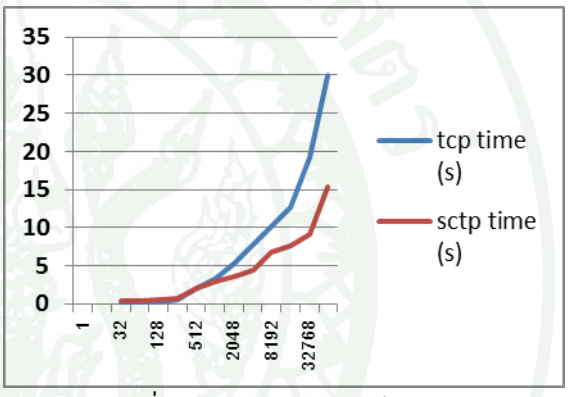
งานวิจัย (Natarajan et al., 2006) ทดลองโดยการจำลอง web server ที่ทำงานบน HTTP และใช้ SCTP ส่งข้อมูลแบบ multi-streaming เพื่อแก้ปัญหา HOL โดยทดลองทั้งแบบส่ง object แต่ละชิ้นแยกไปในแต่ละ stream และส่งรวมใน stream เดียว ผลการทดลองสรุปว่า SCTP สามารถแก้ปัญหา HOL ได้ และในการส่ง object แต่ละชิ้นแยก stream ไป ยังทำให้การส่งข้อมูลทั้งหมดมีประสิทธิภาพเพิ่มขึ้น ใช้เวลาในการส่งน้อยลง

4. การวัดผลประสิทธิภาพการทำงาน

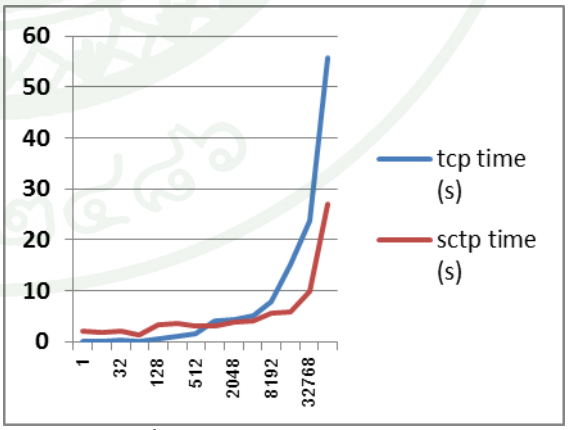
งานวิจัยนี้จัดทำโปรแกรมการทำงานระบบการส่งข้อความด่วน (Instant Messaging) พัฒนาต่อเนื่องจาก Open Source Instant Messenger ซึ่งมีการทำงานในรูปแบบ Peer-to-Peer โดยสร้างโปรแกรมตัวอย่างขึ้นมาสองชุดเพื่อเปรียบทียบประสิทธิภาพการทำงานระหว่างโพรโทคอล TCP และโพรโทคอล SCTP ในสภาพแวดล้อม ที่กำหนดขึ้นและได้ผลการทดสอบแสดงตามแผนภาพ



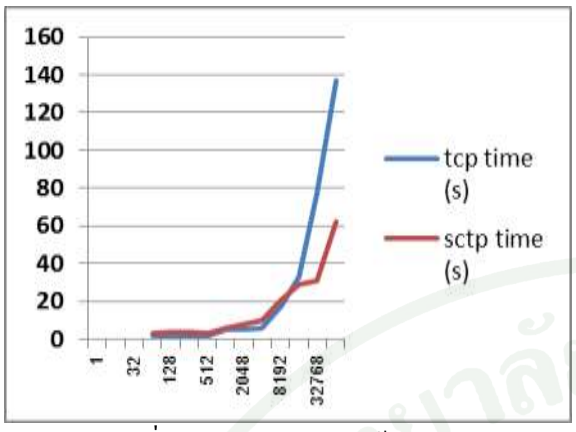
ภาพที่ 4 เปรียบเทียบการส่งข้อมูลแบบ text file บนระบบปิด



ภาพที่ 5 เปรียบเทียบการส่งข้อมูลรูปภาพ jpeg file บนระบบปิด



ภาพที่ 6 เปรียบเทียบการส่งข้อมูลแบบ text บนเครือข่าย traffic 50%



ภาพที่ 7 เปรียบเทียบการส่งข้อมูลรูปภาพ jpeg บนเครือข่าย traffic 50%

จากผลที่ได้แสดงให้เห็นว่าขนาดและประเภทของข้อมูลที่ใช้ในการทำการทดสอบมีความสำพันธ์เกี่ยวเนื่องกับประสิทธิภาพการทำงานที่แตกต่างกันระหว่างโพรโทคอล TCP และโพรโทคอล SCTP ซึ่งพอสรุปได้ดังนี้

ข้อมูลชนิด Text ซึ่งไม่มีการบีบอัดข้อมูลและมีขนาดมีขนาดใหญ่กว่า 1024 byte เมื่อส่งด้วยโพรโทคอล SCTP จะใช้เวลาน้อยกว่าโพรโทคอล TCP ซึ่งประสิทธิภาพที่สูงขึ้นดังกล่าวเป็นอัตราส่วนมากขึ้นเมื่อเพิ่มขนาดของข้อมูล และเป็นที่สังเกตุว่าในระบบเครือข่ายที่มีการจราจรหนาแน่นขึ้นมีผลกระทบเพียงเล็กน้อยต่อประสิทธิภาพการทำงาน

ข้อมูลชนิด Jpeg ซึ่งมีการบีบอัดข้อมูลมีอัตราส่วนประสิทธิภาพและขนาดข้อมูลเป็นไปในแนวทางเดียวกับข้อมูลซึ่งไม่มีการบีบอัด แต่ความหนาแน่นของการจราจรภายในเครือข่ายกลับมีผลต่อประสิทธิภาพอย่างชัดเจนโดยพบว่าเมื่อเครือข่ายมีการจราจรหนาแน่นขึ้นข้อมูลซึ่งถูกส่งด้วยโพรโทคอล SCTP ที่จะใช้เวลาน้อยกว่าโพรโทคอล TCP จะต้องมีขนาดใหญ่ขึ้นด้วย

5.สรุปผล

การทำงานของระบบการส่งข้อความด่วน (Instant Messaging)ซึ่งทำงานบนโพรโทคอล SCTP มีประสิทธิภาพสูงกว่าเมื่อเปรียบเทียบกับการทำงานบนโพรโทคอล TCP เมื่อขนาดของข้อมูลมีขนาดใหญ่ขึ้นและในระบบเครือข่ายที่มีการจราจรหนาแน่นก็เป็นไปในแนวทางเดียวกัน แต่อย่างไรก็ตามการทำงานบนโพรโทคอล SCTP ไม่ได้มีประสิทธิภาพสูงสุดเมื่อข้อมูลมีขนาดเล็ก อันเนื่องมาจากการส่งข้อมูลขนาดเล็กไม่ได้ใช้ประโยชน์จากคุณสมบัติ Multi-Streaming

จากการวิจัยพบว่าเราสามารถเพิ่มประสิทธิภาพการทำงานของระบบการส่งข้อความด่วน (Instant Messaging) ด้วยการปรับใช้โพรโทคอล SCTP และโพรโทคอล TCP อย่างเหมาสมตามลักษณการทำงานและสภาพแวดล้อมเครือข่ายที่เปลี่ยนแปลงไป

งานที่สามารถวิจัยเพิ่มในอนาคตได้แก่การทดสอบประสิทธิภาพการทำงานบนสภาพแวดล้อมที่มีการสูญหายของข้อมูลเพื่อเปรียบเทียบประสิทธิภาพคุณสมบัติ Multi-Homing ร่วมกับ Multi-Streaming ตลอดจนการทดสอบในระบบการส่งข้อความด่วนแบบ Client-Server เป็นต้น

Reference

[1] A. Balk, M. Sigler, M. Gerla, and M.Y. Sanadidi. "Investigation of MPEG-4 video streaming over SCTP," In Proc. IIIS World Multi-Conf. System. Cybern. Inform. (SCI), Orlando, Florida, 2002.

[2] T. Ravier, R. Brennan, and T. Curran, "Experimental studies of SCTP multihoming," First Joint IEI/IEE Symposium on Telecommunications Systems Research, Dublin, Ireland, 2001.

[3] A. Caro, K. Shah, J. Iyengar, P. Amer, R. Stewart, "SCTP and TCP Variants Congestion Control Under Multiple Losses," submitted to ACM Computer Communication Review, 2003.

[4] KJ. Grinnemo, T. Andersson, A. Brunstrom, "Performance Benefits of Avoiding Head-of-Line Blocking in SCTP," Proceedings of the ICAS/ICNS, 2005.

[5] R. Rajamani, S. Kumar, N. Gupta, "SCTP versus TCP Comparing the Performance of Transport Protocols for Web Traffic," University of Wisconsin-Madison, 2002.

ภาคผนวก ข

โค้ดโปรแกรม

ในภาคผนวก ข นี้ประกอบด้วยส่วนของตัวโคดที่มีการแก้ไขดึงมาเฉพาะส่วนที่มีการเพิ่มเติม ซึ่งประกอบไปด้วยฟังก์ชันดังนี้

1. SCTP_Socket
2. SCTP_sender_multi_stream
3. SCTP_receive_multi_stream
4. SCTP_sender_multi_home
5. SCTP_receive_multi_home

1. SCTP_Socket

```
int listenSock, connSock, ret;
struct sockaddr_in servaddr;
char buffer[MAX_BUFFER+1];
time_t currentTime;

/* Create SCTP TCP-Style Socket */
listenSock = socket( AF_INET, SOCK_STREAM, IPPROTO_SCTP );

/* Accept connections from any interface */
bzero( (void *)&servaddr, sizeof(servaddr) );
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl( INADDR_ANY );
servaddr.sin_port = htons(MY_PORT_NUM);

/* Bind to the wildcard address (all) and MY_PORT_NUM */
ret = bind( listenSock,
        (struct sockaddr *)&servaddr, sizeof(servaddr) );

/* Place the server socket into the listening state */
listen( listenSock, 5 );
```

```
    /* Server loop... */
    while( 1 ) {

     /* Await a new client connection */
     connSock = accept( listenSock,
                        (struct sockaddr *)NULL, (int *)NULL );

     /* New client socket has connected */

     /* Grab the current time */
     currentTime = time(NULL);

     /* Send local time on stream 0 (local time stream) */
     snprintf( buffer, MAX_BUFFER, "%s\n", ctime(&currentTime) );

     ret = sctp_sendmsg( connSock,
                         (void *)buffer, (size_t)strlen(buffer),
                         NULL, 0, 0, 0, LOCALTIME_STREAM, 0, 0 );

     /* Send GMT on stream 1 (GMT stream) */
     snprintf( buffer, MAX_BUFFER, "%s\n",
               asctime( gmtime( &currentTime ) ) );

     ret = sctp_sendmsg( connSock,
                         (void *)buffer, (size_t)strlen(buffer),
                         NULL, 0, 0, 0, GMT_STREAM, 0, 0 );

     /* Close the client connection */
     close( connSock );
}
```

2. SCTP_sender_multi_Stream

```
{
 int listenSock, connSock, ret;
 struct sockaddr_in servaddr;
 struct sctp_initmsg initmsg;
 char buffer[MAX_BUFFER+1];
 time_t currentTime;

 /* Create SCTP TCP-Style Socket */
 listenSock = socket( AF_INET, SOCK_STREAM, IPPROTO_SCTP );

 /* Accept connections from any interface */
 bzero( (void *)&servaddr, sizeof(servaddr) );
 servaddr.sin_family = AF_INET;
 servaddr.sin_addr.s_addr = htonl( INADDR_ANY );
 servaddr.sin_port = htons(MY_PORT_NUM);

 ret = bind( listenSock, (struct sockaddr *)&servaddr, sizeof(servaddr) );

 /* Specify that a maximum of 5 streams will be available per socket */
 memset( &initmsg, 0, sizeof(initmsg) );
 initmsg.sinit_num_ostreams = 5;
 initmsg.sinit_max_instreams = 5;
 initmsg.sinit_max_attempts = 4;
 ret = setsockopt( listenSock, IPPROTO_SCTP, SCTP_INITMSG,
                   &initmsg, sizeof(initmsg) );

 /* Place the server socket into the listening state */
 listen( listenSock, 5 );
```

```
  /* Server loop... */
  while( 1 ) {

    /* Await a new client connection */
    printf("Awaiting a new connection\n");
    connSock = accept( listenSock, (struct sockaddr *)NULL, (int *)NULL );

    /* New client socket has connected */

    /* Grab the current time */
    currentTime = time(NULL);

    /* Send local time on stream 0 (local time stream) */
    snprintf( buffer, MAX_BUFFER, "%s\n", ctime(&currentTime) );
    ret = sctp_sendmsg( connSock, (void *)buffer, (size_t)strlen(buffer),
                        NULL, 0, 0, 0, LOCALTIME_STREAM, 0, 0 );

    /* Send GMT on stream 1 (GMT stream) */
    snprintf( buffer, MAX_BUFFER, "%s\n", asctime( gmtime( &currentTime ) ) );
    ret = sctp_sendmsg( connSock, (void *)buffer, (size_t)strlen(buffer),
                        NULL, 0, 0, 0, GMT_STREAM, 0, 0 );

    /* Close the client connection */
    close( connSock );

  }
  return 0;
}
```

3. SCTP_receive_multi_stream

```
{
 int connSock, in, i, ret, flags;
 struct sockaddr_in servaddr;
 struct sctp_status status;
 struct sctp_sndrcvinfo sndrcvinfo;
 struct sctp_event_subscribe events;
 struct sctp_initmsg initmsg;
 char buffer[MAX_BUFFER+1];

 /* Create an SCTP TCP-Style Socket */
 connSock = socket( AF_INET, SOCK_STREAM, IPPROTO_SCTP );

 /* Specify that a maximum of 5 streams will be available per socket */
 memset( &initmsg, 0, sizeof(initmsg) );
 initmsg.sinit_num_ostreams = 5;
 initmsg.sinit_max_instreams = 5;
 initmsg.sinit_max_attempts = 4;
 ret = setsockopt( connSock, IPPROTO_SCTP, SCTP_INITMSG,
                   &initmsg, sizeof(initmsg) );

 /* Specify the peer endpoint to which we'll connect */
 bzero( (void *)&servaddr, sizeof(servaddr) );
 servaddr.sin_family = AF_INET;
 servaddr.sin_port = htons(MY_PORT_NUM);
 servaddr.sin_addr.s_addr = inet_addr( "127.0.0.1" );

 /* Connect to the server */
 ret = connect( connSock, (struct sockaddr *)&servaddr, sizeof(servaddr) );
```

```
/* Enable receipt of SCTP Snd/Rcv Data via sctp_recvmsg */
memset( (void *)&events, 0, sizeof(events) );
events.sctp_data_io_event = 1;
ret = setsockopt( connSock, SOL_SCTP, SCTP_EVENTS,
                (const void *)&events, sizeof(events) );

/* Read and emit the status of the Socket (optional step) */
in = sizeof(status);
ret = getsockopt( connSock, SOL_SCTP, SCTP_STATUS,
                (void *)&status, (socklen_t *)&in );

printf("assoc id  = %d\n", status.sstat_assoc_id );
printf("state     = %d\n", status.sstat_state );
printf("instrms   = %d\n", status.sstat_instrms );
printf("outstrms  = %d\n", status.sstat_outstrms );

/* Expect two messages from the peer */

for (i = 0 ; i < 2 ; i++) {

  in = sctp_recvmsg( connSock, (void *)buffer, sizeof(buffer),
                  (struct sockaddr *)NULL, 0, &sndrcvinfo, &flags );

  if (in > 0) {
    buffer[in] = 0;
    if (sndrcvinfo.sinfo_stream == LOCALTIME_STREAM) {
      printf("(Local) %s\n", buffer);
    } else if (sndrcvinfo.sinfo_stream == GMT_STREAM) {
      printf("(GMT  ) %s\n", buffer);
```

```
    }
   }

  }

  /* Close our socket and exit */
  close(connSock);

  return 0;
 }
```

4. SCTP_sender_multi_home

```
 {
   if((sock = socket(AF_INET, SOCK_SEQPACKET, IPPROTO_SCTP)) < 0)
      perror("socket");

   memset(&addr,       0, sizeof(struct sockaddr_in));
   memset(&event,      1, sizeof(struct sctp_event_subscribe));
   memset(&heartbeat,  0, sizeof(struct sctp_paddrparams));
   memset(&rtoinfo,    0, sizeof(struct sctp_rtoinfo));

   addr.sin_family = AF_INET;
   addr.sin_addr.s_addr = htonl(INADDR_ANY);
   addr.sin_port = htons(PORT);

   from_len = (socklen_t)sizeof(struct sockaddr_in);

   sig_handler.sa_handler = handle_signal;
   sig_handler.sa_flags = 0;
```

```
    heartbeat.spp_flags = SPP_HB_ENABLE;
    heartbeat.spp_hbinterval = 5000;
    heartbeat.spp_pathmaxrxt = 1;

    rtoinfo.srto_max = 2000;

    /*Set Heartbeats*/
    if(setsockopt(sock, SOL_SCTP, SCTP_PEER_ADDR_PARAMS , &heartbeat,
sizeof(heartbeat)) != 0)
        perror("setsockopt");

    /*Set rto_max*/
    if(setsockopt(sock, SOL_SCTP, SCTP_RTOINFO , &rtoinfo, sizeof(rtoinfo)) != 0)
        perror("setsockopt");

    /*Set Signal Handler*/
    if(sigaction(SIGINT, &sig_handler, NULL) == -1)
        perror("sigaction");

    /*Set Events */
    if(setsockopt(sock, IPPROTO_SCTP, SCTP_EVENTS, &event, sizeof(struct
sctp_event_subscribe)) < 0)
        perror("setsockopt");

    /*Set the Reuse of Address*/
    if(setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &reuse, sizeof(int))< 0)
        perror("setsockopt");

    /*Bind the Addresses*/
```

```
    if(bind(sock, (struct sockaddr*)&addr, sizeof(struct sockaddr)) < 0)
        perror("bind");

    if(listen(sock, 2) < 0)
        perror("listen");

    /*Get Heartbeat Value*/
    i = (sizeof heartbeat);
    getsockopt(sock, SOL_SCTP, SCTP_PEER_ADDR_PARAMS, &heartbeat,
(socklen_t*)&i);
    printf("Heartbeat interval %d\n", heartbeat.spp_hbinterval);

    /*Print Locally Binded Addresses*/
    addr_count = sctp_getladdrs(sock, 0, (struct sockaddr**)laddr);
    printf("Addresses binded: %d\n", addr_count);
    for(i = 0; i < addr_count; i++)
        printf("Address %d: %s:%d\n", i +1, inet_ntoa((*laddr)[i].sin_addr),
(*laddr)[i].sin_port);
    sctp_freeladdrs((struct sockaddr*)*laddr);

    while(1)
    {
        flags = 0;

        ret = sctp_recvmsg(sock, buffer, BUFFER_SIZE, NULL, 0, NULL, &flags);

        if(flags & MSG_NOTIFICATION)
        printf("Notification received from %s:%u\n",  inet_ntoa(addr.sin_addr),
ntohs(addr.sin_port));
```

```
        printf("%d bytes received from %s:%u\n", ret, inet_ntoa(addr.sin_addr),
    ntohs(addr.sin_port));
    }

    if(close(sock) < 0)
        perror("close");
}

void handle_signal(int signum)
{
    switch(signum)
    {
        case SIGINT:
            if(close(sock) != 0)
                perror("close");
            exit(0);
            break;
        default: exit(0);
            break;
    }
}
```

5. SCTP_receive_multi_home

```
{
    int i;
    int counter = 0;
    int asconf = 1;
    int ret;
    int addr_count;
```

```
char address[16];
char buffer[MSG_SIZE];
sctp_assoc_t id;
struct sockaddr_in addr;
struct sctp_status status;
struct sctp_initmsg initmsg;
struct sctp_event_subscribe events;
struct sigaction sig_handler;
struct sctp_paddrparams heartbeat;
struct sctp_rtoinfo rtoinfo;

memset(&buffer,     'j', MSG_SIZE);
memset(&initmsg,    0,   sizeof(struct sctp_initmsg));
memset(&addr,       0,   sizeof(struct sockaddr_in));
memset(&events,     1,   sizeof(struct sctp_event_subscribe));
memset(&status,     0,   sizeof(struct sctp_status));
memset(&heartbeat,  0,   sizeof(struct sctp_paddrparams));
memset(&rtoinfo,    0, sizeof(struct sctp_rtoinfo))

if(argc != 2 || (inet_addr(argv[1]) == -1))
{
   puts("Usage: client [IP ADDRESS in form xxx.xxx.xxx.xxx] ");
   return 0;
}

strncpy(address, argv[1], 15);
address[15] = 0;

addr.sin_family = AF_INET;
inet_aton(address, &(addr.sin_addr));
```

```
    addr.sin_port = htons(PORT);

    initmsg.sinit_num_ostreams = 2;
    initmsg.sinit_max_instreams = 2;
    initmsg.sinit_max_attempts = 1;

    heartbeat.spp_flags = SPP_HB_ENABLE;
    heartbeat.spp_hbinterval = 5000;
    heartbeat.spp_pathmaxrxt = 1;

    rtoinfo.srto_max = 2000;

    sig_handler.sa_handler = handle_signal;
    sig_handler.sa_flags = 0;

    /*Handle SIGINT in handle_signal Function*/
    if(sigaction(SIGINT, &sig_handler, NULL) == -1)
       perror("sigaction");

    /*Create the Socket*/
    if((ret = (sock = socket(AF_INET, SOCK_STREAM, IPPROTO_SCTP))) < 0)
       perror("socket");

    /*Configure Heartbeats*/
    if((ret = setsockopt(sock, SOL_SCTP, SCTP_PEER_ADDR_PARAMS , &heartbeat,
sizeof(heartbeat))) != 0)
       perror("setsockopt");

    /*Set rto_max*/
```

```
   if((ret = setsockopt(sock, SOL_SCTP, SCTP_RTOINFO , &rtoinfo, sizeof(rtoinfo))) !=
0)
      perror("setsockopt");

   /*Set SCTP Init Message*/
   if((ret = setsockopt(sock, SOL_SCTP, SCTP_INITMSG, &initmsg, sizeof(initmsg))) !=
0)
      perror("setsockopt");

   /*Enable SCTP Events*/
   if((ret = setsockopt(sock, SOL_SCTP, SCTP_EVENTS, (void *)&events,
sizeof(events))) != 0)
      perror("setsockopt");

   /*Get And Print Heartbeat Interval*/
   i = (sizeof heartbeat);
   getsockopt(sock, SOL_SCTP, SCTP_PEER_ADDR_PARAMS, &heartbeat,
(socklen_t*)&i);

   printf("Heartbeat interval %d\n", heartbeat.spp_hbinterval);

   /*Connect to Host*/
   if(((ret = connect(sock, (struct sockaddr*)&addr, sizeof(struct sockaddr)))) < 0)
   {
      perror("connect");
      close(sock);
      exit(0);
   }

   /*Get Peer Addresses*/
```

```
    addr_count = sctp_getpaddrs(sock, 0, (struct sockaddr**)paddrs);
    printf("\nPeer addresses: %d\n", addr_count);

    /*Print Out Addresses*/
    for(i = 0; i < addr_count; i++)
       printf("Address %d: %s:%d\n", i +1, inet_ntoa((*paddrs)[i].sin_addr),
(*paddrs)[i].sin_port);

    sctp_freepaddrs((struct sockaddr*)*paddrs);

    /*Start to Send Data*/
    for(i = 0; i < NUMBER_OF_MESSAGES; i++)
    {
       counter++;
       printf("Sending data chunk #%d...", counter);

       if((ret = send(sock, buffer, MSG_SIZE, 0)) == -1)
          perror("write");

       printf("Sent %d bytes to peer\n",ret);

       sleep(1);
    }

    if(close(sock) != 0)
       perror("close");
}

void handle_signal(int signum)
```

```
{
   switch(signum)
   {
      case SIGINT:
         if(close(sock) != 0)
            perror("close");
         exit(0);
         break;
      default: exit(0);
         break;
   }
}
```

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	วโรดม ลิมปะพันธุ์
วัน เดือน ปี ที่เกิด	วันที่ 21 พฤจิกายน 2516
สถานที่เกิด	กรุงเทพมหานคร
ประวัติการศึกษา	ระดับปริญญาตรี คณะวิศวกรรมศาสตร์ สาขาวิศวกรรมคอมพิวเตอร์
ตำแหน่งหน้าที่การงานปัจจุบัน	วิศวกรควบคุมระบบ
สถานที่ทำงานปัจจุบัน	บริษัท มติชนจำกัด (มหาชน)
ผลงานดีเด่นและรางวัลทางวิชาการ	
ทุนการศึกษาที่ได้รับ	