ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์)

ปริญญา

วิทยาการคอมพิวเตอร์ | วิทยาการคอมพิวเตอร์

สาขา | **ภาควิชา**

เรื่อง การศึกษาเปรียบเทียบประสิทธิภาพการทำงานร่วมกับ TCP และ SCTP ของ P2P File Sharing Application

Comparison Study on the Performance of P2P File Sharing Application using TCP vs. SCTP

นามผู้วิจัย นายธีรชัย เจนกิจพาณิชย์กุล

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก .....

(ผู้ช่วยศาสตราจารย์ชวลิต ศรีสถาพรพัฒน์, Ph.D.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม .....

(ผู้ช่วยศาสตราจารย์สุขุมาล กิติสิน, Ph.D.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม .....

(อาจารย์ชัยพร ใจแก้ว, Ph.D.)

หัวหน้าภาควิชา .....

(ผู้ช่วยศาสตราจารย์ศิริกร จันทร์นวล, M.S.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

วันที่ ..... **เดือน** ..... **พ.ศ.** .....

วิทยานิพนธ์

เรื่อง

การศึกษาเปรียบเทียบประสิทธิภาพการทำงานร่วมกับ TCP และ SCTP ของ P2P File Sharing Application

Comparison Study on the Performance of P2P File Sharing Application using TCP vs. SCTP

โดย

นายธีรชัย เจนกิจพาณิชย์กุล

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

เพื่อความสมบูรณ์แห่งปริญญาวิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์)

พ.ศ. 2552

ธีรชัย เจนกิจพาณิชย์กุล 2552: การศึกษาเปรียบเทียบประสิทธิภาพการทำงานร่วมกับ TCP และ SCTP ของ P2P File Sharing Application ปริญญาวิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์) สาขาวิทยาการคอมพิวเตอร์ ภาควิชาวิทยาการคอมพิวเตอร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: ผู้ช่วยศาสตราจารย์ชวลิต ศรีสถาพรพัฒน์, Ph.D. 78 หน้า

ระบบเครือข่าย P2P (peer-to-peer) ที่ใช้ TCP ในการทำงาน ในการแลกเปลี่ยนข้อมูล พบว่า มีปัญหา HOL (head-of-line blocking) ซึ่งปัญหานี้ทำให้ TCP ไม่เหมาะต่อการนำมาใช้ในระบบเครือข่ายที่มีการย่อยข้อมูลออกเป็นส่วน ๆ เพื่อแลกเปลี่ยน จึงได้มีการพัฒนา SCTP ขึ้นมาเพื่อลดปัญหาดังกล่าว งานวิจัยนี้ได้ทำการศึกษาเปรียบเทียบการทำงานร่วมกับ TCP และ SCTP ของ P2P File Sharing Application งานวิจัยนี้ประกอบด้วยหกส่วน ได้แก่ ส่วนแรกบทนำและความสำคัญของปัญหา ส่วนที่สองงานวิจัยที่เกี่ยวข้อง ส่วนที่สามแนวทางการปรับปรุง BitTorrent ให้ทำงานร่วมกับ SCTP ส่วนที่สี่ผลการทดลองเบื้องต้น ส่วนที่ห้าและหกสรุปแนวทางการวิจัยในอนาคตและเอกสารอ้างอิง ตามลำดับ ผลการทดลองเมื่อนำเอา SCTP มาปรับใช้ใน P2P พบว่าด้วยคุณสมบัติ multi-streaming ของ SCTP สามารถขจัดปัญหา HOL ได้ อีกทั้งพบว่ามีความเร็วในการดาวน์โหลดไฟล์มากขึ้น 10 ถึง 65 เปอร์เซ็นต์ และมีแนวโน้มว่าจะมีความเร็วเพิ่มขึ้นเมื่อไฟล์ขนาดใหญ่ขึ้น

_____ _____ ___ / ___ / ___

ลายมือชื่อนิสิต ลายมืออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Teerachai Jenkitpanitkul 2009: Comparison Study on the Performance of P2P File Sharing Application using TCP vs. SCTP. Master of Science (Computer Science), Major Field: Computer Science, Department of Computer Science. Thesis Advisor: Assistant Professor Chavallit Srisathapornphat, Ph.D. 78 pages.

P2P file sharing applications using TCP suffer from head-of-line blocking problem. This research attempts to alleviate this by modifying BitTorrent, one of the most popular P2P file sharing applications, to work with SCTP instead of TCP. SCTP allows an application to simultaneously establish more than one Transport layer connection to a target application on the remote machine. This feature is called multi-streaming. Our experimental result indicates 10 to 65% speed up in total file transfer time when using BitTorrent with SCTP over TCP. The speed up tend to increase with the size of file transferred.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จ และสามารถแก้ปัญหาที่เกิดขึ้นได้ ด้วยความช่วยเหลือด้านคำแนะนำ ให้คำปรึกษา ตลอดจนกระบวนการคิดต่าง ๆ ที่เป็นประโยชน์ในการทำงานวิจัย ซึ่งผู้วิจัยขอกราบขอบพระคุณมา ณ โอกาสนี้

ขอขอบคุณ ผศ.ดร.ชวลิต ศรีสถาพรพัฒน์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่คอยให้คำปรึกษา ช่วยเหลือในการแก้ไขปัญหาต่าง ๆ ตลอดจนตรวจทานงานวิจัยให้กับผู้วิจัยมาโดยตลอด

ขอขอบคุณ ผศ.ดร.สุขุมาล กิติสิน และ ดร.ชัยพร ใจแก้ว ผู้ซึ่งมอบความรู้ คำแนะนำ และคอยผลักดันในการทำงานวิจัยมาโดยตลอด

ขอขอบคุณเจ้าหน้าที่ภาควิชาวิทยาการคอมพิวเตอร์ทุกท่าน ที่ให้ความช่วยเหลือและอำนวยความสะดวกในการทำงานวิจัย

สุดท้ายที่ขาดไม่ได้ คือ ขอกราบขอบพระคุณ คุณพ่อและคุณแม่ที่ให้โอกาสทางการศึกษาและคอยเป็นกำลังใจในการทำงานมาโดยตลอด

ธีรชัย เจนกิจพาณิชย์กุล

เมษายน 2552

สารบัญ

หน้า

สารบัญ	(1)
สารบัญตาราง	(2)
สารบัญภาพ	(3)
คำอธิบายสัญลักษณ์และคำย่อ	(4)
คำนำ	1
วัตถุประสงค์	4
การตรวจเอกสาร	5
อุปกรณ์และวิธีการ	27
อุปกรณ์	27
วิธีการ	28
ผลและวิจารณ์	30
ผล	30
วิจารณ์	41
สรุปและข้อเสนอแนะ	42
สรุป	42
ข้อเสนอแนะ	44
เอกสารและสิ่งอ้างอิง	45
ภาคผนวก	49
ภาคผนวก ก แผนผังความคิด เรื่องการดำเนินการโครงงานวิทยานิพนธ์	50
ภาคผนวก ข โปสเตอร์ การเข้าร่วมประชุมประจำปี สวทช. 2550	52
ภาคผนวก ค บทความเสนองานประชุมประจำปี สวทช. 2550	54
ภาคผนวก ง บทความ เสนอ การประชุมวิชาการทางคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ประจำปี 2552	58
ภาคผนวก จ Source Code ส่วนที่ได้ทำการแก้ไข	69
ประวัติการศึกษา และการทำงาน	78

สารบัญตาราง

ตารางที่		หน้า
1	เปรียบเทียบ TCP และ SCTP	13
2	เปรียบเทียบเวลาที่ใช้ในการดาวน์โหลดผ่าน Modem ที่ขนาดไฟล์ต่าง ๆ	33
3	ค่าเฉลี่ยของเวลาที่ใช้ในการดาวน์โหลดของ Modem (56 Kbps)	37
4	ค่าเฉลี่ยของเวลาที่ใช้ในการดาวน์โหลดของ ADSL (2 Mbps)	38
5	ค่าเฉลี่ยของเวลาที่ใช้ในการดาวน์โหลดของ WI-FI (54 Mbps)	39
6	ค่าเฉลี่ยของเวลาที่ใช้ในการดาวน์โหลดของ LAN (100 Mbps)	40

สารบัญภาพ

ภาพที่		หน้า
1	3-way handshake (T. Jones, 2006)	6
2	TCP server lifecycle (J. Kurose, K. Ross, 2002)	8
3	TCP client lifecycle (J. Kurose, K. Ross, 2002)	8
4	Multi-Streaming	10
5	Multi-Homing (T. Jones, 2006)	11
6	4-way handshake (T. Jones, 2006)	12
7	ไดอะแกรมระบบทดลอง (A. Caro, 2003)	18
8	จำนวนแพ็คเกตที่สามารถส่งได้เมื่อมีอัตราการสูญหายเพิ่มขึ้น (A. Caro, 2003)	18
9	เวลาที่ใช้ในการส่งแบบ ordered และ unordered (KJ. Grinnemo, 2005)	19
10	สภาพแวดล้อมเครือข่ายทดลอง (P. Natarajan, 2006)	20
11	เวลาที่ใช้ในการส่ง object แบบ multi-streaming (P. Natarajan, 2006)	21
12	เวลาที่ใช้ในการดาวน์โหลดเมื่ออัตราการสูญหายสูง (R. Rajamani, 2002)	22
13	Response Time ของ TCP (บนระบบปฏิบัติการ Linux/Solaris)	23
14	Response Time ของ SCTP (บนระบบปฏิบัติการ Linux)	24
15	การส่งข้อมูลผ่าน TCP บนระบบปฏิบัตการ Linux (Multi-Connection)	25
16	การส่งข้อมูลผ่าน SCTP บนระบบปฏิบัตการ Linux (Multi-Steaming)	26
17	กราฟเวลาดาวน์โหลดไฟล์	31
18	เวลาดาวน์โหลดข้อมูล 100MB ของไฟล์ขนาดต่างๆ ระหว่าง TCP vs. SCTP	32
19	เวลาที่ใช้ในการดาวน์โหลดผ่าน Modem (56 Kbps) ที่ขนาดไฟล์ต่าง ๆ	34
20	สภาพแวดล้อมในการทดลอง กรณีที่ 2	35
21	เวลาที่ใช้ในการดาวน์โหลดผ่าน Modem (56 Kbps) ระหว่าง TCP และSCTP	37
22	เวลาที่ใช้ในการดาวโหลดผ่าน ADSL (2 Mbps) ระหว่าง TCP และSCTP	38
23	เวลาที่ใช้ในการดาวน์โหลดผ่าน WI-FI (54 Mbps) ระหว่าง TCP และSCTP	39
24	เวลาที่ใช้ในการดาวน์โหลดผ่าน LAN (100 Mbps) ระหว่าง TCP และSCTP	40

คำอธิบายสัญลักษณ์และคำย่อ

GB	=	Gigabyte
HOL	=	Head-of-line blocking
HTTP	=	Hypertext Transfer Protocol
IETF	=	Internet Engineering Task Force
Kbps	=	Kilobit per second
Kbyte	=	Kilobyte
MB	=	Megabyte
Mbps	=	Megabit per second
ms	=	Millisecond
MTU	=	Maximum Transfer Unit
NS-2	=	Network Simulator version 2
P2P	=	Peer-to-Peer
RPM	=	Revolution Per Minute
RTT	=	Round-Trip Time
SCTP	=	Stream Control Transmission Protocol
SIGTRAN	=	Signaling Transport
TCP	=	Transmission Control Protocol

การศึกษาเปรียบเทียบประสิทธิภาพการทำงานร่วมกับ TCP และ SCTP ของ P2P File Sharing Application

Comparison Study on the Performance of P2P File Sharing Application using TCP vs. SCTP

คำนำ

ระบบเครือข่ายคอมพิวเตอร์ในยุคเริ่มแรกมีลักษณะการทำงานเป็นระบบรับ-ให้บริการ หรือเรียกว่า client/server system ในการทำงานของเครื่องลูกข่าย (client) จะต้องติดต่อขอรับบริการจากเครื่องแม่ข่าย (server) โดยตรง ซึ่ง server ในระบบเครือข่ายอาจมีจำนวนน้อย ทำให้ server ต้องรับภาระการทำงานหนักและต้องใช้เครื่องที่มีประสิทธิภาพสูง โดยเฉพาะอย่างยิ่งในกรณีที่มี client จำนวนมาก การส่งข้อมูลระหว่าง client และ server จะเพิ่มมากขึ้นตามไปด้วย ส่งผลให้ต้องใช้ระบบเครือข่ายที่มีประสิทธิภาพสูงขึ้น เพื่อรองรับปริมาณการสื่อสารข้อมูลที่เพิ่มขึ้น นอกจากนี้ถ้า server ขัดข้อง ไม่สามารถให้บริการได้จะทำให้ client ทุกเครื่องไม่สามารถทำงานได้

ในปัจจุบันระบบเครือข่ายคอมพิวเตอร์มีการทำงานในลักษณะ peer-to-peer โดยที่เครื่องคอมพิวเตอร์แต่ละเครื่องมีหน้าที่เป็นทั้ง server และ client ในเวลาเดียวกัน ดังนั้นระบบเครือข่ายจึงสามารถรองรับจำนวนเครื่องคอมพิวเตอร์ที่ใช้งานเครือข่ายได้เพิ่มมากขึ้น โดยที่เครื่องคอมพิวเตอร์แต่ละเครื่องไม่จำเป็นต้องเพิ่มประสิทธิภาพของเครื่องตนเองแต่อย่างใด ยังคงสามารถใช้อุปกรณ์ที่มีในแต่ละเครื่องได้เหมือนเดิม ทั้งนี้เนื่องจากเมื่อมีเครื่องคอมพิวเตอร์ในระบบมาก จะยิ่งมี server มากทำให้ server แต่ละเครื่องไม่ต้องรับภาระให้บริการกับ client จำนวนมาก และข้อดีอีกประการหนึ่งของระบบนี้ คือ client แต่ละเครื่องยังสามารถขอรับบริการจาก server ได้มากกว่าหนึ่งเครื่องในเวลาเดียวกัน ทำให้ได้รับบริการที่รวดเร็วและมีประสิทธิภาพมากขึ้น ลดปัญหาระบบเครือข่ายหยุดทำงาน และแม้ server เครื่องใดขัดข้องไป ก็ยังมี server อื่นเหลืออยู่ในระบบ ทำให้การสื่อสารในเครือข่ายนั้นยังสามารถดำเนินต่อไปได้ รูปแบบการให้บริการแบบ peer-to-peer มี 3 รูปแบบ คือ แบบมี directory เป็นศูนย์กลาง (centralized P2P) แบบไม่มีศูนย์กลาง (pure P2P) และแบบผสม (hybrid)

BitTorrent เป็นเทคโนโลยีการแลกเปลี่ยนไฟล์ โดยระบบเครือข่ายของ BitTorrent เป็นระบบเครือข่าย peer-to-peer แบบผสมที่มี server ศูนย์กลางทำหน้าที่เก็บรายละเอียดไฟล์ที่มีการแลกเปลี่ยนกันในระบบ โดยเครื่องของผู้ใช้แต่ละเครื่องอาจเป็นทั้งผู้ให้บริการและผู้รับบริการได้ในขณะเดียวกัน BitTorrent จะแยกไฟล์ที่แลกเปลี่ยนกันออกเป็นส่วนย่อย เพื่อให้ client เลือกดาวน์โหลดจาก server หลายเครื่องได้พร้อมกัน ซึ่งการแยกไฟล์ออกเป็นส่วนย่อยนี้ เป็นข้อดีที่ทำให้การโอนถ่ายไฟล์ทำได้รวดเร็วขึ้น สามารถเลือกได้ว่าจะดาวน์โหลดจากส่วนย่อยใดก่อนได้ เนื่องจากถ้าส่วนใดของไฟล์ที่มี server ให้บริการน้อย จะถือเป็นส่วนที่มีความสำคัญมาก จำเป็นต้องทำการดาวน์โหลดเป็นอันดับแรก ๆ เพื่อลดปัญหาการดาวน์โหลดไฟล์ไม่เสร็จสมบูรณ์

ปัญหาประการหนึ่งของ BitTorrent คือ BitTorrent ใช้ TCP (transmission Control Protocol) เป็น Transport layer protocol เนื่องจากการทำงานของ TCP มีการสถาปนาการเชื่อมต่อ (connection oriented) มีความน่าเชื่อถือในการส่งข้อมูล (reliable) มีการจัดเรียงลำดับ (ordering) มีการควบคุมความคับคั่ง (congestion control) มีการควบคุมการไหล (flow control) มีการแยกข้อมูลออกเป็นส่วนย่อย (fragmentation) ซึ่งจะส่งข้อมูลให้กับ application ต่อเมื่อข้อมูลที่ได้รับเรียงลำดับอย่างถูกต้องแล้ว ดังนั้นถ้าหากมีการสูญหายหรือล่าช้าของข้อมูลที่ได้รับมา TCP จะไม่ส่งข้อมูลให้ application โดยที่ TCP จะทำการร้องขอข้อมูลส่วนที่สูญหายหรือล่าช้าดังกล่าวจนกว่าจะได้รับอย่างถูกต้อง จึงดำเนินการส่งข้อมูลให้กับ application ปัญหานี้ของ TCP เรียกว่า head-of-line blocking (HOL) ซึ่งทำให้ TCP ไม่เหมาะสมที่จะนำมาใช้สำหรับการแลกเปลี่ยนไฟล์ใน BitTorrent เนื่องจากทำให้ไม่ได้รับประโยชน์จากการแยกไฟล์ออกเป็นส่วนย่อย

นอกจากนี้ปัญหาที่สำคัญของโพรโทคอล TCP อีกประการหนึ่ง คือ ปัญหาการที่ TCP สร้างช่องทางการสื่อสารเพียงช่องทางเดียว ถ้าช่องทางการสื่อสารนั้นเกิดปัญหา HOL แล้ว จะไม่สามารถสื่อสารข้อมูลใด ๆ ได้ในช่วงระยะเวลาหนึ่ง

จากปัญหาที่พบในการใช้งาน BitTorrent บนโพรโทคอล TCP นี้ ทำให้ผู้วิจัยศึกษาแนวทางที่จะนำโพรโทคอลอื่น ที่มีความสามารถสนับสนุนการรับ-ส่งไฟล์แบบแยกส่วนและมีการสร้างช่องทางการสื่อสารมากกว่าหนึ่งช่องทาง ซึ่งทำงานในเลเยอร์เดียวกับ TCP มาเพิ่มประสิทธิภาพให้กับ BitTorrent โดยได้พบว่า SCTP (Stream Control Transmission Protocol) มีความสามารถดังกล่าวครบถ้วน คือ SCTP ทำงานในชั้น Transport layer มีการสถาปนาการเชื่อมต่อ มีความน่าเชื่อถือในการส่งข้อมูล มีการควบคุมความคับคั่ง มีการควบคุมการไหล มีการแยกข้อมูลออกเป็น

ส่วนย่อย และที่แตกต่าง คือ สามารถสร้างช่องทางการสื่อสารได้มากกว่าหนึ่งช่องทาง (multi-streaming) มีระบบความมั่นคงดีกว่า TCP สามารถใช้งาน receive windows ที่มีขนาดถึง 32 บิต ทำให้รองรับประสิทธิภาพของระบบเครือข่ายที่เพิ่มขึ้นได้

ผลที่คาดว่าจะได้รับเมื่อใช้งาน BitTorrent ร่วมกับ SCTP คือ จะสามารถแลกเปลี่ยนไฟล์ได้รวดเร็วขึ้น ลดปัญหา HOL มีระบบความมั่นคงในการเชื่อมต่อสูงขึ้น และสามารถใช้ประโยชน์จาก bandwidth ที่สูงขึ้นจากระบบเครือข่ายได้

วัตถุประสงค์

1. เพื่อศึกษากระบวนการแก้ปัญหา HOL ของโพรโทคอล SCTP

2. เพื่อพัฒนา BitTorrent ให้สามารถทำงานร่วมกับโพรโทคอล SCTP

3. เพื่อศึกษาและเปรียบเทียบประสิทธิภาพการดาวน์โหลดไฟล์ของ BitTorrent เมื่อทำงานกับโพรโทคอล TCP และ SCTP

การตรวจเอกสาร

Transport layer

International Organization for Standardization (1983) เป็นองกรร์ที่คิดค้น OSI (Open System Interconnection) ซึ่ง OSI คือ model มาตรฐานในการสื่อสาร แบ่งออกเป็น 7 layer ด้วยกัน Transport layer ถือเป็น layer ที่ 4 ของ OSI model ชั้นนี้จะคอยตรวจสอบข้อผิดพลาดในการส่งและรับข้อมูล โดยจัดการการส่งข้อมูลจากเครื่องต้นทางจนถึงปลายทาง

ระบบเครือข่ายคอมพิวเตอร์จะสามารถติดต่อสื่อสาร เพื่อส่งข้อมูลระหว่างกันได้นั้น จำเป็นต้องมีผู้ดำเนินการ เพื่อจัดการการติดต่อและส่งข้อมูลระหว่างเครื่องทั้งสองในระบบอินเทอร์เน็ตได้ กำหนดให้ Transport layer ทำหน้าที่ในการจัดการการสื่อสารระหว่างเครื่องคอมพิวเตอร์ต้นทางและปลายทาง (end-to-end) ซึ่งอาจมีการสถาปนาการเชื่อมต่อถึงกัน (connection-oriented) หรือไม่มีการสถาปนาการเชื่อมต่อ (connectionless) ก็ได้ โดย Transport layer จะทำหน้าที่ตรวจสอบความถูกต้องของข้อมูลทั้งก่อนการส่งและหลังจากรับข้อมูลจากเครือข่าย โดยการตรวจสอบข้อมูลก่อนทำการส่งจะเป็นการตรวจสอบขนาดของข้อมูลที่ต้องการส่งว่ามีขนาดใหญ่เกินกว่าขนาดที่ใหญ่ที่สุดของข้อมูลที่เส้นทางนั้นจะรับได้หรือไม่ เรียกว่า MTU (Maximum Transfer Unit) ถ้ามีขนาดใหญ่กว่า MTU แล้ว Transport layer จะทำหน้าที่แบ่งข้อมูล (fragmentation) ที่จะส่งออกเป็นส่วนย่อย ๆ เรียกว่า segment โดยที่ขนาดของแต่ละ segment จะต้องไม่มากกว่า MTU และเมื่อ Transport layer ทางด้านผู้รับได้รับข้อมูลดังกล่าว จะทำการตรวจสอบว่าข้อมูลถูกแบ่งมาหรือไม่ และจะจัดเรียงข้อมูล (ordering) และทำการรวม (reassembly) ข้อมูลที่ได้รับในลำดับที่ถูกต้องก่อนที่จะทำการส่งต่อให้กับ Application layer ต่อไป

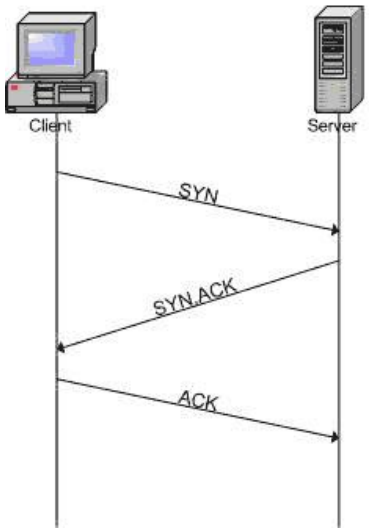
นอกจาก Transport layer จะทำการแบ่งและรวมข้อมูลที่จัดส่งแล้ว ยังทำหน้าที่ตรวจสอบและยืนยันการส่งข้อมูลที่ส่งออกไป (reliability) โดยจะใช้ระบบติดตามตรวจสอบ คือ ข้อมูลที่ถูกส่งออกไป จะมีการระบุหมายเลขลำดับของแต่ละส่วน เมื่อ client ได้รับข้อมูลแต่ละส่วนดังกล่าวแล้ว จะทำการตอบยืนยันการรับข้อมูล (acknowledgement) ให้ server เมื่อ server ได้รับการยืนยันจาก client แล้ว สามารถจัดส่งข้อมูลในลำดับถัดไปได้ทันที แต่ถ้า server ไม่ได้รับข้อมูลยืนยันจาก client ภายในเวลาที่กำหนด Transport layer จะกำหนดให้ server ต้องทำการจัดส่งข้อมูลหมายเลขนั้นออกไปใหม่ จนกว่าจะได้รับการยืนยันจาก client

Transport layer ยังมีความสามารถในการควบคุมการไหล (flow control) โดยจะทำการควบคุมปริมาณการส่งข้อมูลออกไป เพื่อไม่ให้มากเกินกว่าความสามารถของ client ที่จะรับได้ โดยใช้หลักการตอบกลับจาก client ว่า client สามารถรับข้อมูลได้อีกเท่าใดในกระบวนการยืนยันการรับข้อมูล และในระบบเครือข่ายที่มีความหนาแน่นของข้อมูลมาก Transport layer ยังมีการควบคุมความคับคั่ง (congestion control) ซึ่งจะทำการตรวจสอบการสูญหายของข้อมูล เมื่อข้อมูลมีการสูญหาย Transport layer จะปรับลดอัตราการส่งข้อมูลของตน เพื่อคงความมั่นคงของระบบเครือข่ายโดยรวมเอาไว้

โพรโทคอลที่ทำงานอยู่บน Transport layer ที่ได้รับความนิยมมากที่สุด และใช้งานในระบบเครือข่ายอินเทอร์เน็ตปัจจุบัน คือ TCP

1. TCP (Transmission Control Protocol)

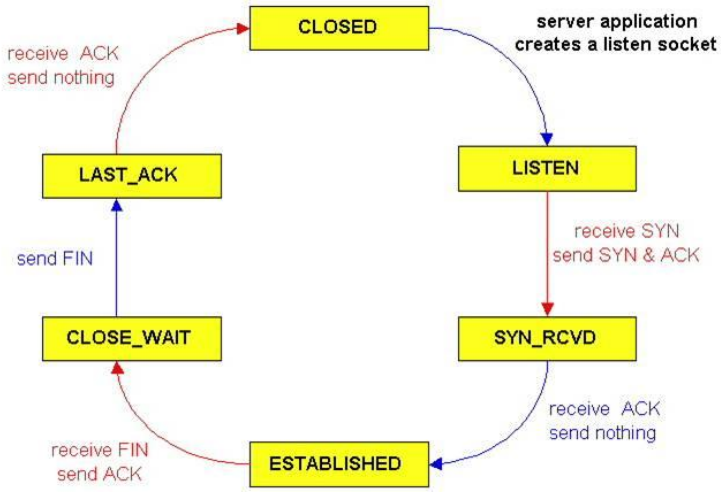
TCP (Department Of Defense, 1969) เป็นโพรโทคอลที่สถาปนาการเชื่อมต่อ (connection-oriented) ระหว่าง server และ client ทุกครั้งที่จะส่งข้อมูลระหว่างกัน เพื่อกำหนดรูปแบบและกฎเกณฑ์ในการติดต่อสื่อสารของทั้งสองฝ่าย ในการสถาปนาการเชื่อมต่อจะใช้หลักการ 3-way handshake



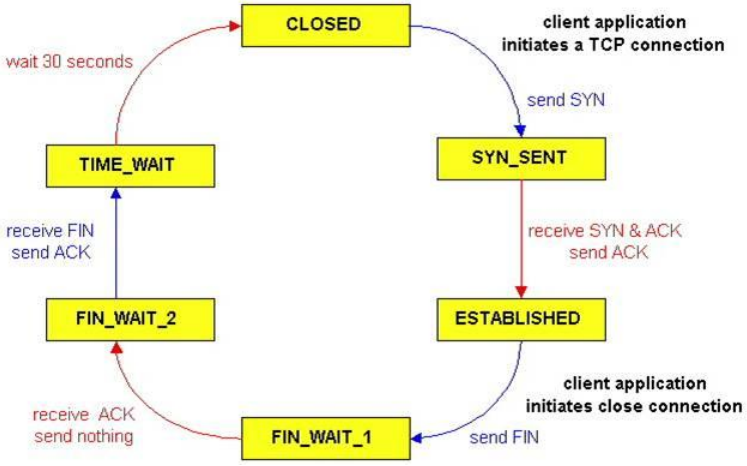
ภาพที่ 1 3-way handshake (T. Jones, 2006)

ภาพที่ 1 แสดงการทำ 3-way handshake ของ TCP โดยก่อนที่ client จะส่งข้อมูลให้กับ server ได้ จะต้องทำการสถาปนาการเชื่อมต่อไปยัง server โดยการส่งคำร้องขอเชื่อมต่อในรูปของ SYN segment ไปยัง server ซึ่ง server จะต้องรออยู่ในสถานะ LISTEN ดังภาพที่ 2 คือ พร้อมที่จะรับการสถาปนาการเชื่อมต่อ เมื่อ server ได้รับ SYN segment จะตอบยืนยันการได้รับ SYN segment พร้อมกับส่งคำร้องขอเชื่อมต่อเพื่อขอสร้างช่องทางการสื่อสารกลับไปยัง client โดยการส่ง SYN-ACK segment ให้กับ client และทำการจองหน่วยความจำที่จำเป็นต้องใช้ในการสื่อสารระหว่างกัน หลังจาก client ได้รับ SYN-ACK segment แล้ว จะส่ง ACK segment ยืนยันกลับให้ server ทราบว่า client พร้อมที่จะส่งข้อมูลให้แล้ว โดยในการส่งข้อมูลยืนยันนั้น ส่งแนบไปกับข้อมูลที่ต้องการส่งให้กับ server ได้เลย ซึ่งเรียกการส่งข้อมูลยืนยันลักษณะนี้ว่า piggy-backed ACK

ภาพที่ 2 แสดงวงจรการทำงานของ TCP ทางด้าน server โดยก่อนที่ server เริ่มทำงานจะอยู่ในสถานะ CLOSED เมื่อจะดำเนินการติดต่อระหว่างกัน server ต้องสร้าง listen socket และเข้าสู่สถานะ LISTEN เมื่อ server ได้รับ SYN segment จะทำการจองหน่วยความจำของระบบและส่ง SYN-ACK segment กลับไปให้ client แล้วเข้าสู่สถานะ SYN_RCVD เพื่อรอรับการยืนยันจาก client หลังจาก server ได้รับ ACK segment แล้ว server จะเข้าสู่สถานะ ESTABLISHED คือ พร้อมที่จะดำเนินการส่งข้อมูลระหว่างกันได้ และการส่งข้อมูลจะดำเนินไปจนกระทั่งได้รับ FIN segment จาก client ซึ่งหมายถึง client ต้องการยุติการติดต่อ server จะทำการส่ง ACK segment เพื่อยืนยันและเข้าสู่สถานะ CLOSE_WAIT เมื่อ server ไม่มีข้อมูลจะส่งให้กับ client แล้วและต้องการยกเลิกการเชื่อมต่อ server จะส่ง FIN segment ไปให้กับ client เพื่อขอยุติการเชื่อมต่อและเข้าสู่สถานะ LAST_ACK รอรับ ACK segment ที่แสดงถึงว่า client รับทราบการขอยกเลิกการเชื่อมต่อจาก server การยุติการเชื่อมต่อจึงเสร็จอย่างสมบูรณ์ และ server เข้าสู่สถานะ CLOSED และทำการคืนหน่วยความจำที่จองไว้ทั้งหมด



ภาพที่ 2 TCP server lifecycle (J. Kurose, K. Ross, 2002)



ภาพที่ 3 TCP client lifecycle (J. Kurose, K. Ross, 2002)

ภาพที่ 3 แสดงการสถาปนาการเชื่อมต่อฝั่ง client โดย client อยู่ในสถานะ CLOSED เมื่อต้องการขอสถาปนาการเชื่อมต่อเพื่อส่งข้อมูลกับ server client จะส่ง SYN segment ไปให้ server และเข้าสู่สถานะ SYN_SENT จากนั้น client จะรอจนกว่าจะได้รับ SYN-ACK segment กลับจาก server จึงทำการส่ง ACK segment ยืนยันให้ server และเข้าสู่สถานะ ESTABLISHED นั้นหมายถึงได้สถาปนาการเชื่อมต่อสำเร็จ พร้อมที่จะส่งข้อมูลระหว่างกันแล้ว และเมื่อ client ต้องการยุติการเชื่อมต่อจะส่ง FIN segment ไปให้ server และเข้าสู่สถานะ FIN_WAIT_1 เพื่อรอการยืนยันการยุติการเชื่อมต่อจาก server เมื่อ server ส่ง ACK segment ยืนยันการยุติกลับมา client จะเข้าสู่สถานะ

FIN_WAIT_2 เพื่อรอข้อมูลบางส่วนที่ทำการส่งระหว่างกันและยังดำเนินการจัดส่งอยู่ และเมื่อรอจนได้รับ FIN segment จาก server แล้ว จึงทำการส่ง ACK segment ยืนยันกลับว่าสามารถทำการยุติการเชื่อมต่อได้แล้ว และเข้าสู่สถานะ TIME_WAIT เพื่อรอข้อมูลบางส่วนที่อาจจะยังคงค้างอยู่ระหว่างทาง ในการหยุดรออยู่ในสถานะ TIME_WAIT นั้น จะทำการหยุดรออยู่ประมาณ 30 วินาที เพื่อความมั่นใจว่า เมื่อข้อมูลส่วนที่ยังคงค้างอยู่ระหว่างทางเข้ามา จะไม่กระทบกับโปรแกรมอื่นที่อาจขอสถาปนาการเชื่อมต่อโดยใช้ช่องบริการนี้ เมื่อครบเวลารอที่กำหนด client จะเข้าสู่สถานะ CLOSED เป็นการยุติการเชื่อมต่อและทำการคืนทรัพยากรให้กับระบบอย่างสมบูรณ์

2. HOL (Head-of-line blocking)

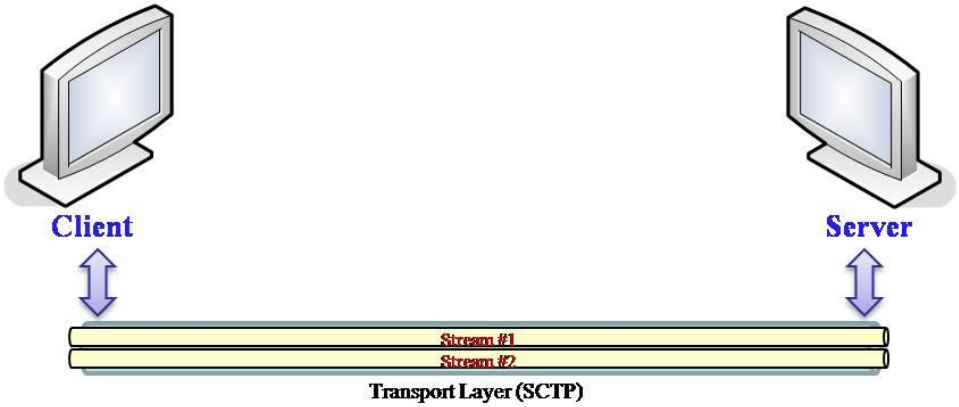
เนื่องจาก TCP ได้มีการกำหนดช่องทางในการส่งข้อมูลภายในเพียงช่องทางเดียวในแต่ละการเชื่อมต่อ ซึ่ง TCP จะจัดเรียงข้อมูลที่ได้รับจาก server ตามหมายเลขลำดับข้อมูล โดยถ้าหากข้อมูลส่วนใดส่วนหนึ่งขาดหายไป TCP จะไม่ส่งข้อมูลส่วนถัดไปให้กับ Application layer แต่จะทำการร้องขอข้อมูลในส่วนที่ขาดไปมาให้ได้ก่อน จึงจะสามารถดำเนินการส่งข้อมูลให้กับ Application layer ได้ เมื่อเป็นเช่นนี้แล้ว เมื่อข้อมูลในส่วนใดของการเชื่อมต่อหายไป TCP จะไม่สามารถดำเนินการต่อไปได้เลย ถ้าไม่ได้รับข้อมูลในส่วนนั้นมาก่อนอย่างถูกต้อง และเนื่องจาก TCP มีช่องทางเพียงช่องทางเดียวในการเชื่อมต่อ จึงทำให้ทั้ง serve และ client ไม่สามารถดำเนินการรับส่งข้อมูลในส่วนอื่นระหว่างกันได้อีกเลย ปัญหานี้เรียกว่า HOL จากปัญหาที่สำคัญของ TCP จึงมีผู้คิดค้นโพรโทคอลใหม่ที่ทำงานบน Transport layer เช่นเดียวกับ TCP เพื่อมาแก้ปัญหาของ TCP ที่ไม่สามารถแก้ไขได้ โพรโทคอลใหม่นี้คือ Stream Control Transmission Protocol

3. SCTP (Stream Control Transmission Protocol)

SCTP (A. Jungmaier, 2001-2003) ถูกพัฒนาขึ้นมาเพื่อรองรับการส่งข้อมูลที่ไม่เหมาะสมที่จะส่งด้วย TCP โดย IETF (Internet Engineering Task Force) SIGTRAN (Signaling Transport) เป็นกลุ่มผู้พัฒนาโพรโทคอลใหม่และได้นำเสนอโพรโทคอลใหม่นี้ออกสู่สาธารณะชนเมื่อเดือนตุลาคม พ.ศ. 2543 โดยเผยแพร่ใน RFC2960 ซึ่ง SCTP มีลักษณะการทำงานที่คล้ายคลึงกับ TCP แต่ได้เพิ่มคุณสมบัติหลายประการเพื่อแก้ปัญหาต่าง ๆ ที่พบใน TCP

3.1 Multi-Streaming

จากปัญหาที่พบใน TCP คือ มีช่องทางในการสื่อสารเพียงช่องทางเดียว SCTP จึงพัฒนาให้สามารถส่งข้อมูลไปพร้อมกันได้ โดยเปรียบเสมือนกับมีหลายช่องทาง (stream) การสื่อสารในการสถาปนาการเชื่อมต่อเพียงครั้งเดียว โดยจะกำหนดค่าเฉพาะให้กับข้อมูลในแต่ละช่องทาง เพื่อให้ข้อมูลสามารถส่งออกไปได้พร้อมกัน เมื่อ client ได้รับข้อมูลในแต่ละช่องทางมาแล้ว client จะดำเนินการจัดเรียงและตรวจสอบข้อมูลในแต่ละช่องทางที่ได้รับ ถ้ามีการสูญหายของข้อมูลในช่องทางใด client จะร้องขอข้อมูลเฉพาะของช่องทางนั้นเท่านั้น โดยที่ไม่มีผลกระทบกับข้อมูลในช่องทางอื่น ทำให้ช่องทางอื่นยังสามารถดำเนินการรับข้อมูลต่อไปได้ ไม่จำเป็นต้องหยุดรอทั้งหมด ซึ่งเป็นการแก้ปัญหา HOL ดังที่พบใน TCP ดังภาพที่ 4 แสดงการส่งข้อมูลระหว่าง client กับ server ซึ่ง client สร้างช่องทางการสื่อสาร 3 ช่องทาง จะเห็นว่าถ้าหากข้อมูลลำดับที่ 3 ซึ่งถูกส่งด้วยช่องทางด้านบนสูญหายไป จะไม่กระทบกับการส่งข้อมูลในช่องทางอื่นเลย ช่องทางอื่นจะยังคงสามารถดำเนินการส่งข้อมูลระหว่างกันได้ มีเฉพาะช่องทางด้านบนเท่านั้นที่หยุดรอเพื่อให้ข้อมูลส่งมาในลำดับที่ถูกต้อง

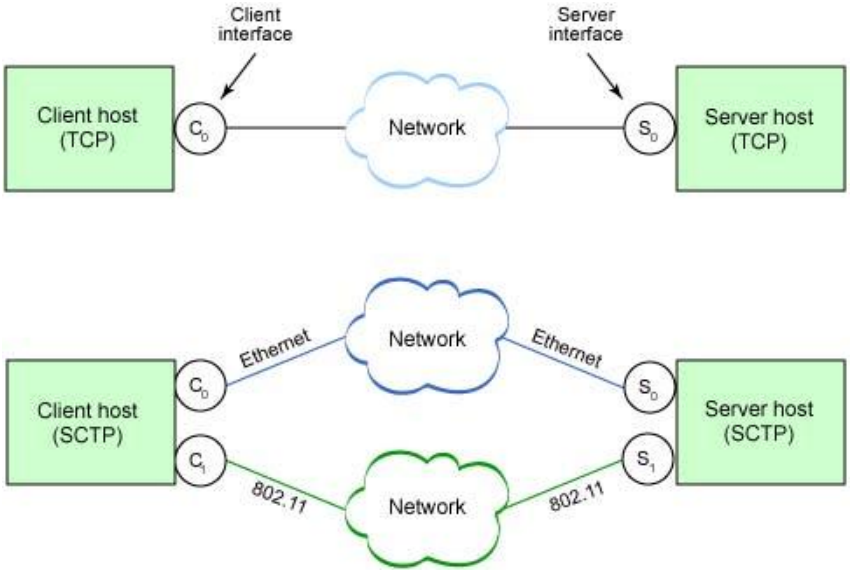


ภาพที่ 4 Multi-Streaming

การสร้างช่องทางการสื่อสารของ SCTP สามารถสร้างได้สูงสุด 65535 ช่องทางทั้งทางด้านส่งออกและ 65535 ช่องทางในด้านรับข้อมูลทำให้การเชื่อมต่อจะสามารถมีช่องทางการสื่อสารได้ถึง 131070 ช่องทาง

3.2 Multi-Homing

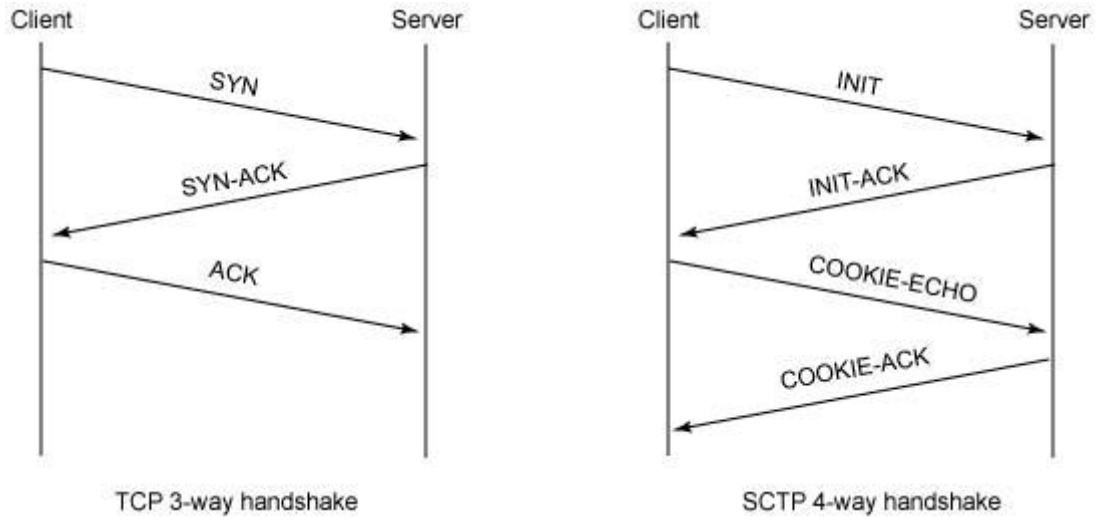
เป็นคุณสมบัติที่ยอมให้มี server ได้มากกว่า 1 เครื่องในเครือข่าย เพื่อขจัดปัญหาเครือข่ายหยุดทำงาน โดยเมื่อ server ใดมีปัญหาไม่สามารถให้บริการได้ client ยังสามารถร้องขอข้อมูลจาก server อื่นได้ โดยไม่จำเป็นต้องเริ่มขอสถาปนาการเชื่อมต่อใหม่ ซึ่ง SCTP มีคุณสมบัติ Multi-Homing ที่สามารถกำหนด server ในเครือข่ายได้หลายเครื่อง จึงสามารถลดปัญหาเครือข่ายหยุดทำงานที่พบใน TCP ซึ่งมี server ได้เพียงเครื่องเดียวได้ ดังรูปที่ 5 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 5 Multi-Homing (T. Jones, 2006)

3.3 4-way handshake

รูปแบบการสถาปนาการเชื่อมต่อใน SCTP จะเป็นแบบ 4-way handshake ดังภาพที่ 6 client จะเริ่มขอสถาปนาการเชื่อมต่อโดยการส่ง INIT segment เมื่อ server ได้รับ INIT segment แล้วจะตอบรับการขอสถาปนาโดยส่ง INIT-ACK segment กลับให้ client ตาม IP Address ที่ client ส่งมาและกลุ่มตัวเลขสุ่มที่ server สร้างขึ้นมาเรียกว่า COOKIE แต่ server จะยังไม่ทำการจองหน่วยความจำของระบบ ซึ่งจะต่างจาก TCP ที่จะจองหน่วยความจำของระบบตั้งแต่กระบวนการนี้ ซึ่งทำให้เสี่ยงต่อการถูกโจมตีด้วย 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 6 4-way handshake (T. Jones, 2006)

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

3.4 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 receive windows ขนาด 32-bit สามารถประกาศหน่วยความจำคงเหลือได้มากกว่า 4 พันล้านไบต์

ตารางที่ 1 เปรียบเทียบ TCP และ SCTP

	TCP	**SCTP**
Connection Type	Connection-Oriented	Connection-Oriented & Non Connection-Oriented
Stream Type	Bytes Stream	Bytes & Message Stream
Reliability	Yes	Yes
Ordered Delivery	Yes	Yes & Unordered Delivery
Multi-Streaming	No	Yes
Multi-Homing	No	Yes
Receive Window Size	16 bits	32 bits
Checksum Size	16 bits	32 bits

Peer-to-Peer Technology

Peer-to-peer (P2P) เป็นรูปแบบการสื่อสารภายในเครือข่าย โดยที่ client แต่ละเครื่องจะทำการรับไฟล์จากเครื่อง client อื่น พร้อมกันได้หลายแหล่ง และในขณะเดียวกันก็ทำหน้าที่ส่งไฟล์ให้ client เครื่องอื่นที่ต้องการไฟล์ด้วย ซึ่งทำให้การส่งไฟล์และรับไฟล์ของ client แต่ละเครื่องทำงานได้เร็วขึ้น จากเดิมที่ client ทุกเครื่องเมื่อต้องการรับไฟล์ใด ๆ ต้องทำการร้องขอไปที่ server เท่านั้น และทำการรับไฟล์จาก server เพียงเครื่องเดียว ซึ่งทำให้ server ต้องทำงานหนักมากในการส่งไฟล์ให้กับ client ทั้งหมดที่มี และเมื่อมีเครื่อง client ที่ต้องการไฟล์จาก server เข้ามาพร้อมกัน ปริมาณข้อมูลในระบบเครือข่ายหนาแน่นทำให้ client แต่ละเครื่องได้รับไฟล์ช้าลง แต่กลับกันกับเทคโนโลยี P2P ซึ่งเครื่องแต่ละเครื่องทำหน้าที่เป็นได้ทั้ง client และ server จึงทำให้ยิ่งมีเครื่องเชื่อมต่อเข้ามามากขึ้น จะยิ่งทำให้ประสิทธิภาพโดยรวมของเครือข่ายเพิ่มสูงขึ้น เนื่องจาก client แต่ละเครื่อง สามารถเลือกได้ว่าจะรับไฟล์จาก client ที่ใกล้หรือมีเส้นทางที่มีความคับคั่งของข้อมูลในเครือข่ายน้อยที่สุด ซึ่งแม้ว่า client ที่เข้ามาเชื่อมต่อแต่ละเครื่อง จะมีประสิทธิภาพที่ต่ำและมี bandwidth ของเครือข่ายที่ต่ำกว่า server ก็ตาม แต่เมื่อนำจำนวน client ที่มีมากขึ้นมาเชื่อมต่อแล้ว จะสามารถทดแทนและทำงานได้ดีกว่า โดยสามารถรับส่งข้อมูลได้เร็วกว่าและในปริมาณที่มากกว่าการเชื่อมต่อที่ต้องพึ่ง server เพียงอย่างเดียว รูปแบบเครือข่าย P2P แบ่งได้ 3 ประเภทคือ

1. แบบมีศูนย์กลาง (Centralized Directory)

P2P (Shawn and Hank, 1987) แบบมีศูนย์กลางเป็นรูปแบบการทำงานของเทคโนโลยีนี้ในยุคแรก ๆ ซึ่งจำเป็นต้องมี server อย่างน้อยหนึ่งเครื่อง ทำหน้าที่เป็นศูนย์กลางในการติดต่อของ client ต่าง ๆ โดย server นี้ ไม่ได้มีหน้าที่ให้บริการส่งไฟล์ให้กับ client แต่มีหน้าที่เป็นเหมือนสมุดหน้าเหลือง (Yellow Pages) ที่คอยบอก client เครื่องใหม่ที่เชื่อมต่อเข้าสู่ระบบเครือข่าย ให้ทราบว่ามีเครื่องใดบ้างที่มีไฟล์ที่ client ต้องการอยู่และแต่ละเครื่องอยู่ที่ใดบ้าง ตัวอย่างโปรแกรม P2P ที่ทำงานแบบมีศูนย์กลาง มีดังนี้

Napster (Shawn and Hank, 1999) เป็นโปรแกรมที่ใช้ในการแลกเปลี่ยนไฟล์แก่กัน ผู้ใช้แต่ละคนที่ต้องการใช้งานเครือข่ายจำเป็นต้องติดตั้งโปรแกรมและทำการเชื่อมต่อเข้ามาที่ server โดยเมื่อทำการติดต่อ server ได้แล้ว โปรแกรมจะส่งรายชื่อของไฟล์ที่ผู้ใช้แลกเปลี่ยนและข้อมูลประจำเครื่องไปไว้ที่ server เพื่อเป็นข้อมูลให้ผู้ใช้คนอื่นทราบ เมื่อผู้ใช้ต้องการไฟล์ใด ผู้ใช้ทำการค้นหา

จากชื่อไฟล์ server จะตรวจสอบว่ามีใครบ้างที่มีไฟล์ที่ผู้ใช้ต้องการ และส่งรายชื่อและรายละเอียดของเครื่องแต่ละเครื่องที่มีไฟล์ที่ผู้ใช้ต้องการกลับมาให้ เมื่อผู้ใช้ได้รับข้อมูลรายชื่อแล้วโปรแกรมจะเริ่มทำการดาวน์โหลดไฟล์

2. แบบไม่มีศูนย์กลาง (Pure peer-to-peer)

จากรูปแบบการให้บริการแบบมี directory เป็นศูนย์กลาง ซึ่งจำเป็นต้องมี server อย่างน้อยหนึ่งเครื่อง ถ้าหาก server ที่ให้บริการมีปัญหาไม่สามารถเข้าใช้งานได้ จะทำให้ระบบทั้งระบบไม่สามารถทำงานได้เลยหรือทำงานได้ช้าลง จึงมีการพัฒนาระบบ P2P แบบไม่มีศูนย์กลางขึ้น เพื่อขจัดปัญหาดังกล่าว โดยมีหลักการทำงานแบบกระจายไป เมื่อถึงเครื่องที่ติดตั้งโปรแกรม P2P แบบไม่มีศูนย์กลางเหมือนกัน เครื่องดังกล่าวจะทำหน้าที่ตรวจสอบในรายการของไฟล์ที่เครื่องตนเองแชร์ไว้ ถ้าพบจะทำการส่งตอบกลับว่ามีไฟล์ที่ต้องการอยู่และข้อมูลที่จำเป็นที่เครื่องต้นทางต้องการใช้เพื่อทำการโหลดไฟล์ได้กลับ แต่ถ้าไม่พบรายชื่อไฟล์ในรายการของตน จะทำการส่งข้อความร้องขอแบบกระจายออกไปอีก เพื่อให้เครื่องอื่นที่ติดตั้งโปรแกรมไว้ทำหน้าที่ตรวจสอบต่อไป โปรแกรมที่ใช้เทคโนโลยี P2P แบบไม่มีศูนย์กลางนี้เช่น

Gnutella (Justin, 2000) ทำงานโดยเมื่อผู้ใช้ต้องการค้นหาไฟล์ที่ต้องการ โปรแกรมทำการส่งข้อความร้องขอออกไปให้เครื่องเพื่อนบ้านในเครือข่ายแบบกระจาย เมื่อแต่ละเครื่องได้รับข้อความร้องขอ จะทำการตรวจสอบในรายการของตนว่ามีไฟล์ที่ต้นทางต้องการหรือไม่ ถ้ามีจะทำการส่งตอบกลับ แต่ถ้าไม่มีจะทำหน้าที่ส่งข้อความร้องขอต่อให้เครื่องเพื่อนบ้านของตนอีกจำนวน 7 เครื่อง และเครื่องที่ได้รับข้อความร้องขอจะทำการตรวจสอบรายการของตนต่อไป โดยการส่งต่อนี้ จะทำการส่งข้อความต่อกันออกไปแต่ไม่เกินจำนวน 10 ระดับ ซึ่งเมื่อครบจำนวน 10 ระดับแล้วและยังไม่พบไฟล์ที่ต้องการ โปรแกรมจะส่งผลการค้นหาว่าไม่สามารถค้นหาไฟล์ที่ต้องการได้กลับให้กับต้นทาง

3. แบบผสม (hybrid)

จากข้อดีและข้อเสียของการให้บริการทั้งแบบมีศูนย์กลางและแบบไม่มีศูนย์กลางจึงมีการคิดค้นรูปแบบการให้บริการรูปแบบใหม่ โดยนำข้อดีของทั้งสองแบบมารวมกัน คือการนำเอาข้อดีของการมี server ศูนย์กลางซึ่งสามารถทำงานได้เร็ว และการกระจายข้อมูลให้ client เพื่อลดปัญหา

server หยุดทำงาน เพื่อพัฒนาระบบ P2P ให้มีประสิทธิภาพสูงขึ้น โปรแกรมที่ทำงานในรูปแบบผสม เช่น

KaZaA (Sharma, 2002) เป็นโปรแกรมที่พัฒนาคุณลักษณะต่างๆ ให้เพิ่มขึ้น เช่นสามารถดาวน์โหลดไฟล์แต่ละไฟล์แบบขนานได้ สามารถสลับการเชื่อมต่อกับเครื่องที่มีไฟล์ที่ต้องการได้โดยอัตโนมัติเมื่อเครื่องที่ดาวน์โหลดอยู่ปัจจุบันมีปัญหา สามารถคำนวณระยะเวลาที่ใช้ในการดาวน์โหลดแต่ละไฟล์ และสามารถกำหนดจำนวนสูงสุดที่โปรแกรมจะทำการดาวน์โหลดไฟล์ในแต่ละเครื่องพร้อมกันได้ นอกจากนี้ KaZaA ยังมีระบบ SuperNodes คือเครื่องที่มีสถิติเวลาในการให้บริการสูงและมีประสิทธิภาพระบบเครือข่ายที่สูง เพื่อให้เครื่องที่ต้องการค้นหาไฟล์ สามารถค้นหาจากเครื่อง SuperNodes ได้ก่อนที่จะทำการค้นหาจากเครื่องทั่วไป ซึ่งช่วยรับประกันความสมบูรณ์และประสิทธิภาพของไฟล์ที่จะทำการโหลดได้มากขึ้น

BitTorrent (Bram, 2004) เป็นเทคโนโลยีการแลกเปลี่ยนไฟล์ที่ได้รับความนิยมมากที่สุด โดยทำงานในแบบผสมที่ต้องมี server ศูนย์กลาง เรียกว่า tracker server เพื่อทำหน้าที่เก็บรายละเอียดไฟล์ที่มีการแลกเปลี่ยนกัน ไฟล์ที่เก็บเรียก torrent file มีนามสกุลไฟล์เป็น .torrent ภายในไฟล์เก็บรายการ tracker server ผู้ให้บริการ รายละเอียดไฟล์ที่แลกเปลี่ยน ค่าจากการคำนวณเพื่อใช้ตรวจสอบความถูกต้อง เมื่อ client ต้องการดาวน์โหลดไฟล์จะติดต่อไปยัง tracker server เมื่อ tracker server รับข้อความร้องขอจะทำการตรวจสอบในฐานข้อมูลของตนว่าในขณะนั้นมี server และ client อยู่ที่ใดบ้างและมีหมายเลขเครื่องใด เมื่อได้รายการทั้งหมดแล้วจะส่งกลับให้ client เมื่อ client ได้รายละเอียดพร้อมแล้ว จะทำการติดต่อตรงไปยัง server ที่ให้บริการอยู่เพื่อขอดาวน์โหลดไฟล์ การขอดาวน์โหลดไฟล์ไปยัง server แต่ละเครื่อง client จะขอดาวน์โหลดเฉพาะบางส่วนของไฟล์ ซึ่งระบบ BitTorrent จะทำการแบ่งไฟล์ออกเป็นส่วน แต่ละส่วนมีขนาดเท่ากัน server ในระบบ BitTorrent จะเรียกอีกอย่างว่า seeder และ client มีชื่อเรียกอีกอย่างว่า leecher โดยเครื่องที่จะเป็น seeder ได้ต้องได้รับไฟล์ทุกชิ้นสมบูรณ์แล้วเท่านั้น ขอดีอีกประการของระบบ BitTorrent คือในการดาวน์โหลดแต่ละชิ้นของไฟล์ leecher สามารถขอดาวน์โหลดจาก leecher อื่นได้ ไม่จำเป็นต้องขอดาวน์โหลดจาก seeder เท่านั้น และในทางกลับกันในขณะที่เครื่องผู้ใช้ซึ่งเป็น leecher กำลังดาวน์โหลดไฟล์จากเครื่องอื่นอยู่ เครื่องอื่นก็สามารถขอดาวน์โหลดไฟล์จากเครื่องผู้ใช้ได้

งานวิจัยที่เกี่ยวข้อง

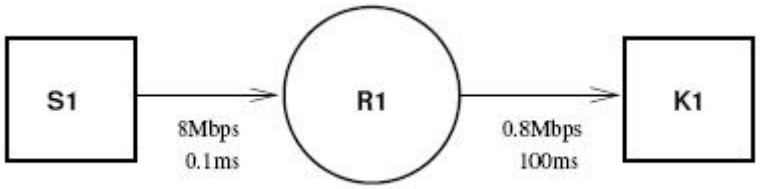
จากปัญหา HOL ที่พบใน BitTorrent เมื่อทำงานร่วมกับ TCP จึงมีงานวิจัยหลายงานที่กล่าวถึงการแก้ปัญหาการทำงาน โดยการนำ SCTP มาใช้เป็น Transport layer protocol เนื่องจากมีความสามารถหลายอย่างเพิ่มขึ้นมา เช่น

งานวิจัย (Balk *et al.*, 2002) ได้ทำการทดลองส่งไฟล์วีดีโอ MPEG-4 เพื่อทดสอบประสิทธิภาพการทำงานแบบ multi-streaming ใน SCTP โดยวัดความเข้ากันได้กับ TCP และเวลาที่ใช้ในการส่งไฟล์ ผลที่ได้ คือ SCTP สามารถทำงานร่วมกับ TCP ได้เป็นอย่างดีและใช้เวลาในการส่งไฟล์น้อยกว่าที่ใช้ในการส่งด้วย TCP

งานวิจัย (Ravier *et al.*, 2001) ทดลองความสามารถ multi-homing ใน SCTP โดยทำการสร้าง server จำนวน 2 เครื่องในการทดลอง ผลการทดลองสรุปว่า เมื่อ server เครื่องใดไม่สามารถให้บริการได้ multi-homing ใน SCTP สามารถสลับการทำงานไปยัง server อีกเครื่องได้ทันที

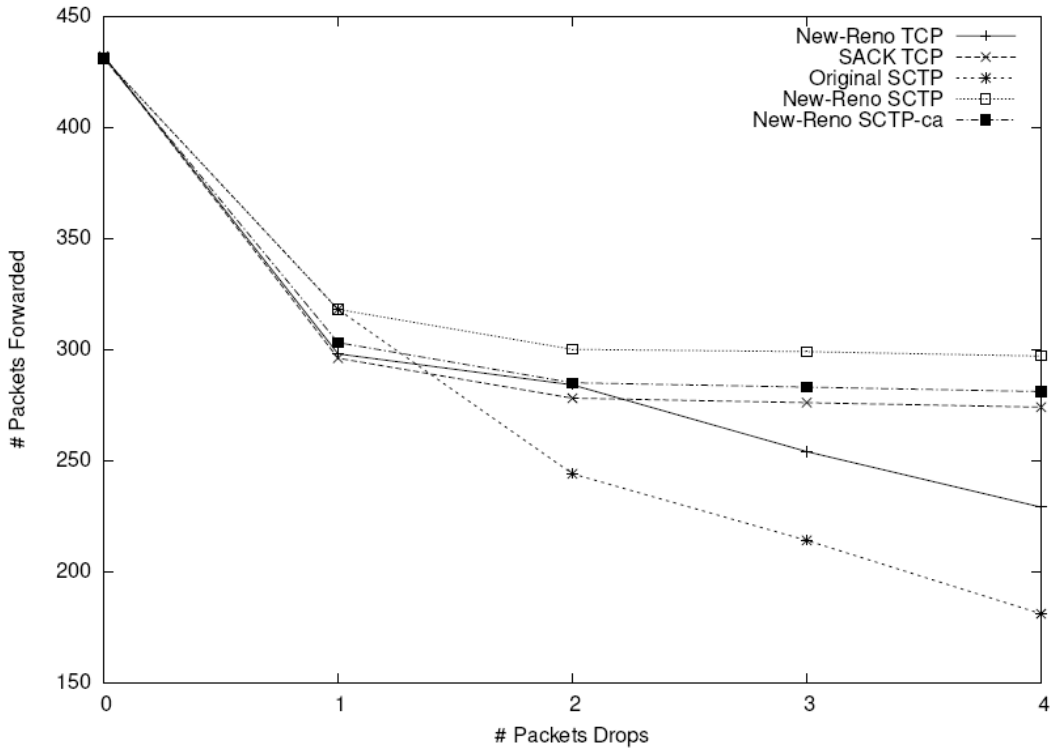
งานวิจัย (Brennan and Curran, 2001) และงานวิจัย (Caro *et al.*, 2003) ได้ทำงานวิจัยในแนวทางเดียวกัน คือ ทดสอบ congestion control ใน SCTP โดยทำการทดสอบทั้งในเครือข่ายที่มีความคับคั่งสูงและเครือข่ายที่มีการสูญหายของข้อมูลสูง โดยกำหนดให้ SCTP ทำงานเพียง stream เดียว และทำการปรับรูปแบบการทำงาน ให้ทำงานสอดคล้องกับการทำงานของ TCP ซึ่งงานวิจัยของ Brennan and Curran ได้จำลองการทำงานบนระบบ OpNet และได้ทำการเปรียบเทียบการทำงานของ SCTP กับ TCP ตรวจสอบประสิทธิภาพในการกู้คืนข้อมูลของ SCTP รวมถึงการสร้างตัวเลือก Gap Ack Block สังเกต SCTP fast retransmit ว่ามีผลกับเครือข่ายในด้านประสิทธิภาพของเครือข่ายเมื่อมีการส่งข้อมูลซ้ำ ๆ หรือไม่ และสามารถทำให้ทราบได้ว่า SCTP มีการตอบรับข้อมูลที่มีสูญหายช้า เนื่องจาก SCTP จะรอข้อมูลให้ได้ครึ่งหนึ่งก่อนที่จะส่งต่อ แต่อย่างไรก็ตามการกู้คืนข้อมูลที่ packet มีจำนวนมากจาก packet loss SCTP ก็สามารถทำได้ดีกว่า TCP ส่วนงานวิจัยของ Caro *et al.* จำลองการทำงานบนระบบ NS-2 ได้แสดงให้เห็นถึงความสามรถของ SCTP ในการจัดการความคับคั่งของข้อมูลเมื่อข้อมูลมีการสูญหาย จึงสร้าง New-Reno SCTP จากนั้นทำการปรับค่าต่าง ๆ ให้สอดคล้องกับการทำงานของ TCP โดยปรับกลไล fast recovery ให้เหมือนกับ New-Reno TCP แก้ไขข้อกำหนดเดิม จากที่เมื่อ packet loss SCTP จะลดเวลาในการส่งข้อมูลลงครึ่งหนึ่งให้ทำการลดเวลาลงไป จากนั้นก็ทำการปรับอัลกอริทึม HTNA (Highest TSN Newly Acked) ของ

SCTP ให้มั่นใจว่าในการ fast retransmits จะไม่มีการรอโดยที่ไม่จำเป็น และสุดท้ายแล้วข้อสรุปที่ได้ คือ New-Reno SCTP ทำงานได้ดีกว่า New-Reno TCP อีกทั้งยังอยู่บนพื้นฐาน AIMD ซึ่งการทดลองได้เป็นไปตามไดอะแกรมภาพที่ 7



ภาพที่ 7 ไดอะแกรมระบบทดลอง (A. Caro, 2003)

ผลการทดลอง



ภาพที่ 8 จำนวนแพ็คเกตที่สามารถส่งได้เมื่อมีอัตราการสูญหายเพิ่มขึ้น (A. Caro, 2003)

จากกราฟข้างต้น แนวแกน x คือ จำนวน packet ที่มีการสูญหาย ส่วนแนวแกน y คือ จำนวน packet ที่ส่งได้ สังเกตแนวโน้มของกราฟที่เกิดขึ้น เมื่อ packet ไม่มีการสูญหายไม่ว่าจะเป็น SCTP หรือ TCP ก็สามารถส่ง packet ได้เท่ากัน แต่เมื่อข้อมูลเริ่มมีการสูญหาย อัตราการส่งข้อมูลก็ลดลงทุกค่าตามจำนวน packet drop ช่วงที่ทำให้เห็นความแตกต่างระหว่าง SCTP และ TCP คือ ตั้งแต่ packets drop ที่ 2% เป็นต้นไป สรุปได้ว่า เมื่อมีการสูญหายของข้อมูล SCTP มีประสิทธิภาพในการส่งข้อมูลมากกว่า TCP และ New-Reno SCTP ที่สร้างขึ้นสามารถทำ fast recovery ได้ดีกว่า TCP ประมาณ 10-30% เนื่องจาก SCTP มี fast retransmits ที่มั่นใจว่าจะไม่รอข้อมูลโดยไม่จำเป็น ถึงแม้เครือข่ายจะมีความคับคั่งของข้อมูลสูง ข้อมูลมีการสูญหาย แต่ SCTP จะไม่รอแต่ข้อมูลที่หายไป สามารถส่งข้อมูลต่อไปได้

งานวิจัย (Grinnemo *et al.*, 2005) ทดลองส่งข้อมูลแบบ ordered และ unordered ใน SCTP ซึ่งการส่งข้อมูลแบบ unordered น่าจะเร็วกว่าการส่งข้อมูลแบบ ordered เนื่องจากไม่ต้องเสียเวลาในการจัดเรียงข้อมูลที่ได้รับมา ไม่สนใจลำดับข้อมูลที่ได้รับ หากข้อมูลมีการสูญหายระหว่างทางก็สามารถรับข้อมูลลำดับต่อไป และทำการทดสอบการขจัดปัญหา HOL ของ SCTP จึงได้ทำการทดลองบน DummyNet โดยกำหนดสภาพแวดล้อมของเครือข่ายดังนี้

1. กำหนดความเร็วเฉลี่ยในการรับ – ส่งข้อมูล 133, 200, 400 Kbps และมากกว่า 400 Kbps
2. กำหนดให้ใช้แพ็คเกตขนาด 500 Bytes
3. กำหนดคิวก่อนส่งข้อมูลด้วยขนาด 12 และ 32 แพ็คเกต
4. ทดลองส่งทั้งหมด 1,000 แพ็คเกตในแต่ละความเร็ว

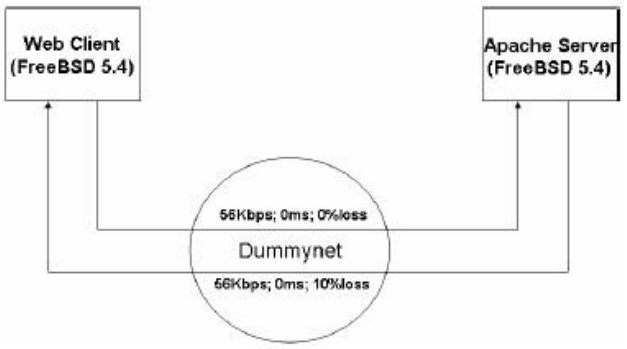
ผลการทดลอง

Msg ID	Send Time	Delivery Service			
		Unordered		Ordered	
		Recv Time	E2E Delay	Recv Time	E2E Delay
1	10	20	10	20	10
2	11	23	12	23	12
3	12	18	6	23	11
4	13	19	6	23	10
5	14	25	11	25	11

ภาพที่ 9 เวลาที่ใช้ในการส่งแบบ ordered และ unordered (KJ. Grinnemo, 2005)

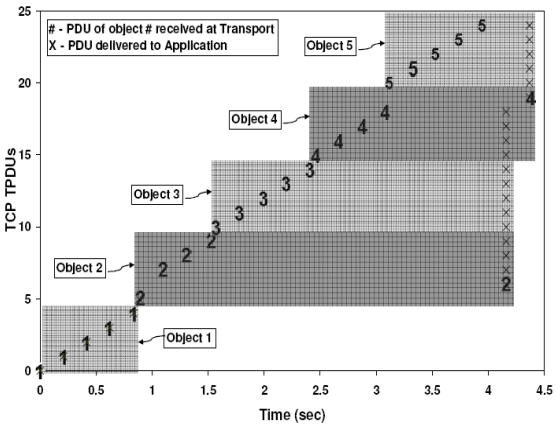
ผลการทดลองดังตารางในภาพที่ 9 นี้ สังเกตได้ว่า เมื่อมีการส่งแค่ message เดียว เวลาในการส่งทั้งแบบ ordered และ unordered จะเท่ากัน แต่เมื่อมีการส่ง message ปริมาณที่เพิ่มขึ้น ผลของเวลาในการส่ง message ที่ได้คือ unordered ใช้เวลาในการส่งน้อยกว่า ordered เมื่อเทียบกันแล้ว unordered มีความเร็วในการส่งมากกว่าแบบ 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 ซึ่งถ้าใช้ SCTP ร่วมกับ HTTP แล้วน่าจะสามารถขจัดปัญหา HOL ของ TCP บน HTTP ได้ งานวิจัยนี้จึงได้ทำการปรับ Apache web server ให้รองรับ SCTP และใช้ DummyNet จำลองระบบเครือข่ายโดยกำหนดสภาพแวดล้อมดังภาพที่ 10 คือ มีการกำหนดความเร็วของ bandwidth เท่ากับ 56 Kbps full duplex กำหนดคิวขนาด 50 Kbytes และไม่จำกัดในการรับ-ส่งข้อมูลและทำการจำลองอัตราการสูญหายของข้อมูลที่ 0 และ 10% แล้วทำการส่ง object แยกไปแต่ละ stream ในการทดลองนี้ได้ใช้ object 5 ตัวด้วยกัน แต่ละ object จะประกอบด้วย 5 packet ตามลำดับ เพื่อทำการเปรียบเทียบการส่งข้อมูลบน HTTP ผ่าน SCTP และ TCP

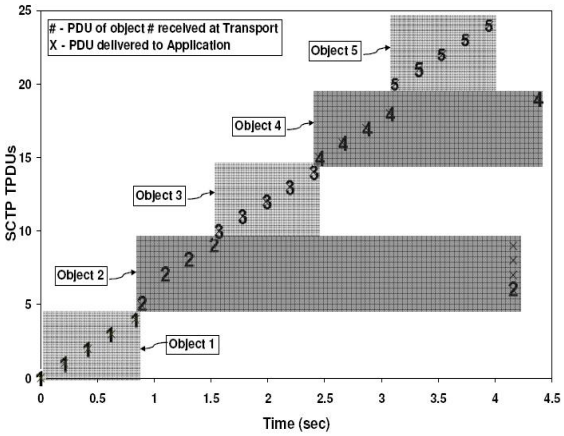


ภาพที่ 10 สภาพแวดล้อมเครือข่ายทดลอง (P. Natarajan, 2006)

ผลการทดลอง



ภาพที่ 11(ก)



ภาพที่ 11(ข)

ภาพที่ 11 เวลาที่ใช้ในการส่ง object แบบ multi-streaming (P. Natarajan, 2006)

ผลจากการทดลองเป็นไปตามกราฟภาพที่ 11(ก) คือ ผลการทดลองของการส่งข้อมูลบน HTTP ผ่าน TCP จะเห็นว่า เมื่อ object ที่ 1 ถูกส่งออกไป ข้อมูลยังไม่มีการสูญหาย แต่ละ packet ที่ได้รับก็เป็นไปตามลำดับ เมื่อส่ง object ที่ 2 สังเกตเห็นว่า packet ที่ 2 เกิดสูญหาย packet ต่อ ๆ ก็ยังไม่หยุดส่ง แต่ไม่สามารถรับได้ จนกว่า packet ที่ 2 จะถูกส่งใหม่จนถึงผู้รับ packet อื่น ๆ จึงจะได้รับ ณ เวลาตามลำดับ ซึ่งการสูญหายของ packet ในแต่ละ object นี้ไม่เพียงกระทบภายใน object เท่านั้น ยังกระทบถึง object อื่นถัดไปอีกด้วย ทำให้เวลาในการได้รับ packet แต่ละตัวล้าช้าไปจากเดิม นี้ก็คือสาเหตุของปัญหา HOL ใน TCP ส่วนกราฟภาพที่ 11(ข) คือ ผลการทดลองของการส่งข้อมูลบน HTTP ผ่าน SCTP ซึ่ง packet ที่ 2 ของ object ที่ 2 มีการสูญหาย packet อื่น ๆ จะได้รับก็ต่อเมื่อ packet ที่ 2 มาถึง แต่ไม่กระทบ object อื่น ๆ สังเกตได้จาก object ที่ 3 ที่ยังสามารถรับส่ง packet ได้ตามเวลาปกติโดยที่ไม่ต้องรอ object ก่อนหน้า ด้วยเหตุนี้ทำให้ SCTP สามารถแก้ปัญหา HOL ที่เกิดขึ้นกับ TCP ได้

งานวิจัย (Rajamani *et al.*, 2002) และ (Henrik, 2005) ทดลองส่งข้อมูลคล้ายการวิจัยที่ผ่านมา โดยส่งข้อมูลกลับจาก web server ผ่าน HTTP โดยส่งข้อมูลแบบ Multi-Streaming เป็นการสร้างช่องทางหลาย ๆ ช่องทางขึ้น ทำให้สามารถส่งข้อมูลได้ดีและเร็วกว่าแบบช่องทางเดียว มีพื้นที่ในการส่งข้อมูลมากขึ้น ทำให้ลดความคับคั่งของเครือข่ายลง งานวิจัยของ Rajamani ใช้ DummyNet

ในการจำลองเครือข่าย โดยได้กำหนดสภาพแวดล้อมไว้ คือ จำกัดให้ RTT อยู่ที่ 80 ms ให้อัตราการสูญหายของข้อมูลเท่ากับ 0 และ 25% และกำหนด bandwidth ที่ 40 400 Kbps, 3, 10 Mbps โดยใช้ค่าตาม modem, cable modem, WIFI, LAN ตามลำดับ จากนั้นก็ทำการทดลอง ประกอบด้วย 2 กรณี คือ ส่งไฟล์เดียวที่ไม่มีการสูญหายของข้อมูล และส่งแบบหลายไฟล์และมีการสูญหายของข้อมูล ส่วนของ Henrik ได้ทำการทดลองด้วยโปรแกรม httperf ซึ่งเป็นโปรแกรมจำลองส่งแพ็คเกต httperf นี้จะทำการส่งแพ็คเกตไปอย่างต่อเนื่องตามที่ได้ตั้งค่าไว้ จากนั้นทำการกำหนดขนาดของไฟล์เป็น 100 Byte, 1, 10, 100 KB, 1, 10 MB ภายใต้การร้องขอทั้งหมด 50 requests โดยร้องขอ 25 requests ต่อวินาที และกำหนดเวลา timeout 15 วินาที โดยทำการเปรียบเทียบ 4 แบบด้วยกัน คือ TCP, SCTP ช่องทางเดียว, SCTP 10 ช่องทาง และ SCTP 50 ช่องทาง เพื่อผลที่แน่ชัดว่าการส่งข้อมูลหลายช่องทางสามารถแก้ปัญหา HOL ได้ โดยในการเพิ่มช่องทางการสื่อสารมากขึ้น จะทำให้ลดความคับคั่งของข้อมูลมากขึ้นด้วย

ผลการทดลอง

Protocol	Loss	File 1	File 2	File 3	File 4	File 5	File 6	File 7	File 8
TCP	0%	679395	768656	3873273	3910344	3942401	4243416	4273269	4708352
SCTP	0%	802931	888264	4468128	4507111	4607180	4834083	4878979	4887073
TCP	1%	4930718	5595977	29598318	31047085	31924275	33460332	34333089	38222168
SCTP	1%	4299919	4775626	24132252	24536217	25106516	26678879	27143072	29628740
TCP	2%	5983277	6725730	35361679	37232434	38509110	40681890	42568811	45179090
SCTP	2%	5506176	6098121	31539786	32164926	32692426	33117396	33981264	41551992

* หน่วย คือ ms

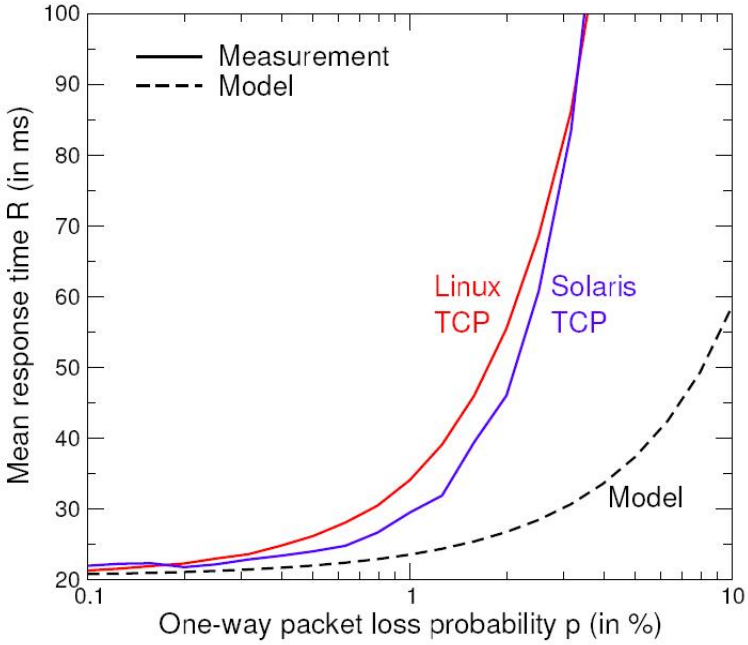
ภาพที่ 12 เวลาที่ใช้ในการดาวน์โหลดเมื่ออัตราการสูญหายสูง (R. Rajamani, 2002)

จากตารางภาพที่ 12 สังเกตค่าที่ได้ในตารางเปรียบเทียบระหว่าง TCP และ SCTP โดยกำหนด loss rate ที่ 0, 1 และ2% ในขณะที่ไม่มีการสูญหายของข้อมูล การส่งข้อมูลผ่าน TCP ใช้เวลาน้อยกว่าการส่งข้อมูลผ่าน SCTP แต่เมื่อข้อมูลเริ่มมีการสูญหายเพิ่มขึ้น SCTP ใช้เวลาในการส่งข้อมูลน้อยกว่า TCP ทุก ๆ ไฟล์ที่ส่งออกไป จึงพิสูจน์ได้ว่า SCTP สามารถแก้ปัญหา HOL ของ TCP ได้จริง ซึ่งในระบบเครือข่ายปัจจุบันเป็นไปไม่ได้ที่ข้อมูลจะไม่เกิดการสูญหาย และเมื่อมีการส่งข้อมูลมากขึ้นเท่าไร ทำให้ความคับคั่งภายในเครือข่ายมากขึ้น ข้อมูลก็มีการสูญหายมากขึ้น SCTP เป็น protocol ที่มีประสิทธิภาพในการขจัดปัญหา HOL ได้เป็นอย่างดี

งานวิจัย (Scharf and Kiesel, 2006) ได้กล่าวไว้ว่าถึงแม้ว่า TCP จะสามารถส่งข้อมูลที่มีความน่าเชื่อถือ และจัดลำดับข้อมูลได้ แต่ก็เกิดความล้าช้า เนื่องจากปัญหา HOL จึงมีการนำ SCTP

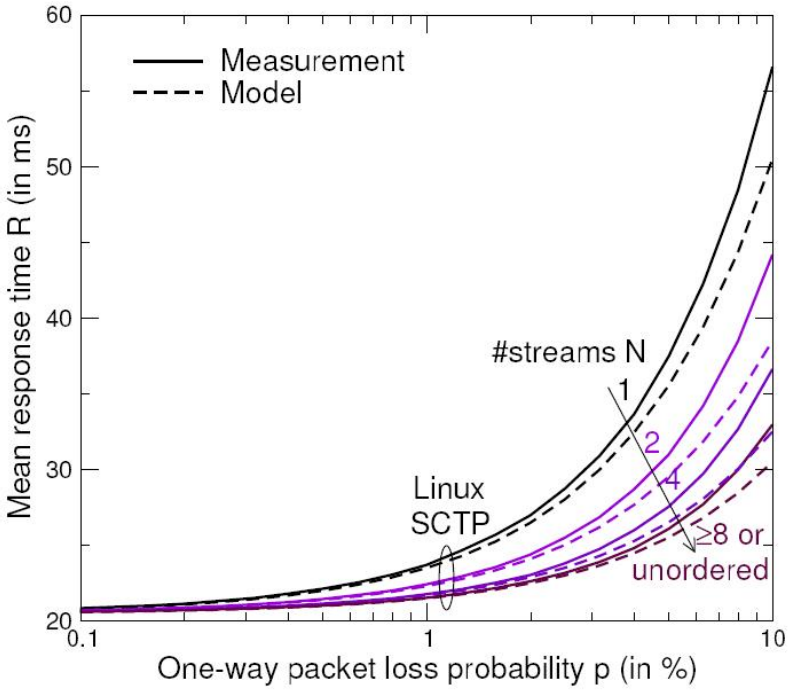
มาใช้แทน ซึ่งประสิทธิภาพของ SCTP เองก็สามารถส่งข้อมูลที่มีความน่าเชื่อถือรวมถึงสามารถจัดเรียงลำดับข้อมูลได้ด้วย จึงมีการเปรียบเทียบการส่งข้อมูลของ SCTP แบบ Multi-Streaming กับการส่งข้อมูลของ TCP แบบ Multi-Connection นอกจากนี้ยังทำการตรวจสอบ โดยใช้หลายระบบปฏิบัติการ เพื่อพิสูจน์ความน่าเชื่อถือ ว่าการส่งข้อมูลไม่ขึ้นกับระบบปฏิบัติ แต่ขึ้นกับโพรโทคอลที่ใช้

ผลการทดลอง



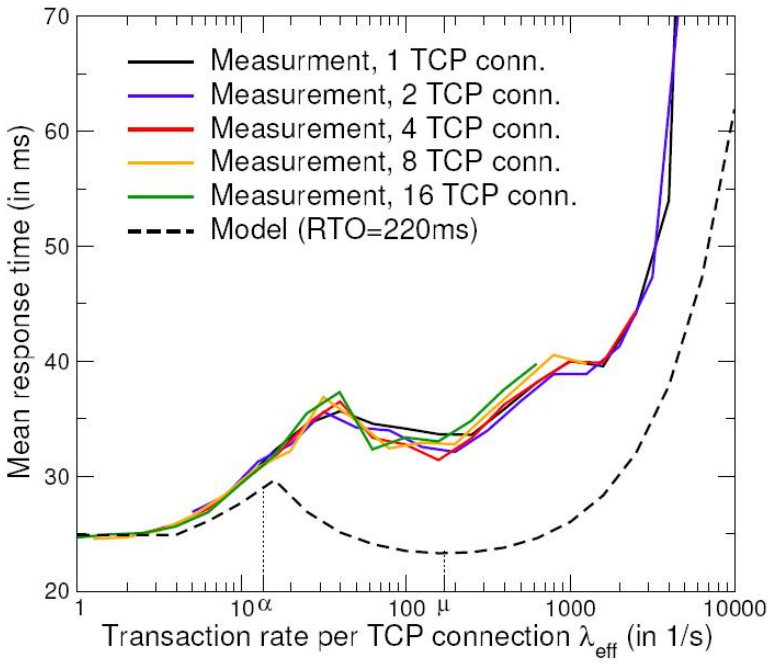
ภาพที่ 13 Response Time ของ TCP (บนระบบปฏิบัติการ Linux/Solaris)

จากกราฟภาพที่ 13 แกน x คืออัตราการสูญหายของ packet แกน y คือค่าเฉลี่ยของ response time ซึ่งเป็นการทดลองส่งข้อมูลผ่าน TCP connection เดียว บนระบบปฏิบัติการ Linux และ Solaris สังเกตเส้นประ คือแบบจำลองตามแนวความคิด แต่ response time ที่เกิดขึ้นจริงกับ TCP มีอัตราสูงกว่า เมื่อ packet เกิดการสูญหายตั้งแต่ 1% ขึ้นไป response time มีอัตราเพิ่มสูงขึ้นประมาณ 40% จากแนวความคิด สรุปได้ว่า TCP เกิดการสูญหายของข้อมูลจาก HOL จริง



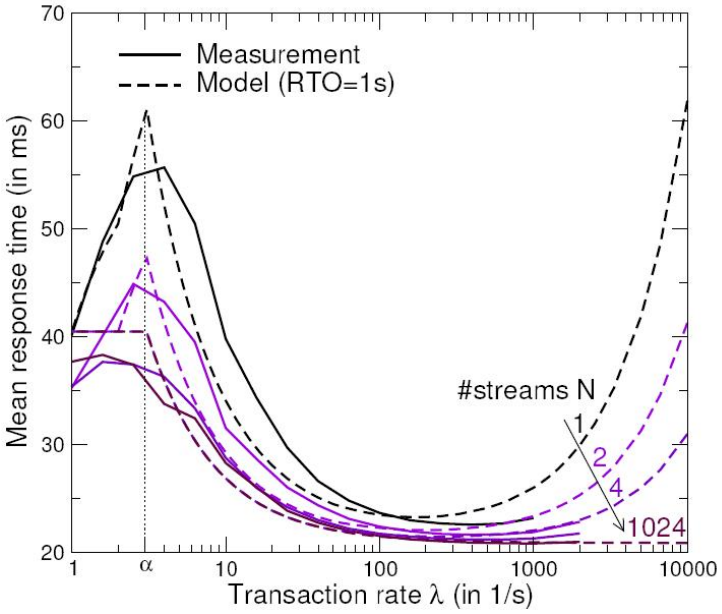
ภาพที่ 14 Response Time ของ SCTP (บนระบบปฏิบัติการ Linux)

จากกราฟภาพที่ 14 แกน x คืออัตราการสูญหายของ packet แกน y คือค่าเฉลี่ยของ response time ซึ่งเป็นการทดลองการส่งข้อมูลผ่าน SCTP บนระบบปฏิบัติการ Linux เส้นประ คือค่าที่เป็นไปตามแนวความคิด สังเกตว่าค่าที่ได้จริงไม่คลาดเคลื่อนไปจากแนวความคิดมาก กล่าวได้ว่า SCTP สามารถส่งข้อมูลในสภาวะที่ packet เกิดการสูญหายได้ดี โดยเฉพาะกับข้อมูลที่ไม่ต้องเรียงลำดับยิ่งสามารถลด response time ได้ ที่ packet loss 1% เปรียบเทียบ response time ของ SCTP และ TCP จากกราฟสรุปได้ว่า SCTP มีประสิทธิในการส่งข้อมูล ถึงแม้จะเกิดปัญหา HOL ก็ตาม



ภาพที่ 15 การส่งข้อมูลผ่าน TCP บนระบบปฏิบัติการ Linux (Multi-Connection)

จากกราฟภาพที่ 15 แกน x คือจำนวน transaction ที่ส่ง แกน y คือค่าเฉลี่ยของ response time แนวเส้นประ คือผลที่เป็นไปตามแนวความคิด เพื่อเปรียบเทียบแนวโน้มของผลที่ได้จริงจากการทดลอง สังเกตได้ว่า TCP จะส่งด้วยจำนวน connection มากเท่าใดก็ตาม แต่ค่าเฉลี่ยของ response time ที่ได้ก็เป็นไปแนวทางเดียวกัน



ภาพที่ 16 การส่งข้อมูลผ่าน SCTP บนระบบปฏิบัติการ Linux (Multi-Steaming)

จากกราฟภาพที่ 16 แกน x คือจำนวน transaction ที่ส่ง แกน y คือค่าเฉลี่ยของ response time แนวเส้นประ คือผลที่เป็นไปตามแนวความคิด เพื่อเปรียบเทียบแนวโน้มของผลที่ได้จริงจากการทดลอง สังเกตค่าที่ได้ SCTP ส่งด้วย stream ที่มีจำนวนมากขึ้น ทำให้สามารถลด response time ได้มาก จนเมื่อใช้ stream จำนวน 1024 แล้ว ไม่มีปัญหาในการส่งข้อมูลเลย response time ที่ได้เป็นค่าที่ต่ำที่สุด สรุปได้ว่า คุณสมบัติของ SCTP ในการส่งข้อมูลแบบ Multi-Streaming มีประสิทธิภาพมากกว่าการส่งด้วย TCP แบบ Multi-Connection และสามารถขจัดปัญหา HOL ได้จริง

อุปกรณ์และวิธีการ

อุปกรณ์

1. คอมพิวเตอร์ตั้งโต๊ะ หน่วยประมวลผล Intel Pentium 4 ความเร็ว 2.8 GHz หน่วยความจำหลัก DDR2 2.5 GB หน่วยความจำสำรอง 160 GB ความเร็วรอบ 7,200 PRM

2. คอมพิวเตอร์โน้ตบุ๊ก หน่วยประมวลผล AMD Turion 64 x2 TL-56 ความเร็ว 1.8 GHz หน่วยความจำหลัก DDR2 2.0 GB หน่วยความจำสำรอง 160 GB ความเร็วรอบ 5,400 PRM

3. คอมพิวเตอร์โน้ตบุ๊ก หน่วยประมวลผล Intel Pentium 4 ความเร็ว 2.4 GHz หน่วยความจำหลัก DDR 256 MB หน่วยความจำสำรอง 30 GB ความเร็วรอบ 5,400 PRM

วิธีการ

ดำเนินการทดลอง พัฒนาโปรแกรมเพิ่มเติม โดยการปรับโพรโทคอลในการเชื่อมต่อจากเดิมที่ใช้ TCP เปลี่ยนมาใช้ SCTP แทน และติดตั้งระบบเครือข่าย โดยแบ่งวิธีการดำเนินงานออกเป็น 3 ส่วน คือ

1. จำลองระบบเครือข่ายโดยมีโปรแกรมที่เกี่ยวข้องดังนี้

1.1 ติดตั้งระบบปฏิบัติการ Linux kernel 2.6 ขึ้นไป ปรับตั้งค่าและสร้างตัวแปรระบบ ด้วยคำสั่ง export LKSCTP_LINUX=/usr/src/linux และ make xconfig ทำการเปิดการใช้งาน SCTP

1.1.1 ติดตั้ง lksctp เครื่องมือพัฒนาโปรแกรม ทำให้สามารถใช้โพรโทคอล SCTP ในการเชื่อมต่อกับระบบเครือข่าย ขั้นตอนการติดตั้งดังนี้

1) bootstrap คำสั่งสร้างไฟล์ตั้งค่าของระบบ
2) configure คำสั่งตั้งค่าระบบ
3) make install clean คอมไพล์โปรแกรมทั้งหมด
4) cp ./src/include/netinet/*.h /usr/include/netinet คัดลอกไฟล์ส่วนหัวเข้าสู่ระบบ
5) cp /usr/local/lib/*.* /usr/lib คัดลอกไฟล์เครื่องมือเข้าสู่ระบบ

1.1.2 ติดตั้งโปรแกรม thttpd เวอร์ชั่น 2.25b ทำหน้าที่เป็น web server ของระบบทดลอง และทำการปรับปรุงโปรแกรม thttpd ด้วย thttpd-2.25b.sctp.nivel1.patch ซึ่งเป็นตัวติดตั้งปรับปรุงให้สามารถทำงานร่วมกับ SCTP ได้

1.1.3 ติดตั้งตัวแปลภาษา PHP เวอร์ชั่น 4.4.4 เป็นภาษาหลักในการพัฒนาโปรแกรมส่วน tracker server ซึ่งทำหน้าที่เป็น server เก็บข้อมูลการเชื่อมต่อ

1.2 ส่วนของ tracker server ติดตั้งโดยทำการติดตั้งโปรแกรม Torrent Trader Lite 1.0.3 ซึ่งพัฒนาด้วยภาษา php และทำการปรับแต่งค่าตัวแปรของระบบ ดังนี้

1.2.1 ปรับค่าตัวแปร $sitetitle และ $siteurl ให้เป็นชื่อและตำแหน่งของ server ของระบบทดลอง

1.2.2 ปรับค่าตัวแปรอาเรย์ $announce_urls[] โดยใส่ชื่อ server ทั้งหมดของระบบทดลอง เพื่อให้โปรแกรม client ใช้ในการอ้างอิงชื่อเมื่อทำการสร้างไฟล์ torrent

1.3 พัฒนาปรับปรุงโปรแกรม BitTorrent client ให้สามารถทำงานบน SCTP ได้ โดยทำการติดตั้งโปรแกรม cTorrent-dnh เวอร์ชั่น 1.3.4 พัฒนาด้วยภาษา C++ และทำการปรับแก้โค้ดของโปรแกรมดังนี้

1.3.1 ปรับแก้โปรแกรมส่วนติดต่อกับ tracker server ให้ใช้โพรโทคอล SCTP ในการติดต่อ โดยทำการปรับแก้ไฟล์ tracker.cpp บรรทัดที่ 390 จากเดิม

socket(AF_INET,SOCK_STREAM, IPPROTO_TCP); ซึ่งทำงานบน TCP ให้เป็น socket(AF_INET,SOCK_STREAM, IPPROTO_SCTP); เพื่อทำงานบน SCTP และทำการกำหนดค่าเบื้องต้นให้กับ socket โดยเพิ่มส่วนกำหนดค่า

```
memset(&initmsg, 0, sizeof(initmsg));
initmsg.sinit_num_ostreams = 65535;
initmsg.sinit_max_instreams = 65535;
initmsg.sinit_max_attempts = 4;
```

และทำการเซ็ตค่าให้ socket ด้วยคำสั่ง

ret = setsockopt(m_sock, IPPROTO_SCTP, SCTP_INITMSG, &initmsg, sizeof(initmsg));

1.3.2 ปรับแก้โปรแกรมส่วนของ seeder ในขั้นตอนรอการเชื่อมต่อ เพื่อให้รอการเชื่อมต่อด้วยโพรโทคอล SCTP ด้วยการปรับแก้ไฟล์ peerlist.cpp บรรทัดที่ 558 จากเดิม

socket(AF_INET,SOCK_STREAM, IPPROTO_TCP); ให้เป็น socket(AF_INET,SOCK_STREAM, IPPROTO_SCTP);

ผลและวิจารณ์

ผล

ในการดำเนินการทดลอง ได้พัฒนาโปรแกรมเพิ่มเติม และติดตั้งระบบเครือข่าย โดยแบ่งวิธีการดำเนินงานออกเป็น 3 ส่วน คือ

1. จำลองระบบเครือข่าย

2. ส่วนของ tracker server ติดตั้งโดยทำการติดตั้งโปรแกรม *Torrent Trader Lite 1.0.3* ซึ่งพัฒนาด้วยภาษา PHP และทำการปรับแต่งค่าตัวแปรของระบบ

3. พัฒนาปรับปรุงโปรแกรม BitTorrent Client ให้สามารถทำงานบน SCTP ได้ โดยติดตั้งโปรแกรม cTorrent-dnh เวอร์ชั่น 1.3.4 พัฒนาด้วยภาษา C++ และปรับแก้โค้ดของโปรแกรม

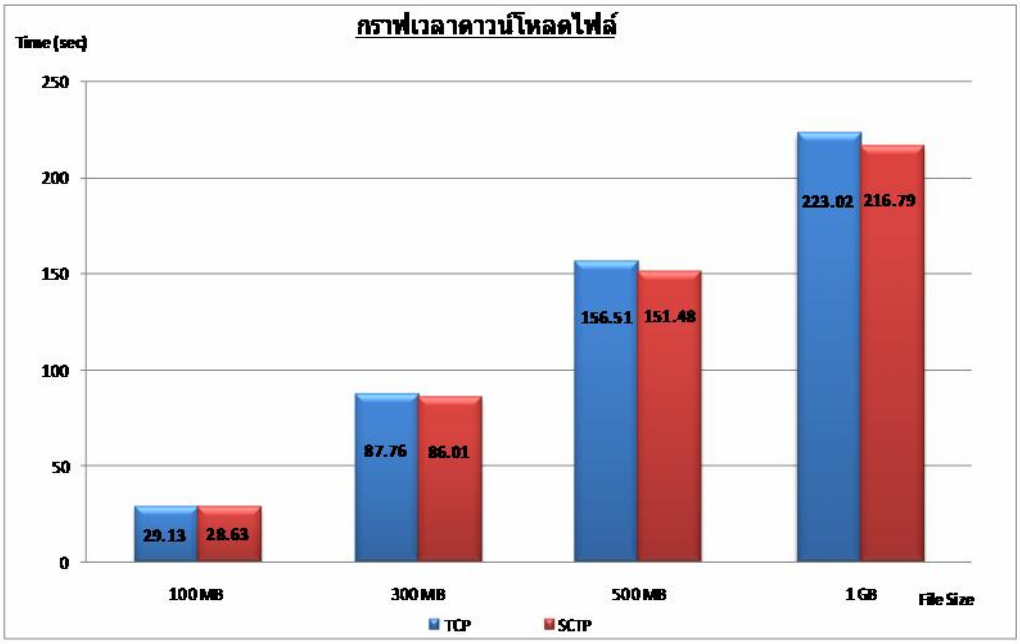
สภาพแวดล้อมในการทดลอง กรณีที่ 1

เนื่องจากการทำงานของ BitTorrent ผ่านโพรโทคอล TCP เกิดปัญหา HOL ทำให้ผู้วิจัยศึกษาการทำงานของ BitTorrent เพื่อหาโพรโทคอลที่จะนำมาประยุกต์ และโพรโทคอลที่นำมาใช้คือ SCTP ซึ่งมีคุณสมบัติในการขจัดปัญหา HOL การทดลองเริ่มต้นจากนำไฟล์ขนาด 100, 300, 500 MB และ 1 GB มาทำการดาวน์โหลดในสภาวะแวดล้อมจริง โดยผ่านระบบเครือข่าย LAN ซึ่งในการทดลองนี้ไม่มีการกำหนดอัตราการสูญหายของข้อมูล มีการเกิด cross traffic ตามสภาพการใช้งานของเครือข่ายจริง ดังกราฟภาพที่ 17 เพื่อศึกษาว่าเมื่อขนาดไฟล์เพิ่มขึ้น ประสิทธิภาพการทำงานของ SCTP และ TCP ต่างกันอย่างไร

1. หลังจากติดตั้งระบบทดลองเรียบร้อยแล้ว ทำการสร้างไฟล์ตัวอย่างจำนวน 4 ไฟล์ แต่ละไฟล์มีขนาด 100, 300, 500 MB และ 1 GB นำไฟล์ตัวอย่างทั้งหมดมาสร้างไฟล์ .torrent ตามขนาดไฟล์ตัวอย่างนั้น ๆ แล้วทำการอัพโหลดขึ้น tracker server เพื่อใช้ในการดาวน์โหลดต่อไป

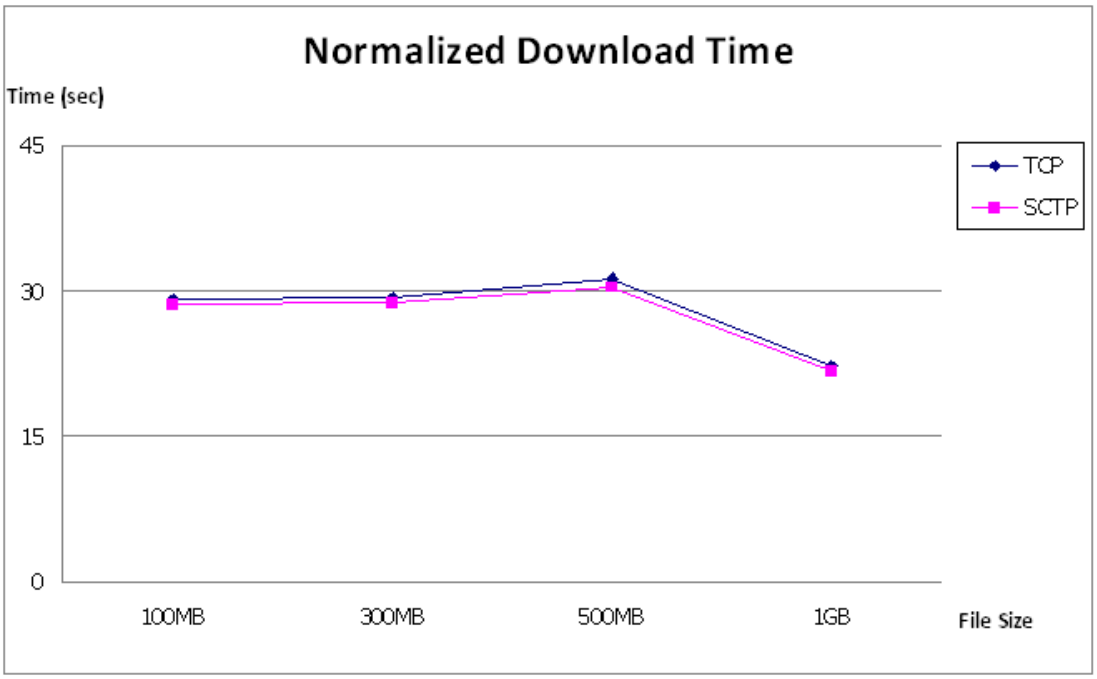
2. เริ่มทำการดาวน์โหลด โดยให้เครื่องคอมพิวเตอร์ 1 เครื่องทำหน้าที่เป็น seeder ของระบบ จากนั้นให้ leecher โหลดไฟล์ .torrent ทั้ง 4 ไฟล์เพื่อใช้เป็นข้อมูลในการดาวน์โหลดไฟล์ตัวอย่าง

3. ขั้นตอน leecher ดาวน์โหลด จะให้โหลดทีละขนาดไฟล์ แต่ละไฟล์ทำการดาวน์โหลดจำนวน 5 ครั้งเพื่อหาค่าเฉลี่ย โดยทำการสลับการดาวน์โหลดแต่ละไฟล์ระหว่างการใช้ TCP และ SCTP โหลด



ภาพที่ 17 กราฟเวลาดาวน์โหลดไฟล์

จากกราฟเวลาดาวน์โหลดไฟล์ข้างต้นนี้ ค่าเฉลี่ยของการดาวน์โหลด ที่ไฟล์มีขนาด 100, 300, 500 MB และ 1 GB TCP ใช้เวลาเฉลี่ย 29.13, 87.76, 156.51 และ 223.02 ตามลำดับ ส่วน SCTP ใช้เวลาเฉลี่ย 28.63, 86.01, 151.48 และ 216.79 ตามลำดับ ผลที่ได้แสดงให้เห็นว่า SCTP ใช้เวลาในการดาวน์โหลดไฟล์น้อยกว่า TCP ทุก ๆ ขนาดไฟล์ที่เพิ่มขึ้น ซึ่งส่วนต่างของค่าเฉลี่ยเวลาในการดาวน์โหลดไฟล์ของ TCP เทียบกับ SCTP คือ ที่ขนาดไฟล์ 100 MB ส่วนต่างเท่ากับ 0.05 วินาที ขนาดไฟล์ 300 MB ส่วนต่างเท่ากับ 1.75 วินาที ขนาดไฟล์ 500 MB ส่วนต่างเท่ากับ 5.03 วินาที และขนาดไฟล์ 1 GB ส่วนต่างเท่ากับ 6.23 วินาที แนวโน้มของเวลาในการดาวน์โหลดไฟล์มีส่วนต่าง ๆเพิ่มขึ้น แสดงให้เห็นว่า SCTP ทำงานได้อย่างมีประสิทธิภาพเพิ่มขึ้น



ภาพที่ 18 เปรียบเทียบเวลาดาวน์โหลดข้อมูล 100MB ของไฟล์ขนาดต่างๆ ระหว่าง TCP vs. SCTP

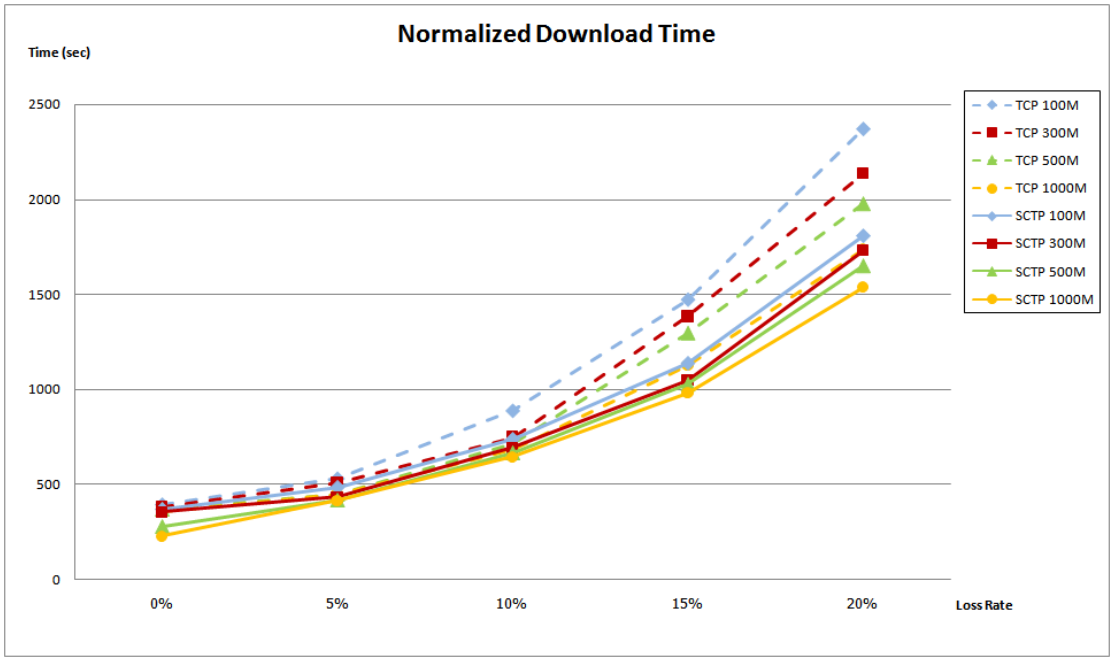
ภาพที่ 18 แสดงการเปรียบเทียบเวลาที่ใช้ในการดาวน์โหลดข้อมูลจำนวน 100 MB ของไฟล์ขนาดต่างๆ ระหว่างโพรโทคอล TCP และ SCTP ซึ่งจะพบว่าขนาดไฟล์ที่ใหญ่ขึ้น ทั้งสองโพรโทคอลมีแนวโน้มที่จะใช้เวลาในการดาวน์โหลดน้อยลง ทั้งนี้เนื่องจากจากพฤติกรรมของ Transport layer protocol ที่มีการเพิ่มอัตราการส่งข้อมูลที่สูงขึ้นอย่างต่อเนื่อง ในกรณีที่ไม่มีการสูญหายของข้อมูล อย่างไรก็ตาม ในกรณีที่มีการสูญหายของข้อมูลแล้ว คาดการณ์ได้ว่าเวลาที่ใช้ต่อหน่วยของข้อมูลสำหรับขนาดไฟล์ที่ใหญ่ขึ้น ควรจะสูงขึ้น

สภาพแวดล้อมในการทดลอง กรณีที่ 2

จากการทดลองที่ 1 เป็นการทดลองดาวน์โหลดไฟล์ขนาดต่าง ๆ ผ่านเครือข่าย TCP และ STCP โดยไม่มีการสูญหายของข้อมูล สังเกตว่าขนาดไฟล์ไม่มีผลต่อการดาวน์โหลดข้อมูล อ้างอิงจากการทดลองในกรณีที่ 1 เนื่องจากไม่ว่าขนาดไฟล์จะใหญ่ขึ้นเท่าไร แนวโน้มการดาวน์โหลดไฟล์ผ่าน SCTP ก็ทำได้เร็วกว่า TCP เพราะฉะนั้น เพื่อยืนยันว่าขนาดไฟล์ไม่มีผล จึงได้ทำการทดลองในกรณีที่ 2 โดยทำการทดลองบนระบบ DummyNet ควบคุมสภาพแวดล้อมให้มี loss rate ที่ 0, 5, 10, 15 และ 20% จากนั้นทำการดาวน์โหลดไฟล์ขนาดต่าง ๆ ตั้งแต่ 100, 300, 500 MB และ 1 GB เริ่มจาก loss rate 0% ตามลำดับ ซึ่งได้ค่าดังตารางที่ 2

ตารางที่ 2 เปรียบเทียบเวลาที่ใช้ในการดาวน์โหลดผ่าน Modem ที่ขนาดไฟล์ต่าง ๆ

loss rate	เวลาที่ใช้ในการดาวน์โหลดผ่าน Modem (56 Kbps) (sec)				
Protocal	0%	5%	10%	15%	20%
TCP					
100MB	396.368	533.363	890.309	1,475.241	2,373.070
300MB	384.134	512.398	749.595	1,386.365	2,138.457
500MB	372.393	451.386	713.387	1,297.359	1,975.910
1GB	359.277	447.393	683.867	1,128.357	1,738.865
SCTP					
100MB	367.303	483.098	736.169	1,135.945	1,806.303
300MB	355.109	434.250	692.443	1,048.735	1,731.357
500MB	281.537	420.529	668.794	1,030.239	1,652.377
1GB	229.633	416.325	647.487	982.357	1,539.367

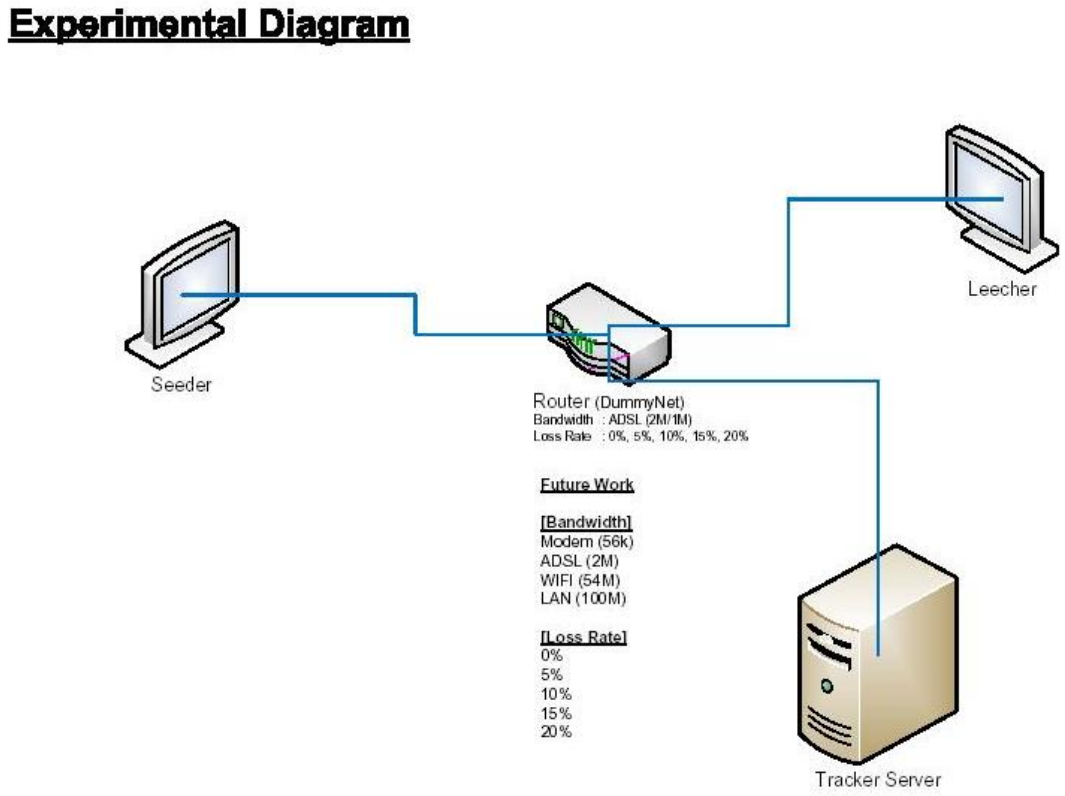


ภาพที่ 19 เปรียบเทียบเวลาที่ใช้ในการดาวน์โหลดผ่าน Modem (56 Kbps) ที่ขนาดไฟล์ต่าง ๆ

จากการทดลองได้กราฟดังภาพที่ 19 แต่ละเส้นกราฟแสดงถึงเวลาเฉลี่ยที่ได้จากการส่งข้อมูลจำนวน 100 MB ของไฟล์ขนาดต่าง ๆ กันผ่าน SCTP และ TCP ที่ loss rate 0-20% สังเกตได้ว่า ไม่ว่าขนาดไฟล์จะใหญ่เท่าไร แต่แนวโน้มของเวลาที่ใช้ก็เป็นไปในแนวทางเดียวกันกับที่ไฟล์ขนาด 100 MB เปรียบเทียบเวลาและแนวโน้ม ระหว่างที่ผ่านโพรโทคอล SCTP และ TCP เวลาที่ได้จากการดาวน์โหลดไฟล์ผ่าน SCTP ก็น้อยกว่าเวลาที่ได้จากดาวน์โหลดไฟล์ผ่าน TCP การทดลองนี้จึงพิสูจน์ว่า ขนาดของไฟล์ไม่มีผลทำให้การดาวน์โหลดไฟล์บนเครือข่าย ผ่านโพรโทคอล SCTP ด้อยลง ไม่ว่าขนาดของไฟล์จะใหญ่ขึ้นเพียงใด ประสิทธิในการทำงานของ SCTP ก็ยังคงสูงกว่า TCP ยังสามารถจัดการกับไฟล์ได้ดี แม้ว่าจะมี loss rate ก็ตาม ผู้วิจัยจึงทำการทดลองต่อไป โดยกำหนดขนาดไฟล์ที่ 100 MB เท่านั้น ควบคุมสภาพแวดล้อมของเครือข่ายผ่านระบบ DummyNet และปรับขนาด bandwidth ให้ได้ตามที่ต้องการ

สภาพแวดล้อมในการทดลอง กรณีที่ 3

จากการทดลองที่ 1 ทำการทดลองบนระบบ LAN เกิดปัญหา cross traffic มีการสูญหายของข้อมูล แต่ไม่สามารถควบคุมสภาพแวดล้อมให้เป็นไปตามที่ต้องการได้ ผลที่ได้มีความคลาดเคลื่อน จึงเปลี่ยนมาทำการทดลองบนระบบ DummyNet ทำให้สามารถควบคุมสภาพแวดล้อมของระบบให้เป็นไปตามที่ต้องการ และได้วางโครงสร้างของระบบไว้ดังภาพที่ 20



ภาพที่ 20 สภาพแวดล้อมในการทดลอง กรณีที่ 2

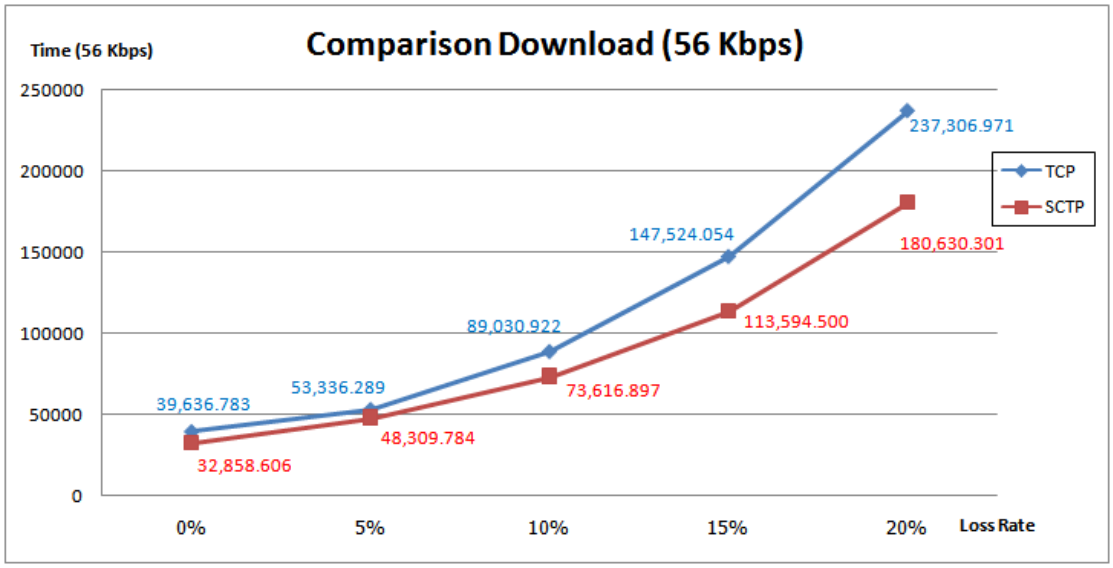
จากโครงสร้างใน diagram ประกอบด้วย tracker server ทำหน้าที่เป็นตัวกลางในการเก็บ IP Address ของผู้ที่กำลังดาวน์โหลดไฟล์ เพื่อให้ผู้ดาวน์โหลดไฟล์ สามารถติดต่อกับคนที่กำลังดาวน์โหลดไฟล์ เดียวกันบน Tracker เดียวกันได้ และจัดเก็บข้อมูลด้านสถิติด้วยว่าเครื่องใดทำการดาวน์โหลดไฟล์ใดเท่าไรแล้วบ้าง ส่วน seeder คือเครื่องที่เป็นเจ้าของไฟล์ที่เป็นเครื่องเริ่มต้นแจกจ่ายไฟล์หรือเครื่องที่ดาวน์โหลดไฟล์ได้เสร็จสมบูรณ์แล้ว แต่ยังคงเปิดต่อไปเพื่อแบ่งปันข้อมูลให้เครื่องอื่น ๆ หรือเรียกได้ว่าเครื่องที่มีไฟล์ครบนั้นเอง และ leecher คือเครื่องที่กำลัง download ไฟล์อยู่ หรือเรียกว่าเครื่องที่มีไฟล์ไม่ครบนั้นเอง จากนั้นทำการตั้งค่าโปรแกรม DummyNet ให้เป็น

Router เพื่อเป็นตัวกลางในการเชื่อมต่อ tracker server, leecher และ seeder เข้าด้วยกัน การทดลองนี้ใช้ไฟล์ขนาด 100 MB เนื่องจากการทดลองข้างต้น ทำให้เห็นว่าขนาดของไฟล์ไม่มีผลต่อการส่งข้อมูลของ SCTP ถึงแม้ไฟล์จะมีขนาดใหญ่ขึ้น ความสามารถในการขจัดปัญหา HOL ของ SCTP ยังมีประสิทธิภาพมากอยู่เช่นกัน โดยทำการส่งข้อมูลในแต่ละการทดลองจำนวน 5 ครั้ง เพื่อหาค่าเฉลี่ยของเวลาในการดาวน์โหลดไฟล์ กำหนดให้มีการเกิด lost rate ที่ 0, 5, 10, 15 และ 20% ตามลำดับ แบ่งการทดลองเป็น 4 การทดลอง โดยจำลองทำการดาวน์โหลดไฟล์ผ่าน Modem ที่ความเร็ว 56 Mbps, ADSL ความเร็ว 2 Mbps, WI-FI ความเร็ว 54 Mbps และ LAN ความเร็ว 100 Mbps จากนั้นนำค่าที่ได้ทำการสร้างกราฟ เพื่อดูแนวโน้มและเปรียบเทียบเวลาที่ใช้ในการดาวน์โหลดระหว่าง SCTP กับ TCP ภายใต้สภาพแวดล้อมที่ถูกควบคุมทั้ง loss rate และ bandwidth ผลที่ได้จึงเป็นค่าเปรียบเทียบประสิทธิภาพระหว่าง SCTP และ TCP แบบไม่มี cross traffic อื่นเกี่ยวข้อง

ผลการทดลอง

ตารางที่ 3 ค่าเฉลี่ยของเวลาที่ใช้ในการดาวน์โหลดของ Modem (56 Kbps)

loss rate / protocol	เวลาเฉลี่ยที่ใช้ในการดาวน์โหลดผ่าน Modem (56 Kbps) (sec)				
	0 %	5 %	10 %	15 %	20 %
TCP	39,636.783	53,336.289	89,030.922	147,524.054	237,306.971
SCTP	32,858.606	48,309.784	73,616.897	113,594.500	180,630.301

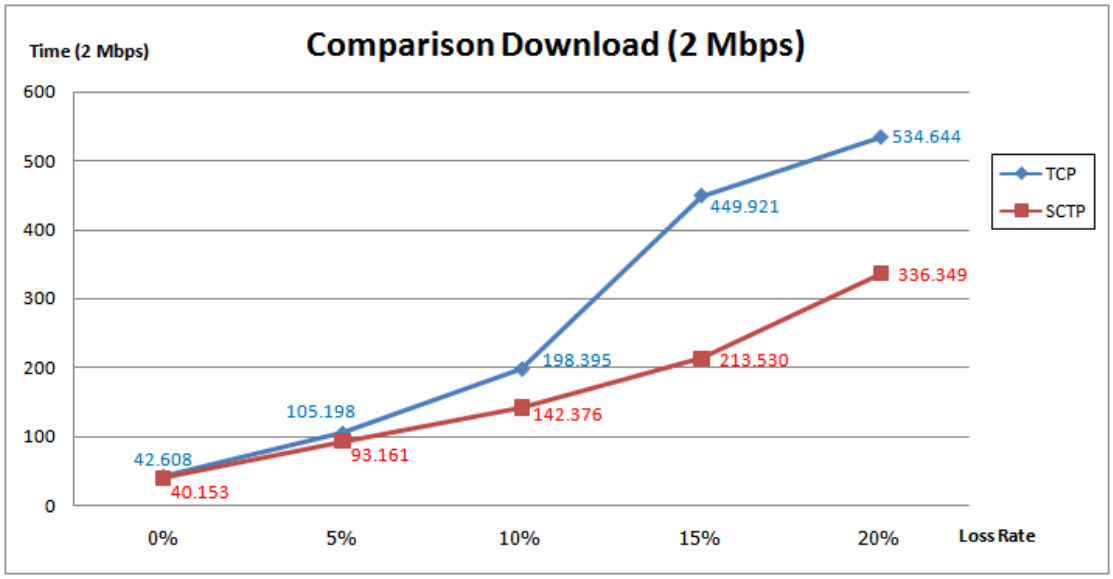


ภาพที่ 21 เวลาที่ใช้ในการดาวน์โหลดผ่าน Modem (56 Kbps) ระหว่าง TCP และSCTP

จากการทดลองส่งข้อมูล 5 ครั้ง ได้ค่าเฉลี่ยของเวลาที่ใช้ในการดาวน์โหลดผ่าน Modem ทุก ๆ loss rate ตามตารางที่ 3 สังเกต loss rate ที่ 0 และ 5% ความแตกต่างของเวลายังไม่ชัดเจน เนื่องจากมีการสูญหายของข้อมูลน้อย ก็ไม่กระทบกับ traffic มาก แต่เมื่อข้อมูลเริ่มมีการสูญหายตั้งแต่ 10% ขึ้นไป จะเห็นประสิทธิภาพในการจัดการส่งข้อมูลระหว่าง SCTP และ TCP เด่นชัดขึ้น แนวโน้มของกราฟทำให้เห็นว่า SCTP ใช้เวลาในการดาวน์โหลดไฟล์น้อยกว่า TCP

ตารางที่ 4 ค่าเฉลี่ยของเวลาที่ใช้ในการดาวน์โหลดของ ADSL (2 Mbps)

loss rate / Protocol	เวลาเฉลี่ยที่ใช้ในการดาวน์โหลดผ่าน ADSL (2 Mbps) (sec)				
	0 %	5 %	10 %	15 %	20 %
TCP	428.47	1040.80	2006.13	3516.78	5629.20
SCTP	394.09	916.63	1404.92	2094.96	3296.66

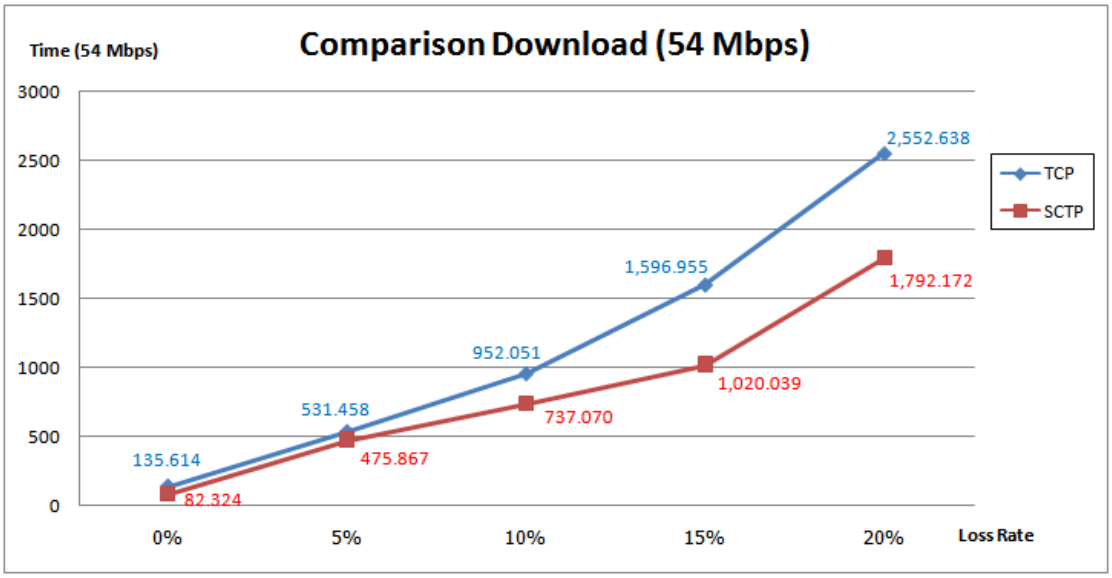


ภาพที่ 22 เวลาที่ใช้ในการดาวโหลดผ่าน ADSL (2 Mbps) ระหว่าง TCP และSCTP

จากการทดลองนี้ ได้เปลี่ยนค่าของ bandwidth ให้มีความเร็วขึ้น โดยทดลองส่งข้อมูล ADSL จากนั้นทดลองส่งข้อมูล 5 ครั้ง เพื่อหาค่าเฉลี่ยของแต่ละ loss rate ได้ผลตามตารางที่ 4 สังเกต loss rate ที่ 0 และ 5% เวลาที่ได้ยังคาบเกี่ยวกัน เนื่องจากมีการสูญหายของข้อมูลในอัตราที่น้อย แต่เมื่อข้อมูลเริ่มมีการสูญหายตั้งแต่ 10% ขึ้นไป จะเห็นประสิทธิภาพในการจัดการส่งข้อมูลระหว่าง SCTP และ TCP เด่นชัดขึ้น แนวโน้มของกราฟทำให้เห็นว่า SCTP ใช้เวลาในการดาวน์โหลดไฟล์น้อยกว่า TCP

ตารางที่ 5 ค่าเฉลี่ยของเวลาที่ใช้ในการดาวน์โหลดของ WI-FI (54 Mbps)

loss rate / protocol	เวลาเฉลี่ยที่ใช้ในการดาวน์โหลดผ่าน WI-FI (54 Mbps) (sec)				
	0 %	5 %	10 %	15 %	20 %
TCP	135.614	531.458	952.051	1,596.955	2,552.638
SCTP	82.324	475.867	737.070	1,020.039	1,792.172

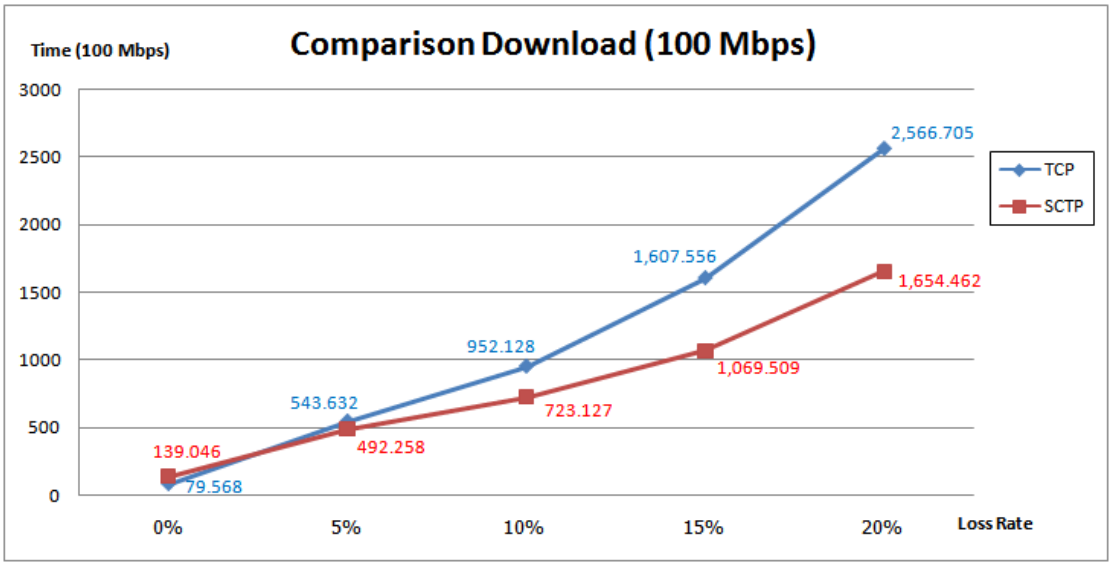


ภาพที่ 23 เวลาที่ใช้ในการดาวน์โหลดผ่าน WI-FI (54 Mbps) ระหว่าง TCP และSCTP

จากการทดลองนี้ได้กำหนด bandwidth 54 Mbps ส่งข้อมูล 5 ครั้ง ผ่าน WI-FI ได้ค่าเฉลี่ยของเวลาที่ใช้ในการดาวน์โหลด ตามตารางที่ 5 สังเกต loss rate ที่ 0 และ 5% เวลาไม่ต่างกันมาก แต่เมื่อข้อมูลเริ่มมีการสูญหายตั้งแต่ 10% ขึ้นไป จะเห็นประสิทธิภาพในการจัดการส่งข้อมูลระหว่าง SCTP และ TCP เด่นชัดขึ้น แนวโน้มของกราฟทำให้เห็นว่า SCTP ใช้เวลาในการดาวน์โหลดไฟล์น้อยกว่า TCP

ตารางที่ 6 ค่าเฉลี่ยของเวลาที่ใช้ในการดาวน์โหลดของ LAN (100 Mbps)

loss rate / protocol	เวลาเฉลี่ยที่ใช้ในการดาวน์โหลดผ่าน LAN (100 Mbps) (sec)				
	0 %	5 %	10 %	15 %	20 %
TCP	79.568	543.632	952.128	1,607.556	2,566.705
SCTP	139.046	492.258	723.127	1,069.509	1,654.462



ภาพที่ 24 เวลาที่ใช้ในการดาวน์โหลดผ่าน LAN (100 Mbps) ระหว่าง TCP และSCTP

จากการทดลองส่งข้อมูล 5 ครั้ง ได้ค่าเฉลี่ยของเวลาที่ใช้ในการดาวน์โหลดผ่าน LAN ทุก ๆ loss rate ตามตารางที่ 6 สังเกต loss rate ที่ 0% TCP ใช้เวลาในการดาวน์โหลดน้อยกว่า SCTP ทำให้เกิดการตัดกันของกราฟ การกำหนดให้ bandwidth มีขนาดถึง 100 Mbps ถือเป็นเครือข่ายที่มีความเร็วสูง ทำให้ SCTP อาจดูด้อยกว่า TCP แต่ที่นำ SCTP มาใช้เนื่องจากในความเป็นจริงแล้ว ไม่มีทางที่ข้อมูลจะไม่เกิดการสูญหายเลย นอกจากอยู่ภายใต้การควบคุมอย่างดี เพราะฉะนั้นการเลือกใช้ protocol ใด ก็ต้องคำนึงถึงความเหมาะสมในการใช้งานด้วย แต่เมื่อข้อมูลเริ่มมีการสูญหายตั้งแต่ 5% ขึ้นไป จะเห็นประสิทธิภาพในการจัดการส่งข้อมูลระหว่าง SCTP และ TCP เด่นชัดขึ้น แนวโน้มของกราฟทำให้เห็นว่า SCTP ใช้เวลาในการดาวน์โหลดไฟล์น้อยกว่า TCP

วิจารณ์

ในปัจจุบันการแลกเปลี่ยนข้อมูลเป็นสิ่งที่ทำได้ง่าย BitTorrent ก็เป็นอีกหนึ่งเทคโนโลยีที่ช่วยทำให้การแลกเปลี่ยนข้อมูลรวดเร็วขึ้น การนำ SCTP มาประยุกต์ร่วมกับ BitTorrent สามารถตอบสนองความต้องการของผู้ใช้ได้เพียงพอกับแบบเดิมที่ใช้ TCP เป็น Transport layer protocol และสามารถขจัดปัญหาช่องทางการสื่อสารของ TCP ที่สามารถสร้างได้เพียงช่องทางเดียว หากเกิดปัญหา HOL จะไม่สามารถสื่อสารข้อมูลใด ๆ ได้ในช่วงระยะเวลาหนึ่ง แต่การที่จะนำ SCTP เข้ามาใช้ร่วมกับ BitTorrent นั้น ต้องอาศัยการปรับแก้โค้ดในส่วนการติดต่อระหว่างเครื่อง และทำการกำหนดค่าระบบปฏิบัติการให้รองรับ SCTP

จุดแรกของการเริ่มต้นการทดลอง คือ ทดลองโดยการส่งข้อมูลขนาดต่าง ๆ กัน ที่ 100 MB, 300 MB, 500 MB และ 1 GB ส่งผ่านระบบ LAN จริง โดยไม่มี loss rate ได้ข้อสรุปว่า SCTP สามารถดาวน์โหลดไฟล์เร็วกว่า TCP ถึงแม้ขนาดไฟล์จะเพิ่มขึ้นก็ตาม เพื่อยืนยันว่าขนาดไฟล์ไม่มีผลต่อการส่งข้อมูล จึงได้ทำการทดลองในกรณีที่ 2 ขึ้น โดยทำการดาวน์โหลดไฟล์ขนาดต่าง ๆ กัน ที่ loss rate 0-20% ผ่าน Modem (56 Kbps) แล้วนำผลที่ได้มาเปรียบเทียบความแตกต่างของเวลาในการดาวน์โหลดข้อมูลผ่าน SCTP และ TCP สรุปได้ว่า ณ loss rate ต่าง ๆ ที่ขนาดไฟล์เท่ากัน การดาวน์โหลดข้อมูลผ่าน SCTP ก็ทำเวลาได้ดีกว่า TCP เพราะฉะนั้นไม่ว่าขนาดไฟล์จะใหญ่ขึ้นเพียงใด ประสิทธิภาพในการทำงานของ SCTP ยังคงควบคุม loss rate ได้ดี เมื่อเป็นดังนี้ จึงกำหนดขนาดไฟล์ที่ 100 MB เพื่อทำการทดลองต่อไปว่า bandwidth ผลต่อเวลาในการดาวน์โหลดอย่างไร โดยทำงานบนระบบ DummyNet ตั้งค่า loss rate ที่ 0, 5, 10, 15 และ 20% ตามลำดับ และกำหนด bandwidth เป็น Modem(56 Kbps), ADSL(2 Mbps), WI-FI(54 Mbps) และ LAN(100 Mbps) และผลที่ได้ก็คือ ประสิทธิภาพการจัดการ loss rate ของ SCTP ดีกว่า TCP จากการทดลองทั้งหมดที่กล่าวมา ได้แสดงถึงประสิทธิของโพรโทคอล SCTP ว่าสามารถทำได้มากว่า TCP อีกทั้งยังแก้ปัญหา HOL ที่เกิดกับ TCP เมื่อมีคนใช้งานเครือข่ายสูง

สรุปและข้อเสนอแนะ

สรุป

เหตุผลของการทำการทดลอง คือเนื่องจากผู้ใช้งานระบบเครือข่ายต้องการความเร็วในการดาวน์โหลดไฟล์ที่เพิ่มขึ้นและต้องการแก้ปัญหา HOL อีกทั้งเห็นว่าแนวคิดของ BitTorrent น่าสนใจเป็นการนำหลักการของ Peer-to-Peer มาประยุกต์ใช้ มีการแบ่งไฟล์เป็นชิ้น ๆ เป็นผลให้สะดวกและรวดเร็วในการดาวน์โหลด ผู้วิจัยจึงศึกษาเพื่อพัฒนาให้การดาวน์โหลดไฟล์ผ่านระบบเครือข่ายสะดวกราบรื่นมากขึ้น โดยพยายามศึกษาการทำงานของ BitTorrent ทำให้ทราบว่า BitTorrent ทำงานบนชั้น Transport layer ซึ่งเป็น layer ที่ 4 ของ OSI model โพรโทคอลที่ทำงานใน layer นี้มีหลายตัวด้วยกัน หนึ่งในนั้นคือ SCTP ซึ่งคุณสมบัติของ SCTP ที่ทำให้เลือกมาประยุกต์กับ BitTorrent คือ Multi-Streaming, Multi-Homing, 4-way handshake และ 64bit Receive Windows ซึ่งคุณสมบัติที่กล่าวมานี้ เมื่อนำมาประยุกต์ใช้กับ BitTorrent ทำให้ BitTorrent สามารถทำงานได้อย่างมีประสิทธิภาพมากขึ้น

โครงงานวิทยานิพนธ์เล่มได้ทำการทดลองไว้ 2 กรณีด้วยกัน คือกรณีแรกทำการทดลองบนระบบ LAN จริง เกิด loss rate อัตราตามสภาพเครือข่ายจริง เป็น cross traffic ที่ไม่สามารถควบคุมสภาพแวดล้อมได้ ทำการดาวน์โหลดไฟล์ขนาดต่าง ๆ กัน ที่ขนาด 100, 300, 500 MB และ 1 GB จากนั้นหาค่าเฉลี่ยของเวลาในการดาวน์โหลด ได้ข้อสรุปจากการทดลองว่า ในการดาวน์โหลดไฟล์ผ่านโพรโทคอล SCTP ใช้เวลาในการดาวน์โหลดน้อยกว่าการดาวน์โหลดไฟล์ผ่านโพรโทคอล TCP แต่ค่าที่ได้ไม่ค่อยสมบูรณ์นัก เนื่องจากจะแปรผันตาม cross traffic จึงได้ทำการทดลองในกรณีที่ 2 โดยทำการจำลองระบบโครงสร้าง BitTorrent โดยใช้เครื่องคอมพิวเตอร์โน้ตบุ๊กทั้งหมด 3 เครื่อง และคอมพิวเตอร์ตั้งโต๊ะ 1 เครื่อง ซึ่งมีสเปกดังนี้ หน่วยประมวลผล Intel Pentium 4 ความเร็ว 2.8 GHz หน่วยความจำหลัก DDR2 2.5 GB หน่วยความจำสำรอง 160 GB ความเร็วรอบ 7,200 PRM นำมาทำการตั้งค่าให้เป็น tracker server ทำหน้าที่เป็นแม่ข่ายเสมือนตัวกลางระหว่าง client และทำหน้าที่เก็บรวบรวม torrent file ไว้ เพื่อให้ client ทราบว่ามีไฟล์ใดที่สามารถดาวน์โหลดและอยู่ที่ใดบ้าง คอมพิวเตอร์โน้ตบุ๊ก หน่วยประมวลผล Intel Pentium 4 ความเร็ว 2.4 GHz หน่วยความจำหลัก DDR 256 MB หน่วยความจำสำรอง 30 GB ความเร็วรอบ 5,400 PRM ทั้งหมด 2 เครื่อง เครื่องคอมพิวเตอร์โน้ตบุ๊กเครื่องที่ 1 ทำการตั้งค่าให้เป็น seeder คือ จำลองให้เป็น client ที่มีไฟล์ให้ดาวน์โหลดฉบับสมบูรณ์ และพร้อมที่จะให้ client ผู้อื่นมาดาวน์โหลดไฟล์ เครื่อง

คอมพิวเตอร์โน๊ตบุ๊คเครื่องที่ 2 ทำการเซ็ตค่าให้เป็น leecher คือ จำลองให้เป็น client ที่ต้องการดาวน์โหลดไฟล์ ซึ่งอาจมีไฟล์ที่ต้องการบางส่วนหรือไม่มีก็ได้ leecher เป็น client ที่ขณะดาวน์โหลดไฟล์ก็สามารถเปิดบริการให้ผู้อื่นเข้ามาดาวน์โหลด และเมื่อ leecher ดาวน์โหลดไฟล์ที่ต้องการจนสมบูรณ์ก็จะเปลี่ยนสถานะเป็น seeder ได้เช่นกัน ซึ่งคอมพิวเตอร์โน๊ตบุ๊คทั้ง 2 เครื่องนี้ในตอนแรกให้ทำการเชื่อมเครือข่ายผ่านโพรโทคอล TCP แล้วทำการบันทึกเวลาที่ได้ โดยทดลองให้ดาวน์โหลดทั้งหมด 5 ครั้ง เพื่อหาค่าเฉลี่ยของเวลาไว้ จากนั้นได้มีการปรับแก้โค้ดบางส่วน ภาษาที่ใช้ในการปรับแก้โค้ด คือภาษา C ให้ทำการเชื่อมต่อเครือข่ายผ่านโพรโทคอล STCP ทำการทดลองเช่นเดียวกับการทดลองบน TCP เพื่อหาค่าเฉลี่ยเวลาแล้วนำมาเปรียบเทียบ เครื่องคอมพิวเตอร์โน๊ตบุ๊คเครื่องที่ 3 มีสเปกดังนี้ หน่วยประมวลผล Intel Pentium 4 ความเร็ว 2.4 GHz หน่วยความจำหลัก DDR 256 MB หน่วยความจำสำรอง 30 GB ความเร็วรอบ 5,400 PRM ทำการเซ็ตค่าให้เป็น Router คือ ตัวกลางสำหรับให้ leecher ติอต่อกับ seeder ใช้โปรแกรม DummyNet ในการควบคุมสภาพแวดล้อมของระบบ โดยกำหนดให้มี loss rate 0, 5, 10, 15 และ 20% bandwith เท่ากับ Modem ความเร็ว 56 Mbps, ADSL ความเร็ว 2 Mbps, WI-FI ความเร็ว 54 Mbps และ LAN ความเร็ว 100 Mbps ขนาดไฟล์ในการดาวน์โหลดเท่ากับ 100 MB เหตุที่ใช้ขนาดไฟล์เดียว เป็นผลมาจากการทดลองกรณีแรกที่แนวโน้มไม่แตกต่าง ถึงแม้ขนาดของไฟล์จะใหญ่ขึ้น SCTP ก็ใช้เวลาในการดาวน์โหลดน้อยกว่า TCP การใช้ไฟล์ที่มีขนาดเดียวทำให้ทราบว่า ความเร็วของเครือข่ายที่ต่างกัน มีผลต่อการดาวน์โหลดไฟล์อย่างไร สรุปได้ว่า bandwidth ต่ำ ทำให้เกิดความคับคั่งของข้อมูลสูง ยิ่งกำหนดให้การสูญหายของข้อมูลสูง TCP ทำงานได้ประสิทธิภาพลดลง เป็นเหตุให้ BitTorrent ที่ทำงานบน Transport layer ผ่านโพรโทคอล TCP ด้อยประสิทธิไปด้วย

สรุปสุดท้ายจากผลการทดลองในระบบเครือข่ายทดสอบพบว่า BitTorrent บน SCTP สามารถลดเวลาในการดาวน์โหลดได้ ประมาณ 10 ถึง 65% เมื่อเทียบกับการใช้ BitTorrent ร่วมกับโพรโทคอล TCP เพราะฉะนั้นเมื่อนำ BitTorrent มาใช้งานร่วมกับโพรโทคอล SCTP สามารถทำงานได้อย่างมีประสิทธิภาพมากกว่าและตอบสนองความต้องการในด้านความเร็วในการดาวน์โหลดไฟล์อย่างแท้จริง ทำให้การทดลองที่นำโพรโทคอล SCTP มาปรับใช้เพื่อพัฒนากับ BitTorrent นอกจากจะเพิ่มความเร็วแล้ว ยังสามารถขจัดปัญหา HOL ที่เกิดขึ้นกับโพรโทคอล TCP อีกด้วย เพราะฉะนั้นการเปรียบเทียบประสิทธิภาพการดาวน์โหลดไฟล์ของ BitTorrent ร่วมกับโพรโทคอล SCTP ให้ผลดีกว่าการทำงานกับโพรโทคอล TCP

ข้อเสนอแนะ

งานวิจัยชิ้นนี้ได้ศึกษาการทำงานของ TCP และ SCTP ในการนำมาประยุกต์ในเทคโนโลยี BitTorrent เพื่อเปรียบเทียบประสิทธิภาพที่ได้ ซึ่งผลสรุป คือ การนำ SCTP มาใช้ ให้ประสิทธิภาพมากกว่าการใช้ TCP เป็น Transport layer protocol ซึ่งผู้สนใจสามารถศึกษาหรือทำการวิจัยต่อได้ในด้านการนำตัวชี้วัดอื่นมาศึกษาเพิ่มเติม ซึ่งได้แก่ การนำจำนวนครั้งในการเชื่อมต่อมาเปรียบเทียบหรือเปรียบเทียบเวลาที่ใช้ในการดาวน์โหลดข้อมูลแต่ละชิ้น เพื่อเพิ่มประสิทธิภาพในการทำงานให้กับเทคโนโลยี BitTorrent หรือการนำจำนวน stream ในการส่งข้อมูลของโพรโทคอล SCTP เพื่อหาจำนวน Stream ที่ทำให้ SCTP ทำงานได้อย่างเต็มประสิทธิภาพ และศึกษากรณีการเกิด HOL ที่มีผลต่อประสิทธิภาพโดยรวมของเทคโนโลยี BitTorrent โดยทำการศึกษาเวลาที่รอทำการส่งข้อมูลซ้ำเมื่อเกิดการสูญหายของข้อมูล หรือทำการศึกษาโดยอธิบายลักษณะการเกิด HOL ว่ามีลักษณะการเกิดปัญหาดังกล่าวในรูปแบบใด เช่นใช้เวลาในการแบบลักษณะหรือใช้วิธีตรวจสอบลำดับของแพ็คเกตที่ส่งและรับว่ามีการส่งและรับสอดคล้องกันหรือไม่ มีการเสียเวลาในการหยุดรอข้อมูลในส่วนใดบ้าง รวมถึงตรวจสอบการสูญหายของข้อมูลในรูปแบบต่าง ๆ

เอกสารและสิ่งอ้างอิง

J. Teerachai, 2007. Improvement BitTorrent Performance by SCTP, ใน **การเข้าร่วมประชุมประจำปี สวทช. 2550**. สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ, กรุงเทพฯ.

J. Teerachai, 2009. การศึกษาเปรียบเทียบประสิทธิภาพการทำงานร่วมกับ TCP และ SCTP ของ P2P File Sharing Application, ใน **การประชุมวิชาการทางคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ประจำปี 2552**. มหาวิทยาลัยขอนแก่น วิทยาเขตหนองคาย, หนองคาย.

A. Jungmaier, 2001-2003. **Stream Control Transmission Protocol.** Available Source: http://tdrwww.exp-math.uni-essen.de/inhalt/forschung/sctp_fb, Oct 25, 2006.

Balk *et al.*, 2002. **Investigation of MPEG-4 Video Streaming over SCTP**. Available Source: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.9125, Jan 6, 2009.

Brennan and Curran, 2001. **SCTP Congestion Control Initial Simulation Studies**. Available Source: http://elm.eeng.dcu.ie/~opnet/papers/ITC01_SCTPCongestionControl3.pdf , Jan 16, 2009.

Caro *et al.*, 2003. **SCTP and TCP Variants: Congestion Control Under Multiple Losses.** Available Source: http://www.armandocaro.net/publications/2003.TR2003-04.acaro.pdf, Jan 16, 2009**.**

CERT advisory CA-1996-21. 1996. **TCP SYN flooding and IP spoofing attacks**. Available Source:http://www.cert.org/advisories/CA-1996-21.html, September 19, 1996.

Cheng-feng Tai, Lin-huang Chang, Ting-wei Hou. 2008. **Improvement of SCTP Perfomance during Handshake Process**. AINAW 22nd, March 25, 2008.

Department Of Defense, 2003. **What Is TCP/IP?**. Available Source: http://technet.microsoft.com/en-us/library/cc775418.aspx, July 10, 2008.

Grinnemo *et al.*, 2005. **Performance Benefits of Avoiding Head-of-Line Blocking in SCTP**. Available Source: https://dbkn.ismb.it/DBKN%20Content/6/publications/conferences/performance-benefits-of-avoiding-head-of-line-blocking-in-sctp, Jan 16, 2009.

Henrik Osterdahl, 2005.**A comparison of TCP and SCTP performance using the HTTP protocol**. Available Source: http://www.it.kth.se/courses/2G1305/Sample-papers/Henrik_Oesterdahl-sctp-20050525.pdf, March 3, 2008.

IBM, 2006. **Better networking with SCTP**. Available Source: http://www.ibm.com/developerworks/linux/library/l-sctp/?ca=dgr-lnxw03SCTP, Dec 25, 2007.

International Organization for Standardization, 2000. **ISO OSI 7 Layer Model forced with TCP/IP**. Available Source: http://mike.passwall.com/networking/netmodels/isoosi7layermodel.html, July 08, 2008.

J. Kurose and K. Ross, 2002. **TCP lifecycle**. Available Source: www.cse.unr.edu/~mgunes/cpe400/Lecture10.ppt, July 10, 2008.

Jian Liang, Rakesh Kumar and Keith W. Ross, **Understanding KaZaA**. Available Source: http://nsl.cs.surrey.sfu.ca/teaching/05/880_p2p/Papers/LKR04.pdf, Jan 6, 2009.

Johan Pouwelse, Paweł Garbacki, Dick Epema and Henk Sips, 2005. **The Bittorrent P2P File-Sharing System: Measurements and Analysis.** Available Source: http://www.springerlink.com/content/l251rj12233u05l4/, Jan 16, 2009.

Jon C. R. Bennett, Craig Partridge, Nicholas Shectman. 1999. **Packet reordering is not pathological network Behavior**. IEEE/ACM TON, Dec 6, 1999.

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

M. Allman, V. Paxson and W. Stevens. 1999. **TCP Congestion Control**. RFC-2581, April 1999.

Natarajan *et al.*,2006. **SCTP: An innovative transport layer protocol for the web**. Available Source: http://www.cis.udel.edu/~nataraja/papers/www2006.pdf , Jan 16, 2009.

P. Natarajan, JR. Iyengar, PD. Amer, R. Stewart, **SCTP an Innovative Transport Layer Protocol for the Web**. Proceedings of the 15th international conference on World Wide Web, 2006.

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

R. Rajamani, S. Kumar, N. Gupta, 2002.**SCTP versus TCP Comparing the Performance of Transport Protocols for Web Traffic.** University of Wisconsin-Madison, March 3, 2008.

Rajamani *et al.*, 2002. SCTP versus TCP: Comparing the Performance of Transport Protocols for Web **Traffic**.Available Source: http://pages.cs.wisc.edu/~raj/sctp/report.pdf, Jan 16, 2009.

Ravier *et al.*, 2001. **Experimental studies of SCTP multi-homing**. Available Source: http://www.sigtran.net/sigtran-papers/sctp-multi-homing.pdf , Jan 16, 2009.

R.S. Talab, 2002. Napster, **distributed peer sharing, and it's chronology: "you say you want a revolution?"**. Available Source: http://www.springerlink.com/content/042u48762vk3v1x0/ , Jan 6, 2009.

Scharf and Kiesel, 2006. **Head-of-line Blocking in TCP and SCTP: Analysis and Measurements**. Available Source : http://www.ikr.unistuttgart.de/Content/Publications/Archive/Sf_GLOBECOM2006_36516.pdf, Dec 25, 2006.

Sukumal, 2005. **Peer-to-Peer Technology Overview.** Available Source: http://www.ku.ac.th/netday2005/document/Sukumal.pdf, Dec 25, 2006.

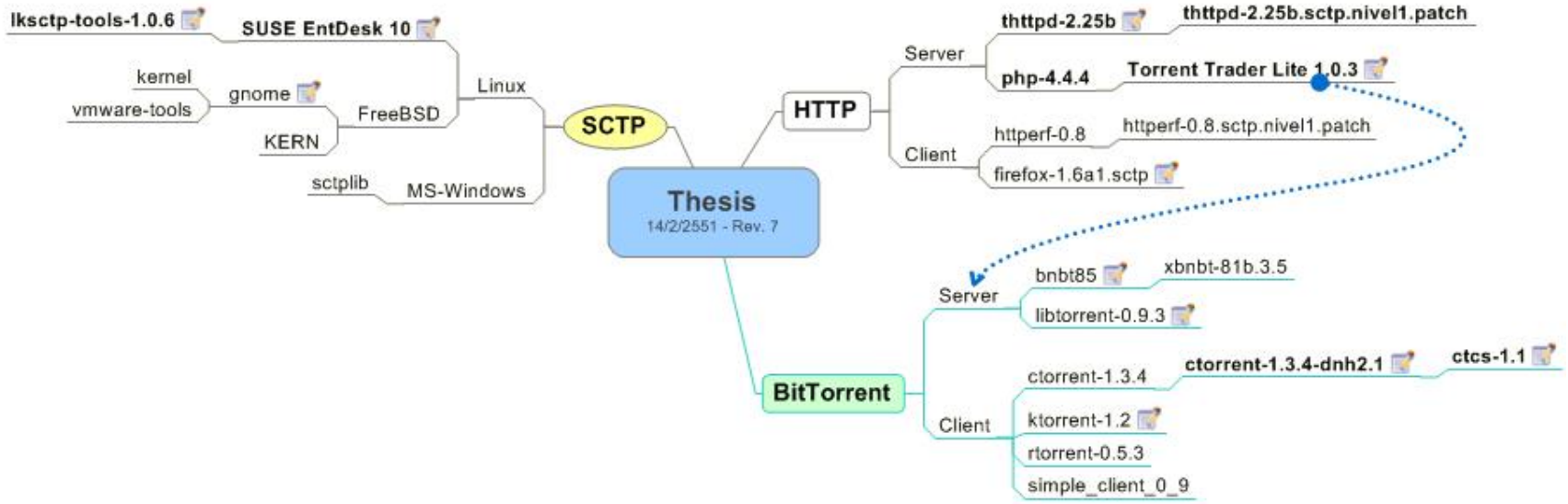
Torrent Trader Lite 1.0.3, 2005. Available Source: http://sourceforge.net/ projects/torrenttrader, March 3, 2008.

William Acosta and Surendar Chandra, **Understanding the Practical Limits of the Gnutella P2P System: An Analysis of Query Terms and Object Name Distributions**. Available Source: http://www.cse.nd.edu/~surendar/papers/mmcn08-long.pdf, Jan 16, 2009.

ภาคผนวก

ภาคผนวก ก

แผนผังความคิด เรื่องการดำเนินการโครงงานวิทยานิพนธ์


lksctp-tools-1.0.6
SUSE EntDesk 10
kernel
vmware-tools
gnome
FreeBSD
KERN
Linux
sctplib
MS-Windows
SCTP
Thesis
14/2/2551 - Rev. 7
HTTP
Server
thttpd-2.25b
thttpd-2.25b.sctp.nivel1.patch
php-4.4.4
Torrent Trader Lite 1.0.3
Client
httperf-0.8
httperf-0.8.sctp.nivel1.patch
firefox-1.6a1.sctp
BitTorrent
Server
bnbt85
xbnbt-81b.3.5
libtorrent-0.9.3
Client
ctorrent-1.3.4
ctorrent-1.3.4-dnh2.1
ctcs-1.1
ktorrent-1.2
rtorrent-0.5.3
simple_client_0_9

ภาคผนวก ข

โปสเตอร์ การเข้าร่วมประชุมประจำปี สวทช. 2550

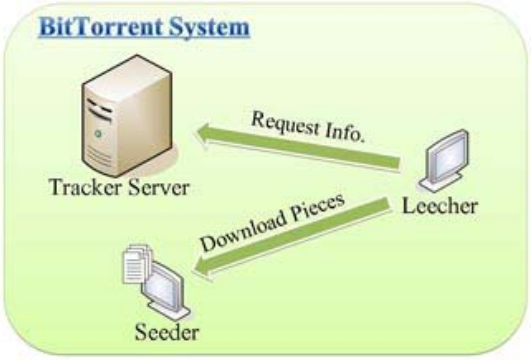
(NSTDA Annual Conference 2007)

Improvement BitTorrent Performance by SCTP

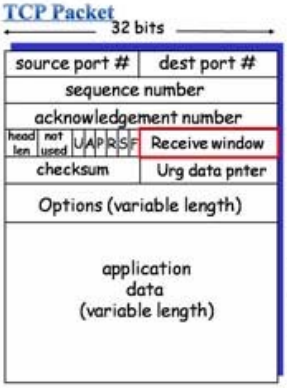
ธีรชัย เจนกิจพาณิชย์กุล, ผศ.ดร.สุขุมาล กิติสิน, ดร.ชาลิต ศรีสถาพรพัฒน์
วิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเกษตรศาสตร์
{g4864197, fscismi, fscicls}@ku.ac.th

บทคัดย่อ

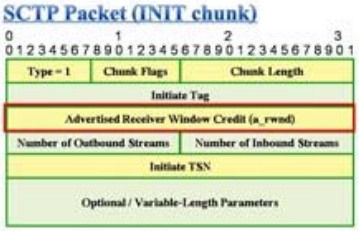
ในเครือข่ายปัจจุบันโพรโทคอล TCP มีปัญหาเรื่อง HOL ซึ่งเมื่อนำมาใช้กับระบบ BitTorrent ที่ทำงานในลักษณะ Peer-to-Peer และมีการแบ่งไฟล์ข้อมูลออกเป็นชิ้นเล็กๆ ทำให้ประสิทธิภาพในเครือข่ายลดลง และในโพรโทคอล TCP ขนาดของบัฟเฟอร์ที่ประกาศจำกัดอยู่เพียง 16 bit แต่ในระบบเครือข่ายที่มีประสิทธิภาพสูงขึ้น การจำกัดขนาดของบัฟเฟอร์ ทำให้ประสิทธิภาพของระบบเครือข่ายลดลง แม้ว่าในระบบเครือข่ายจะยังสามารถรองรับปริมาณของข้อมูลเพิ่มขึ้นได้อีกก็ตาม โพรโทคอล SCTP จึงเป็นโพรโทคอลที่ออกมาลดข้อจำกัดต่างๆ เหล่านี้



TCP Packet

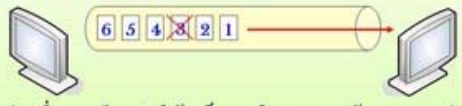


SCTP Packet (INIT chunk)



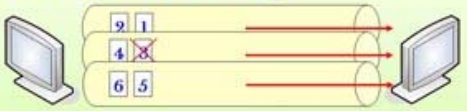
บัฟเฟอร์ที่เพิ่มจากขึ้นของโพรโทคอล SCTP ทำให้สามารถใช้ขีดความสามารถของระบบเครือข่ายปัจจุบันได้อย่างคุ้มค่า และเพิ่มประสิทธิภาพโดยรวมของระบบ

HOL on TCP



ช่องสื่อสารเดียว ทำให้แพ็คเกตใดสูญหาย ต้องรอจนกว่าจะได้รับแพ็คเกตนั้น จึงสามารถรับแพ็คเกตต่อไปได้

SCTP (Multi-Streaming)



SCTP สามารถสร้างช่องทางการสื่อสารได้หลายช่องทางในการสถาปนาการเชื่อมต่อเดียว สามารถขจัด HOL ได้

Attribute	UDP	TCP	SCTP
Reliability	Unreliable	Reliable	Reliable
Connection Management	Connectionless	Connection-Oriented	Connection-Oriented
Transmission	Message-Oriented	Byte-Oriented	Message-Oriented
Flow Control	No	Yes	Yes
Congestion Control	No	Yes	Yes
Fault Tolerance	No	No	Yes
Data Delivery	Unordered	Strictly Ordered	Partially Ordered
Security	Yes	Yes	Improved

WiCoS

Wireless Communication and System Networking Research Lab

ภาคผนวก ค

บทความเสนองานประชุมประจำปี สวทช. 2550

(NSTDA Annual Conference 2007)

Improvement BitTorrent Performance by SCTP

นายธีรชัย เจนกิจพาณิชย์กุล
มหาวิทยาลัย
เกษตรศาสตร์

ดร.ชวลิต ศรีสถาพรพัฒน์
มหาวิทยาลัย
เกษตรศาสตร์

ผศ.ดร.สุขุมาล กิติสิน
มหาวิทยาลัย
เกษตรศาสตร์

บทคัดย่อ -- ในเครือข่ายอินเทอร์เน็ตปัจจุบันโพรโตคอลที่ใช้ในการส่งข้อมูลในชั้น Transport Layer ใช้โพรโตคอล TCP โดยการทำงานของโพรโตคอล TCP มีปัญหาเรื่อง HOL ซึ่งเมื่อนำมาใช้กับระบบ BitTorrent ที่ทำงานในลักษณะ Peer-to-Peer และมีการแบ่งไฟล์ข้อมูลออกเป็นชิ้นเล็กๆ ก่อนส่งให้ผู้รับ จึงทำให้ประสิทธิภาพในการรับ-ส่งข้อมูลในเครือข่ายลดลง ในการต้องรอแพ็คเกจก่อนหน้าที่สูญหายในระบบให้ถูก Retransmit มาก่อนจึงสามารถดำเนินการรับแพ็คเกจอื่นต่อไปได้ และในโพรโตคอล TCP ขนาดของบัฟเฟอร์ที่ประกาศออกไปในระบบเครือข่ายได้ จำกัดอยู่เพียงแค่ 16 bit แต่ในระบบเครือข่ายที่มีประสิทธิภาพสูงขึ้น มีขนาดเส้นทางที่ใหญ่ขึ้น การจำกัดขนาดของบัฟเฟอร์เพียงแค่นั้น ทำให้ประสิทธิภาพโดยรวมของระบบเครือข่ายลดลง แม้ว่าในระบบเครือข่ายจะยังสามารถรองรับปริมาณของข้อมูลเพิ่มขึ้นได้อีกก็ตาม โพรโตคอล SCTP จึงเป็นโพรโตคอลที่ออกมาลดข้อจำกัดต่าง ๆ เหล่านี้ออกไปได้

งานวิจัยที่ทดลองวัดประสิทธิภาพของโพรโตคอล TCP เทียบกับโพรโตคอล SCTP มีออกมาให้เห็นเป็นอย่างมากเช่น

1. Innovative transport layer protocol for the web [1] ทำการปรับแก้ Web Server ให้ทำงานกับทั้งโพรโตคอล TCP และโพรโตคอล SCTP และทำการร้องขอข้อมูล โดยทดลองทั้งการร้องขอด้วย HTTP 1.0 แบบเปิดการเชื่อมต่อหลาย Connection และทดลองกับ HTTP 1.1 ทั้งแบบใช้ Piped lining และไม่ใช้ Piped lining ผลการทดลองโดยร่วมโพรโตคอล SCTP จะทำการได้ใกล้เคียงกับโพรโตคอล TCP แต่จะมีบางขนาดของไฟล์ที่ร้องขอ ที่โพรโตคอล SCTP จะมีประสิทธิภาพมากกว่าอย่างมาก
2. Experimental studies of SCTP multi-homing [2] ทดลองประสิทธิภาพโพรโตคอล SCTP ในเรื่อง Fault tolerance โดยใช้คุณสมบัติ Multi-homing ที่สามารถกำหนดการเชื่อมต่อกับเครื่องเซิร์ฟเวอร์ได้มากกว่าหนึ่งเครื่อง ในการทดลองทำการทดลองทั้งแบบมีการสูญหายของแพ็คเกจและแบบไม่มีการสูญหาย ผลการทดลองวัดประสิทธิภาพโดยรวมของระบบเครือข่ายที่ใช้โพรโตคอล SCTP มีเสถียรภาพสูงกว่าระบบเครือข่ายที่ใช้โพรโตคอล TCP

3. Investigation of MPEG4 Video Streaming over SCTP [3] ทดลองการทำงานของ Multi-streaming โดยทำการส่งข้อมูลวีดีโอรูปแบบ MPEG4 ที่มีคุณภาพต่างกันให้ผู้รับแบบ Multi-streaming ทำการทดลองโดยใช้โปรแกรมจำลองระบบเครือข่าย (NS-2) ผลการทดลองพบว่าโพรโตคอล SCTP ที่ใช้การส่งแบบ Multi-streaming ให้คุณภาพของวีดีโอที่ bit rate สูงกว่าการส่งด้วยโพรโตคอล TCP
4. SCTP and TCP Variants: Congestion Control Under Multiple Losses [4] ทดลองกระบวนการ Congestion control ในสภาพแวดล้อมที่มีการสูญหายของแพ็คเกจข้อมูลสูง ผลการทดลองโพรโตคอล SCTP มีการจัดการกับกระบวนการควบคุมความคับคั่งได้ดีกว่า โดยทำการส่ง SACK ออกไปน้อยกว่าโดยทำการรวมเป็นแพ็คเกจเดียว

บทความนี้ทำการวัดประสิทธิภาพของระบบ BitTorrent ที่ใช้โพรโตคอล TCP และระบบ BitTorrent ที่ใช้โพรโตคอล SCTP โดยตั้งสมมุติฐานว่า คุณสมบัติต่างๆ ของโพรโตคอล SCTP จะสามารถเพิ่มประสิทธิภาพให้กับระบบเครือข่าย BitTorrent ที่นำมาใช้ได้ เนื่องจากระบบ BitTorrent มีการแบ่งไฟล์ออกเป็นส่วนๆ ซึ่งถ้าส่งด้วยโพรโตคอล TCP จะทำการส่งข้อมูลแบบ Byte stream เมื่อมีแพ็คเกจใดสูญหาย จำเป็นต้องหยุดรอจนกว่าจะได้รับการ Retransmit แพ็คเกจนั้น ก่อนที่จะสามารถดำเนินการรับข้อมูลแพ็คเกจอื่นต่อไปได้ และแม้จะใช้คุณสมบัติ Piped lining ของโพรโตคอล TCP แต่ในการสถาปนาการเชื่อมต่อแต่ละครั้ง จำเป็นต้องทำการร้องขอซึ่งต้องเสีย Overhead อย่างสูง แต่ถ้าทำการปรับปรุงระบบ BitTorrent ให้ทำการส่งด้วยโพรโตคอล SCTP ซึ่งมีคุณสมบัติของ Multi-streaming ระบบสามารถนำข้อมูลแต่ละชิ้นที่ทำการแบ่ง ส่งออกมาในแต่ละ Stream ได้โดยไม่ต้องทำการสถาปนาการเชื่อมต่อใหม่ และในแต่ละ Stream ยังเป็นอิสระต่อกันในลักษณะของ Message stream เมื่อแพ็คเกจใน Stream ใดมีการสูญหาย จะไม่มีผลกระทบกับข้อมูลใน Stream อื่น จึงไม่มีปัญหาในการหยุดรอข้อมูล

การปรับปรุงระบบ BitTorrent เพื่อให้ใช้โพรโตคอล SCTP จะทำการปรับปรุงในชั้น Transport layer โดยทำงานอยู่บนระบบปฏิบัติการ Linux ซึ่งเป็นระบบปฏิบัติการที่รองรับการทำงานของโพรโตคอล SCTP ด้วยการปรับแก้ kernel ของตัวระบบปฏิบัติการ ในเครื่อง BitTorrent Server ต้องทำการติดตั้งโปรแกรม tracker server ที่ได้รับการปรับแก้ให้สามารถใช้งานกับโพรโตคอล SCTP แล้ว ซึ่งบทความนี้ดำเนินการทดลองและปรับแก้โปรแกรม tracker server เพื่อใช้กับโพรโตคอล SCTP เสร็จเรียบร้อยแล้ว ส่วนเครื่องที่ต้องการใช้บริการระบบ BitTorrent จำเป็นต้องทำการติดตั้งโปรแกรม BitTorrent Client บทความนี้ทำการทดลองปรับแก้โปรแกรม BitTorrent Client เพื่อใช้ในการทดลองแล้วหลายโปรแกรม โดยทำการทดลองในสภาพแวดล้อม

จริง ที่ยังไม่ได้กำหนดอัตราการสูญหายของแพ็คเกจในเครือข่าย ผลการทดลองเบื้องต้น ระบบ BitTorrent ที่ทำงานบนโพรโตคอล SCTP ให้ประสิทธิภาพการทำงานใกล้เคียงกับระบบที่ทำงานบนโพรโตคอล TCP แต่เนื่องจากระบบเครือข่ายที่ทำการทดลอง เป็นระบบเครือข่ายภายในพื้นที่ (LAN) จึงทำให้อัตราการสูญหายของแพ็คเกจต่ำ แต่ในสภาพแวดล้อมในระบบเครือข่ายอินเทอร์เน็ตจริง จะมีความคับคั่งของระบบเครือข่ายและมีอัตราการสูญหายของแพ็คเกจข้อมูลที่สูงกว่า ผลการทดลองกับระบบที่ทำงานบนโพรโตคอล SCTP จึงน่าที่จะให้ประสิทธิภาพที่สูงกว่า แนวทางในการทดลองในอนาคตบนระบบเครือข่ายภายในพื้นที่คือ ทำการกำหนดอัตราการสูญหายในระบบเครือข่ายให้สูงขึ้น โดยทำการกำหนดอัตราการสูญหายในตัว Router หรือทำการปรับแก้ระบบปฏิบัติการ Linux ให้มีการสูญหายสูงขึ้น

คำสำคัญ – BitTorrent, SCTP, TCP, HOL

เอกสารอ้างอิง

[1] P. Natarajan, J. Iyengar, P. Amer, R. Stewart. “An innovative transport layer protocol for the web”, May 2006.

[2] T. Ravier, R. Brennan, and T. Curran. "Experimental studies of SCTP multihoming”. Nov 2001.

[3] A. Balk, M. Sigler, M. Gerla, and M.Y. Sanadidi. “Investigation of MPEG-4 video streaming over SCTP”. 2002.

[4] A. Caro, K. Shah, J. Iyengar, P. Amer, R. Stewart. “SCTP and TCP Variants: Congestion Control Under Multiple Losses”. 2003.

โปรดระบุรูปแบบที่ท่านต้องการนำเสนอ

☑ นำเสนอในรูปแบบบทความ ☐ นำเสนอในรูปแบบโปสเตอร์

ภาคผนวก ง

บทความ เสนอ การประชุมวิชาการทางคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ประจำปี 2552

(Conference on Computer Information Technologies 2009)

การศึกษาเปรียบเทียบประสิทธิภาพการทำงานร่วมกับ TCP และ SCTP ของ P2P File Sharing Application

Comparison Study on the Performance of P2P File Sharing Application using TCP vs. SCTP

นายธีรชัย เจนกิจพาณิชย์กุล *ผศ.ดร.ชวลิต ศรีสถาพรพัฒน์* *ผศ.ดร.สุขุมาล กิติสิน*
มหาวิทยาลัยเกษตรศาสตร์ *มหาวิทยาลัยเกษตรศาสตร์* *มหาวิทยาลัยเกษตรศาสตร์*
g4864197@ku.ac.th *fscicls@ku.ac.th* *fscismi@ku.ac.th*

Abstract

P2P file sharing applications using TCP suffer from head-of-line blocking problem. This research attempts to alleviate this by modifying BitTorrent, one of the most popular P2P file sharing applications, to work with SCTP instead of TCP. SCTP allows an application to simultaneously establish more than one Transport layer connection to a target application on the remote machine. This feature is called multi-streaming. Our experimental result indicates 1 to 3% speed up in total file transfer time when using BitTorrent with SCTP over TCP. The speed up tend to increase with the size of file transferred.

Keywords: P2P, BitTorrent, SCTP, TCP, HOL

บทคัดย่อ

ระบบเครือข่าย P2P ที่ใช้ TCP ในการทำงาน ในการแลกเปลี่ยนข้อมูลพบว่ามีปัญหา HOL ซึ่งปัญหานี้ทำให้ TCP ไม่เหมาะต่อการนำมาใช้ในระบบเครือข่าย ที่มีการย่อยข้อมูลออกเป็นส่วนๆ เพื่อแลกเปลี่ยน จึงได้มีการพัฒนา SCTP ขึ้นมาเพื่อลดปัญหาดังกล่าว งานวิจัยนี้จึงทำการศึกษาเปรียบเทียบการทำงานร่วมกับ TCP และ SCTP ของ P2P File Sharing Application บทความนี้ประกอบด้วยหกส่วน ได้แก่ ส่วนแรกเป็นบทนำและความสำคัญของปัญหา ส่วนที่สองเป็นงานวิจัยที่เกี่ยวข้อง ส่วนที่สามเป็นแนวทางการปรับปรุง BitTorrent ให้ทำงานร่วมกับ SCTP ส่วนที่สี่เป็นผลการทดลองเบื้องต้น ส่วนที่ห้าและหกเป็นสรุปแนวทางการวิจัยในอนาคตและเอกสารอ้างอิงตามลำดับ ผลการทดลองเมื่อนำเอา SCTP มาปรับใช้ใน P2P พบว่าด้วยคุณสมบัติ Multi-Streaming

ของ SCTP ทำให้ขจัดปัญหา HOL ได้ อีกทั้งยังพบว่ามีความเร็วในการดาวน์โหลดไฟล์มากขึ้น 1 ถึง 3% และมีแนวโน้มมีความเร็วเพิ่มขึ้นเมื่อไฟล์มีขนาดใหญ่ขึ้น

คำสำคัญ P2P, BitTorrent, SCTP, TCP, HOL

1. บทนำและความสำคัญของปัญหา

การสื่อสารผ่านเครือข่ายคอมพิวเตอร์ที่มีลักษณะเป็นแบบรับ/ให้บริการ (client/server system) นั้น เครื่องลูกข่าย (client) จะติดต่อขอรับบริการจากเครื่องแม่ข่าย (server) โดยตรง ซึ่ง server ในระบบเครือข่ายอาจมีจำนวนน้อย ทำให้ server ต้องรับภาระการทำงานหนัก จึงต้องใช้เครื่องที่มีประสิทธิภาพสูง โดยเฉพาะกรณีที่มี client จำนวนมาก เมื่อมีการส่งข้อมูลระหว่าง client และ server เพิ่มมากขึ้นจะส่งผลให้ต้องใช้ระบบเครือข่ายที่มีประสิทธิภาพสูงขึ้น นอกจากนี้ถ้า server ไม่สามารถให้บริการได้จะทำให้ client ทุกเครื่องไม่สามารถติดต่อเพื่อขอข้อมูลจาก server ได้

สำหรับการสื่อสารผ่านเครือข่ายคอมพิวเตอร์ในลักษณะ peer-to-peer นั้น เครื่องคอมพิวเตอร์แต่ละเครื่องทำหน้าที่เป็นทั้ง server และ client ในเวลาเดียวกัน ดังนั้นระบบเครือข่ายจึงสามารถรองรับจำนวนเครื่องคอมพิวเตอร์ที่ใช้งานเครือข่ายได้เพิ่มมากขึ้น โดยที่เครื่องคอมพิวเตอร์แต่ละเครื่องไม่จำเป็นต้องมีประสิทธิภาพสูงมาก สามารถใช้อุปกรณ์ในลักษณะของเครื่อง client ได้ ทั้งนี้เนื่องจากเมื่อมีเครื่องคอมพิวเตอร์ในระบบมาก จะยิ่งมี server มาก ทำให้ server แต่ละเครื่องไม่ต้องรับภาระในการให้บริการกับ client จำนวนมาก ข้อดีอีกประการหนึ่งของระบบนี้คือ client แต่ละเครื่องยังสามารถขอรับบริการจาก server ได้มากกว่าหนึ่งเครื่องในเวลาเดียวกัน เมื่อ server เครื่องใดขัดข้องไป ก็ยังมี server อื่นเหลืออยู่ในระบบ ทำให้การสื่อสารในเครือข่ายนั้นยังสามารถดำเนินต่อไปได้ การให้บริการแบบ peer-to-peer มี 3 รูปแบบ คือ แบบมีศูนย์กลาง (centralized P2P) แบบไม่มีศูนย์กลาง (pure P2P) และแบบผสม (hybrid P2P)

BitTorrent [2] เป็นเทคโนโลยีการแลกเปลี่ยนไฟล์ ในรูปแบบ peer-to-peer แบบผสมที่มี server ศูนย์กลางทำหน้าที่เก็บรายละเอียดไฟล์ที่มีการแลกเปลี่ยนกันในระบบ โดยเครื่องของผู้ใช้แต่ละเครื่องอาจเป็นทั้งผู้ให้บริการและผู้รับบริการได้ในขณะเดียวกัน BitTorrent จะแยกไฟล์ที่แลกเปลี่ยนกันออกเป็นส่วนย่อยหลายส่วน เพื่อให้ client เลือกดาวน์โหลดจาก server หลายเครื่องได้พร้อมกัน ซึ่งวิธีนี้ยังผลให้การโอนถ่ายไฟล์ทำได้รวดเร็วขึ้น

ปัญหาประการหนึ่งของ BitTorrent คือ BitTorrent ใช้ TCP (Transmission Control Protocol) เป็น Transport layer protocol เนื่องจากการทำงานของ TCP มีการสถาปนาการเชื่อมต่อ (connection oriented) มีความน่าเชื่อถือในการส่งข้อมูล (reliable) มีการจัดเรียงลำดับ (ordering) ซึ่ง

จะส่งข้อมูลให้กับ application ต่อเมื่อข้อมูลที่ได้รับเรียงลำดับอย่างถูกต้องแล้ว ดังนั้นถ้าหากมีการสูญหายหรือล่าช้าของข้อมูลที่ได้รับมา TCP จะไม่ส่งข้อมูลให้ application โดย TCP จะร้องขอข้อมูลส่วนที่สูญหายหรือล่าช้าดังกล่าวจนกว่าจะได้รับอย่างถูกต้องจึงดำเนินการส่งข้อมูลให้กับ application ปัญหานี้ของ TCP เรียกว่า *Head-of-line blocking (HOL)* ซึ่งทำให้ TCP ไม่เหมาะสมที่จะนำมาใช้สำหรับการแลกเปลี่ยนไฟล์ใน BitTorrent เนื่องจากทำให้ไม่ได้รับประโยชน์จากการแยกไฟล์ออกเป็นส่วนย่อย

นอกจากนี้ปัญหาที่สำคัญอีกประการหนึ่งของ TCP คือ TCP สร้างช่องทางการสื่อสารเพียงช่องทางเดียว ถ้าช่องทางการสื่อสารนั้นเกิดปัญหา HOL แล้ว จะไม่สามารถสื่อสารข้อมูลได้ในช่วงระยะเวลาหนึ่ง

จากปัญหาดังกล่าวของ BitTorrent ทำให้ผู้วิจัยสนใจศึกษาแนวทางที่จะนำโพรโทคอลอื่นที่มีความสามารถสนับสนุนการรับ-ส่งไฟล์แบบแยกส่วนและมีการสร้างช่องทางการสื่อสารมากกว่าหนึ่งช่องทาง ซึ่งสามารถทำงานทดแทน TCP ได้ มาเพิ่มประสิทธิภาพให้กับ BitTorrent โดยได้พบว่า SCTP (Stream Control Transmission Protocol) มีความสามารถของ TCP ครบถ้วน อีกทั้งยังมีการแยกข้อมูลออกเป็นส่วนย่อย (fragmentation) สามารถสร้างช่องทางการสื่อสารได้มากกว่าหนึ่งช่องทาง (multi-streaming) มีระบบความมั่นคงดีกว่า TCP และสามารถใช้งาน receive windows ที่มีขนาดถึง 32 บิต ทำให้รองรับประสิทธิภาพของระบบเครือข่ายที่เพิ่มขึ้นได้ งานวิจัยนี้ได้ศึกษาวิธีการนำ SCTP มาใช้งานร่วมกันกับ BitTorrent และเปรียบเทียบให้เห็นถึงประสิทธิภาพที่เพิ่มขึ้น โดยหัวข้อที่ 2 จะได้กล่าวถึงงานวิจัยที่เกี่ยวข้อง และนำเสนอแนวคิดการปรับปรุงในหัวข้อที่ 3 พร้อมชี้ให้เห็นถึงประสิทธิภาพที่เพิ่มขึ้นในหัวข้อที่ 4 และในหัวข้อสุดท้ายได้กล่าวถึงผลสรุปและงานวิจัยต่อเนื่อง รวมถึงเอกสารที่ใช้อ้างอิง

2. งานวิจัยที่เกี่ยวข้อง

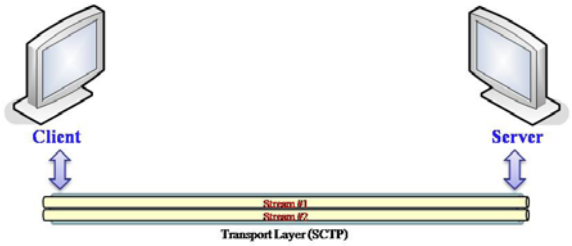
2.1 SCTP (Stream Control Transmission Protocol)

SCTP ถูกพัฒนาขึ้นเพื่อรองรับการส่งข้อมูลที่ไม่เหมาะที่จะส่งด้วย TCP โดย IETF (Internet Engineering Task Force) SIGTRAN (Signaling Transport) ซึ่งเป็นกลุ่มผู้พัฒนาโพรโทคอลใหม่ โดยได้เผยแพร่ออกสู่สาธารณะชนเมื่อเดือนตุลาคม พ.ศ.2543 ใน RFC2960 SCTP มีลักษณะการทำงานที่คล้ายคลึงกับ TCP แต่ได้เพิ่มคุณสมบัติหลายประการเพื่อแก้ปัญหาต่างๆ ที่พบใน TCP ซึ่งคุณสมบัติเหล่านั้นได้แก่

2.1.1 Multi-Streaming

จากปัญหาที่พบใน TCP ที่มีช่องทางในการสื่อสารเพียงช่องทางเดียว SCTP จึงถูกพัฒนาให้สามารถส่งข้อมูลไปพร้อมกันได้ โดยเปรียบเสมือนกับมีหลายช่องทาง (stream) การสื่อสารในการสถาปนาการเชื่อมต่อเพียงครั้งเดียว โดยจะกำหนดค่าเฉพาะให้กับข้อมูลในแต่ละช่องทาง เพื่อให้ข้อมูลสามารถส่งออกไปได้พร้อมกัน เมื่อ client ได้รับข้อมูลในแต่ละช่องทางมาแล้ว client จะดำเนินการจัดเรียงและตรวจสอบข้อมูลในแต่ละช่องทางที่ได้รับ ถ้ามีการสูญหายของข้อมูลในช่องทางใด client จะร้องขอข้อมูลเฉพาะของช่องทางนั้นเท่านั้น โดยที่ไม่มีผลกระทบกับข้อมูลในช่องทางอื่น ทำให้ช่องทางอื่นยังสามารถดำเนินการรับข้อมูลต่อไปได้ ไม่จำเป็นต้องหยุดรอทั้งหมด ซึ่งถือได้ว่าเป็นการแก้ปัญหา HOL ที่พบใน TCP

รูปที่ 1 แสดงการส่งข้อมูลระหว่าง client กับ server ซึ่ง client สร้างช่องทางการสื่อสาร 2 ช่องทาง จะเห็นว่าหากข้อมูลลำดับใด ซึ่งถูกส่งด้วยช่องทางด้านที่ 1 สูญหาย จะไม่กระทบต่อการส่งข้อมูลในช่องทางอื่นเลย ช่องทางอื่นยังสามารถดำเนินการส่งข้อมูลระหว่างกันได้ มีเฉพาะช่องทางด้านที่ 1 เท่านั้นที่หยุดรอเพื่อให้ข้อมูลส่งมาในลำดับที่ถูกต้อง จึงดำเนินการรับ-ส่งต่อ



รูปที่ 1 Multi-Streaming

2.1.2 Multi-Homing

Multi-Homing เป็นคุณสมบัติที่ยอมให้มีการเชื่อมต่อกับ server ได้มากกว่า 1 เครื่องในเวลาเดียวกัน เพื่อขจัดปัญหาการสื่อสารขัดข้อง เครือข่ายหยุดทำงาน โดยเมื่อ server ใดมีปัญหาไม่สามารถให้บริการได้ client สามารถร้องขอข้อมูลจาก server อื่นได้ทันทีโดยไม่จำเป็นต้องเริ่มขอสถาปนาการเชื่อมต่อใหม่

2.1.3 4-way Handshake

การสถาปนาการเชื่อมต่อใน SCTP เป็นแบบ *4-way Handshake [3]* ซึ่งสามารถขจัดปัญหา *SYN-flood [4]* ที่เกิดกับ TCP ได้

2.1.4 Receive Windows ขนาด 32-bit

Receive Windows [5] ใน SCTP มีขนาดเพิ่มขึ้นเป็น 32-bit จากเดิมใน TCP ซึ่งมีเพียง 16-bit ทำให้ SCTP สามารถใช้ทรัพยากรเครือข่ายได้อย่างเต็มประสิทธิภาพ

2.2 P2P แบบผสม (Hybrid P2P)

จากข้อดีและข้อเสียของการให้บริการทั้งแบบมีศูนย์กลางและแบบไม่มีศูนย์กลางจึงมีการคิดค้นรูปแบบการให้บริการรูปแบบใหม่ โดยนำข้อดีของทั้งสองแบบมารวมกัน คือการนำเอาข้อดีของการมี server ศูนย์กลางซึ่งสามารถทำงานได้เร็ว และการกระจายข้อมูลให้ client เพื่อลดปัญหา server หยุดทำงาน เพื่อพัฒนาระบบ P2P ให้มีประสิทธิภาพสูงขึ้น Hybrid P2P ทำงานโดยใช้หลักการกระจาย server ที่ให้บริการในการค้นหาแหล่งข้อมูลที่มีในแต่ละ peer ออกไปเป็นหลายเครื่อง ทำให้ได้ระบบ P2P ที่มีประสิทธิภาพสูงขึ้น โปรแกรมที่ทำงานในรูปแบบผสมเช่น KaZaA และ BitTorrent

KaZaA เป็นโปรแกรมที่พัฒนาคุณลักษณะต่างๆ ให้เพิ่มขึ้น เช่น สามารถดาวน์โหลดไฟล์แต่ละไฟล์แบบขนานได้ สามารถสลับการเชื่อมต่อกับเครื่องที่มีไฟล์ที่ต้องการได้โดยอัตโนมัติเมื่อเครื่องที่ดาวน์โหลดอยู่ปัจจุบันมีปัญหา สามารถคำนวณระยะเวลาที่ใช้ในการดาวน์โหลดแต่ละไฟล์ และสามารถกำหนดจำนวนสูงสุดที่โปรแกรมจะทำการดาวน์โหลดไฟล์ในแต่ละเครื่องพร้อมกันได้ นอกจากนี้ KaZaA ยังมีระบบ SuperNodes คือเครื่องที่มีสถิติเวลาในการให้บริการสูงและมีประสิทธิภาพระบบเครือข่ายที่สูง เพื่อให้เครื่องที่ต้องการค้นหาไฟล์ สามารถค้นหาจากเครื่อง SuperNodes ได้ก่อนที่จะทำการค้นหาจากเครื่องทั่วไป ซึ่งช่วยรับประกันความสมบูรณ์และประสิทธิภาพของไฟล์ที่จะทำการโหลดได้มากขึ้น

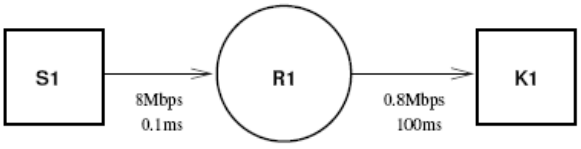
BitTorrent เป็นเทคโนโลยีการแลกเปลี่ยนไฟล์ที่ได้รับความนิยมมากที่สุด โดยทำงานในแบบผสมที่ต้องมี server ศูนย์กลาง เรียกว่า tracker server เพื่อทำหน้าที่เก็บรายละเอียดไฟล์ที่มีการแลกเปลี่ยนกัน ซึ่งเรียกว่า torrent file มีนามสกุลไฟล์เป็น .torrent ภายในไฟล์เก็บรายการ tracker server ของผู้ให้บริการ รายละเอียดไฟล์ที่แลกเปลี่ยน ค่าจากการคำนวณเพื่อใช้ตรวจสอบความถูกต้อง เมื่อ client ต้องการดาวน์โหลดไฟล์จะติดต่อไปยัง tracker server เมื่อ tracker server รับข้อความร้องขอจะทำการตรวจสอบในฐานข้อมูลของตนว่าในขณะนั้นมี peer อยู่ที่ใดบ้าง เมื่อได้รายการทั้งหมดแล้วจะส่งกลับให้ client เมื่อ client ได้รายละเอียดพร้อมแล้ว จะทำการติดต่อตรงไปยัง peer ที่ให้บริการอยู่เพื่อขอดาวน์โหลดไฟล์ การขอดาวน์โหลดไฟล์ไปยัง peer แต่ละเครื่อง client จะขอดาวน์โหลดเฉพาะบางส่วนของไฟล์ ซึ่งระบบ BitTorrent จะทำการแบ่งไฟล์ออกเป็นส่วน แต่ละส่วนมีขนาดเท่ากัน สำหรับ peer ที่ให้บริการ file นี้ ในระบบ BitTorrent เรียกอีกอย่างหนึ่งว่า seeder และ client มีชื่อเรียกอีกอย่างว่า leecher โดยเครื่องที่จะเป็น seeder ได้ต้องได้รับไฟล์ทุกชิ้นสมบูรณ์แล้วเท่านั้น ข้อดีอีกประการของระบบ BitTorrent คือ ในการดาวน์โหลดแต่ละชิ้นของไฟล์ leecher สามารถขอดาวน์โหลดจาก leecher อื่นได้ ไม่จำเป็นต้องขอดาวน์โหลดจาก

seeder เท่านั้น และในทางกลับกันในขณะที่เครื่องผู้ใช้ซึ่งเป็น leecher กำลังดาวน์โหลดไฟล์จากเครื่องอื่นอยู่ เครื่องอื่นก็สามารถขอดาวน์โหลดไฟล์จากเครื่องผู้ใช้ได้

2.3 การแก้ปัญหา Head-of-line blocking

จากปัญหา HOL ที่พบใน BitTorrent เมื่อทำงานร่วมกับ TCP จึงมีงานวิจัยหลายชิ้นที่กล่าวถึงการแก้ปัญหาการทำงาน โดยการใช้ SCTP เป็น Transport layer protocol เนื่องจากมีความสามารถหลายอย่างเพิ่มขึ้นมาเช่น

Brennan and Curran, 2001 [6] และ *Caro et al., 2003 [7]* ได้ทำการทดลอง congestion control ใน SCTP โดยทำการทดสอบทั้งในเครือข่ายที่มีความคับคั่งสูงและเครือข่ายที่มีการสูญหายของข้อมูลสูง โดยกำหนดให้ SCTP ทำงานเพียง stream เดียว และทำการปรับรูปแบบการทำงาน ให้ทำงานสอดคล้องกับการทำงานของ TCP แนวคิดของงานวิจัยคือ SCTP น่าจะมีประสิทธิภาพดีกว่า TCP เมื่อมีการสูญหายของข้อมูลสูง ทำการทดลองโดยจำลองการทำงานบนระบบ NS-2 กำหนด bandwidth ระหว่าง server และ router ที่ 8Mbps delay เท่ากับ 0.1ms และระหว่าง router และ client ที่ 0.8Mbps delay เท่ากับ 100ms ดังรูปที่ 2



รูปที่ 2 ไดอะแกรมระบบทดลอง (A. Caro, 2003)

ผลการทดลองสรุปว่า เมื่อมีการสูญหายของข้อมูลสูง SCTP มีประสิทธิภาพมากกว่า TCP ประมาณ 10 ถึง 30% เนื่องจากความสามารถในการทำ fast recovery ของ SCTP ดีกว่า TCP งานวิจัยทำการทดลองทำงานเพียง stream เดียว ไม่ได้นำความสามารถของ SCTP มาใช้อย่างเต็มที่ แนวทางในการวิจัยต่อไปคือ ศึกษาประสิทธิภาพเพื่อให้ SCTP ทำงานแบบ multi-streaming

Grinnemo et al., 2005 [8] ได้ทดลองส่งข้อมูลแบบ ordered และ unordered ใน SCTP เพื่อเปรียบเทียบและทดสอบประสิทธิภาพการแก้ปัญหา HOL แนวคิดของงานนี้คือ การส่งข้อมูลแบบ unordered น่าจะมีความเร็วมากกว่าการส่งแบบ ordered และ SCTP น่าจะขจัดปัญหา HOL ได้ โดยจำลองการทำงานบนระบบ DummyNet และกำหนดความเร็วในการรับ-ส่งที่ 133, 200, 400 Kbps และแบบไม่จำกัดความเร็ว แพ็คเกตขนาด 500 bytes และ queue ขนาด 12 และ 32 แพ็คเกต โดยทำการทดลองส่งข้อมูลทั้งหมด 1,000 แพ็คเกตในแต่ละความเร็ว

ผลการทดลองพบว่า การส่งข้อมูลแบบ unordered มีความเร็วกว่าการส่งแบบ ordered ประมาณ 0 ถึง 18% แนวทางในการทดลองต่อไปคือ กำหนดอัตราการสูญหายของข้อมูล เพื่อทดสอบปัญหา HOL ที่ SCTP สามารถขจัดได้เปรียบเทียบกับ TCP

Natarajan et al., 2006 [9] ได้ทดลองโดยการจำลอง web server ที่ทำงานบน HTTP และใช้ SCTP ส่งข้อมูลแบบ multi-streaming เพื่อแก้ปัญหา HOL โดยทดลองทั้งแบบส่งแต่ละ object แยกไปในแต่ละ stream และรวมส่งใน stream เดียวกัน แนวคิดของงานวิจัยคือ ถ้าใช้ SCTP ร่วมกับ HTTP น่าจะสามารถขจัดปัญหา HOL ได้ วิธีการทดลองทำโดยปรับ Apache Web Server ให้รองรับ SCTP จำลองระบบ DummyNet และกำหนดความเร็วที่ 56 Kbps แบบ full duplex โดยใช้ queue ขนาด 50 KBytes และไม่จำกัดขนาดคิว อัตราการสูญหาย 0 และ 10%

ผลการทดลองสรุปว่าสามารถขจัดปัญหา HOL ที่พบใน TCP ได้ และการส่ง object แยกไปแต่ละ stream ทำให้ใช้เวลาในการส่งลดลงประมาณ 10 ถึง 40%

Rajamani et al., 2002 [10] และ *Henrik, 2005 [11]* ได้ทดลองส่งข้อมูลด้วย SCTP แบบ multi-streaming บน HTTP แนวคิดของงานนี้คือ การส่งข้อมูลแบบหลายช่องทาง น่าจะมีความเร็วกว่าการส่งแบบช่องทางเดียว ทำการทดลองโดยกำหนด RTT ที่ 80ms อัตราการสูญหายที่ 0 ถึง 25% ความเร็วในการส่งที่ 40, 400 Kbps, 3 และ 10 Mbps โดยทำการทดลอง 2 กรณีคือ ส่งแบบไฟล์เดียวไม่มีการสูญหายของข้อมูล และส่งแบบหลายไฟล์มีการสูญหายของข้อมูล

ผลการทดลองพบว่า TCP สามารถส่งข้อมูลได้เร็วกว่า เมื่อไม่มีการสูญหายของข้อมูล เนื่องมาจาก SCTP มี overhead สูงกว่า แต่เมื่อมีการสูญหายของข้อมูล SCTP สามารถส่งข้อมูลได้เร็วกว่า TCP ประมาณ 15 ถึง 20% และมีอัตราการส่งที่เร็วกว่าเพิ่มขึ้น เมื่อมีการสูญหายของข้อมูลเพิ่มขึ้น

3. แนวทางการปรับปรุง BitTorrent ให้ทำงานร่วมกับ SCTP

จากผลการทดลองของงานวิจัยที่ศึกษาสรุปได้ว่า SCTP สามารถขจัดปัญหา HOL ที่เกิดกับ TCP ได้ และคุณสมบัติ multi-streaming ของ SCTP น่าจะเหมาะสำหรับการรับ-ส่งไฟล์แบบแยกส่วนของ BitTorrent งานวิจัยนี้จึงเห็นว่า สามารถพัฒนาให้ BitTorrent ทำงานบน SCTP ได้ โดยทำการปรับระบบ BitTorrent client ให้รองรับการทำงานกับ SCTP แบบ multi-streaming ทำการส่งข้อมูลแต่ละชิ้นของ BitTorrent ไปในแต่ละ stream และมีหมายเลข stream กำกับไปในชิ้นส่วนที่ส่ง

จากแนวคิดในการปรับใช้ SCTP ร่วมกับ BitTorrent งานวิจัยนี้ผู้วิจัยเลือก cTorrent ซึ่งเป็น BitTorrent client พัฒนาด้วยภาษา C เพื่อทำการปรับแก้ โดยใส่โค้ดส่วนการติดต่อระหว่างเครื่องจากเดิมทำการติดต่อด้วย TCP ให้ติดต่อด้วย SCTP แทน และทำการตั้งค่าระบบปฏิบัติการให้รองรับ SCTP โดยเลือกใช้ระบบปฏิบัติการ Linux ในการทดลองและ kernel ต้องเวอร์ชั่นมากกว่า 2.4 ขึ้นไป ซึ่งถ้าเวอร์ชั่นต่ำกว่านี้จะไม่รองรับ SCTP

4. ผลการทดลองเบื้องต้น

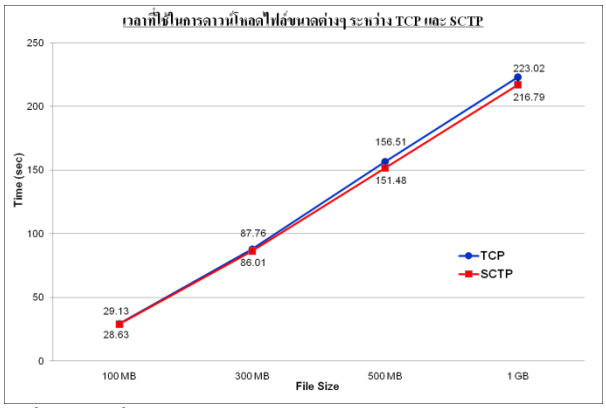
ในการดำเนินการทดลอง ได้พัฒนาโปรแกรมเพิ่มเติม และติดตั้งระบบเครือข่าย โดยแบ่งวิธีการดำเนินงานออกเป็น 3 ส่วนคือ

4. จำลองระบบเครือข่าย
5. ส่วนของ tracker server ติดตั้งโดยทำการติดตั้งโปรแกรม *Torrent Trader Lite 1.0.3 [9]* ซึ่งพัฒนาด้วยภาษา PHP และทำการปรับแต่งค่าตัวแปรของระบบ
6. พัฒนาปรับปรุงโปรแกรม BitTorrent Client ให้สามารถทำงานบน SCTP ได้ โดยทำการติดตั้งโปรแกรม cTorrent-dnh เวอร์ชั่น 1.3.4 พัฒนาด้วยภาษา C++ และทำการปรับแก้โค้ดของโปรแกรม

สภาพแวดล้อมในการทดลอง

7. หลังจากติดตั้งระบบทดลองเรียบร้อยแล้ว ทำการสร้างไฟล์ตัวอย่างจำนวน 4 ไฟล์ แต่ละไฟล์มีขนาด 100MB, 300MB, 500MB และ 1GB นำไฟล์ตัวอย่างทั้งหมดมาสร้างไฟล์ .torrent ตามขนาดไฟล์ตัวอย่างนั้นๆ แล้วทำการอัพโหลดขึ้น tracker server เพื่อใช้ในการดาวน์โหลดต่อไป
8. เริ่มทำการดาวน์โหลด โดยให้เครื่องคอมพิวเตอร์ 1 เครื่องทำหน้าที่เป็น seeder ของระบบ จากนั้นให้ leecher โหลดไฟล์ .torrent ทั้ง 4 ไฟล์เพื่อใช้เป็นข้อมูลในการดาวน์โหลดไฟล์ตัวอย่าง
9. ขั้นตอน leecher ดาวน์โหลด จะให้โหลดทีละขนาดไฟล์ แต่ละไฟล์ทำการดาวน์โหลดจำนวน 5 ครั้งเพื่อหาค่าเฉลี่ย โดยทำการสลับการดาวน์โหลดแต่ละไฟล์ระหว่างการใช้ TCP และ SCTP โหลด

ผลการทดลองเบื้องต้น



รูปที่ 3 ค่าเฉลี่ยเวลาในการดาวน์โหลดระหว่าง TCP และ SCTP

รูปที่ 3 แสดงเวลาที่ใช้ในการดาวน์โหลด จะเห็นว่าค่าเฉลี่ยเวลาที่ใช้ในการดาวน์โหลด BitTorrent โดยใช้ SCTP ใช้เวลาน้อยกว่าการทำงานร่วมกับ TCP ที่ทุกขนาดไฟล์ตัวอย่าง

5. สรุปและแนวทางการวิจัยในอนาคต

เนื่องจากผู้ใช้งานระบบเครือข่ายต้องการความเร็วในการดาวน์โหลดไฟล์ที่เพิ่มขึ้นและต้องการแก้ปัญหา HOL ผู้วิจัยจึงได้ศึกษาทดลองแนวทางการแก้ปัญหา โดยใช้ SCTP ร่วมกับ BitTorrent

โดยจากผลการทดลองในระบบเครือข่ายจำลองพบว่า BitTorrent บน SCTP สามารถลดเวลาในการดาวน์โหลดได้ ประมาณ 1 ถึง 3% เมื่อเทียบกับการใช้ BitTorrent ร่วมกับ TCP

แนวทางที่ผู้สนใจสามารถนำไปศึกษาวิจัยต่อได้ เช่นการนำตัวชี้วัดอื่นมาศึกษาเพิ่มเติม ซึ่งได้แก่ จำนวนครั้งการเชื่อมต่อ หรือเวลาที่ใช้ในการดาวน์โหลดข้อมูลแต่ละชิ้น เป็นต้น

6. เอกสารอ้างอิง

[1] Jon C. R. Bennett, Craig Partridge, Nicholas Shectman, “Packet reordering is not pathological network behavior”, IEEE/ACM TON, Dec 1999.

[2] Bram Cohen, “BitTorrent peer-to-peer file sharing protocol”, http://en.wikipedia.org/wiki/BitTorrent, July 2, 2001.

[3] Cheng-feng Tai, Lin-huang Chang, Ting-wei Hou, "Improvement of SCTP Perfomance during Handshake Process", AINAW 22nd, March 25, 2008.

[4] "CERT advisory CA-1996-21 TCP SYN flooding and IP spoofing attacks", http://www.cert.org/advisories/CA-1996-21.html, September 1996.

[5] M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control", RFC-2581, April 1999.

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

[7] 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.

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

[9] P. Natarajan, JR. Iyengar, PD. Amer, R. Stewart, “SCTP An Innovative Transport Layer Protocol for the Web”, Proceedings of the 15th international conference on World Wide Web, 2006.

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

[11]Henrik Osterdahl, “A comparison of TCP and SCTP performance using the HTTP protocol", http://www.it.kth.se/courses/2G1305/Sample-papers/Henrik_Oesterdahl-sctp-20050525.pdf, 2005.

[12]Torrent Trader Lite 1.0.3, http://sourceforge.net/ projects/torrenttrader, March 3, 2005.

ภาคผนวก จ

Source Code ส่วนที่ได้ทำการแก้ไข

ไฟล์ **ctcs.cpp**

```
int Ctcs::Connect()
{
  ssize_t r;
  m_last_timestamp = now;

  if(_s2sin(m_host,m_port,&m_sin) < 0) {
    fprintf(stderr,"warn, get CTCS ip address failed.");
    return -1;
  }

  m_sock = socket(AF_INET,SOCK_STREAM, IPPROTO_SCTP);
  if(INVALID_SOCKET == m_sock) return -1;

  if(setfd_nonblock(m_sock) < 0) {CLOSE_SOCKET(m_sock); return -1; }

  r = connect_nonb(m_sock,(struct sockaddr*)&m_sin);

  if( r == -1 ){ CLOSE_SOCKET(m_sock); return -1;}
  else if( r == -2 ) m_status = T_CONNECTING;
  else{
    m_status = T_READY;
    if( Send_Protocol() != 0 && errno != EINPROGRESS ){
      fprintf(stderr,"warn, send protocol to CTCS failed.
%s\n",strerror(errno));
      return -1;
    }
    if( Send_Auth() != 0 && errno != EINPROGRESS ) {
      fprintf(stderr,"warn, send password to CTCS failed.
%s\n",strerror(errno));
      return -1;
    }
    if( Send_Torrent(BTCONTENT.GetPeerId(), arg_metainfo_file) != 0
&&
        errno != EINPROGRESS ){
      fprintf(stderr,"warn, send torrent to CTCS failed.
%s\n",strerror(errno));
      return -1;
    }
  }
  return 0;
}
```

ไฟล์ **PeerList.cpp**

```
int PeerList::NewPeer(struct sockaddr_in addr, SOCKET sk)
{
  PEERNODE *p;
  btPeer *peer = (btPeer*) 0;
  int r;
      struct sctp_initmsg initmsg;

  if( m_peers_count >= cfg_max_peers ){
    if( INVALID_SOCKET != sk ) CLOSE_SOCKET(sk);
    return -4;
  }
```

```
  if( Self.IpEquiv(addr) ){
    if(INVALID_SOCKET != sk) CLOSE_SOCKET(sk); return -3;} // myself

  for(p = m_head; p; p = p->next){
    if(PEER_IS_FAILED(p->peer)) continue;
    if( p->peer->IpEquiv(addr)){  // already exist.
      if( INVALID_SOCKET != sk) CLOSE_SOCKET(sk);
      return -3;
    }
  }

  if( INVALID_SOCKET == sk ){
    if( INVALID_SOCKET == (sk = socket(AF_INET,SOCK_STREAM,
IPPROTO_SCTP)) ) return -1;

      memset(&initmsg, 0, sizeof(initmsg));
      initmsg.sinit_num_ostreams = 65535;
      initmsg.sinit_max_instreams = 65535;
      initmsg.sinit_max_attempts = 4;
      r = setsockopt(m_listen_sock, IPPROTO_SCTP, SCTP_INITMSG,
&initmsg, sizeof(initmsg));

    if( setfd_nonblock(sk) < 0) goto err;

    if(arg_verbose) fprintf(stderr, "Connecting to %s:%hu\n",
        inet_ntoa(addr.sin_addr), ntohs(addr.sin_port));
    if( -1 == (r = connect_nonb(sk,(struct sockaddr*)&addr)) ) return
-1;

    peer = new btPeer;

#ifndef WINDOWS
    if( !peer ) goto err;
#endif

    peer->SetConnect();
    peer->SetAddress(addr);
    peer->stream.SetSocket(sk);
    peer->SetStatus( (-2 == r) ? P_CONNECTING : P_HANDSHAKE );

  }else{
    if( setfd_nonblock(sk) < 0) goto err;

    peer = new btPeer;

#ifndef WINDOWS
    if( !peer ) goto err;
#endif

    peer->SetAddress(addr);
    peer->stream.SetSocket(sk);
    peer->SetStatus(P_HANDSHAKE);
  }

  if( P_HANDSHAKE == peer->GetStatus() )
    if( peer->Send_ShakeInfo() != 0 ) { delete peer; return -1; }

  p = new PEERNODE;
```

```
#ifndef WINDOWS
  if( !p ){ delete peer; return -1;}
#endif
  m_peers_count++;

  p->peer = peer;

  p->next = m_head;
  m_head = p;

  return 0;
 err:
  CLOSE_SOCKET(sk);
  return -1;
}


int PeerList::Initial_ListenPort()
{
  int r = 0;
  struct sockaddr_in lis_addr;
      struct sctp_initmsg initmsg;

  memset(&lis_addr,0, sizeof(sockaddr_in));
  lis_addr.sin_family = AF_INET;
  lis_addr.sin_addr.s_addr = INADDR_ANY;

  m_listen_sock = socket(AF_INET,SOCK_STREAM, IPPROTO_SCTP);
  if( INVALID_SOCKET == m_listen_sock ) return -1;

      memset(&initmsg, 0, sizeof(initmsg));
      initmsg.sinit_num_ostreams = 65535;
      initmsg.sinit_max_instreams = 65535;
      initmsg.sinit_max_attempts = 4;
      r = setsockopt(m_listen_sock, IPPROTO_SCTP, SCTP_INITMSG,
&initmsg, sizeof(initmsg));

  if ( cfg_listen_ip != 0 )
    lis_addr.sin_addr.s_addr = cfg_listen_ip;

  if(cfg_listen_port && cfg_listen_port != LISTEN_PORT_MAX){
    lis_addr.sin_port = htons(cfg_listen_port);
    if(bind(m_listen_sock,(struct sockaddr*)&lis_addr,sizeof(struct
sockaddr_in)) == 0)
      r = 1;
    else
      fprintf(stderr,"warn,couldn't bind on specified port %d: %s\n",
        cfg_listen_port,strerror(errno));
  }

  if( !r ){
    r = -1;
    cfg_listen_port = cfg_max_listen_port;
    for( ; r != 0; ){
      lis_addr.sin_port = htons(cfg_listen_port);
      r = bind(m_listen_sock,(struct
sockaddr*)&lis_addr,sizeof(struct sockaddr_in));
      if(r != 0){
```

```
        cfg_listen_port--;
        if(cfg_listen_port < cfg_min_listen_port){
          CLOSE_SOCKET(m_listen_sock);
          fprintf(stderr,"error,couldn't bind port from %d to %d:
%s\n",
            cfg_min_listen_port,cfg_max_listen_port,strerror(errno));
          return -1;
        }
      }
    } /* end for(; r != 0;) */
  }

  if(listen(m_listen_sock,5) == -1){
    CLOSE_SOCKET(m_listen_sock);
    fprintf(stderr,"error, couldn't listen on port %d: %s\n",
      cfg_listen_port,strerror(errno));
    return -1;
  }

  if( setfd_nonblock(m_listen_sock) < 0){
    CLOSE_SOCKET(m_listen_sock);
    fprintf(stderr,"error, couldn't set socket to nonblock mode.\n");
    return -1;
  }

  printf("Listening on %s:%d\n", inet_ntoa(lis_addr.sin_addr),
    ntohs(lis_addr.sin_port));

  return 0;
}
```

ไฟล์ **Tracker.cpp**

```
int btTracker::Connect()
{
  ssize_t r;
  time(&m_last_timestamp);
  struct sctp_initmsg initmsg;

  if(_s2sin(m_host,m_port,&m_sin) < 0) {
    fprintf(stderr,"warn, get tracker's ip address failed.");
    if(arg_ctcs) CTCS.Send_Info("warn, get tracker's ip address
failed.");
    return -1;
  }

  m_sock = socket(AF_INET,SOCK_STREAM, IPPROTO_TCP);
  if(INVALID_SOCKET == m_sock) return -1;

  memset(&initmsg, 0, sizeof(initmsg));
  initmsg.sinit_num_ostreams = 65535;
  initmsg.sinit_max_instreams = 65535;
  initmsg.sinit_max_attempts = 4;
  r = setsockopt(m_sock, IPPROTO_SCTP, SCTP_INITMSG, &initmsg,
sizeof(initmsg));

  // we only need to bind if we have specified an ip
  // we need it to bind here before the connect!!!!
```

```
  if ( cfg_listen_ip != 0 ) {
    struct sockaddr_in addr;
    // clear the struct as requested in the manpages
    memset(&addr,0, sizeof(sockaddr_in));
    // set the type
    addr.sin_family = AF_INET;
    // we want the system to choose port
    addr.sin_port = 0;
    // set the defined ip from the commandline
    addr.sin_addr.s_addr = cfg_listen_ip;
    // bind it or return...
    if(bind(m_sock,(struct sockaddr*)&addr,sizeof(struct
sockaddr_in)) != 0){
      fprintf(stderr, "warn, can't set up tracker connection: %s\n",
        strerror(errno));
      return -1;
    }
  }

  if(setfd_nonblock(m_sock) < 0) {CLOSE_SOCKET(m_sock); return -1; }

  r = connect_nonb(m_sock,(struct sockaddr*)&m_sin);

  if( r == -1 ){ CLOSE_SOCKET(m_sock); return -1;}
  else if( r == -2 ) m_status = T_CONNECTING;
  else{
    if( 0 == SendRequest()) m_status = T_READY;
    else{ CLOSE_SOCKET(m_sock); return -1;}
  }
  return 0;
}
```

BTContent.cpp

```
int btContent::SeedTimeout(const time_t *pnow)
{
  u_int64_t dl;
  if( pBF->IsFull() ){
    if( !m_seed_timestamp ){
      Tracker.Reset(1);
      Self.ResetDLTimer();
      Self.ResetULTimer();
      ReleaseHashTable();
      m_seed_timestamp = *pnow;
      FlushCache();

  extern struct timeval timeStart, timeStop;
  extern long sec, usec;

  gettimeofday(&timeStop, NULL);
  printf("\n--> Finished at %d.%d\n", timeStop.tv_sec,
timeStop.tv_usec);
  usec = timeStop.tv_usec - timeStart.tv_usec;
  sec  = timeStop.tv_sec - timeStart.tv_sec;
  if(usec < 0)  //borrow sec
  {
      usec = 1000000 - usec;
      sec--;
```

```
  }
  printf("--> amount = %d.%d\n", sec, usec);
      printf("\nDownload complete.\n");
      printf("Total time used: %lu minutes.\n",(*pnow -
m_start_timestamp) / 60);
      printf("Seed for other %lu hours", cfg_seed_hours);
      if(cfg_seed_ratio) printf(" or to ratio of %f",
cfg_seed_ratio);
      printf(".\n\n");

  FILE *fp = fopen("result.txt", "a+");
  char output[150];
  sprintf(output, "File = %s\nStart = %d.%d\nStop = %d.%d\namount =
%d.%d\n\n", arg_metainfo_file, timeStart.tv_sec, timeStart.tv_usec,
timeStop.tv_sec, timeStop.tv_usec, sec, usec);
  fputs(output, fp);
  fclose(fp);

  if (!arg_flg_force_seed_mode)
  {
      Tracker.SetStoped();
      WORLD.CloseAll();
      exit(0);
  }

    }
    dl = (Self.TotalDL() > 0) ? Self.TotalDL() :
GetTotalFilesLength();
    if( (cfg_seed_ratio == 0 && cfg_seed_hours == 0) ||
        (cfg_seed_hours > 0 &&
          (*pnow - m_seed_timestamp) >= (cfg_seed_hours * 60 * 60))
||
        (cfg_seed_ratio > 0 &&
          cfg_seed_ratio <= Self.TotalUL() / dl) ) return 1;
  }
  return 0;
}
```

BTStream.cpp

```
ssize_t btStream::Send_Piece(size_t idx,size_t off,char
*piece_buf,size_t len)
{
  size_t q = htonl(len + H_PIECE_LEN);
  unsigned char t = M_PIECE;
  ssize_t r;
      size_t x = idx;
  idx = htonl(idx);
  off = htonl(off);
  if( (r = out_buffer.Put(sock,(char*)&q,4, x)) < 0 ) return r;
  if( (r = out_buffer.Put(sock,(char*)&t,1, x)) < 0 ) return r;
  if( (r = out_buffer.Put(sock,(char*)&idx,4, x)) < 0) return r;
  if( (r = out_buffer.Put(sock,(char*)&off,4, x)) < 0) return r;
  return out_buffer.PutFlush(sock,piece_buf,len, x);
}
```

bufIO.cpp

```
ssize_t BufIo::Put(SOCKET sk, char *buf,size_t len, uint16_t idx)
{
  ssize_t r;
  if( _left_buffer_size < len ){ //no enough space
    r = FlushOut(sk, idx);
    if( r < 0 ) return r;
    for( ; _left_buffer_size < len; ) // still no enough space
      if(_realloc_buffer() < 0) return -3;
  }
  memcpy(b + p, buf, len);
  p += len;
  return 0;
}
```

btconfig.h

```
#ifndef BTCONFIG_H
#define BTCONFIG_H

extern size_t cfg_req_slice_size;

#define MAX_METAINFO_FILESIZ  4194304
// According to specs the max slice size is 128K.  But most clients
do not
// accept a value that large, so we limit to 64K.  Note that there is
a
// comparison in RequestQueue::IsValidRequest() (see btrequest.cpp)
that
// doubles the value so that we will accept a request for 128K.
#define cfg_max_slice_size 65536
extern size_t cfg_req_queue_length;
#define MAX_PF_LEN 8
#define PEER_ID_LEN 20
#define PEER_PFX "-CD0201-"

extern size_t cfg_cache_size;

extern size_t cfg_max_peers;
extern size_t cfg_min_peers;

extern unsigned long cfg_listen_ip;
extern int cfg_listen_port;
extern int cfg_max_listen_port;
extern int cfg_min_listen_port;

extern time_t cfg_seed_hours;
extern double cfg_seed_ratio;

extern int cfg_max_bandwidth;
extern int cfg_max_bandwidth_down;
extern int cfg_max_bandwidth_up;

// arguments global value
extern char *arg_metainfo_file;
extern char *arg_bitfield_file;
extern char *arg_save_as;
```

```
extern char *arg_user_agent;

extern unsigned char arg_flg_force_seed_mode;
extern unsigned char arg_flg_check_only;
extern unsigned char arg_flg_exam_only;
extern unsigned char arg_flg_make_torrent;
extern size_t arg_file_to_download;
extern unsigned char arg_verbose;

extern size_t arg_piece_length;
extern char *arg_announce;

extern char *arg_ctcs;
extern int cfg_exit_zero_peers;
#endif
```

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	นายธีรชัย เจนกิจพาณิชย์กุล
วัน เดือน ปี ที่เกิด	วันที่ 25 เมษายน 2522
สถานที่เกิด	จังหวัดกรุงเทพมหานคร
ประวัติการศึกษา	ระดับปริญญาตรี คณะวิทยาศาสตร์ วิชาเอกวิทยาการคอมพิวเตอร์ มหาวิทยาลัยราชภัฏพระนคร
ตำแหน่งหน้าที่การงานปัจจุบัน	โปรแกรมเมอร์
สถานที่ทำงานปัจจุบัน	บริษัท บุญรอดบริวเวอรี่ จำกัด
ผลงานดีเด่นและรางวัลทางวิชาการ	-
ทุนการศึกษาที่ได้รับ	-