ระบบแคชสำหรับโปรแกรมบิตทอเรนต์

BITTORRENT CACHE SYSTEM

นายปรัชญา ติงสะ

นายพชร ปัทมาลัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2548

ระบบแคชสำหรับโปรแกรมบิตทอเรนต์

BITTORRENT CACHE SYSTEM

โดย

นายปรัชญา ดิงสะ

นายพชร ปัทมาฉัย

อาจารย์ที่ปรึกษา

อ. ธนัญชัย ตรีภาค

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบแคชสำหรับโปรแกรมบิตทอเรนต์

BITTORRENT CACHE SYSTEM

ผู้จัดทำ

1. นายปรัชญา ติงสะ รหัสนักศึกษา 45010445

2. นายพชร ปัทมาลัย รหัสนักศึกษา 45010493

อาจารย์ที่ปรึกษา

(อาจารย์ธนัญชัย ตรีภาค)

ระบบแคชสำหรับโปรแกรมบิตทอเรนต์

นายปรัชญา ติงสะ	45010445
นายพชร ปัทมาลัย	45010493
อ.ธนัญชัย ตรีภาค	อาจารย์ที่ปรึกษา
ปีการศึกษา 2548	

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้เสนอวิธีสร้างระบบการทำงานที่จำกัดการใช้งานเครือข่ายสำหรับโปรแกรมเชื่อมต่อแบบบิตทอเรนต์ (Cache System for BitTorrent) โดยใช้การเปลี่ยนแปลงรูปแบบข้อมูลมูลฐานในการดำเนินงานแลกเปลี่ยนข้อมูล (.torrent file metadata) การกำหนดค่าลงในฐานข้อมูลเพื่อให้สอดคล้องกัน (Database) การแก้ไขวิธีการดำเนินงานของโปรแกรมบิตทอเรนต์ (BitTorrent Tweak) และการทำงานของพร็อกซี่เซิร์ฟเวอร์ (Proxy Server) โดยลักษณะเด่นของวิธีการที่นำเสนอในวิทยานิพนธ์นี้คือ การลดการเชื่อมต่อออกสู่ภายนอกเครือข่ายจากโปรแกรมบิตทอเรนต์ได้โดยส่งผลกระทบต่อการทำงานของโปรแกรมบิตทอเรนต์ให้น้อยที่สุด ความสามารถที่จะแลกเปลี่ยนข้อมูลกันภายในได้เมื่อเกิดปัญหากับตัวกลางการแลกเปลี่ยนข้อมูล (Tracker) และการแลกเปลี่ยนข้อมูลที่เคยดึงมาแล้วสามารถทำได้ง่ายขึ้น ในวิทยานิพนธ์เล่มนี้จะใช้อัตราการดึงข้อมูลเฉลี่ยและอัตราสูงสุดมาเปรียบเทียบกรณีการดึงข้อมูลต่างๆกัน ระหว่างที่ใช้ระบบ กับการทำงานตามปกติ ซึ่งในวิทยานิพนธ์เล่มนี้ได้แสดงผลของการทดสอบกรณีต่างๆ เพื่อเปรียบเทียบวิธีการที่นำเสนอกับวิธีการแบบเดิมเพื่อดูประสิทธิภาพและข้อได้เปรียบของแต่ละแบบ

BITTORRENT CACHE SYSTEM

Pratchaya Tingsa	45010445
Pochara Patthamalai	45010493
Thanunchai Threepak	Advisor
Academic Year 2005	

ABSTRACT

This thesis proposes a cache system for BitTorrent Protocol by integrating torrent metadata modification, Database insertion and modification, BitTorrent Client tweak, and Proxy Server concept. The outstanding feature of proposed method is to reduce outside accessing of BitTorrent with minimum influence to current method while giving extra abilities to allow file sharing to work within network when tracker has problem and easy to access files which were downloaded before. This thesis employs the average download rate and maximum download rate to compare various situations between certain standard and proposed method. This thesis presents the various testing results of proposed method comparing with current using method to determine efficiency and advantage.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้อย่างดีด้วยคำแนะนำและคำปรึกษาจากอาจารย์ธนัญชัย ตรีภาค ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์และอาจารย์ที่ปรึกษาของโครงงาน ข้าพเจ้ารู้สึกซาบซึ้งในความอนุเคราะห์จากท่านอาจารย์ และขอขอบพระคุณเป็นอย่างสูง

ขอกราบพระคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกๆ ท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า

ขอกราบพระคุณคณาจารย์ทุกท่านที่ได้ให้ความรู้ความเข้าใจในเรื่องต่างๆ แก่ข้าพเจ้านอกเหนือจากที่ใช้ในการทำวิทยานิพนธ์ฉบับนี้

ขอขอบคุณหอสมุดกลาง สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ได้สนับสนุนหนังสือต่างๆ ที่ใช้ในการทำวิจัย

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกคนที่ให้คำแนะนำต่างๆ และคอยให้กำลังใจเสมอมา

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ชุมนุมวิชาการ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่คอยดูแลและเป็นกำลังใจมาโดยตลอดเวลาสี่ปีที่ศึกษาในสถาบันแห่งนี้

ขอขอบคุณบัณฑิตศึกษาและบัณฑิตวิทยาลัย คณะวิศวกรรมศาสตร์ที่ให้ความช่วยเหลือในเรื่องต่างๆ

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจและให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

นายปรัชญา ติงสะ

นายพชร ปัทมาลัย

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

ในทุกวันนี้มีการใช้งานอินเตอร์เน็ตกันอย่างแพร่หลายในทุกๆด้านไม่ว่าจะเป็นการใช้งานเว็บไซต์, การรับส่ง e-mail, การเล่นเกมออนไลน์, การซื้อขายผ่านทางอินเตอร์เน็ต รวมไปถึงการแลกเปลี่ยนไฟล์ข้อมูล ซึ่งการแลกเปลี่ยนไฟล์ข้อมูลก็มีวิธีการหลายวิธี หนึ่งในนั้นก็คือการแลกเปลี่ยนไฟล์ข้อมูลแบบ P2P (Peer-to-Peer) และโปรโตคอลในการแลกเปลี่ยนไฟล์ข้อมูลแบบ P2P ที่ได้รับความนิยมอยู่ในขณะนี้ คือ โปรโตคอลบิตทอเรนต์

เมื่อระบบเครือข่ายหนึ่งมีการใช้งานโปรแกรมบิตทอเรนต์เป็นจำนวนมากนั้น จะทำให้เกิดการเชื่อมต่อระหว่างผู้ใช้งานโปรแกรมบิตทอเรนต์ภายในกับผู้ใช้งานโปรแกรมบิตทอเรนต์ภายนอกอย่างมากมายมหาศาล ทำให้การใช้งานอื่นๆในระบบอินเตอร์เน็ตได้รับผลกระทบ และบางครั้งผู้ใช้โปรแกรมบิตทอเรนต์ที่อยู่ในระบบเครือข่ายเดียวกันนั้น ก็ดาวน์โหลดไฟล์ข้อมูลไฟล์เดียวกันมา ซึ่งทำให้เกิดการเชื่อมต่อออกไปภายนอกเครือข่ายอย่างไม่เกิดประโยชน์ ยกตัวอย่างเช่น ระบบเครือข่ายของผู้ให้บริการอินเตอร์เน็ต (ISP)

แต่ถ้าในระบบเครือข่ายนั้นมีระบบแคชสำหรับโปรแกรมบิตทอเรนต์ จะทำให้เวลาผู้ใช้โปรแกรมบิตทอเรนต์ดาวน์โหลดไฟล์ข้อมูลที่มีผู้อื่นเคยดาวน์โหลดมาแล้ว ระบบแคชก็จะไปนำไฟล์ข้อมูลในแคชส่งไปให้ผู้ต้องการ และจะไม่ทำการดาวน์โหลดไฟล์ข้อมูลจากอินเตอร์เน็ต ซึ่งจะสามารถลดการเชื่อมต่อออกไปภายนอกเครือข่ายได้ และเวลาที่ใช้ในการดาวน์โหลดไฟล์ข้อมูลที่ซ้ำกันก็จะใช้เวลาน้อยลง

1.2 วัตถุประสงค์ของโครงงาน

ปริญญานิพนธ์ฉบับนี้มุ่งหวังเพื่อศึกษาและออกแบบระบบแคชสำหรับโปรแกรมบิดทอเรนต์ เนื่องด้วยปัจจุบันนี้การเชื่อมต่อเพื่อแลกเปลี่ยนข้อมูลแบบ Peer-to-peer ด้วยโปรโตคอลของบิตทอเรนต์นั้นเป็นที่นิยมใช้กันอย่างแพร่หลายในแทบจะทุกองค์กร ทำให้การเชื่อมต่อระหว่างภายในองค์กรไปยังภายนอกองค์กรนั้นเต็มไปด้วยการใช้งานโปรแกรมบิตทอเรนต์ จึงทำให้ไม่สามารถใช้งานอินเตอร์เน็ตในงานด้านอื่นได้อย่างมีประสิทธิภาพเท่าที่ควร ซึ่งถ้ามีการออกแบบระบบแคชสำหรับโปรแกรมบิตทอเรนต์มาใช้งานได้นั้น จะเป็นหนึ่งวิธีในการลดการเชื่อมต่อจากภายในออกไปยังเครือข่ายภายนอก และทำให้เวลาดาวน์โหลดไฟล์ที่เหมือนกันโดยใช้โปรแกรมบิตทอเรนต์ทำได้เร็วขึ้น

1.3 ขอบเขตของโครงงาน

ในปริญญานิพนธ์ฉบับนี้ได้นำเสนอวิธีการออกแบบระบบแคชสำหรับโปรแกรมบิตทอเรนต์ โดยกล่าวถึงทฤษฎีที่ใช้ในการออกแบบระบบ แบบจำลองและขั้นตอนการทำงานของระบบ และผลการทดสอบระบบแคชสำหรับโปรแกรมบิตทอเรนต์

สำหรับโครงงานระบบแคชสำหรับโปรแกรมบิตทอเรนต์นี้ ได้กำหนดขอบเขตของการศึกษาและออกแบบระบบเอาไว้ดังนี้

1. ระบบแคชสำหรับโปรแกรมบิตทอเรนต์นี้ถูกออกแบบมาสำหรับเครือข่ายที่มีการกำหนดหมายเลขไอพีจริงให้กับอุปกรณ์ในระบบเอาไว้อย่างตายตัว
2. ระบบแคชสำหรับโปรแกรมบิตทอเรนต์นี้สามารถทำงานได้สำหรับการดาวน์โหลดไฟล์ .torrent ที่ปรากฏอยู่บน URL เท่านั้น
3. ระบบแคชสำหรับโปรแกรมบิดทอเรนต์นี้ไม่ได้รวมกระบวนและวิธีการเคลียร์ข้อมูลที่มีอยู่ออกจากระบบแคช

1.4 วิธีการดำเนินการ

1. ศึกษาทฤษฎีและการทำงานของโปรโตคอลบิดทอเรนต์ การทำงานของแทรคเกอร์ การทำงานของโปรแกรมบิตทอเรนต์ ซึ่งนำมาใช้ในการออกแบบระบบและเขียนเว็บเซอร์วิสของระบบ
2. ศึกษาทฤษฎีของพร็อกซี่เซิร์ฟเวอร์ Transparent Proxy และโปรแกรมที่เกี่ยวข้องกับการทำ Redirect Proxy ได้แก่ โปรแกรม Squid และโปรแกรม SquidGuard
3. ศึกษาหลักการทำงาน การติดตั้ง และการปรับแต่งค่าของเว็บเซิร์ฟเวอร์
4. ศึกษาการทำงาน คำสั่งต่างๆ การกำหนด Permission และ Owner ให้กับไฟล์ ระบบไฟล์ และไดเรกทอรี่ของระบบปฏิบัติการ Linux หรือ UNIX ที่นำมาใช้งาน
5. เตรียมอุปกรณ์ที่จำเป็นต้องใช้ในการติดตั้งระบบแคชสำหรับโปรแกรมบิตทอเรนต์
6. วิเคราะห์และออกแบบระบบ
7. ทำการติดตั้งและปรับแต่งค่าสำหรับองค์ประกอบต่างๆของระบบ ได้แก่ การติดตั้ง Squid และ SquidGuard สำหรับ Redirect Proxy การติดตั้งแทรคเกอร์และโปรแกรมบิตทอเรนต์ไคลเอนต์
8. พัฒนาเว็บเซอร์วิสของระบบแคชสำหรับโปรแกรมบิดทอเรนต์
9. ทำการทดสอบและบันทึกผลของระบบแคชสำหรับโปรแกรมบิตทอเรนต์
10. สรุปและวิจารณ์ผลของการทำโครงงาน รวมไปถึงเสนอแนะแนวทางการพัฒนาต่อไป

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รับความรู้และความเข้าใจเกี่ยวกับวิธีการทำงานของโปรโตคอลบิตทอเรนต์ การทำงานของแทรคเกอร์และการทำงานของโปรแกรมบิตทอเรนต์ไคลเอนต์
2. ได้รับความรู้ ความเข้าใจทั้งในด้านทฤษฎีและด้านปฏิบัติเกี่ยวกับการศึกษาและออกแบบในการทำระบบแคชสำหรับโปรแกรมบิตทอเรนต์
3. สามารถออกแบบระบบแคชสำหรับโปรแกรมบิตทอเรนต์ให้ทำงานได้

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 5 บทด้วยกันคือ

บทที่ 1 กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของปริญญานิพนธ์

บทที่ 2 กล่าวถึงทฤษฎีที่เกี่ยวข้องที่ใช้ในโครงงาน ซึ่งประกอบด้วยทฤษฎีที่เกี่ยวกับบิตทอเรนต์ ทฤษฎีของพร็อกซี่เซิร์ฟเวอร์ โปรแกรม Squid โปรแกรม SquidGuard และทฤษฎีของเว็บเซอร์วิส

บทที่ 3 กล่าวถึงการออกแบบและแบบจำลองระบบแคชสำหรับโปรแกรมบิตทอเรนต์ โดยอธิบายหน้าที่และการทำงานขององค์ประกอบต่างๆที่ใช้ในระบบแคชสำหรับโปรแกรมบิตทอเรนต์

บทที่ 4 กล่าวถึงเรื่องของการทดลองระบบแคชสำหรับโปรแกรมบิตทอเรนต์ เกณฑ์ที่ใช้ในการประเมิน และผลจากการทดลองระบบ

บทที่ 5 เป็นบทวิจารณ์และสรุป ซึ่งกล่าวถึงบทสรุปของโครงงาน วิจารณ์สิ่งที่ได้รับจากโครงงาน รวมไปถึงข้อเสนอแนะสำหรับเป็นแนวทางในการพัฒนาระบบต่อ

บทที่ 2
ทฤษฎีพื้นฐานที่ใช้ในโครงงาน

ก่อนที่จะทำการออกแบบหรือพัฒนาระบบใดๆ ขึ้นมานั้น การศึกษาหาข้อมูลของทฤษฎีที่เกี่ยวข้องนั้นเป็นสิ่งที่จำเป็นต้องกระทำอย่างหลีกเลี่ยงไม่ได้ เพื่อให้ระบบที่ต้องการออกแบบหรือพัฒนานั้นประสบผลสำเร็จสูงสุดและเกิดความผิดพลาดน้อยที่สุด

ในบทนี้จะกล่าวถึงทฤษฎีต่างๆที่เกี่ยวข้องกับการทำระบบแคชสำหรับโปรแกรมบิตทอเรนต์ ได้แก่ โปรโตคอลบิตทอเรนต์ องค์ประกอบและการทำงานของบิตทอเรนต์ พร็อกซี่เซิร์ฟเวอร์ โปรแกรม Squid โปรแกรม SquidGuard และเว็บเซอร์วิส ซึ่งทฤษฎีที่เกี่ยวข้องนี้จะนำไปใช้ในการออกแบบระบบและพัฒนาระบบในบทต่อไป

2.1 บิตทอเรนต์

2.1.1 บิตทอเรนต์คืออะไร

BitTorrent เป็นชื่อของโปรโตคอลและอุปกรณ์การแจกจ่ายข้อมูลแบบ Peer-to-peer ที่ถูกพัฒนาโดยผู้พัฒนา Bram Cohen โดยเริ่มเป็นที่รู้จักครั้งแรกในงาน CodeCon ปี ค.ศ. 2002 ในครั้งแรกการทำงานเขียนโดยภาษาไพตอนและจดสิทธิบัตรในแบบโอเพ่นซอร์สภายใต้เวอร์ชัน 4.0 คำว่า BitTorrent สามารถใช้อ้างถึงตัวโปรโตคอลในการแจกจ่ายไฟล์ ตัวโปรแกรมประยุกต์ส่วนลูกข่าย และไฟล์นามสกุล .torrent ได้

โปรโตคอล BitTorrent จะทำการแบ่งไฟล์ออกเป็นหลายๆส่วนย่อยๆ โดยประมาณคือหนึ่งในสี่ของเมกะไบต์ แต่ละส่วนย่อยจะถูกแจกจ่ายไปยังเครื่อง peer โดยสุ่มลำดับ ซึ่งจะไปประกอบกลับเป็นไฟล์หนึ่งๆในเครื่องที่ขอไฟล์ไป แต่ละ peer จะใช้ประโยชน์จากการเชื่อมต่อที่ดีที่สุดไปยังส่วนที่ยังขาดไป ในขณะที่ส่งส่วนที่มีอยู่ให้เครื่องอื่นๆได้รับ รูปแบบการทำงานอย่างนี้มีประโยชน์อย่างมากในการแลกเปลี่ยนไฟล์ขนาดใหญ่ อย่างเช่นวิดีโอหรือโค้ดต้นแบบของโปรแกรม ในการดาวน์โหลดโดยทั่วไปแล้ว ไฟล์ที่มีความต้องการสูงมักก่อให้เกิดสภาวะคอขวด ซึ่งผู้ที่ต้องการไฟล์จะใช้แบนวิดท์ของเครื่องให้บริการจนหมด แต่ใน BitTorrent นั้นไฟล์ที่เป็นที่ต้องการอย่างมากจะถูกช่วยในการส่งไฟล์เมื่อแบนวิดท์และจำนวนเครื่องปล่อยไฟล์ (seed) ที่สมบูรณ์มีมากขึ้น

การทำงานของ BitTorrent เครื่องลูกข่ายจำเป็นต้องมีสิ่งหลักๆพื้นฐานก็คือโปรแกรมประยุกต์ที่รองรับโปรโตคอล BitTorrent ไฟล์นามสกุล .torrent และการเชื่อมต่อกับอินเตอร์เน็ต

2.1.2 รายละเอียดของไฟล์ .torrent

ไฟล์นามสกุล .torrent จะมีข้อมูลที่เกี่ยวข้องกับไฟล์จริงๆที่จะทำการดาวน์โหลด เรียกข้อมูลนี้ว่า Metainfo ซึ่งถูกจัดรูปแบบในลักษณะ Bencode (อ่านว่า บี-เอนโค้ด) โดยมีรายละเอียดของรูปแบบดังนี้

- ข้อมูลที่เป็น String จะอยู่ในรูป <ความยาวของstringในเลขฐานสิบ>:<ข้อมูลstring> เช่น 4:spam
- ข้อมูลที่เป็นจำนวนเต็มจะอยู่ในรูป i<จำนวนเต็มในAsciiฐานสิบ>e โดยสามารถใส่เครื่องหมายบอกจำนวนลบไว้ก่อนตัวเลขได้ จำนวนเต็มศูนย์ให้ใส่เพียงเลขศูนย์ เช่น i7e i-3e i0e
- ชุดข้อมูล (list) จะอยู่ในรูป l<ข้อมูลในรูปแบบbencode>e ซึ่งในชุดข้อมูลสามารถประกอบไปด้วยข้อมูลในรูป bencode ที่เป็น string จำนวนเต็ม พจนานุกรม และ/หรือชุดข้อมูลอื่นๆก็ได้
- พจนานุกรม จะอยู่ในรูป d<bencoded string><bencoded element>e โดยตัวตั้งต้นจะต้องเป็น string ส่วนค่านั้นจะเป็นอะไรก็ได้ที่อยู่ในรูปแบบ bencode และหากมีมากกว่าหนึ่งตัวตั้งต้นจะต้องทำการจัดลำดับตัวตั้งต้นตามลำดับ alphabet ของ string เช่น d4:bone7:calcium3:cow3:mooe ซึ่งก็คือ bone=calcium และ cow=moo

โครงสร้างของ Metainfo ทุกอย่างจะอยู่ในรูปของ bencode โดย metainfo ในไฟล์นามสกุล .torrent เป็น bencoded dictionary ที่มีข้อมูลกล่าวไว้ด้านล่าง ทุกตัวอักษร string อยู่ในรูป UTF-8 โดยส่วนใดไม่ได้ระบุว่าเป็นตัวเลือกคือข้อมูลที่จำเป็น

- info: เป็น dictionary ที่บรรยายลักษณะของไฟล์หนึ่งๆหรือหลายๆไฟล์ใน torrent ซึ่งสามารถมีได้สองลักษณะ คือแบบไฟล์เดี่ยวที่ไม่ต้องมีโครงสร้าง Directory กับแบบไฟล์ torrent ที่บอกถึงหลายไฟล์

สำหรับไฟล์เดี่ยว รูปแบบพจนานุกรม info จะมีโครงสร้างดังรูปที่ 2.1 นี้

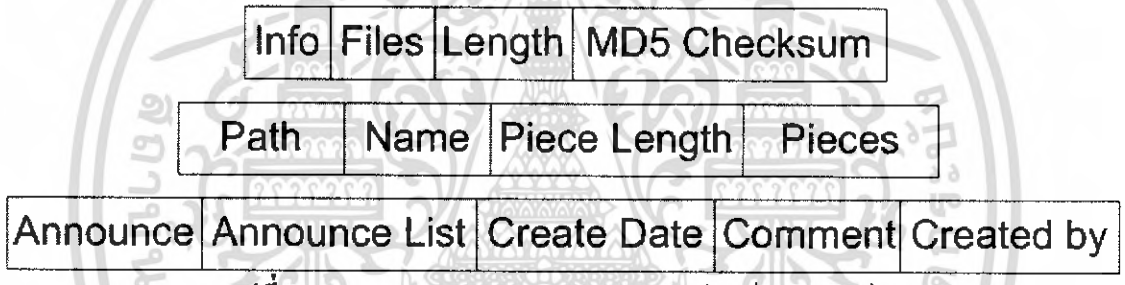
Info	Length	MD5 Checksum	Name	Piece	Pieces

Announce	Announce List	Create Date	Comment	Created by

รูปที่ 2.1 รูปแบบพจนานุกรม info สำหรับไฟล์เดี่ยว

- length: ความยาวของไฟล์ (จำนวนเต็ม)
- md5sum: (ตัวเลือก) string ความยาว 32 ตัวอักษรในรูปเลขฐาน 16 ที่ตรงกับ MD5 Sum ของไฟล์ ข้อมูลนี้แทบไม่ถูกใช้โดยโปรแกรม BitTorrent เลย แต่ถูกรวมไว้ในบางโปรแกรมเพื่อเพิ่มความยืดหยุ่นในการใช้งาน
- name: ชื่อของไฟล์ (string)
- piece length: ความยาวของแต่ละชิ้นส่วนของไฟล์ในหน่วยไบต์ ตัวเลขจะเป็นกำลังของ 2 ขึ้นไป โดยปกติจะตั้งไว้ให้น้อยกว่าหรือเท่ากับ 512 KB (จำนวนเต็ม)
- pieces: string ที่มีค่า SHA-1 hash ของแต่ละชิ้น โดยค่าเป็น 20 bytes ต่อหนึ่งชิ้นทั้งหมดต่อกันเป็น string เดี่ยวที่มีความยาวเป็นผลคูณของ 20

สำหรับไฟล์หลายไฟล์ รูปแบบพจนานุกรม info จะมีโครงสร้างดังรูปที่ 2.2 นี้



รูปที่ 2.2 รูปแบบพจนานุกรม info สำหรับไฟล์หลายไฟล์

- files: ชุดข้อมูลของพจนานุกรม แต่ละอันสำหรับแต่ละไฟล์ แต่ละพจนานุกรมจะมีชุดข้อมูลดังนี้
  - length: ความยาวของไฟล์ (จำนวนเต็ม)
  - md5sum: (ตัวเลือก) string ความยาว 32 ตัวอักษรในรูปเลขฐาน 16 ที่ตรงกับ MD5 Sum ของไฟล์ ข้อมูลนี้แทบไม่ถูกใช้โดยโปรแกรม BitTorrent เลย แต่ถูกรวมไว้ในบางโปรแกรมเพื่อเพิ่มความยืดหยุ่นในการใช้งาน
  - path: ชุดข้อมูลที่บันทึกหนึ่ง string หรือมากกว่าเพื่อบอก path และชื่อไฟล์ แต่ละค่าจะบอกถึงชื่อของ directory หรือ (ค่าสุดท้ายในชุดข้อมูล) ชื่อไฟล์
  - name: ชื่อของ directory บนสุดของโครงสร้าง—Directory ที่มีไฟล์ทุกไฟล์ตามที่กล่าวไว้ในชุดข้อมูล files

- piece length: ความยาวของแต่ละชิ้นส่วนของไฟล์ในหน่วยไบต์ ตัวเลขจะเป็นกำลังของ 2 ขึ้นไป โดยปกติจะตั้งไว้ให้น้อยกว่าหรือเท่ากับ 512 KB (จำนวนเต็ม)
- pieces: string ที่มีค่า SHA-1 hash ของแต่ละชิ้น โดยค่าเป็น 20 bytes ต่อหนึ่งชิ้น ทั้งหมดต่อกันเป็น string เดี่ยวที่มีความยาวเป็นผลคูณของ 20

หลังจาก info จะเป็นรูปแบบที่ทั้งไฟล์เดี่ยวและหลายไฟล์ใช้เหมือนกัน

- announce: URL ของ tracker (string)
- announce-list: (ตัวเลือก) ส่วนขยายของข้อมูลปกติเพื่อบอก tracker สำรองในกรณีที่ tracker หลักล่มหรือติดต่อไม่ได้
- create date: (ตัวเลือก) เวลาที่สร้างไฟล์นามสกุล .torrent นี้ขึ้น ใช้รูปแบบ Unix epoch โดยนับเวลาเป็นวินาทีจาก 1 Jan 1970 ณ เวลา 00:00:00 UTC (จำนวนเต็ม)
- comment: (ตัวเลือก) ข้อความของผู้สร้าง (string)
- created by: (ตัวเลือก) ชื่อและรุ่นของโปรแกรมที่ใช้สร้างไฟล์ .torrent นี้ (string)

2.1.3 การทำงานของแทรคเกอร์

ในการเชื่อมต่อระหว่างแต่ละเครื่อง Peer นั้น จะมี BitTorrent Tracker เป็นตัวช่วยในการแลกเปลี่ยนข้อมูลสถานะของแต่ละ peer เพื่อให้ทราบว่า มีเครื่องใดบ้างที่ต้องการหรือมีไฟล์ดังกล่าวที่ตรงกับไฟล์นามสกุล .torrent บอกไว้ แทรคเกอร์มีหลายแบบ ที่พัฒนาแรกสุดก็คือ HTTP/HTTPS protocol Tracker พัฒนาในภาษาไพตอน ซึ่งตอบสนองต่อคำสั่ง HTTP GET คำสั่งที่ส่งมาจะมี ค่า metrics จากเครื่องลูกข่ายที่ช่วยแทรคเกอร์ในด้านสถิติของ torrent นั้นๆ การตอบสนองจะมี peer list ที่ช่วยให้เครื่องลูกข่ายทราบถึงเครื่องอื่นๆที่เข้าร่วมงาน torrent นี้ ปกติแล้วคำสั่งจะใช้ URL จาก announce ใน metainfo ตามด้วย Parameter โดยใช้วิธีปกติของ CGI (ใช้ ? หลัง announce URL ตามด้วยชุด parameter=value ซึ่งขั้นจากกันด้วย &) ซึ่งค่าทั้งหมดนั้น ถ้าไม่ได้อยู่ในรูป 0-9 a-z A-Z และ $-_.+!*'() จะต้องอยู่ในรูป %nn โดยที่ nn เป็นค่าเลขฐานสิบหกที่ตรงกับไบต์ตัวอักษรนั้นๆ

Parameter ที่ใช้ใน คำสั่ง GET จากเครื่องลูกข่ายไปยัง Tracker จะมีข้อมูลดังรูปที่ 2.3 นี้

Info_Hash	Peer_ID	Port	Uploaded	Downloaded

Left	Compact	Event	IP	Numwant	Key	TrackerID

รูปที่ 2.3 พารามิเตอร์ที่ใช้ส่งจากเครื่อง Client ไปยัง Tracker

- info_hash: 20 ไบต์ SHA-1 Hash ของค่าของ info ตามในพจนานุกรมใน metainfo แต่ string ที่ใช้นี้จะแปลงเป็น urlencoded string
- peer_id: 20 ไบต์ string ใช้เป็นหมายเลขเฉพาะของเครื่องลูกข่าย สร้างโดยโปรแกรมลูกข่ายตอนที่เริ่มโปรแกรม สามารถเป็นค่าใดๆก็ได้
- port: หมายเลขพอร์ตที่เครื่องลูกข่ายรับฟังอยู่พอร์ตที่จัดไว้สำหรับ BitTorrent โดยเฉพาะโดยปกติคือ 6881 ถึง 6889 เครื่องลูกข่ายอาจต้องยกเลิกหากไม่สามารถใช้พอร์ตในช่วงนี้ได้
- uploaded: จำนวนข้อมูลที่ได้ส่งออกไปให้เครื่องอื่นในรูปแบบ ASCII ฐานสิบ
- downloaded: จำนวนข้อมูลที่ได้โหลดเข้ามาในรูปแบบ ASCII ฐานสิบ
- left: จำนวนข้อมูลคงเหลือที่เครื่องลูกข่ายจำเป็นต้องโหลดในรูปแบบ ASCII ฐานสิบ
- compact: บ่งบอกว่าเครื่องลูกข่ายรับข้อมูลตอบรับที่กระชับ Peers list จะถูกแทนด้วย Peers string ข้อมูลที่ใช้คือ 6 ไบต์ต่อหนึ่ง peer โดย 4 ไบต์แรกเป็น host และอีกสองไบต์เป็นพอร์ต
- event: ถ้ามี จะต้องเป็นหนึ่งใน started, stopped หรือ completed ถ้าไม่ระบุ จะดำเนินการกระทำตามปกติ
    - started: คำขอครั้งแรกที่ติดต่อกับแทรคเกอร์จะต้องมี event ที่มีค่า started เสมอ
    - stopped: จะต้องส่งไปให้แทรคเกอร์ หากโปรแกรมลูกข่ายปิดลงด้วยดี
    - completed: จะต้องส่งไปให้แทรคเกอร์ เมื่อการดาวน์โหลดเสร็จสิ้นโดยสมบูรณ์ แต่ไม่ต้องส่งหากในตอนที่เริ่มโปรแกรมลูกข่ายนั้นงานเสร็จสิ้นอยู่แล้ว
- ip: ตัวเลือก เป็นค่าไอพีแอดเดรสจริงของเครื่องลูกข่ายในรูปแบบ dotted quad หรือ hexed IPv6 ปกติไม่จำเป็นเพราะสามารถดูไอพีที่ request มาได้ เว้นแต่เป็นการติดต่อภายใต้พร็อกซี่ บางแทรคเกอร์ยังถือเป็นการบอกลักษณะการติดต่อแบบ IPv6 ด้วย
- numwant: ตัวเลือก ระบุจำนวนของ peer ที่เครื่องลูกข่ายต้องการรับจากแทรคเกอร์ ซึ่งสามารถเป็นค่าศูนย์ได้ หากไม่ระบุ ปกติจะรับ 50 peers

- key: ตัวเลือก ข้อมูลระบุตัวนี้จะไม่ใช้ร่วมกับผู้ใช้อื่น ถูกกำหนดมาเพื่อระบุตัวตนของเครื่องลูกข่ายหากมีการเปลี่ยนแปลงไอพีแอดเดรส
- trackerid: ตัวเลือก หากข้อมูลใน announce เคยระบุ tracker id ก็ควรจะกำหนดไว้ที่นี่

แทรคเกอร์จะตอบสนองด้วยเอกสาร "text/plain" ซึ่งมีพจนานุกรม bencode ที่มีค่าตั้งต้นดังรูป 2.4 ต่อไปนี้

Failure Reason	Warning Message	Interval	Min Interval

Tracker ID	Complete	Incomplete	Peers

รูปที่ 2.4 รูปแบบการตอบสนองของ Tracker

- failure reason: ถ้ามี จะอยู่ในรูปข้อความที่คนอ่านได้ (string)
- warning message: คล้ายกับ failure reason แต่กระบวนการจะดำเนินไปตามปกติ ข้อความจะถูกแสดงขึ้นเหมือน failure reason
- interval: ช่วงระยะห่างเป็นวินาทีที่เครื่องลูกข่ายควรรอก่อนจะส่งคำขอครั้งต่อไป
- min interval: ช่วงระยะอย่างต่ำในการประกาศ ถ้าระบุเครื่องลูกข่ายจะต้องไปประกาศในช่วงที่ถี่กว่านี้
- tracker id: string ที่เครื่องลูกข่ายควรส่งมาในการประกาศครั้งต่อไป หากไม่มีและในการประกาศรอบก่อนได้ส่ง tracker id มา ไม่ต้องทิ้งค่าเก่า ให้ใช้ต่อไป
- complete: จำนวนของ peer ที่มีไฟล์โดยสมบูรณ์ หรือ seeder (จำนวนเต็ม)
- incomplete: จำนวนของ peer ที่มีไฟล์ไม่สมบูรณ์ หรือ lecher (จำนวนเต็ม)
- peers: มีค่าเป็นชุดข้อมูลของพจนานุกรม แต่ละชุดจะมีค่าตั้งต้นดังนี้
    - peer id: ค่าที่ตั้งเองของ peer ตามที่กล่าวไว้เบื้องต้น
    - ip: IP Address ของ peer (IPv4 หรือ IPv4) หรือ DNS name
    - port: Port number ของ Peer

ในปัจจุบันนอกจาก HTTP Tracker แล้ว ยังมีแบบที่เขียนขึ้นด้วย PHP และ C++ ซึ่งแตกต่างกันไปตามวิธีการทำงานของโค้ดและการพัฒนา เช่น XBNBT BNBT CBTT ที่เป็นตัวหลักๆในปัจจุบัน ส่วนแทรคเกอร์ที่เป็นโค้ดเสรี (Open Source) อื่นๆนั้น มาจากการการพัฒนาต่อจากแทรคเกอร์เหล่านี้

นอกจากนี้ ในโปรแกรมประยุกต์บางตัวยังรองรับ BitTorrent แบบ Trackerless ซึ่งใช้วิธีติดต่อกับ Peer อื่นๆที่ใช้โปรแกรมที่รองรับวิธีนี้เหมือนกัน โดยใช้การติดต่อค้นหา Distributed Hash Table ที่ตรงกันในขณะที่ทำงาน ซึ่ง peer เครื่องอื่นที่กำลังดำเนิน Torrent เดียวกันจะตอบรับ Query และสร้าง Peers Table ระหว่างกันโดยไม่ต้องมีแทรคเกอร์มาช่วย

2.1.4 การทำงานในรูปแบบ Peer-2-Peer ของเครื่องลูกข่าย

หลังจากมี Peers Table แล้ว โปรแกรมลูกข่าย(Client) จะติดต่อระหว่างกันโดยใช้ Peer Wire Protocol (TCP) เพื่อแลกเปลี่ยนชิ้นส่วนของข้อมูลที่ Torrent ที่ทำงานอยู่นั้นกล่าวถึง ซึ่งในจำนวน peer ที่มีทั้งหมดใน peer table นั้น โปรแกรมลูกข่ายจะไม่ได้ติดต่อไปทั้งหมดเพื่อรักษา Load Balance โดยมีการระบุสถานะกับแต่ละ peer ไว้ ซึ่งมี

- choked: บ่งบอกว่าเครื่องลูกข่ายนั้นถูกปิดกั้นโดย peer หรือไม่ ซึ่งในขณะที่ถูกปิดกั้น คำร้องขอต่างๆจะไม่ได้รับการตอบรับจนกว่าจะยุติการปิดกั้น เครื่องลูกข่ายควรระงับการร้องขอชิ้นส่วน และรู้ไว้ว่าคำร้องที่ไม่ได้รับการตอบจะถูกละเลยจาก peer
- interested: บ่งบอกว่า Peer สนใจที่จะขอข้อมูลจากเครื่องลูกข่าย ซึ่งทันทีที่ได้ยุติการปิดกั้น peer จะเริ่มส่งคำขอไปยังเครื่องลูกข่ายทันที

ซึ่งในความเป็นจริง ทั้ง Peer และเครื่องลูกข่ายนั้นนับเป็น peer เช่นกัน ดังนั้นข้อมูลสถานะจริงๆจะเป็นดังนี้

- am_choking: เครื่องลูกข่ายนี้กำลังปิดกั้น peer
- am_interested: เครื่องลูกข่ายนี้สนใจ peer
- peer_choking: เครื่อง peer กำลังปิดกั้นเครื่องลูกข่ายนี้
- peer_interested: เครื่อง peer สนใจเครื่องลูกข่าย

สถานะในการเริ่มการเชื่อมต่อจะเป็นการปิดกั้นและไม่สนใจจากทั้งสองฝ่าย ชิ้นส่วนจะถูกส่งมาสู่เครื่องลูกข่ายก็ต่อเมื่อ peer ไม่ได้ปิดกั้นเครื่องลูกข่าย และเครื่องลูกข่ายสนใจในตัว peer สิ่งสำคัญที่เครื่องลูกข่ายควรทำก็คือการบอก peer ทุก peer ถึงความสนใจที่มีต่อแต่ละ peer ซึ่งควรบอกไว้ทุกครั้งที่มีการเปลี่ยนแปลงแม้ว่า peer จะปิดกั้นอยู่ เพื่อให้ peer สามารถรับรู้ได้ว่าเครื่องลูกข่ายจะโหลดข้อมูลจาก peer ทันทีที่ยุติการปิดกั้น

ตารางที่ 2.1 สภาวะการติดต่อกันของ Client กับ Peer

Peer to Client	**Client to Peer**			
Status	**Interested**	**Not Interested**	**Choking**	**Unchoking**
Interested	ต่างก็มีชิ้นที่อีกฝั่งยังขาด	มีชิ้นที่ peer ยังขาด	พิจารณาการ unchoke Peer	ส่งชิ้นส่วนให้ Peer
Not Interested	Peer มีชิ้นที่ยังขาด	หลังเวลาหนึ่งจะเลิกติดต่อ	ไม่จำเป็นต้อง unchoke	x
Choking	รอส่ง request	ไม่มีชิ้นส่วน	สถานะเริ่มต้น	x
Unchoking	รับชิ้นส่วนจาก Peer	x	x	x

สำหรับเลขจำนวนเต็มที่ส่งใน Peer Wire Protocol ถ้าไม่ได้ระบุไว้ จะอยู่ในรูป 4 byte big endian ซึ่งจะรวมถึงตัวระบุความยาวทุกตัวหลังจากการทำ handshake ใน Peer Wire Protocol จะมีการทำ handshake เพื่อเริ่มต้น หลังจากนั้น จะติดต่อกันผ่านข้อความที่ระบุความยาวไว้ด้านหน้าตามรูปแบบที่กล่าวไป

การ Handshake เป็นข้อความที่จำเป็นต้องส่งเป็นข้อความแรก มีลักษณะดังรูปที่ 2.5 ข้างล่างนี้

รูปที่ 2.5 รูปแบบของ Handshake

- handshake: <pstrlen><pstr><reserved><info_hash><peer_id>
  - pstrlen: ความยาวของ pstr ในรูป single raw byte
  - pstr: String Identifier of the Protocol
  - reserved: 8 bytes ที่สงวนไว้ ในการใช้งานปัจจุบันให้ทุกไบต์เป็น 0

- info_hash: 20 bytes SHA1 hash ของส่วน info ใน metainfo file ซึ่งเป็น info_hash อันเดียวกับที่ใช้ส่งคำขอไปยังแทรคเกอร์
- peer_id: 20 bytes string ที่เป็นหมายเลขระบุตัวตนของเครื่อง อันเดียวกับที่ส่งไปบอกแทรคเกอร์
- ใน BitTorrent Protocol เวอร์ชัน 1.0 นั้น pstrlen=19 และ pstr="BitTorrent protocol" ผู้ที่เริ่มการเชื่อมต่อจะถูกคาดหวังให้ส่ง handshake ทันที ผู้รับอาจจะรอดู handshake ของผู้เริ่มหากสามารถรองรับการทำงาน torrent หลายงานได้พร้อมกัน ซึ่งผู้รับจะตอบสนองทันทีเมื่อได้ตรวจ info_hash ที่ส่งมา ถ้า info_hash ไม่ตรงกับ torrent ใดๆที่เครื่องลูกข่ายกำลังดำเนินอยู่ การเชื่อมต่อจะต้องยุติไป ถ้าหากผู้เริ่มได้รับการตอบที่มี peer_id ไม่ตรงกับที่คาดไว้ การเชื่อมต่อก็ยุติอีกเช่นกัน ทั้งนี้ peer_id ที่คาดนั้นนำมาจากที่ได้จากแทรคเกอร์

นอกจากนี้ยังมีส่วนของข้อความระหว่าง Peer เพื่อบอกสถานะระหว่างกัน รูปแบบจะเป็น <length prefix><message ID><payload> มีข้อความต่างๆ ดังรูปที่ 2.6 นี้

รูปที่ 2.6 รูปแบบของข้อความที่ส่งระหว่าง Peer

- keep-alive: <len=0000>: เป็นข้อความที่ค่าเป็นศูนย์ ดังที่บอกไว้ในความยาวของข้อความ โดยไม่มีส่วนของ message ID และ payload ปกติ peer จะยุติการเชื่อมต่อหากไม่ได้รับข้อความใดๆเลยเป็นเวลาหนึ่ง ข้อความ keep-alive สามารถส่งได้เพื่อรักษาการเชื่อมต่อ โดยทั่วไป keep-alive message จะส่งทุกๆสองนาที
- choke: <len=0001><id=0>: ข้อความปิดกั้นมีความยาวจำกัดและไม่มี payload
- unchoke: <len=0001><id=1>: ข้อความยุติการปิดกั้นมีความยาวจำกัดและไม่มี payload
- interested: <len=0001><id=2>: ข้อความระบุสนใจมีความยาวจำกัดและไม่มี payload
- not interested: <len=0001><id=3>: ข้อความระบุไม่สนใจมีความยาวจำกัดและไม่มี payload
- have: <len=0005><id=4><piece index>: ข้อความ have มีความยาวจำกัด payload เป็นดัชนีฐานศูนย์ของชิ้นส่วนที่โหลดโดยสมบูรณ์และตรวจสอบค่า hash แล้ว
- bitfield: <len=0001+x><id=5><bitfield>: ข้อความ bitfield จะถูกส่งทันทีหลังจากการทำ handshake ก่อนข้อความอื่นๆเพื่อระบุชิ้นส่วนที่มีอยู่แล้วตามในดัชนี ซึ่งหากเครื่อง

ลูกข่ายไม่มีอะไรเลยก็สามารถละข้อความนี้ไว้ไม่ต้องส่งได้ ข้อความจะตั้งดัชนีที่ตรงกับชิ้นส่วนที่มีแล้วให้ค่าบิตเป็น 1 ส่วนที่ไม่มีจะเป็น 0 โดยนับจากบิตสูงในไบต์แรกลงไป บิตที่เหลือท้ายข้อความจะตั้งเป็น 0

- request: <len=0013><id=6><index><begin><length>: ข้อความร้องขอมีความยาวจำกัด ใช้ร้องขอ block ในส่วนของ payload จะมีข้อมูลดังต่อไปนี้
  - index: จำนวนเต็มระบุตำแหน่งชิ้นส่วนในดัชนีฐานศูนย์
  - begin: จำนวนเต็มระบุตำแหน่งไบต์ที่ถัดเข้าไปในชิ้นส่วน
  - length: จำนวนเต็มระบุความยาวที่ขอ ค่าปกติมักเป็น 2^15 ไบต์ ค่าที่น้อยกว่าอาจนำมาใช้แต่มันจะไม่จำเป็นนอกจากในกรณีที่ชิ้นส่วนหารด้วย 2^15 ไม่ลงตัว
- piece: <len=0009+x><id=7><index><begin><block>: ข้อความ piece มีความยาวหลากหลาย ซึ่ง x เป็นความยาวของ block ส่วน payload จะมีข้อมูลดังนี้
  - index: จำนวนเต็มระบุตำแหน่งชิ้นส่วนในดัชนีฐานศูนย์
  - begin: จำนวนเต็มระบุตำแหน่งไบต์ที่ถัดเข้าไปในชิ้นส่วน
  - block: block ของข้อมูล ซึ่งเป็นหน่วยย่อยของชิ้นส่วนที่ระบุไว้ในดัชนี
- cancel: <len=0013><id=8>><index><begin><length>: ข้อความยกเลิกมีความยาวจำกัด ใช้เพื่อยกเลิกการร้องขอ block ค่าใน payload จะตรงกับข้อความ request ที่ใช้มักใช้ในช่วง End game algorithm
- port: <len=0003><id=9><listen-port>: ข้อความพอร์ตนี้ใช้ในรุ่นหลังๆของ Mainline ที่รองรับ DHT tracker (Trackerless) เพื่อระบุว่า peer รับฟัง DHT ที่พอร์ตไหน

2.2 พร็อกซี่เซิร์ฟเวอร์

2.2.1 พร็อกซี่เซิร์ฟเวอร์คืออะไร

พร็อกซี่เซิร์ฟเวอร์หรืออาจเรียกกันว่าแคชเป็นอุปกรณ์เครือข่ายชนิดหนึ่ง ซึ่งจะทำการเก็บข้อมูลของหน้าเว็บเพจต่างๆ ที่เราเรียกมาดูนั่นเอง โดยจะทำหน้าที่เป็นเสมือนที่เก็บข้อมูลหน้าเว็บเพจที่มีผู้เคยเรียกดูหน้าเว็บนั้นไว้ในระบบของเซิร์ฟเวอร์และเมื่อมีผู้อื่นเรียกดูข้อมูลที่เป็นหน้าเว็บเพจตัวเดียวกันกับที่เคยมีเก็บไว้แล้วก็ไม่จำเป็นที่จะต้องไปเรียกอ่านข้อมูลมาจากเซิร์ฟเวอร์ตัวต้นฉบับจริง ๆ ก็ได้ เพียงแค่ค้นหาข้อมูลเว็บเพจนั้นจากพร็อกซี่ก่อน ซึ่งถ้าหากหาพบหรือมีผู้อื่นเคยเรียกอ่านแล้วระบบพร็อกซี่เซิร์ฟเวอร์ก็สามารถส่งข้อมูลนั้นมาให้เราได้ดูได้ทันที ซึ่งวิธีนี้จะทำให้เราสามารถเรียกดูข้อมูลได้เร็วกว่าการเรียก มาจากเว็บไซต์ต้นฉบับจริงๆ แต่ข้อเสียของการตั้งใช้ พร็อกซี่เซิร์ฟเวอร์ ก็อาจจะมีบ้างเหมือนกัน คือ ข้อมูลที่มีการอัพเดตเร็วๆ

เช่นพวกเว็บบอร์ดต่าง ๆ อาจจะไม่มีการอัพเดตตามได้ทัน เพราะเราจะได้ข้อมูลที่มาจากพร็อกซี่ไม่ใช่มาจากต้นฉบับหรือแหล่งกำเนิดจริงๆ ในกรณีเช่นนี้ อาจจะทำการแก้ไขเบื้องต้น ได้โดยการกดปุ่ม Ctrl พร้อม ๆ กับการใช้เมาส์คลิกที่ปุ่ม Refresh จะเป็นการร้องขอ ข้อมูล หน้าเว็บเพ็จนั้นๆ ใหม่จากเซิร์ฟเวอร์ต้นฉบับ

2.2.2 หลักการทำงานของพร็อกซี่เซิร์ฟเวอร์

เมื่อมีผู้ใช้บริการทำการเรียกข้อมูลของเว็บไซต์โดยผ่านพร็อกซี่เซิร์ฟเวอร์ ในครั้งแรกพร็อกซี่เซิร์ฟเวอร์ จะทำการตรวจสอบว่า มีข้อมูลของเว็บไซต์นั้นมีอยู่หรือไม่ หากพบว่าไม่มีข้อมูล พร็อกซี่เซิร์ฟเวอร์ จะทำการเรียกข้อมูลนั้นจากเว็บไซต์แล้วเก็บไว้ในเครื่อง และเมื่อมีผู้ใช้บริการทำการเรียกเว็บไซต์นี้อีกครั้ง พร็อกซี่เซิร์ฟเวอร์จะทำการส่งข้อมูลไปยังเครื่องของผู้ใช้บริการทันที ในกรณีที่เว็บไซต์มีการอัพเดตข้อมูล พร็อกซี่เซิร์ฟเวอร์จะทำการตรวจสอบข้อมูลที่มีอยู่ว่าอัพเดตหรือไม่ และจะทำการอัพเดตข้อมูลใหม่ทันที ในกรณีที่มีผู้เรียกใช้บริการก็จะได้ข้อมูลที่อัพเดตอยู่เสมอ

2.2.3 การจัดการแคชของพร็อกซี่เซิร์ฟเวอร์

ในการจัดการแคชของ พร็อกซี่เซิร์ฟเวอร์ จะมีอยู่ 2 แบบ คือ

2.2.3.1 Passive Caching

จะเป็นการจัดการแคชแบบทั่วไป คือจะบันทึกข้อมูลลงในแคชของพร็อกซี่เฉพาะเมื่อไคลเอนต์มีการขอใช้ข้อมูลผ่านทางพร็อกซี่เซิร์ฟเวอร์เท่านั้น

2.2.3.2 Active Caching

จะใช้เทคนิคที่สูงกว่า โดย พร็อกซี่เซิร์ฟเวอร์ จะพิจารณาเองว่าควรจะเก็บข้อมูลจากเว็บเซิร์ฟเวอร์ใด และเก็บไว้ในแคชเมื่อใด โดยไม่จำเป็นต้องรอให้ไคลเอนต์ขอใช้ข้อมูลเข้ามาก่อน

2.2.4 โปรโตคอลที่ใช้ในการจัดการแคชของพร็อกซี่

ส่วนโปรโตคอลที่ใช้ในการจัดการแคชของพร็อกซี่นั้นมีอยู่2 แบบด้วยกันคือ

2.2.4.1 Internet Cache Protocol (ICP : RFC 2186)

Internet Cache Protocol หรือ ICP ได้รับการพัฒนามาจากมหาวิทยาลัย Southern California ซึ่งปัจจุบัน ICP ได้ถูกพัฒนาขึ้นเพื่อใช้งานร่วมกับ IPv6 ได้ด้วย (แต่ยังไม่ได้รับการรับรองเป็นมาตรฐาน) การใช้งาน ICP นี้จะใช้โปรโตคอลพื้นฐานร่วมกัน 3 ชุด โดยแยกหมายเลขพอร์ตให้ต่างกัน คือหมายเลขพอร์ตแรกจะใช้กับโปรโตคอล

HTTP เพื่อให้ไคลเอนต์ติดต่อเข้ามายัง พร็อกซี่เซิร์ฟเวอร์ และหมายเลขพอร์ตที่สองจะใช้กับโปรโตคอล UDP เพื่อสื่อสารแบบ ICP ระหว่าง พร็อกซี่เซิร์ฟเวอร์ ด้วยกัน ส่วนหมายเลขพอร์ตสุดท้ายจะใช้กับโปรโตคอล TCP เพื่ออ่านข้อมูลเว็บเพ็จจากแคชของ พร็อกซี่เซิร์ฟเวอร์ อย่างไรก็ตาม ICP ก็ยังมีจุดอ่อนอยู่ตรงที่ไม่สามารถรองรับการขยายตัวของระบบได้ดีพอ เนื่องจากข้อมูลที่แลกเปลี่ยนกันระหว่าง พร็อกซี่เซิร์ฟเวอร์ จะเพิ่มขึ้นอย่างมากและซ้ำซ้อนกัน เมื่อมีการเพิ่มจำนวน พร็อกซี่เซิร์ฟเวอร์ มากขึ้น ซึ่งเมื่อมีการร้องขอใช้ข้อมูลจาก พร็อกซี่เซิร์ฟเวอร์ นั้น ถ้าหากค้นหาไม่พบในเครื่องแรก ก็จะต้องค้นหาในเครื่องที่สองและต่อไปเรื่อยๆ ทำให้ประสิทธิภาพการทำงานลดลง

2.2.4.2 Cache Array Routing Protocol (CARP : RFC 3040)

Cache Array Routing Protocol หรือ CARP ได้พัฒนาให้มีความสามารถมากขึ้น โดยมุ่งที่จะลดภาระในการติดต่อสื่อสารระหว่าง พร็อกซี่เซิร์ฟเวอร์ ให้น้อยลง โดยการให้พร็อกซี่หลายๆชุดจัดเรียงตัวกัน โดยมีตารางเก็บรายละเอียดและหมายเลขไอพีของพร็อกซี่เซิร์ฟเวอร์ทั้งหมดไว้ ส่วนในการค้นหาข้อมูลนั้นไคลเอนต์ก็จะส่ง URL ที่ต้องการใช้งานผ่านเข้ามายัง พร็อกซี่เซิร์ฟเวอร์ ซึ่งจะใช้ฟังก์ชั่นการ Hash ที่ใช้หาตำแหน่งของข้อมูลที่ต้องการ นำมาใช้หา URL ดังกล่าวว่าอยู่ในแคชของ พร็อกซี่เซิร์ฟเวอร์ใด แต่ถ้าหากไม่พบก็จะไปดึงมาจากเว็บเซิร์ฟเวอร์โดยตรง ซึ่ง CARP นี้สามารถเรียกได้อีกอย่างว่า Queryless Distributed Caching และในปัจจุบันผลิตภัณฑ์ทั้งของบริษัทไมโครซอฟท์และเน็ตสเคปก็ได้สนับสนุนโปรโตคอลแบบ CARP ทั้งสิ้น

2.2.5 ประโยชน์ของการใช้พร็อกซี่เซิร์ฟเวอร์

ผู้ใช้บริการสามารถเรียกดูข้อมูลจากเว็บไซต์ต่าง ๆ ได้รวดเร็วขึ้น และช่วยประหยัดเวลาในการใช้งานอินเทอร์เน็ต ทั้งนี้ก็เพราะพร็อกซี่เซิร์ฟเวอร์ ก็จะสามารถใช้ข้อมูลที่เก็บไว้จากการร้องขอของผู้ใช้รายแรกมาส่งให้แก่ผู้ใช้รายอื่นๆ ได้เลยโดยไม่จำเป็นต้องทำการร้องขอไปยังเว็บเซิร์ฟเวอร์อีกครั้ง ทำให้สามารถประหยัดได้ทั้งเวลาและแบนวิดของเครือข่าย นอกจากนี้ ยังได้ข้อมูลที่มีความถูกต้องและทันสมัยอยู่ตลอดเวลา

นอกจากนี้พร็อกซี่ยังเกี่ยวข้องกับเรื่องของความปลอดภัยของผู้ใช้ด้วย ตามปกติแล้วถ้าผู้ใช้งานที่มีความรู้พื้นฐานเกี่ยวกับอินเตอร์เน็ตบ้าง ก็คงจะเคยได้ยินหรือรู้จักคำว่าไอพีแอดเดรสมาบ้าง คอมพิวเตอร์ที่อยู่บนเน็ตเวิร์ค (ที่ใช้โปรโตคอล TCP/IP) หรือบนอินเตอร์เน็ตจะมีไอพีแอดเดรสเอาไว้เพื่อเป็นการบอกที่อยู่ของเครื่องคอมพิวเตอร์แต่ละเครื่องบนเน็ตเวิร์คนั้นๆ หรือบนอินเตอร์เน็ต ในบางครั้งเวลาที่เราเปิดเว็บบราวน์เซอร์แล้วไปยังเว็บไซต์ต่างๆนั้น พวกเว็บไซต์เหล่านี้จะเก็บข้อมูลหลายๆอย่างของผู้ที่เข้ามาเยี่ยมชมเว็บไซต์ โดยส่วนมากเก็บ log เป็น

ไฟล์ไว้ด้วยความสามารถของตัวเว็บเซิร์ฟเวอร์เองที่มีอยู่แล้วหรือ Script ต่างๆ ข้อมูลหลายๆอย่างที่พูดมานั้น หนึ่งในนั้นก็มีไอพีแอดเดรสรวมอยู่ด้วย ในบางครั้งตัวพร็อกซี่เซิร์ฟเวอร์สามารถช่วยปิดบัง ข้อมูลต่างๆรวมทั้งไอพีแอดเดรสของเรา เช่น การเรียกดูเว็บเพจจะเป็นพร็อกซี่เซิร์ฟเวอร์เองที่ไปเรียกดูจากเว็บไซต์นั้นๆ แทนที่จะเป็นเครื่องคอมพิวเตอร์ของเราที่ไปเรียกดูโดยตรง เว็บไซต์เหล่านั้นจึงได้แต่ข้อมูลของพร็อกซี่เซิร์ฟเวอร์ไปแทนที่จะได้จากเครื่องคอมพิวเตอร์ของผู้ใช้งานที่เรียกดู

อีกประการหนึ่งพร็อกซี่ยังมีคุณสมบัติในด้านการจำกัดสิทธิที่จะเข้าถึงเว็บไซต์บางแห่งที่มีเนื้อหาไม่สมควรเข้าชม การจำกัดยูสเซอร์ใช้งานในเวลาที่นอกเหนือจากเวลางาน หรือข้อจำกัดอื่น ๆ ที่ทำให้สิ้นเปลืองค่าใช้จ่ายไปโดยไม่ก่อให้เกิดประโยชน์ คุณสมบัติเช่นนี้สำพังอินเตอร์เน็ตเกตเวย์คงยังไม่เพียงพอ จึงจะต้องอาศัยพร็อกซี่เซิร์ฟเวอร์ (พร็อกซี่เซิร์ฟเวอร์) เข้ามาช่วยเสริมอีกแรงหนึ่ง ซึ่งภายในลีนุกซ์ทุกดิสทริบิวชั่นจะมีโปรแกรมพร็อกซี่เซิร์ฟเวอร์ที่มีประสิทธิภาพสูงให้มาพร้อมแล้ว คือ โปรแกรม Squid

โปรแกรม Squid เป็น พร็อกซี่เซิร์ฟเวอร์ ที่มีคุณสมบัติในการจำกัด ควบคุมการแอกเซสเข้าสู่เว็บไซต์ภายนอกองค์กรได้เป็นอย่างดีและมีประสิทธิภาพ ที่เรียกว่า Access Control List (ACL) ซึ่งเป็นการนิยามชื่อลิสต์ขึ้นแทนคุณสมบัติของสิ่งที่ต้องการอ้างอิง จากนั้นจึงตั้งข้อกำหนดลงไปว่าต้องการให้ลิสต์นั้นสามารถแอกเซสผ่านพร็อกซี่ได้หรือไม่ ดังนั้นการที่เสริมการทำงานของอินเตอร์เน็ตเซิร์ฟเวอร์ด้วย Squid พร็อกซี่เซิร์ฟเวอร์ จึงเป็นการควบคุมการเข้าสู่อินเตอร์เน็ตของผู้ใช้งานในองค์กรได้ตามต้องการ และยังช่วยเพิ่มประสิทธิภาพให้แก่ระบบอีกด้วยเพราะ Squid จะมีคุณสมบัติเป็น HTTP Object cache ที่ช่วยเก็บข้อมูลจากเว็บไซต์ภายนอกไว้ในหน่วยความจำ (RAM และฮาร์ดดิสก์) ของตัวเซิร์ฟเวอร์เองอีกด้วย ช่วยให้การเรียกเว็บไซต์ที่เคยเข้าถึงมาก่อนทำได้รวดเร็วยิ่งขึ้น เนื่องจากมีข้อมูลบางส่วนของเว็บเพจที่ ยังคงอยู่ในแคชนั่นเอง

2.2.6 Transparent Proxy คืออะไร

พร็อกซี่เซิร์ฟเวอร์นั้นผู้ใช้งานทั่วไปจำเป็นจะต้องเข้าไปเซ็ตค่าพร็อกซี่เซิร์ฟเวอร์ ที่ตัวบราวเซอร์ก่อน ซึ่งอาจจะมีผู้ใช้บางคนที่ไม่ได้ทำตามนั้น ส่วน Transparent Proxy นั้นผู้ใช้งานไม่จำเป็นต้องเซ็ตค่าอะไรในเครื่องคอมพิวเตอร์ของตัวเอง ตัวของ Transparent Proxy นั้นจะใช้หลักของ iptables ในการ redirects ผู้ที่ติดต่อผ่านมาทางพอร์ต http ให้ไปยัง พร็อกซี่เซิร์ฟเวอร์ได้เองโดยอัตโนมัติ

2.3 โปรแกรม Squid

บนระบบปฏิบัติการลีนุกซ์ มีโปรแกรมพร็อกซี่เซิร์ฟเวอร์ที่มีประสิทธิภาพสูงมาก คือ โปรแกรม Squid ซึ่งเป็นโปรแกรมประเภท Proxy Caching Server สำหรับการให้บริการ Web Caching Service คือ จะคอยรับคำร้องขอบริการจากเครื่องลูกข่าย และส่งผ่านไปยังเซิร์ฟเวอร์ปลายทางที่เหมาะสม ข้อมูลต่าง ๆ ที่ผ่านเข้ามาจะถูกสำเนาเก็บไว้ในหน่วยความจำแคช และดิสก์ ดังนั้นเมื่อมีการร้องขอข้อมูลซ้ำอีกในครั้งต่อมาจะสามารถนำข้อมูลในแคชมาให้บริการได้รวดเร็วกว่าการติดต่อไปยังเซิร์ฟเวอร์โดยตรง ช่วยให้ลดการใช้ช่องทางสื่อสารข้อมูลลงได้

2.3.1 ความสามารถของโปรแกรม Squid

- Proxying และ caching ของ HTTP, FTP และ URLs อื่นๆ
- Proxying สำหรับ SSL
- Cache hierarchies
- ICP, HTCP, Cache Digests
- Transparent caching
- WCCP (squid v2.3 และเวอร์ชั่นที่ใหม่กว่านั้น)
- Extensive access controls
- HTTP server acceleration
- SNMP
- Caching of DNS lookups

2.3.2 Access Log ของโปรแกรม squid

เมื่อเครื่องลูกข่ายเรียกใช้งานเว็บไซต์ภายนอก จะทำการติดต่อผ่านพอร์ต 8080 มายังเครื่องเซิร์ฟเวอร์ลีนุกซ์ที่รันโปรแกรม Squid ไว้ เราสามารถมอนิเตอร์ดูความเคลื่อนไหวของการเรียกใช้งานดังกล่าวได้จาก LOG ไฟล์ของ squid ด้วยคำสั่ง

tail -f /var/log/squid/access.log (แล้วแต่จะกำหนดตำแหน่งของ LOG ไฟล์)

จะเห็นข้อความแสดงเหตุการณ์ต่าง ๆ ที่เกิดขึ้นเกี่ยวกับโปรแกรม Squid ได้ดังรูป ซึ่งเราสามารถตีความหมายได้ เช่น

TCP_MISS หมายถึง การร้องขอนั้นไม่มีการแคชข้อมูลไว้ กรณีนี้จะต้องไป GET มาจากเว็บไซต์ปลายทางมาจริง

TCP_HIT หรือ TCP_MEM_HIT หมายถึง มีข้อมูลที่เครื่องลูกข่ายร้องขอมาอยู่แล้วในแคช กรณีนี้จะเอาข้อมูลที่อยู่ในแคชส่งกลับไปให้ผู้ร้องขอ

1025162259.349 6 192.168.0.254 TCP_MEM_HIT/200 440 GET
http://www.thairath.co.th/picuse/rmore_it.gif - NONE/- image/gif
1025162259.454 11 192.168.0.254 TCP_MEM_HIT/200 653 GET
http://www.thairath.co.th/picuse/foot_it.gif - NONE/- image/gif
1025162259.758 3 192.168.0.254 TCP_HIT/200 9413 GET
http://www.thairath.co.th/link/schedule.jpg - NONE/- image/jpeg
1025162259.970 17 192.168.0.254 TCP_HIT/200 8550 GET
http://www.thairath.co.th/link/IT.banner.gif - NONE/- image/gif
1025162263.053 3321 192.168.0.254 TCP_MISS/200 400 GET
http://adserve.inet.co.th/jnserver/SITE=thairath.co.th/AREA=thairath.index/A
AMSZ=195x39 - DIRECT/203.150.14.102 application/x-javascript
1025162263.165 2992 192.168.0.254 TCP_MISS/200 430 GET
http://www.thairath.co.th/pichead/dotrd.gif - DIRECT/203.151.217.25
image/gif

ไฟล์ access.log ของโปรแกรม Squid นี้จะมีขนาดเพิ่มขึ้นเรื่อย ๆ จนอาจทำให้เนื้อที่ดิสก์เต็ม และสร้างปัญหาให้กับระบบได้ ดังนั้นจึงควรใช้กำหนดให้ Squid ทำการเปลี่ยนไฟล์ Log ด้วยคำสั่ง

squid -k rotate

โดยอาจจะเขียนเป็นเชลล์สคริปต์เพื่อทำให้ Log ไฟล์เกิดการหมุนเวียนกันไป ในแต่ละสัปดาห์ก็ได้ แต่ถ้าไม่จำเป็นต้องใช้ข้อมูลจาก Log ก็สามารถยกเลิกการเก็บ Log ไว้ได้โดยใช้คีย์เวิร์ด ดังนี้

cache_access_log /dev/null

นอกจาก access.log แล้ว ไฟล์ที่สำคัญอีกไฟล์หนึ่งก็คือ cache.log มีหน้าที่เก็บสถานะและข้อมูลเกี่ยวกับการดีบั๊ก โปรแกรม Squid ซึ่งมีประโยชน์อย่างมากในการแก้ไขปัญหาต่าง ๆ ที่เกิดขึ้น

2.3.3 Cache Dir แคชของโปรแกรม Squid

พื้นที่เก็บข้อมูลแคชบนดิสก์ของ Squid จะอยู่ที่ /var/spool/squid (อาจไม่อยู่ในตำแหน่งนี้เสมอไป ขึ้นอยู่กับการกำหนดของผู้ติดตั้งโปรแกรม) พื้นที่นี้ควรแยกออกจากพาร์ทิชั่นของระบบอย่างเด็ดขาด เนื่องจากพื้นที่นี้จะมีข้อมูลอ่าน เขียนอยู่ตลอดเวลาทำให้มีโอกาสเสื่อมสภาพมากกว่าส่วนอื่นๆ หากเกิดข้อผิดพลาดขึ้นแล้วจะได้ไม่สร้างความเสียหายไปยังส่วนอื่น ๆ ที่สำคัญต่อระบบด้วย นอกจากนี้หากพิจารณาเรื่องของความเร็ว และประสิทธิภาพแล้ว ควรสร้างพื้นที่นี้ในฮาร์ดดิสก์ที่มีความเร็วสูง และหากเป็นไปได้ควรใช้ฮาร์ดดิสก์ชนิด SCSI จะเหมาะสมกว่าชนิด IDE

หากเราต้องการตรวจดูว่า พื้นที่ที่เราใช้งานเป็นดิสก์แคชมีการใช้เนื้อที่ไปแล้วมากน้อยแค่ไหน สามารถตรวจสอบได้จากคำสั่ง

du -sh /var/spool/squid

หรือกรณีที่แบ่งแยกพาร์ทิชั่นนี้ออกมา ดังที่กล่าวไว้ข้างต้น สามารถตรวจสอบได้ด้วยคำสั่ง df -h อีกคำสั่งหนึ่ง

2.3.4 โปรแกรมยูทิลิตี้ของโปรแกรม Squid

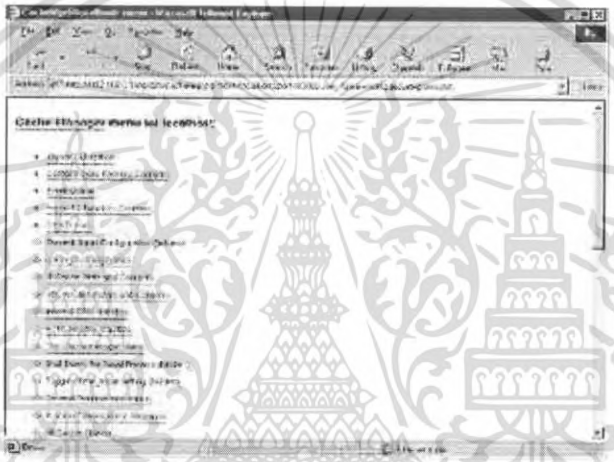
โปรแกรม cachemgr.cgi เป็นยูทิลิตี้ของ Squid ที่ช่วยในด้านการรายงานสถานะ ค่าสถิติต่าง ๆ เกี่ยวกับคอนฟิกและประสิทธิภาพของโปรแกรม Squid โดยจะมีติดตั้งมาพร้อมกันกับ Squid แล้ว โปรแกรมนี้ถูกออกแบบให้ใช้งานผ่านเว็บ (Web Interface)

โปรแกรมนี้จะอยู่ที่ไดเร็คทอรี่ /usr/lib/squid และเราจะต้องสำเนาไปไว้ที่พื้นที่ CGI ของเว็บเซิร์ฟเวอร์ คือ /var/www/cgi-bin เพื่อให้สามารถเรียกใช้งานได้

cp /usr/lib/squid/cachemgr.cgi /var/www/cgi-bin

เมื่อได้ติดตั้งโปรแกรมนี้ไว้ที่พื้นที่ CGI ของเว็บเซิร์ฟเวอร์แล้ว ให้เปิดโปรแกรม Web Browser ไปที่แอดเดรส http://192.168.0.1/cgi-bin/cachemgr.cgi โดยที่ 192.168.0.1 เป็นไอพีแอดเดรสของเครื่องเว็บเซิร์ฟเวอร์และ Squid ของเรา จากนั้นจะต้องป้อนชื่อของ cache manger และรหัสผ่านตามที่ได้กำหนดเอาไว้ในไฟล์ squid.conf เมื่อคลิ๊กที่ปุ่ม Continue แล้วเห็นรายละเอียดต่าง ๆ เป็นข้อ ๆ ดังรูป ซึ่งเป็นข้อมูลสถิติทั้งหมดเกี่ยวกับ Squid Proxy Server ของเรา ดังรูปที่ 2.7 ถึง 2.9

รูป 2.7 การล็อกอินในฐานะ Cache Manager

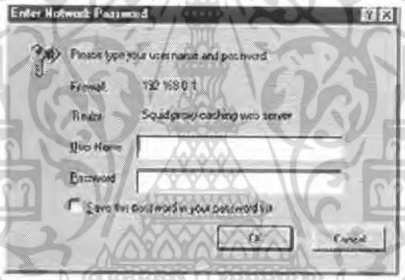


รูป 2.8 เมนูหลักของโปรแกรม Cache Manager

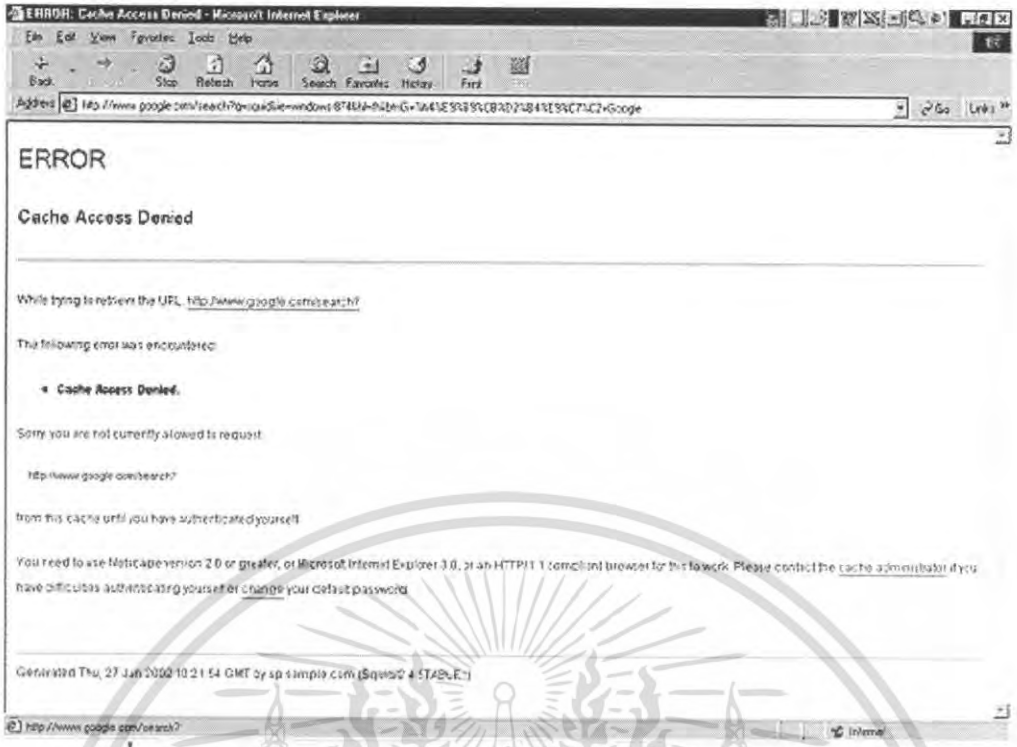
รูป 2.9 สถิติการเข้าสู่เว็บไซต์ต่าง ๆ

2.3.5 User Authentication ตรวจสอบสิทธิ์ของผู้ใช้งานเว็บ

คุณสมบัติที่น่าสนใจอีกประการหนึ่งของ Squid ก็คือการตรวจสอบสิทธิ์ของผู้ใช้งานด้วยชื่อ และรหัสผ่าน ซึ่งจะช่วยให้ผู้จัดการระบบสามารถควบคุมการใช้งานเว็บของผู้ใช้แต่ละคนได้ เมื่อนำมารวมเข้ากับคุณสมบัติด้าน Filewall หรือ Access Control List แล้ว จะสามารถกำหนดเงื่อนไขได้หลากหลายมาก เช่น กำหนดให้นายสมชายสามารถเข้าได้เฉพาะเว็บไซต์ที่เกี่ยวกับงานของบริษัทเท่านั้น นางสาวสมศรีสามารถเข้าเว็บได้เฉพาะเวลา 12.00 น. - 13.00 น. เท่านั้น เป็นต้น ซึ่งการตรวจสอบรายชื่อของผู้ใช้จะต้องอาศัยโปรแกรมภายนอกเข้ามาเสริมการทำงาน โดยที่ใน Squid เองก็มีโปรแกรมจำพวกนี้ให้มาบ้างแล้วจำนวนหนึ่ง ซึ่งจะอยู่ภายในไดเร็คทอรี่ /usr/lib/squid ได้แก่ smb_auth.sh, pam_auth เป็นต้น ซึ่งแต่ละโปรแกรมก็จะมีเทคนิคการตรวจสอบรายชื่อผู้ใช้ที่แตกต่างกันออกไป บ้างก็ใช้ระบบรายชื่อจาก Windows NT บ้างก็ใช้ระบบรายชื่อจากตัว Linux เอง ซึ่งผู้จัดการระบบสามารถเลือกใช้ได้ตามความต้องการดังรูปที่ 2.10 และรูปที่ 2.11



รูปที่ 2.10 แสดงกรอบโต้ตอบของโปรแกรม mySquid_Auth



รูปที่ 2.11 แสดงข้อความของ Squid เมื่อผู้ใช้เข้าสู่เว็บไซต์ที่ไม่ได้รับอนุญาต

อย่างไรก็ตามเมื่อได้นำโปรแกรมเหล่านี้มาใช้งานจึงได้พบข้อบกพร่อง และข้อจำกัดที่ไม่ตรงตามความต้องการใช้งาน ผู้เขียนจึงต้องเขียนโปรแกรมขึ้นเอง ชื่อโปรแกรม mySquid_Auth ซึ่งมีระบบจัดเก็บรายชื่อของผู้ใช้เป็นของตนเองโดยอิสระ และสามารถจัดเก็บสถิติการใช้งานของผู้ใช้เพื่อตรวจสอบได้ในภายหลัง โปรแกรมนี้สามารถดาวน์โหลดได้จากเว็บไซต์ http://www.itdestination.com

Squid เป็นโปรแกรมพร็อกซี่เซิร์ฟเวอร์ที่มีประสิทธิภาพสูง มีคุณสมบัติน่าสนใจมากมาย สามารถช่วยให้การใช้แบนวิธของเครือข่ายเป็นไปอย่างคุ้มค่า เรื่องราวของ Squid ยังมีสิ่งที่ต้องศึกษาอีกมากมาย ไม่ว่าจะเป็นเรื่องของการจัดสรรโควตาเพื่อจำกัดการใช้แบนวิธ การใช้งาน Squid ในโหมด HTTPd Accelerator การทำงานในสภาพแวดล้อมแบบ Multi-Level Web Caching การทำงานในแบบ Transparent Proxy และเรื่องราวอื่นๆ อีกมากมาย โดยสามารถศึกษาเพิ่มเติมได้จากเว็บไซต์ของโปรเจ็ค Squid ที่ http://www.squid-cache.org

2.4 โปรแกรม SquidGuard

โปรแกรม SquidGuard คือโปรแกรมที่เป็น plugin สำหรับโปรแกรม Squid ซึ่งมีความสามารถของการทำ filter, redirect และการทำ access controller โดยโปรแกรม SquidGuard นั้น เป็นโปรแกรมฟรีแวร์ที่ทำงานบนระบบปฏิบัติการ UNIX

2.4.1 โปรแกรม SquidGuard นำมาใช้ในด้าน

- จำกัดการเข้าเว็บไซต์สำหรับผู้ใช้บางคนโดยพิจารณาจากไอพีแอดเดรส และ/หรือ URLs
- บล็อกการเข้าเว็บไซต์ที่อยู่ใน list หรือ blacklist
- บล็อกการเข้า URLs ที่ตรงกับ list ของ regular expression หรือ words
- Redirect จากเว็บไซต์ที่บล็อกไปยังหน้าเว็บไซต์ที่ต้องการได้

2.4.2 ความสามารถของโปรแกรม SquidGuard

- define different time spaces ได้แก่
  - time of day (00:00-08:00 17:00-24:00)
  - day of the week (sa)
  - date (1999-05-13)
  - date range (1999-04-01-1999-04-05)
  - date wildcards (*-01-01 *-05-17 *-12-25)
- group source (users/clients) ได้แก่
  - IP address range with
    - Prefix notation (172.16.0.0/12)
    - Netmask notation (172.16.0.0/255.240.0.0)
    - First-last notation (172.16.0.11-172.16.0.35)
  - Address lists (172.16.134.54 172.16.156.23 ...)
  - Domain lists (foo.bar.com ...) *)
  - User id lists (weho sdgh dfhj asef ...) **)
  - Positively (within business-hours)
  - Negatively (outside leisure-time)
- group destinations (URLs/servers) ได้แก่
  - Domains รวมทั้ง subdomains (foo.bar.com)
  - Hosts (host.foo.bar.com)
  - Directory URLs รวมทั้ง subdirectories (foo.bar.com/some/dir)
  - File URLs (foo.bar.com/somewhere/file.html)
  - Regular expression ((expr1|expr2|...))
  - Positively (within business-hours)
  - Negatively (without leisure-time)

- rewrite/redirect URLs ได้แก่
  - string/regular expression editing โดย
    - silent squid redirecting rewrite (s@from@to@[i])
    - visible client redirecting rewrite (s@from@to@[i]r) ***)
  - URL replacement โดย
    - Silent squid redirect to a common URL(redirect "new_url")
    - Visible client redirect to a common URL (redirect "302:new_url") ***)
- define access control lists (acl)
  - giving each source (user/client) group
    - a pass list
      - acceptable destination groups (good-dests …)
      - unacceptable destination groups (!bad-dests …)
      - block IP address URLs (enforce the use of domain names) (!in-addr)
    - wildcards/nothing (any|all|none)
    - optionally a common rewrite rule set สำหรับ source group
    - optionally a default replacement URL สำหรับบล็อก destination สำหรับ source group
  - link the acl to a given time space
  - positively (within business-hours)
    - negatively (outside leisure-time)
  - defining a fallback/default ruleset

- มีตัวเลือกสำหรับ logging ดังนี้
  - Source/client group declaration เป็น log สำหรับทุกๆ translation ของ group (log "file")
  - Destination group declaration. เป็น log สำหรับที่ match กับ blacklist (log "file")
  - Rewrite rule group declaration เป็น log สำหรับทุกๆ translation ของ rule set

2.5 เว็บเซอร์วิส

2.5.1 ที่มาของเว็บเซอร์วิส

ปัจจุบันท่ามกลางสภาวะเศรษฐกิจที่เปลี่ยนแปลงประกอบกับการแข่งขันทางธุรกิจที่สูงขึ้น ปัจจัยหลายๆ อย่างเข้ามามีอิทธิพลต่อรูปแบบการดำเนินธุรกิจ ทว่าแนวโน้มที่เห็นได้ชัดเจนสำหรับธุรกิจในยุคใหม่ คือ การทำธุรกิจในลักษณะอิเล็กทรอนิกส์หรือที่เรียกว่า พาณิชย์อิเล็กทรอนิกส์ (E-Business) ซึ่งเป็นวิธีการทำธุรกิจที่น่าสนใจและกำลังแพร่หลายในหลายๆ ธุรกิจ รวมทั้งยังเป็นปัจจัยหนึ่งที่มีส่วนทำให้องค์กรทางธุรกิจนั้นๆยังยืนหยัดอยู่ได้ องค์กรทางธุรกิจต่างๆ ได้เรียนรู้ที่จะนำข้อมูลและทรัพยากรด้านเทคโนโลยีสารสนเทศ มาใช้เป็นเครื่องมือในการพัฒนาธุรกิจของตนมากยิ่งขึ้น เพื่อเป็นฐานที่มั่นคงและการเติบโตทางเศรษฐกิจในอนาคต

องค์กรทางธุรกิจต่างๆ เริ่มมองเห็นวิธีที่ดีที่สุดสำหรับใช้ในการพัฒนาประสิทธิภาพของการปฏิบัติงาน ลดค่าใช้จ่าย และสานสัมพันธ์ที่มีกับลูกค้ารวมถึงซัพพลายเออร์รายสำคัญให้แน่นแฟ้นยิ่งขึ้น โดยสร้างวิธีการดำเนินธุรกิจแบบใหม่ที่ใช้เทคโนโลยีผสมผสานกับเครือข่ายอินเทอร์เน็ตที่แพร่หลายในปัจจุบัน ซึ่งเทคโนโลยีที่กำลังได้รับความสนใจและยังเป็นเครื่องมือที่สนับสนุนความต้องการดังกล่าวในธุรกิจยุคใหม่ก็คือ เว็บเซอร์วิส

เป็นที่ยอมรับกันว่าเทคโนโลยีสารสนเทศและเครือข่ายอินเทอร์เน็ตเป็นเครื่องมือธุรกิจที่จำเป็นในการดำเนินธุรกิจยุคใหม่ไปแล้ว ซึ่งเว็บแอพพลิเคชันเป็นพระเอกหรือเป็นเครื่องมือสำคัญในการทำธุรกิจบนโลกของอินเทอร์เน็ต แต่ปัจจุบันเว็บแอพพลิเคชันกำลังจะกลายเป็นเพียงตัวประกอบ ด้วยการที่องค์กรธุรกิจเริ่มหันมาสนใจเทคโนโลยีใหม่อย่างเว็บเซอร์วิส ซึ่งเข้ามาช่วยเพิ่มประสิทธิภาพให้กับระบบการดำเนินธุรกิจขององค์กร และยังสามารถทำงานร่วมกับเว็บแอพพลิเคชันเดิมได้เป็นอย่างดี

เพื่อให้องค์กรธุรกิจต่างๆ ได้เห็นแนวทางในการนำเว็บเซอร์วิสไปใช้ในธุรกิจของตน แต่ละองค์กรจำเป็นต้องเข้าใจความหมายของเว็บแอพพลิเคชันและเว็บเซอร์วิส เพื่อที่จะได้เห็นความแตกต่างและนำไปประยุกต์ใช้งานได้อย่างถูกต้องและมีประสิทธิภาพมากที่สุด

2.5.2 เว็บแอพพลิเคชั่นคืออะไร

เว็บแอพพลิเคชั่น (Web application) คือ โปรแกรมที่อยู่ในเว็บเซิร์ฟเวอร์ที่คอยให้บริการสิ่งที่ร้องขอ (request) จากทางไคลเอนต์ผ่านโปรโตคอล HTTP ซึ่งจะแสดงผลที่ร้องขอในรูปของ HTML page ผ่านทางบราวเซอร์ ซึ่งก็คือเว็บไซต์ต่างๆ ที่เราใช้บริการอยู่นั่นเอง

เว็บแอพพลิเคชันสามารถตอบสนองความคิด Distributed Processing ได้ในระดับหนึ่ง ซึ่งก็คือ การแบ่งการประมวลผลไว้ที่ฝั่งไคลเอนต์และฝั่งเซิร์ฟเวอร์ และมักจะมีการใช้ฐานข้อมูล (database) ควบคู่กับการทำเว็บแอพพลิเคชันไปด้วยตามความต้องการในการทำ E-Business และ

E-Commerce ที่กำลังเป็นที่นิยมในปัจจุบัน แต่ปัญหาที่ตามมาคือ เรื่องของการจ่ายเงินหรือที่เรียกว่า E-payment หรือ Payment-Gateway ซึ่งเว็บแอพพลิเคชันที่ทำ E-Commerce ต้องใช้บริการจากธนาคารออนไลน์ ในการจัดเก็บเงินกับลูกค้าด้วยเทคโนโลยีนี้ การใช้บริการเก็บเงินจากธนาคารออนไลน์ จำเป็นที่ผู้ค้าจะต้องไปทำการตกลงกับธนาคารและเขียนโปรแกรมให้ตรงตามมาตรฐานที่ธนาคารออนไลน์กำหนดไว้

2.5.3 เว็บเซอร์วิสคืออะไร

เว็บเซอร์วิส คือ แอพพลิเคชันหรือโปรแกรมที่ทำงานอย่างใดอย่างหนึ่ง ในลักษณะให้บริการ โดยจะถูกเรียกใช้งานจากแอพพลิเคชันอื่นๆ ในรูปแบบ RPC(Remote Procedure Call) ซึ่งการให้บริการจะมีเอกสารที่อธิบายคุณสมบัติของบริการกำกับไว้ โดยภาษาที่ถูกใช้เป็นสื่อในการแลกเปลี่ยนคือ XML ทำให้เราสามารถเรียกใช้ส่วนประกอบใดๆ ก็ได้ ในระบบหรือ platform ใดๆ ก็ได้ บนโปรโตคอล HTTP ซึ่งเป็นโปรโตคอลสำหรับอินเทอร์เน็ต อันเป็นช่องทางที่ได้รับการยอมรับทั่วโลกในการติดต่อสื่อสารกันระหว่างแอพพลิเคชันกับแอพพลิเคชันในปัจจุบัน

การทำงานของเว็บเซอร์วิส ประกอบไปด้วย มาตรฐานหลัก 4 อย่าง ซึ่งสามารถอธิบายอย่างง่ายๆ ได้ดังนี้

1. XML (Extensible Markup Language) เป็นภาษามาตรฐานที่ทุกระบบสนับสนุน ทำให้ข้อมูลที่มีโครงสร้างของภาษา XML จะถูกนำไปประมวลผลต่ออย่างอัตโนมัติได้อย่างง่ายดาย ภาษา XML จึงถูกนำมาใช้เป็นภาษามาตรฐานในการแลกเปลี่ยนข้อมูลของเว็บเซอร์วิส
2. SOAP (Simple Object Access Protocol) หรือโซพ เป็นมาตรฐานของเทคโนโลยี Distributed Objects แบบหนึ่ง โดยทำหน้าที่ส่งข้อมูลผ่านอินเทอร์เน็ต ในรูปแบบของ XML ทำให้เรียกใช้งานโปรแกรมข้ามระบบผ่านทางอินเทอร์เน็ตได้
3. WSDL (Web Services Description Language) เป็นภาษามาตรฐานที่ใช้สำหรับอธิบายการใช้งานโปรแกรมที่เปิดให้บริการ ซึ่งเขียนขึ้นตามแบบมาตรฐาน XML ดังนั้น WSDL จึงเป็นเสมือนคู่มือให้กับระบบ เพื่อเรียนรู้วิธีการเรียกใช้งานเว็บเซอร์วิส
4. UDDI (Universal Description, Discovery, and Integration) เป็นระบบมาตรฐานในการอธิบายและค้นหาเว็บเซอร์วิส โดยเป็นตัวกลางให้ Provider มาลงทะเบียนไว้ โดยใช้ไฟล์ WSDL บอกรายละเอียดของบริษัทและบริการที่มีให้ ทำให้ Requestor สามารถค้นหาและทราบว่าบริษัทมีผลิตภัณฑ์และบริการอะไรบ้าง สามารถติดต่อขอดำเนินธุรกิจการค้ากับบริษัทได้โดยอัตโนมัติผ่านทางเว็บเซอร์วิส

จากมาตรฐานทั้ง 4 อย่างที่กล่าวข้างต้นสามารถสรุปลำดับขั้นของการทำงานของเว็บเซอร์วิส ได้ดังนี้

1. Provider จัดทำระบบหรือบริการที่เป็นเว็บเซอร์วิสขึ้นมา
2. ทำการลงทะเบียนเว็บเซอร์วิสกับหน่วยงานที่ให้บริการระบบ UDDI
3. นำ WSDL ไฟล์ไปไว้ในระบบ UDDI ที่ได้ลงทะเบียนไว้
4. Requestor ทำการค้นหาระบบหรือบริการที่ต้องการจากระบบ UDDI
5. เมื่อ Requestor ได้พบระบบหรือบริการที่ต้องการจะนำไฟล์ WSDL ไปเรียนรู้วิธีการเรียกใช้ผ่านระบบของตน
6. Requestor ทำการติดต่อและเรียกใช้ระบบหรือบริการจาก Provider ได้โดยตรงผ่าน SOAP ในระบบของตน

ถ้าพิจารณาลักษณะการทำงานและความสามารถของเว็บเซอร์วิสแล้ว เราสามารถสรุปได้ว่าเว็บเซอร์วิสก็คือ วิวัฒนาการอีกก้าวหนึ่งของเว็บแอพพลิเคชันนั่นเอง

2.5.4 เปรียบเทียบระหว่างเว็บแอพพลิเคชันและเว็บเซอร์วิส

เมื่อเราเข้าใจความหมายและการทำงานเว็บแอพพลิเคชันและเว็บเซอร์วิสแล้ว จะเห็นว่าเครื่องมือทั้งสองต่างใช้ HTTP Protocol หรืออินเทอร์เน็ตเป็นช่องทางในการสื่อสารเหมือนกัน แต่มีวัตถุประสงค์ต่างกัน โดยเว็บแอพพลิเคชันใช้เพื่อการแลกไฟล์ HTML ระหว่าง เว็บเซิร์ฟเวอร์ แต่เว็บเซอร์วิสเป็นการแลก “บริการ” ระหว่างระบบสารสนเทศ ผ่านเว็บเซิร์ฟเวอร์

ในเรื่องของความสามารถโดยส่วนใหญ่จะใช้เว็บแอพพลิเคชัน ในการติดต่อกับผู้ใช้ผ่านทางเว็บบราวเซอร์ เพื่อนำเสนอข้อมูลและการทำธุรกรรมต่างๆ ส่วนเว็บเซอร์วิสจะทำหน้าที่ในการติดต่อกับเว็บเซิร์ฟเวอร์ เพื่อแลกเปลี่ยนข้อมูลและการทำงานหรือใช้บริการข้ามระบบกันโดยใช้ Web Application หรือ Application Interface ในการติดต่อกับผู้ใช้ นอกจากนี้เว็บเซอร์วิสยังสามารถทำงานกับระบบต่างๆ ได้มากกว่า 1 ระบบ ในขณะที่เว็บแอพพลิเคชันไม่สามารถทำได้โดยตรง ซึ่งสามารถสรุปการเปรียบเทียบได้ดังตารางที่ 2.2

ตารางที่ 2.2 สรุปการเปรียบเทียบระหว่างเว็บเซอร์วิสกับเว็บแอพพลิเคชัน

หัวข้อ	**Web Services**	**Web Applications**
การเชื่อมต่อ	program-program	human-program
ภาษาที่ใช้	XML	HTML
รายชื่อการให้บริการ	ค้นหาผ่าน UDDI	ค้นหาผ่าน search engine
ขอบเขตการใช้งาน	Business-to-Business (B2B)	Business-to-Customer (B2C)
โปรโตคอล(Protocol)	SOAP+HTTP	HTTP

หลังจากที่ได้ทำความเข้าใจความหมายและข้อเปรียบเทียบระหว่างเว็บแอพพลิเคชันและเว็บเซอร์วิสแล้ว ความแตกต่างระหว่างเครื่องมือทั้งสองจะเห็นได้ชัดในเรื่องของความสามารถ แต่ในการนำไปใช้งานจำเป็นจะต้องประยุกต์ใช้เครื่องมือทั้งสองชนิดร่วมกัน เพื่อให้เกิดประสิทธิภาพมากที่สุด

บทที่ 3
การออกแบบและพัฒนาระแบบแคชสำหรับโปรแกรมบิตทอเรนต์

ก่อนที่จะทำการออกแบบและพัฒนาระบบแคชสำหรับโปรแกรมบิตทอเรนต์ขึ้นมาเพื่อใช้งานนั้น จำเป็นอย่างยิ่งที่จะต้องมีการศึกษาหาข้อมูลต่างๆที่เกี่ยวข้องกับระบบแคชสำหรับโปรแกรมบิตทอเรนต์ ดังที่ได้นำเสนอในบทที่ผ่านมา ในการออกแบบระบบแคชสำหรับโปรแกรมบิตทอเร้นนั้น จะต้องคำนึงถึงหลักของโปรโตคอลบิตทอเรนต์ หลักของพร็อกซี่เซิร์ฟเวอร์ หลักการของเว็บเซอร์วิส รวมไปถึงกระบวนการทำงานของตัวโปรแกรมบิตทอเรนต์ที่ใช้ในการดาวน์โหลดไฟล์

บทนี้จะกล่าวถึงการออกแบบและพัฒนาระบบแคชสำหรับโปรแกรมบิตทอเรนต์ โดยจะบอกถึงแบบจำลองของระบบ องค์ประกอบต่างๆที่สำคัญของระบบ รวมไปถึงขั้นตอนการทำงานของระบบแคชสำหรับโปรแกรมบิตทอเรนต์

3.1 การทำงานของบิตทอเรนต์

ก่อนที่จะทำการออกแบบระบบแคชสำหรับโปรแกรมบิตทอเรนต์นั้น จะต้องศึกษาทฤษฎีหลักการทำงานของโปรโตคอลบิตทอเรนต์เสียก่อน โดยพบว่าระบบการทำงานของโปรโตคอลบิตทอเรนต์นั้นมีองค์ประกอบที่สำคัญ ได้แก่

1. Torrent Server

   Torrent Server คือ เครื่องเซิร์ฟเวอร์ที่เก็บไฟล์ .torrent เอาไว้ โดย Client จะมาโหลดไฟล์ .torrent จากที่นี่ ซึ่งส่วนมากแล้ว Torrent Server จะทำหน้าที่เป็น Tracker ในตัวเดียวกันอีกด้วย

2. Tracker

   Tracker มีหน้าที่ที่สำคัญ คือ คอยแจกจ่ายหมายเลขไอพีของผู้ที่มีไฟล์ข้อมูลที่ Client ต้องการกลับไปยัง Client และคอยทำหน้าที่เก็บข้อมูลสถิติต่างๆของ seeder และ peer ที่เคยติดต่อกับ Tracker นี้

3. Seeder

   Seeder คือ ผู้ที่มีไฟล์ข้อมูลที่สมบูรณ์และเป็นผู้ที่คอยส่งชิ้นของไฟล์ข้อมูลที่มีอยู่ให้กับ Client ที่ต้องการไฟล์ข้อมูลนั้นๆ

4. Peer

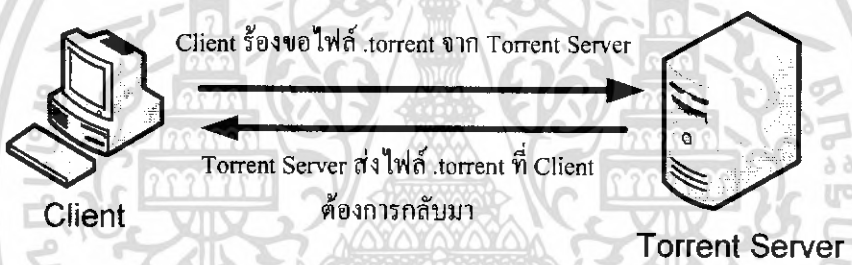
Peer คือ ผู้ที่มีความต้องการที่จะได้ไฟล์ข้อมูล โดยการรับชิ้นส่วนของไฟล์ข้อมูลนั้น เป็นไปได้ทั้งการรับมาจากตัว Seeder และการแลกเปลี่ยนชิ้นส่วนของไฟล์ข้อมูลระหว่าง Peer ด้วยกันเอง

5. Client

Client คือ ผู้ที่ใช้งานโปรแกรมบิตทอเรนต์เพื่อดาวน์โหลดไฟล์ข้อมูลที่ตนเองต้องการ

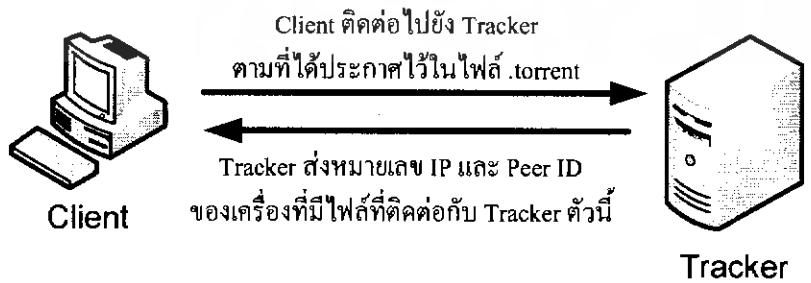
โดยการทำงานของโปรโตคอลบิตทอเรนต์ที่เป็นปกตินั้น จะมีขั้นตอนการทำงานดังต่อไปนี้

1. เริ่มต้อง Client จะต้องมีไฟล์ .torrent ของไฟล์ที่ต้องการเสียก่อน โดย Client จะต้องไปดาวน์โหลดไฟล์ .torrent ที่ต้องการจาก Torrent Server ดังรูปที่ 3.1



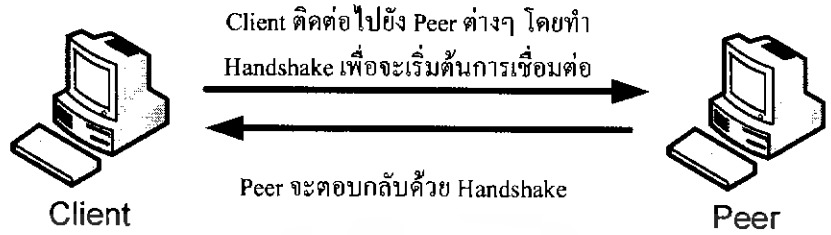
รูปที่ 3.1 การติดต่อกันระหว่าง Client และ Torrent Server

2. เมื่อ Client ได้ไฟล์ .torrent จาก Torrent Server มาแล้วก็จะทำการรันโปรแกรมบิตทอเรนต์ขึ้นมา จากนั้นClient ก็จะทำการติดต่อกับ Tracker ตามที่ประกาศเอาไว้ในไฟล์ .torrent ที่ได้โหลดมา จากนั้นตัว Tracker จะส่งหมายเลขไอพีและ Peer ID ของผู้ที่มีไฟล์ที่เครื่อง Client ต้องการกลับมาให้ Client ดังรูปที่ 3.2



รูปที่ 3.2 การติดต่อกันระหว่าง Client กับ Tracker

3. เมื่อ Client ได้หมายเลขไอพีและ Peer ID มาแล้ว ก็จะทำการติดต่อไปยังแต่ละ Peer ตามหมายเลขไอพีที่ได้มา โดยจะส่งข้อความไปทำ Handshake กับ Peer นั้นๆ ดังรูปที่ 3.3

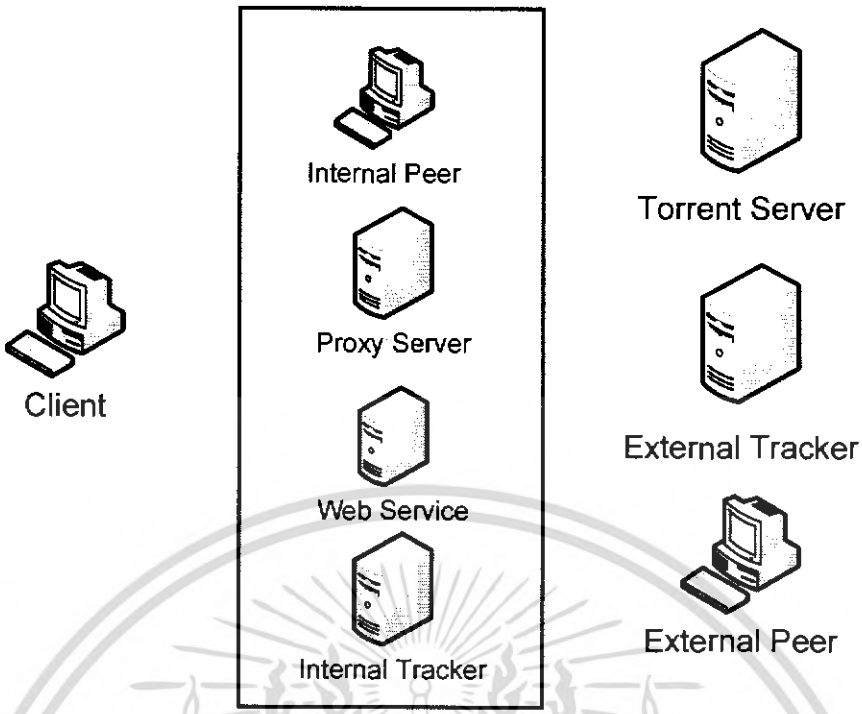


รูปที่ 3.3 การติดต่อกันระหว่าง Client กับ Peer

4. จากนั้น Client ก็จะทำการดาวน์โหลดและแลกเปลี่ยนชิ้นส่วนของไฟล์กับ Peer หรือ Seed ที่ติดต่อได้ ระหว่างนั้น Client ก็จะทำการติดต่อกับ Tracker เป็นระยะๆ เพื่ออัพเดตข้อมูลของตัวเองกับ Tracker และรับข้อมูลใหม่ๆจาก Tracker

3.2 แบบจำลองของระบบแคชสำหรับโปรแกรมบิตทอเรนต์

จากความต้องการของโครงงานระบบแคชสำหรับโปรแกรมบิตทอเรนต์นี้ คือ การลดการเชื่อมต่อจากเครือข่ายภายในที่มีการใช้โปรแกรมบิตทอเรนต์ออกไปยังภายนอกเครือข่าย ดังนั้นเราจำเป็นที่จะต้องจำลองระบบการทำงานของบิตทอเรนต์ขึ้นมาภายในเครือข่ายของเราเอง และจะต้องทำให้เวลาที่เครื่องภายในเครือข่ายของเราต้องการจะใช้งานโปรแกรมบิตทอเรนต์จะต้องผ่านระบบแคชของเรา ในส่วนนี้จะต้องนำหลักการของพร็อกซี่เซิร์ฟเวอร์มาใช้งาน ซึ่งจากการวิเคราะห์แล้วจึงทำแบบจำลองของระบบแคชสำหรับโปรแกรมบิตทอเรนต์เป็นดังรูป 3.4 ข้างล่างนี้



รูปที่ 3.4 แบบจำลองของระบบแคชสำหรับโปรแกรมบิตทอเรนต์

3.3 องค์ประกอบของระบบแคชสำหรับโปรแกรมบิตทอเรนต์

จากแบบจำลองของระบบแคชสำหรับโปรแกรมบิตทอเรนต์ที่ได้ออกแบบมานั้น สามารถแบ่งองค์ประกอบของระบบออกเป็น 4 ส่วนได้ดังนี้

3.3.1 Proxy Server

Proxy Server นี้ทำหน้าที่ filter URL ที่มี request ต้องการไฟล์ .torrent เมื่อพบแล้วจะทำการ Redirect ส่งไปยังตัวของ Web Service ซึ่งการทำงานของ Proxy Server นี้สามารถส่งให้ทำงานแบบ Transparent Proxy ได้ ขึ้นอยู่กับความต้องการและการคอนฟิก

Proxy Server นี้จะทำงานโดยใช้โปรแกรม Squid รวมกับโปรแกรม SquidGuard บนระบบปฏิบัติการลินุกซ์ ซึ่งขั้นตอนการติดตั้งและการคอนฟิกสำหรับการทำงานเป็นระบบแคชสำหรับโปรแกรมบิตทอเรนต์นั้นจะอยู่ในคู่มือการติดตั้ง

3.3.2 Web Service

Web Service นี้ทำหน้าที่ คือ ดาวน์โหลดไฟล์ .torrent ที่ Client ต้องการจะดาวน์โหลด โดยส่งผ่านมาทาง Proxy Server เมื่อดาวน์โหลดไฟล์เสร็จเรียบร้อยแล้ว ก็จะทำการ

เปลี่ยนค่า announce ของไฟล์ torrent ให้เป็น URL ของ Internal Tracker จากนั้นจึงส่งไฟล์ที่แก้ไขแล้วกลับไปยัง Client ผู้ที่ต้องการดาวน์โหลดไฟล์

จากนั้น Web Service นี้ก็จะทำการลงทะเบียนข้อมูลของไฟล์ torrent ที่แก้ไขแล้วลงใน Internal Tracker จากนั้นก็จะสั่งให้ Internal Peer ทำงาน

Web Service นี้เขียนขึ้นเองโดยใช้ภาษา PHP โดยในระบบแคชสำหรับโปรแกรมบิตทอเรนต์นี้ จะให้ Web Service ทำงานบนระบบปฏิบัติการลินุกซ์ ซึ่งรายละเอียดของ Source Code และวิธีการติดตั้งเพื่อใช้งานในระบบแคชสำหรับโปรแกรมบิดทอเรนต์นั้นจะอยู่ในคู่มือการติดตั้งและคู่มือโปรแกรมเมอร์

3.3.3 Internal Tracker

เป็นแทรคเกอร์ของระบบแคชสำหรับโปรแกรมบิตทอเรนต์ ทำหน้าที่คือเก็บข้อมูลของไฟล์ torrent ที่แก้ไขแล้วจาก Web Service และเก็บหมายเลขไอพีของ Internal Peer เมื่อ Client ใช้งานโปรแกรมบิตทอเรนต์ติดต่อมายัง Internal Tracker ตัว Internal Tracker ก็จะส่งหมายเลขไอพีของ Internal Peer กลับไปยัง Client ที่ติดต่อมา

Internal Tracker ที่ใช้ในระบบแคชสำหรับโปรแกรมบิตทอเรนต์นี้ ใช้ตัว Torrentpier ซึ่งเป็นแทรคเกอร์แบบ Forum โดยใช้ร่วมกับ phpBB2 สาเหตุที่เลือกใช้แทรคเกอร์ตัวนี้ก็เพราะว่ามีการกำหนดไม่ให้ใช้งาน passkey ได้ ซึ่งรายละเอียดการติดตั้งเพื่อใช้งานในระบบแคชสำหรับโปรแกรมบิตทอเรนต์นั้นจะอยู่ในคู่มือการติดตั้ง

3.3.4 Internal Peer

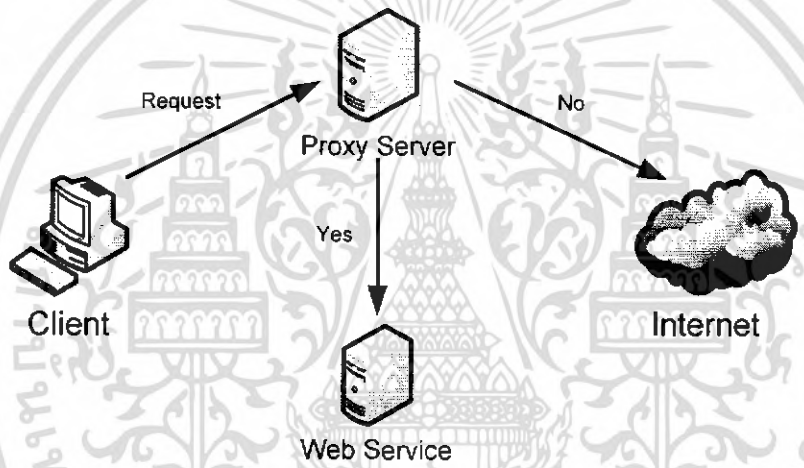
Internal Peer คือโปรแกรม Bittorent Client ที่ทำหน้าที่ดาวน์โหลดไฟล์ข้อมูลจากเครือข่ายภายนอก และทำหน้าที่เป็น Seeder ให้กับเครื่อง Client ภายในเครือข่ายที่ใช้งานโปรแกรมบิตทอเรนต์เพื่อดาวน์โหลดไฟล์

Internal Peer ที่ใช้ในระบบแคชสำหรับโปรแกรมบิดทอเรนต์นี้ ใช้โปรแกรมบิตทอเรนต์ สาเหตุที่เลือกใช้เพราะเป็นโอเพ่นซอร์ส ซึ่งใช้ภาษาไพตอนในการพัฒนา การติดตั้งและการคอนฟิกตัว Internal Peer สำหรับใช้งานโปรแกรมบิตทอเรนต์นั้นจะอยู่ในคู่มือการติตตั้ง

3.4 ขั้นตอนการทำงานของระบบแคชสำหรับโปรแกรมบิตทอเรนต์

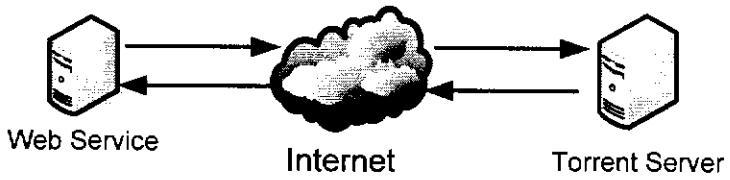
หลังจากที่ได้ทราบถึงแบบจำลองและองค์ประกอบต่างๆของระบบแคชสำหรับโปรแกรมบิตทอเรนต์แล้ว ในหัวข้อนี้จะกล่าวถึงขั้นตอนการทำงานของระบบแคชสำหรับโปรแกรมบิตทอเรนต์ ซึ่งมีขั้นตอนการทำงานดังนี้

1. ในระบบนี้เครื่องที่อยู่ภายในระบบเมื่อต้องการที่จะใช้งานอินเตอร์เน็ตก็จะต้องผ่าน Proxy Server ก่อน ดังนั้นเมื่อ Client ต้องการที่จะดาวน์โหลดไฟล์นามสกุล .torrent จากอินเตอร์เน็ตนั้น ก็จะต้องผ่านตัว Proxy Server ก่อนซึ่งตัว Proxy Server นี้จะทำการตรวจดู URL ว่าเป็นการเรียกไฟล์ .torrent หรือไม่ ถ้าเป็นการเรียกไฟล์ .torrent ก็จะ redirect URL นั้นไปยังตัว Web Service แต่ถ้าไม่ใช่การเรียกไฟล์ .torrent ก็จะปล่อยไป ซึ่งแสดงดังรูปที่ 3.5 ดังต่อไปนี้



รูปที่ 3.5 การทำงานของ Proxy Server

2. เมื่อ Proxy Server ตรวจสอบแล้วว่าเป็นการเรียกไฟล์ .torrent ก็จะส่ง URL ไปยัง Web Service ซึ่ง Web Service เมื่อได้รับ URL มาแล้วก็จะทำการดาวน์โหลดไฟล์ .torrent ที่ถูก redirect มายังเครื่องของตัวเอง เมื่อดาวน์โหลดมาแล้วก็จะทำการตรวจสอบค่า infohash ของไฟล์ .torrent ที่ดาวน์โหลดมากับฐานข้อมูลของ Internal Tracker ว่ามีไฟล์นี้อยู่ใน Internal Tracker แล้วหรือยัง ถ้าพบว่ามีอยู่แล้วก็จะส่งไฟล์ที่เคยแก้ไขแล้วกลับไปยัง Client ผู้ร้องขอ ถ้าไม่พบก็จะทำการทำสำเนา (copy) ไฟล์ต้นฉบับไว้ แล้วทำการแก้ไขไฟล์ .torrent ที่ทำการสำเนาไว้ โดยเปลี่ยนการประกาศ URL Tracker เป็น URL ของ Internal Tracker แทน ดังรูปที่ 3.6 ถึง 3.8



Web Service จะทำการดาวน์โหลดไฟล์ .torrent จาก Torrent Server
มาไว้ในเครื่อง แล้วทำการเปลี่ยนค่า URL Tracker ภายในไฟล์

รูปที่ 3.6 การทำงานของ Web Server

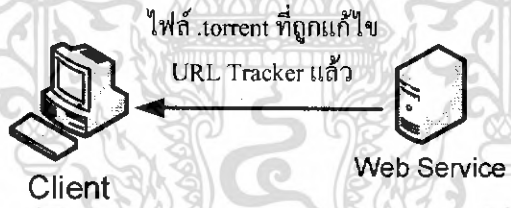
```
d8:announce41:http://tracker.bitcomet.net:8080/announce10:created by13:BitComet/0.6013
```

รูปที่ 3.7 URL ของ Tracker จากไฟล์ต้นฉบับ

```
d8:announce43:http://161.246.5.225/phpBB2/bt/announce.php10:created by13:BitComet/0.6013
```

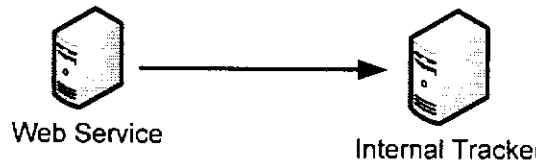
รูปที่ 3.8 URL ของ Tracker จากไฟล์ที่ถูกแก้ไขโดย Web Service

3. เมื่อ Web Service ทำการแก้ไขไฟล์ .torrent ที่ดาวน์โหลดมาเรียบร้อยแล้ว ก็จะส่งไฟล์ .torrent ที่แก้ไขแล้ว กลับไปยัง Client ที่ร้องขอมา ดังรูปที่ 3.9



รูปที่ 3.9 การส่งไฟล์ .torrent ที่แก้ไขแล้วกลับไปยัง Client

4. จากนั้น Web Service ก็จะติดต่อกับ Internal Tracker เพื่อลงทะเบียนไฟล์ .torrent ที่แก้ไขแล้วลงฐานข้อมูล Internal Tracker เพื่อให้ Internal Tracker มีข้อมูลสำหรับการติดต่อกับ Client ดังรูปที่ 3.10 และ 3.11



ติดต่อ Internal Tracker เพื่อลงทะเบียนไฟล์
.torrent ที่แก้ไขแล้ว

รูปที่ 3.10 การติดต่อกันของ Web Server กับ Internal Tracker เพื่อลงทะเบียนไฟล์ .torrent

←T→	torrent_id	info_hash	post_id	poster_id	topic_id	attach_id	size	piece_length	reg_time	complete_count	seeder_last_seen	last_seed
	1	????oF? qO+ ?□	4	4	3	1	16748964	4194304	1117378225	0	0	
	2	- ??G ?O□ ?-~?±□	5	2	4	2	2554734	32768	1138362456	0	1138382563	
	3	\0?}?%?????□A□????& o	6	2	5	3	356352	32768	1138622200	0	1138523477	
	7	●@?? E' '?U?	9	2	8	6	2910782	32768	1139314022	0	0	

รูปที่ 3.11 ตารางฐานข้อมูลของ Internal Tracker ที่ได้ลงทะเบียนไฟล์ .torrent แล้ว

5. เมื่อลงทะเบียนไฟล์ .torrent กับ Internal Tracker แล้ว Web Service ก็จะเพิ่มข้อมูล seeder ให้กับฐานข้อมูลของ Internal Tracker เพื่อเป็นข้อมูลให้ Internal Tracker ส่งกลับไปเมื่อ Client ติดต่อมา โดยข้อมูลของ seeder ที่เพิ่มเข้าไปนั้นก็คือหลายเลขไอพี พอร์ตที่ใช้ในการเชื่อมต่อและ Peer ID ของ Internal Peer นั้นเอง ดังรูปที่ 3.12 และ 3.13

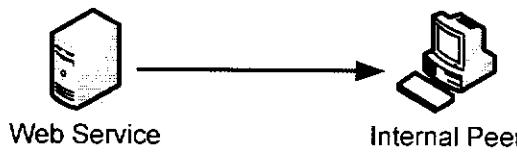
ติดต่อ Internal Tracker เพื่อเพิ่มข้อมูลของ
seeder ลงในฐานข้อมูล

รูปที่ 3.12 การติดต่อของ Web Service กับ Internal Tracker เพื่อเพิ่มข้อมูล seeder

←T→	torrent_id	peer_id	user_id	ip	port	uploaded	downloaded	complete_percent	seeder
	3	-BC0062-£³ÇÚ□ ¬"¦ ÜÃ£"	-1	a1f605e1	20889	0	0	0	1

รูปที่ 3.13 รายละเอียดของ Peer และ Seeder ในฐานข้อมูลของ Internal Tracker

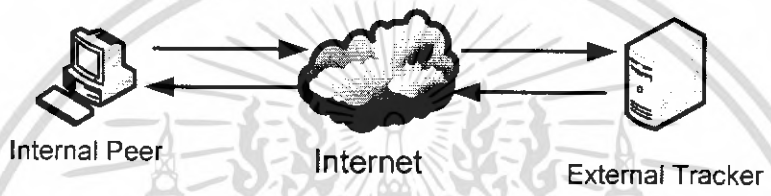
6. จากนั้น Web Service ก็จะทำการสั่งให้โปรแกรมบิตทอเรนต์ในเครื่อง Internal Peer ทำงาน โดยให้ Internal Peer ทำงานจากไฟล์ .torrent ที่เป็นไฟล์ต้นฉบับ เพื่อไปดาวน์โหลดชิ้นส่วนของไฟล์ข้อมูลจากอินเตอร์เน็ต ดังรูปที่ 3.14



สั่งให้ Internal Peer ทำงานจากไฟล์ .torrent ต้นฉบับ

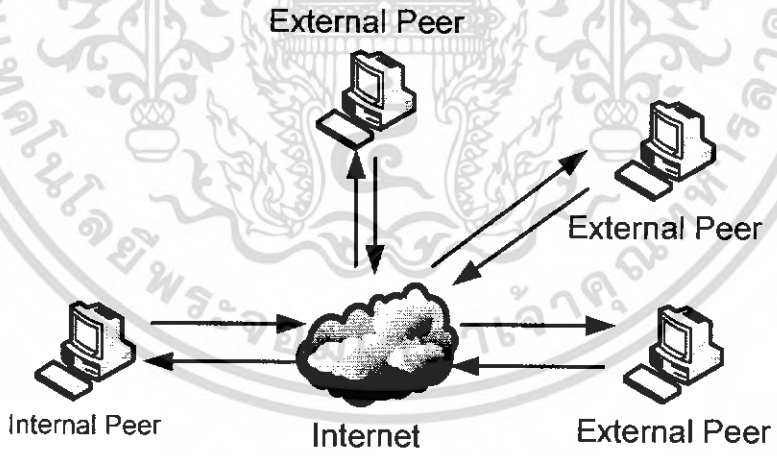
รูปที่ 3.14 การสั่งให้ Internal Peer ทำงานจาก Web Service

7. เมื่อ Internal Peer ถูกสั่งให้ทำงาน ก็จะไปติดต่อกับ External Tracker เพื่อรับข้อมูลของผู้ที่ไฟล์ที่ต้องการมา แล้วทำการเชื่อมต่อกับ External Peer เพื่อแลกเปลี่ยนชิ้นส่วนไฟล์ข้อมูลระหว่างกัน ดังรูปที่ 3.15 และ 3.16



Internal Peer จะติดต่อกับ External Tracker เพื่อรับข้อมูลของ Peer และ Seeder ที่มีไฟล์ข้อมูลที่ต้องการ

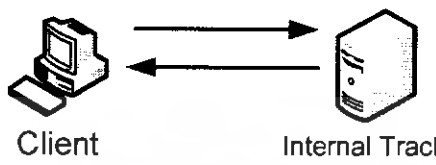
รูปที่ 3.15 การติดต่อระหว่าง Internal Peer กับ External Tracker



Internal Peer จะติดต่อกับ External Peer
เพื่อแลกเปลี่ยนชิ้นส่วนไฟล์ข้อมูลระหว่างกัน

รูปที่ 3.16 การติดต่อกันระหว่าง Internal Peer กับ External Peer

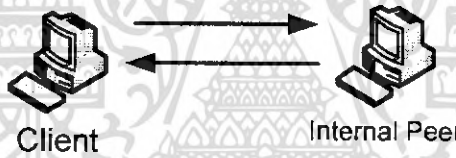
8. ส่วน Client นั้นเมื่อสั่งให้โปรแกรมบิตทอเรนต์ทำงานจากไฟล์ .torrent ที่ถูกแก้ไขแล้ว ก็จะไปติดต่อกับตัว Internal Tracker ซึ่งเมื่อติดต่อกับ Internal Tracker ได้แล้วก็จะได้ข้อมูลของ seeder มา ซึ่งข้อมูลนี้ก็เป็นข้อมูลของตัว Internal Peer ที่ Web Service ได้ทำการเพิ่มลงฐานข้อมูลของ Internal Tracker ดังรูปที่ 3.17



Client ติดต่อกับ Internal Tracker เพื่อรับข้อมูล seeder

รูปที่ 3.17 การติดต่อระหว่าง Client กับ Internal Tracker

9. เมื่อได้ข้อมูลของ seeder มาแล้ว Client ก็จะทำการติดต่อไปยัง Internal Peer เพื่อที่จะดาวน์โหลดชิ้นส่วนของไฟล์ข้อมูลที่ต้องการมายังเครื่องของตัวเอง ดังรูปที่ 3.18



Client ติดต่อกับ Internal Peer เพื่อดาวน์โหลดไฟล์ข้อมูล

รูปที่ 3.18 การติดต่อระหว่าง Client กับ Internal Peer

บทที่ 4

การทดลองและผลการทดลอง

ในบทนี้จะกล่าวถึงแบบจำลองและผลที่ได้จากการทดลอง ซึ่งแสดงความสามารถในการจำกัดการใช้งานเครือข่ายสู่ภายนอก โดยส่งผลกระทบต่อการทำงานของโปรแกรมบิตทอเรนต์น้อยที่สุด โดยทดสอบดูประสิทธิภาพการทำงานผ่านระบบแคชเทียบกับการทำงานแบบปกติของโปรแกรมบิตทอร์เรนท์ รวมถึงการลดจำนวนการรับส่งข้อมูลที่เกิดจากการโหลด นอกจากนี้ยังทดสอบการทำงานว่าเป็นไปตามที่ได้ออกแบบไว้หรือไม่ ปัญหาที่พบและการแก้ไขได้บันทึกไว้ที่ช่วงท้ายของบทนี้

4.1 แบบจำลองและคุณสมบัติของอุปกรณ์ที่ใช้ทดลอง

เนื่องจากข้อจำกัดด้านทรัพยากร ความแตกต่างของประสิทธิภาพเครื่อง ในการทดสอบนี้แบบจำลองใช้เครื่องคอมพิวเตอร์ทั้งหมดสี่เครื่องตั้งอยู่บนเครือข่ายเดียวกัน และอีกหนึ่งเครื่องอยู่คนละเครือข่าย โดยมีหน้าที่แตกต่างกันไป ซึ่งก็มี เครื่องที่ทำหน้าที่เป็นแคชคอยโหลดงาน Torrent เมื่อเครื่องภายในเครือข่ายเรียกไฟล์มาใช้ จะตั้งเป็นพร็อกซี่ของเครื่องภายในเนื่องจากอยู่ในเครือข่ายเดียวกัน เครื่องภายในทั้งสามที่ทำหน้าที่โหลด โดยทั้งสามเครื่องนี้เป็นเครื่องในเครือข่ายเครือข่ายหนึ่ง ที่ติดต่อผ่านเครื่องแคชเวลามีการโหลดไฟล์ที่เป็น Metadata และเครื่องที่อยู่ภายนอกเครือข่าย ทำหน้าที่ปล่อยไฟล์ให้กับเครือข่าย ทำงานเป็นแทรคเกอร์ของงานแลกเปลี่ยนไฟล์

- คุณสมบัติของเครื่องที่ทำงานเป็นแคช
    - Central Processing Unit: Intel® Pentium® 4 CPU 3.2 GHz
    - Main board: ASUS® P4P800 Special Edition
    - Memory: 1024 MB Dual Channel at 800 MHz
    - Operating System: Fedora Core 4 Linux, Kernel 2.6
    - Network Adapter: SMC EZ Card 10/100 PCI (SMC1211 series), connected at 100.0 Mbps
    - IP Address: 161.246.7.191
    - BitTorrent Client: BitTorrent 4.4
    - Web Server: Lampp Web Service pack for Linux. Included: Apache, MySql and PhPMyAdmin

- คุณสมบัติของเครื่องที่ทำงานเป็นเครื่องภายนอก
  - Central Processing Unit: AMD Athlon™ XP 2200+ 1.8 GHz
  - Main board: ASUS® A7V8X-X
  - Memory: 768 MB
  - Operating System: Microsoft® Windows® XP Professional, Service Pack 2
  - Network Adapter: VIA Rhine 2 Fast Ethernet Adapter, connected at 100.0 Mbps
  - IP Address: 161.246.5.120
  - BitTorrent Client: BitComet 0.59]
  - Web Server: Xampp Web Service pack for Windows. Included: Apache, MySql and PhPMyAdmin

- ไฟล์ที่ใช้ทดสอบ
  - ไฟล์ขนาด 11.3 MB โดยมี Metadata เป็น test10.torrent ขนาด 7.30 KB ตั้งให้ดาวน์โหลดไว้ที่ http://161.246.5.120/test10.torrent
  - ไฟล์ขนาด 53.04 MB โดยมี Metadata เป็น test50.torrent ขนาด 8.59 KB ตั้งให้ดาวน์โหลดไว้ที่ http://161.246.5.120/test50.torrent
  - ไฟล์ขนาด 92.48 MB โดยมี Metadata เป็น test100.torrent ขนาด 14.7 KB ตั้งให้ดาวน์โหลดไว้ที่ http://161.246.5.120/test100.torrent
  - ไฟล์ขนาด 200.27 MB โดยมี Metadata เป็น test200.torrent ขนาด 15 KB ตั้งให้ดาวน์โหลดไว้ที่ http://161.246.5.120/test200.torrent
  - ไฟล์ขนาด 386.01 MB โดยมี Metadata เป็น test400.torrent ขนาด 31 KB ตั้งให้ดาวน์โหลดไว้ที่ http://161.246.5.120/test400.torrent

ไฟล์ทุกไฟล์เปิดงาน Torrent โดยใช้เครื่องที่อยู่ภายนอก ทำงานภายใต้ BitComet 0.59 เครื่องที่อยู่เครื่องภายนอก ทำหน้าที่เป็น Tracker, Seeder และผู้ให้ Metadata files ไปด้วย เครื่องแคช ทำหน้าที่เป็น Gateway, Tracker, Seeder, Peer และผู้ให้ Metadata files เมื่อมีการ Redirect ไปด้วย เครื่องโหลด ทำหน้าที่เป็น Peer และผู้โหลดไฟล์ Metadata ทั้งโดยตรงและผ่านพร็อกซี่

4.2 การตั้งค่าต่างๆในการทดลอง

- เครื่องภายนอก

สร้าง Metadata ของแต่ละไฟล์ขึ้นมาตามที่กำหนด โดยใช้โปรแกรม BitComet สร้างไฟล์แบบรับแต่ Peers จากในแทรคเกอร์เท่านั้น กำหนด URL ไปที่ไอพีแอดเดรสของเครื่องภายนอก ทำการเก็บไฟล์ Metadata ไว้ที่โฟลเดอร์ htdocs ของโปรแกรม Xampp เพื่อให้สามารถเข้าถึงได้จากอินเตอร์เน็ต เมื่อเริ่ม Apache Web Service จากนั้น เอา Metadata ของแต่ละไฟล์ไปลงทะเบียนเข้ากับแทรคเกอร์

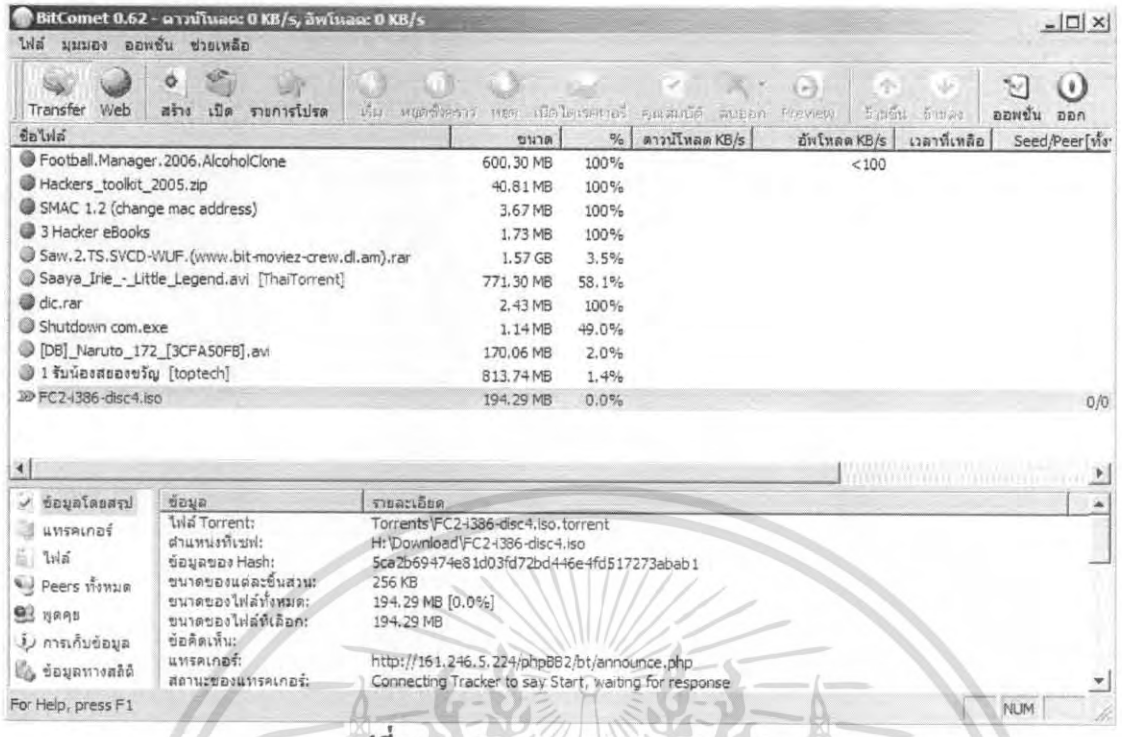
แทรคเกอร์ที่ใช้คือ Torrent Pier บน phpBB ซึ่งต้องติดตั้งเมื่อเริ่ม Apache Web Service และ MySQL กับ PhpMyAdmin แล้ว โดยติดตั้ง phpBB ก่อน เพื่อความสะดวกในการทดสอบ ทำการติดตั้ง phpBB และ Torrent Pier ที่โฟลเดอร์ htdocs เช่นกัน การลงทะเบียนกับแทรคเกอร์ที่ง่ายที่สุด คือการ Attach file ที่ Web Board Download ของ Torrent Pier ซึ่งจะมีการยืนยันการลงทะเบียนเข้ากับแทรคเกอร์โดยสมบูรณ์ ให้ลงทะเบียนให้ครบทั้งสามไฟล์ ทั้งนี้สามารถตรวจสอบการลงทะเบียนกับแทรคเกอร์ได้ โดยเรียก PhpMyAdmin ขึ้นมา เข้าไปดูฐานข้อมูลของ Torrent Pier ที่ตาราง Torrentfile หรือ bt_torrent

จากนั้นให้เรียก Metadata ทั้งสามไฟล์มาทำงานในโปรแกรม BitComet เพื่อทำการปล่อยไฟล์ให้กับเครือข่ายเมื่อมีการติดต่อเข้ามา ทั้งนี้ เพื่อป้องกันการติดต่อกับเครื่องภายในผ่าน UDP ให้ทำการปิด DHT Function ของโปรแกรม BitComet ออก และตั้งค่า Upload speed ให้สูงสุดไว้ เพื่อการทดสอบ

ปิด Firewall ของเครื่องออกเพื่อให้แทรคเกอร์สามารถติดต่อกับเครื่องอื่นๆได้ และการส่งชิ้นส่วนไฟล์ใน Bittorrent มีพอร์ตที่ไม่แน่นอน จึงเป็นการยากที่จะทำ Allow Access

- เครื่องภายใน

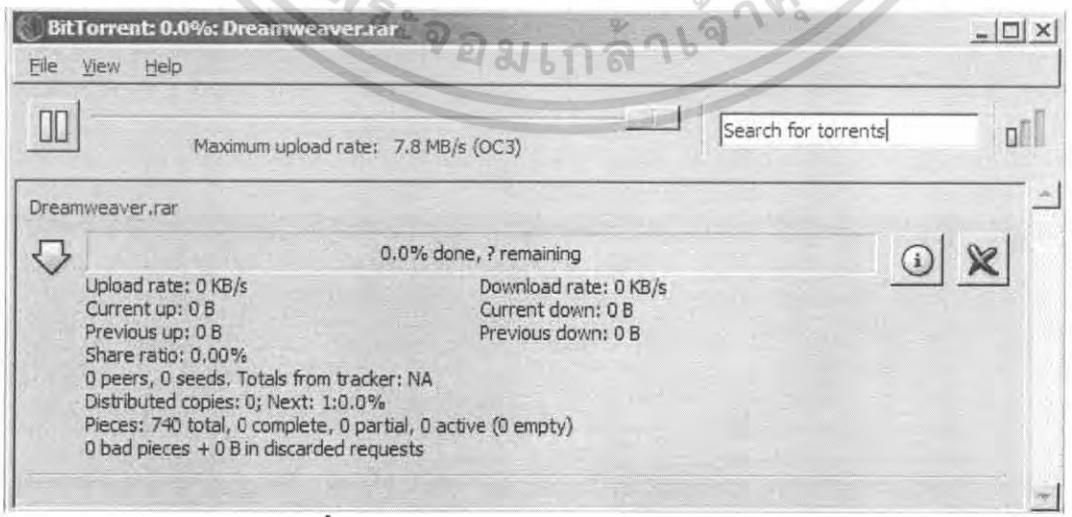
ทำการโหลดไฟล์ Metadata จากเครื่องภายนอกไว้ก่อน เพื่อใช้ทดสอบการทำงานแบบไม่ผ่านแคช จากนั้น ตั้ง Gateway ของเครื่องให้ผ่านแคช เพื่อเปิดระบบแคชสำหรับการทดลองอื่นๆ ปิดระบบ DHT function ของ BitComet ออกเพื่อป้องกันการโหลดจากแหล่งอื่นนอกเหนือจากการทดลอง



รูปที่ 4.1 User Interface ของ BitComet

- เครื่องแคช

เปิด Web Service ของ Lampp ขึ้นมา นำไฟล์ที่ต้องใช้ในการ redirect แก้ไขและโหลดไฟล์ metadata ไปติดตั้งไว้ที่โฟลเดอร์ htdocs ของโปรแกรมเพื่อเรียกใช้ได้สะดวกเมื่อ SquidGaurd redirect มาให้ ทำการเตรียมแทรคเกอร์ให้พร้อมโดยเปิด MySQL ให้พร้อมให้บริการ ทำการลบข้อมูลไฟล์ที่เคยโหลดไว้ก่อนออกจากแคช ซึ่งก็คือโฟลเดอร์ Original, Modified และ Return ใน htdocs/torrent แล้วเรียก BitTorrent ที่ได้ปรับแต่งไว้แล้วมาพร้อมทำงาน ติดตั้งโปรแกรม Ntop สำหรับตรวจจับการเชื่อมต่อไว้



รูปที่ 4.2 User Interface ของ BitTorrent 4.4.0

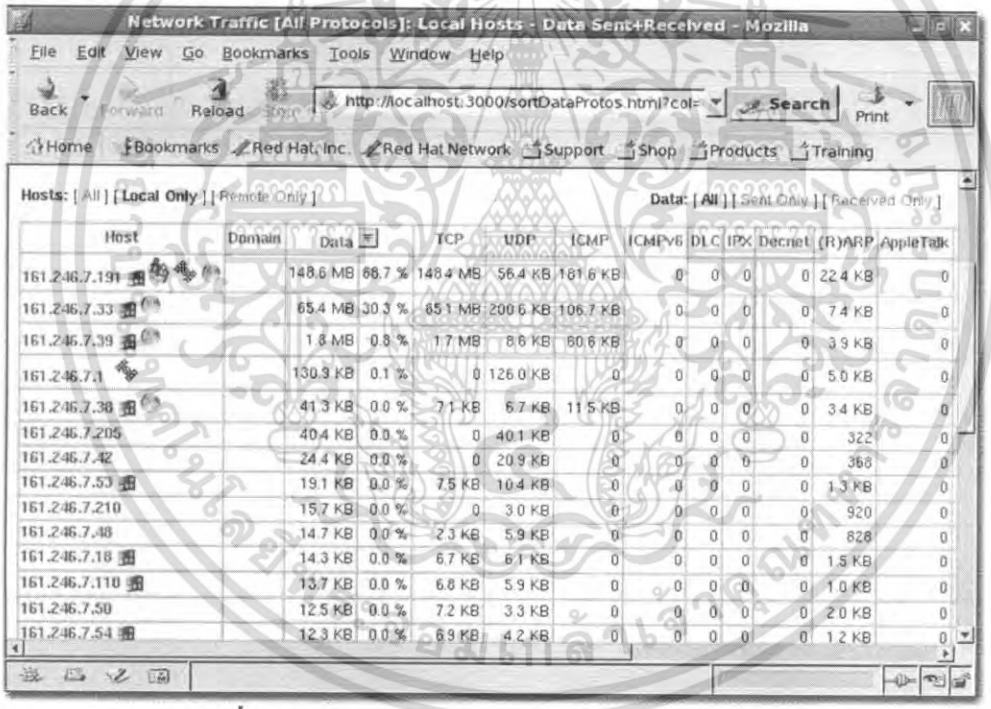
4.3 การทดลองและผลการทดลอง

4.3.1 การทดลองที่ 1 ประสิทธิภาพการลดจำนวนการรับส่งข้อมูลที่ติดต่อออกสู่ภายนอก

ทำการทดสอบว่า แบบจำลองที่ใช้สามารถลดการติดต่อออกภายนอกเครือข่ายได้แค่ไหน โดยเปรียบเทียบระหว่างการติดต่อโดยตรงและการติดต่อผ่านแคช

เนื่องจากจำนวนของเครื่องและจำนวนของไฟล์ที่ทำงานพร้อมกันมีผลต่อประสิทธิภาพของระบบพอๆกับ ขนาดของไฟล์ จึงแยกเป็นกรณีต่างๆ มาเปรียบเทียบความสามารถของระบบ

การทดสอบนี้ ใช้โปรแกรม Ntop ซึ่งมีความสามารถในการดูจำนวนข้อมูลและการรับส่งข้อมูลต่างๆ ของเครื่องภายในเครือข่ายที่ติดต่อผ่านอุปกรณ์ของเครื่องที่มีโปรแกรมนี้ โดยติดตั้งโปรแกรมไว้ที่เครื่องแคชที่อยู่กับ Gateway จึงสามารถตรวจสอบการรับส่งข้อมูลของเครื่องทั้งสามได้ โปรแกรม Ntop นี้ใช้ WinPCap เป็นตัวจับข้อมูล รายละเอียดของโปรแกรมนี้ สามารถดูได้ที่ http://www.ntop.org/



รูปที่ 4.3 หน้าจอการดูจำนวนการเชื่อมต่อของโปรแกรม Ntop

ทุกครั้งก่อนการทดสอบทุกกรณี จะมีการลบข้อมูลเก่าและเตรียมโปรแกรม Ntop ใหม่ โดยทำการลบงานที่โหลดออกจากโปรแกรม BitComet และหยุดการทำงานของ Ntop เพื่อลบสถิติที่มีอยู่แล้ว ก่อนการทดสอบกรณีที่โหลดพร้อมกับแคช จะทำการลบข้อมูลออกจากแทรคเกอร์และ BitTorrent และไฟล์ Metadata ที่อยู่ในเครื่อง

การทดสอบที่ 1.1 ดูขนาดการรับส่งข้อมูลของเครื่องสองเครื่อง ที่ทำงานผ่านระบบแคช ด้วยไฟล์ขนาดต่างๆ

ทำการเรียก Ntop ให้ทำงานก่อนเริ่มการโหลดข้อมูล ใช้เครื่องสองเครื่องในเครือข่ายโหลดไฟล์ metadata ผ่านแคชเพื่อเรียกให้แคชโหลดไปด้วย ระหว่างที่โหลด ให้ยุติการใช้งานเครือข่ายทั้งหมดเพื่อให้ได้ผลที่ใกล้เคียงกับการใช้งานที่เกิดจากการโหลดของ BitTorrent อย่างเดียว เก็บผลต่างๆเมื่อโหลดเสร็จ

ผลการทดสอบ

ตารางที่ 4.1 ขนาดของการรับส่งข้อมูลที่เกิดจากการติดต่อออกนอกเครือข่าย ภายใต้ระบบแคช

ขนาดไฟล์	ขนาดข้อมูลส่งออก	ขนาดข้อมูลรับเข้า
11.3 MB	218.2 KB	12.3 MB
53.04 MB	739.8 KB	56.6 MB
92.48 MB	2.1 MB	96.8 MB

สังเกตได้ว่า จำนวนข้อมูลขาออกที่เกิดขึ้น เป็นส่วนที่เกิดขึ้นจากการขอชิ้นส่วนจากเครื่องภายนอก และการติดต่อกับ Tracker ส่วนข้อมูลขาเข้า เป็นชิ้นส่วนไฟล์ที่ทำการโหลดมาจากเครื่องภายนอก ขนาดการโหลดที่เกิดขึ้นนั้น ใหญ่กว่าขนาดของไฟล์เล็กน้อย เพราะมีการติดต่อจาก Tracker และการติดต่อที่เกิดจาก Protocol ของเครือข่ายเอง

การทดสอบที่ 1.2 ดูขนาดการรับส่งข้อมูลของเครื่องสองเครื่อง ที่ทำงานตามการโหลดปกติ ด้วยไฟล์ขนาดต่างๆ

ลบข้อมูลเก่าในเครื่องทั้งสามเครื่อง เริ่มโปรแกรม Ntop เมื่อพร้อมที่จะทดสอบ โหลดไฟล์จากเครื่องภายนอกโดยตรงเพื่อทำงานแบบไม่ผ่านแคช แล้วเก็บข้อมูลการใช้งานเครือข่ายเมื่อโหลดเสร็จ

ผลการทดสอบ

ตารางที่ 4.2 ขนาดของการรับส่งข้อมูลที่เกิดจากการติดต่อออกนอกเครือข่าย โดยการทำงานปกติ

ขนาดไฟล์	ขนาดข้อมูลส่งออก	ขนาดข้อมูลรับเข้า
11.3 MB	558.4 KB	24.7 MB
53.04 MB	1.7 MB	114.2 MB
92.48 MB	4.9 MB	192.6 MB

จากตารางที่ 4.1 และ 4.2 เห็นได้ชัดว่า ความแตกต่างที่เกิดจากการโหลดในแบบไม่ใช้แคช กับใช้ระบบแคช คือจำนวนการใช้งานเครือข่ายทั้งขาเข้าและขาออก มีขนาดประมาณสองเท่าของขนาดไฟล์ที่โหลดจริง ซึ่งมาจากการโหลดไฟล์ของเครื่องทั้งสองเครื่องแยกกัน ไม่เหมือนในตอนที่หนึ่งที่เครื่องแคชเครื่องเดียวติดต่อออกภายนอก ซึ่งขนาดการรับส่งไฟล์ มากกว่าขนาดของไฟล์จริงเพียงเล็กน้อย

การทดลองที่ 1.3 ดูประสิทธิภาพการลดขนาดการใช้งานเครือข่ายออกภายนอกของระบบแคช เมื่อจำนวนเครื่องเพิ่มขึ้น

เพื่อดูว่า ประสิทธิภาพการลดการใช้งานเครือข่าย จะสามารถทำงานได้ดีแม้ว่าจะมีจำนวนเครื่องที่โหลดไฟล์หนึ่งหนึ่งมากขึ้น จึงให้เพิ่มจำนวนเครื่องที่ทำงานในเครือข่ายภายใต้ระบบแคช แล้วเก็บข้อมูลมาเปรียบเทียบ โดยทดลองที่สามเครื่อง และสี่เครื่อง

การทดลอง ลบค่าเก่าในเครื่องแคชออกเพื่อดูการโหลดไฟล์พร้อมกับเครื่องอื่นๆ ทำการเริ่ม Ntop เมื่อพร้อม แล้วให้ทั้งสามเครื่อง โหลดไฟล์ metadata ของไฟล์ขนาดประมาณ 100 MB ผ่านระบบแคช แล้วเริ่มทำงาน เก็บข้อมูลเมื่อการโหลดเสร็จสิ้น

ทำซ้ำอีกรอบโดยเพิ่มจำนวนเครื่องเป็นสี่เครื่อง

นำผลการโหลดไฟล์ขนาด 100 MB จากการทดลองในตอนที่หนึ่งมาเปรียบเทียบ

ผลการทดสอบ

ตารางที่ 4.3 ขนาดของการรับส่งข้อมูลที่เกิดจากการติดต่อออกนอกเครือข่าย เปรียบเทียบจำนวนเครื่อง โดยใช้ไฟล์ขนาด 100 MB ผ่านระบบแคช

จำนวนเครื่อง	ขนาดข้อมูลส่งออก	ขนาดข้อมูลรับเข้า
2	2.1 MB	96.8 MB
3	2.0 MB	96.2 MB
4	2.2 MB	92.9 MB

การทดลองที่ 1.4 ดูประสิทธิภาพการใช้งานเครือข่ายออกภายนอกแบบปกติ เมื่อจำนวนเครื่องเพิ่มขึ้น

ทำการทดลองเหมือนกับการทดสอบที่ 1.3 แต่ให้ใช้การโหลดโดยตรงจากเครื่องภายนอก เพื่อไม่ต้องทำงานผ่านระบบแคช แล้วนำผลการโหลดไฟล์ขนาด 100 MB จากการทดลองในตอนที่ 2 มาเปรียบเทียบ

ผลการทดสอบ

ตารางที่ 4.4 ขนาดของการรับส่งข้อมูลที่เกิดจากการติดต่อออกนอกเครือข่าย เปรียบเทียบจำนวนเครื่อง โดยใช้ไฟล์ขนาด 92.48 MB โดยการทำงานแบบปกติ

จำนวนเครื่อง	ขนาดข้อมูลส่งออก	ขนาดข้อมูลรับเข้า
2	4.9 MB	192.6 MB
3	6.7 MB	286.3 MB
4	9.3 MB	354.2 MB

จากตารางที่ 4.3 และ 4.4 สังเกตได้ว่า จำนวนเครื่องนั้น แทบจะไม่ส่งผลต่อการใช้งานเครือข่ายของ BitTorrent ภายใต้การทำงานของแคชเลย ที่เป็นเช่นนี้ก็เพราะเครื่อง Internal Peer ของระบบจะเป็นเครื่องเดียวที่ติดต่อออกนอกเครือข่าย ทำให้ขนาดการส่งและรับข้อมูล มากกว่าขนาดของไฟล์เพียงเล็กน้อย (จากความคลาดเคลื่อนที่มาจาก Protocol ต่างๆของเครือข่าย) ซึ่งในการทำงานแบบปกติของ BitTorrent ขนาดของการใช้งานเครือข่ายโดยเฉพาะที่ข้อมูลเข้า มีขนาดใกล้เคียงกับผลคูณของขนาดของไฟล์ที่โหลดกับจำนวนเครื่อง ซึ่งแสดงให้เห็นอย่างชัดเจนว่าระบบแคชสามารถลดการใช้งานเครือข่ายได้มากขนาดไหนเมื่อนำมาเปรียบเทียบกัน

4.3.2 การทดลองที่ 2 เปรียบเทียบการโหลดไฟล์ขนาดต่างๆกัน

การทดสอบประสิทธิภาพของการโหลดไฟล์ ที่เปลี่ยนแปลงไประหว่างการใช้และไม่ใช้แคช และข้อได้เปรียบของแคชที่เก็บไฟล์ไว้ให้โหลดซ้ำได้ทันที ผลการทดลองที่ได้ในตาราง ใช้ผลเฉลี่ยของการทดลองสามครั้ง ในบางการทดลองมีปัญหาทำให้ผลการทดลองผิดพลาดไปหลายวินาที ซึ่งบางกรณีนั้นมากกว่าผลอื่นๆที่ได้จากการทดลองเดียวกัน แต่เนื่องจากทราบสาเหตุที่แน่นอนว่ามาจากปัญหา Endgame จึงตัดผลการทดลองที่ติดปัญหา Endgame ออกในภายหลัง เนื่องจากได้พบวิธีแก้ระหว่างการทดลอง

ตอนที่ 1 การโหลดโดยไม่ใช้แคช

ให้เครื่องที่ทำหน้าที่เป็นเครื่องภายในเครือข่าย เรียก Metadata ที่เก็บไว้ออกมาทำงาน เพราะไฟล์ metadata เหล่านี้ไม่ได้ถูกโหลดผ่านพร็อกซี่จึงทำให้โปรแกรม BitComet ติดต่อไปยังเครื่องภายนอกโดยตรง ทำการโหลดทีละไฟล์เพื่อให้ประสิทธิภาพสูงสุด

ผลการทดสอบ

```
8:announce43:http://161.246.5.120/phpBB2/bt/announce.php10:created by13:BitComet/0.59
```

รูปที่ 4.4 URL ของ Tracker ที่ใช้โหลดโดยไม่ผ่านแคช

ตารางที่ 4.5 Client BitComet โหลดไม่ผ่านแคช

File Size	Max DL Rate	Avg DL Rate	Elapse Time
92.48MB	1279 KB/s	679 KB/s	2 min 21 s
200.27 MB	1294 KB/s	637 KB/s	5 min 22 s
386.01 MB	1304 KB/s	740 KB/s	8 min 54 s

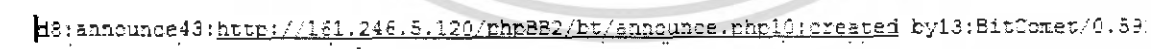
สิ่งที่น่าสนใจ มีบางช่วงที่ความเร็วตกลงจากสูงสุดไปยังหยุดนิ่ง เนื่องมาจากการใช้งานเครือข่ายระหว่างแต่ละลง ทำให้มีอุปสรรคการติดต่อต่างๆ ความเร็วการโหลดนั้น จะค่อยๆเพิ่มขึ้นจนถึงความเร็วสูงสุด ดังนั้นในไฟล์ที่มีขนาดเล็ก ความเร็วสูงสุดในการโหลดไฟล์จึงขึ้นไม่ถึงประสิทธิภาพสูงสุดที่ทำได้ อีกทั้งช่วง 30 MB แรก เป็นช่วงเร่งความเร็ว จึงทำให้ยิ่งโหลดนานอัตราเฉลี่ยจึงยิ่งสูงขึ้น

ตอนที่ 2 การโหลดโดยใช้แคช

เนื่องจากไฟล์เป็นไฟล์เดียวกัน ให้ทำการลบงานและไฟล์ที่โหลดมาแล้วทั้งสามไฟล์ออกจาก BitComet เพื่อเตรียมเครื่องให้พร้อมที่จะโหลดใหม่ การโหลดผ่านแคช ใช้ Internet Explorer ใส่ URL ของ metadata ไฟล์ตามที่กล่าวไว้เบื้องต้น SquidGuard จะทำการ Redirect ไปที่เว็บเซอร์วิสของแคชเอง ซึ่งตัวแคชจะเริ่มทำงานและโหลดไฟล์ไปก่อนที่เครื่องภายในเครือข่ายจะโหลด ขณะเดียวกันก็โหลด metadata ที่เครื่องภายในจะโหลดมาแก้ไขและ redirect ไฟล์ที่แก้ไขแล้วไปให้แทน และสร้าง Seeder หลอกขึ้นมาในแทรคเกอร์โดยชี้ไอพีแอดเดรสและพอร์ตไปที่ตัวโปรแกรม BitTorrent ของแคช เพื่อให้เครื่องภายในติดต่อกับแทรคเกอร์และโหลดจาก Bittorrent ของแคชเอง

ผลการทดสอบ

Cache



รูปที่ 4.5 URL ของ Tracker ที่แคชใช้ทำการโหลด

ตารางที่ 4.6 Cache BitTorrent โหลดพร้อมกับ CIient

File Size	Max DL Rate	Avg DL Rate	Elapse Time
92.48MB	1.2 MB/s	1203 KB/s	1 min 20 s
200.27 MB	1.3 MB/s	1207 KB/s	2 min 56 s
386.01 MB	1.3 MB/s	867 KB/s	7 min 25 s

สิ่งที่น่าสนใจ เครื่องแคชเริ่มทำงานโหลดไฟล์ตั้งแต่ช่วงที่เครื่องภายในยังขึ้น Dialog ถามว่าจะเก็บไฟล์ metadata ไว้ที่ไหน ซึ่งทำให้สามารถเริ่มทำการโหลดไปล่วงหน้าได้ เนื่องจากแคชนั้น อยู่ในสถานการณ์เดียวกับเครื่องภายในในการทดสอบแรก ความเร็วในการโหลดของแคชเป็นที่น่าพอใจกว่ามาก ทั้งนี้อาจมีส่วนมาจากประสิทธิภาพของเครื่อง ความแตกต่างของโปรแกรม BitTorrent และการใช้งานเครือข่าย แต่ในไฟล์ขนาดใหญ่ ก็มีช่วงที่เกิดความเร็วตกเช่นกัน

Client BitComet

d8:announce43:http://161.246.5.224/phpBB2/bt/announce.php10:created by13:BitComet/0.53

รูปที่ 4.6 URL ของ Tracker ที่ใช้โหลดผ่านแคช

ตารางที่ 4.7 Client BitComet โหลดพร้อมกับแคช

File Size	Max DL Rate	Avg DL Rate	Elapse Time
92.48MB	1407 KB/s	1155 KB/s	1 min 22 s
200.27 MB	1340 KB/s	1159 KB/s	2 min 57 s
386.01 MB	1887 KB/s	847 KB/s	7 min 27 s

สิ่งที่น่าสนใจ โปรแกรม BitComet นั้นแสดงผลว่าติดต่อกับเครื่องแคชอย่างเดียว และ URL ของแทรคเกอร์ชี้ไปที่เครื่องแคชตามที่ได้ออกแบบไว้ การติดต่อในช่วงแรกเริ่มต้นช้ากว่าการติดต่อไม่ผ่านแคช ทั้งนี้มาจากการทำ Handshake กับไคลเอนต์ที่ต่างชนิดกัน การติดต่อกับฐานข้อมูลของแทรคเกอร์ภายในที่เพิ่งทำการอัพเดต เมื่อเริ่มโหลดไฟล์ Metadata การที่ตัวแคชยังไม่มีข้อมูลที่จะส่งให้เครื่องภายใน และการใช้งานเครือข่ายและทรัพยากรของเครื่องแคชที่ต้องรับข้อมูลจากเครื่องภายนอก และส่งข้อมูลให้เครื่องนี้ไปพร้อมๆกัน แต่ในภายหลัง เมื่อเครื่องแคชทำการโหลดเสร็จสิ้น ความเร็วในการโหลดก็กลับมาเร็วขึ้นมาก ซึ่งความเร็วสูงสุดในการโหลดนั้น สูงกว่าไม่ผ่านแคช

ตอนที่ 3 การโหลดไฟล์เดิมซ้ำผ่านแคช

ทำการลบงานและไฟล์ที่โหลดมาแล้วออกจาก BitComet ของเครื่องภายใน แล้วโหลด metadata ไฟล์อีกครั้งเช่นเดียวกับในตอนที่ 2 แต่ที่จะต่างกันก็คือ ตัวแคชจะตรวจสอบว่างานนี้มีอยู่แล้วหรือไม่ โดยตรวจ Info_Hash ของไฟล์ Metadata เทียบกับฐายข้อมูลของแทรคเกอร์ ซึ่งถ้าพบว่ามีอยู่แล้ว แคชจะไม่แก้ไขฐานข้อมูล เพื่อสร้าง seeder ปลอมอีก และงานโหลดนั้นยังทำงานอยู่ใน Bittorrent ของแคช

ผลการทดสอบ

```
d8:announce43:http://161.246.5.224/phpBB2/bt/announce.php10:created by13:BitComet/0.59
```

รูปที่ 4.7 URL ของ Tracker ที่ใช้โหลดผ่านแคช

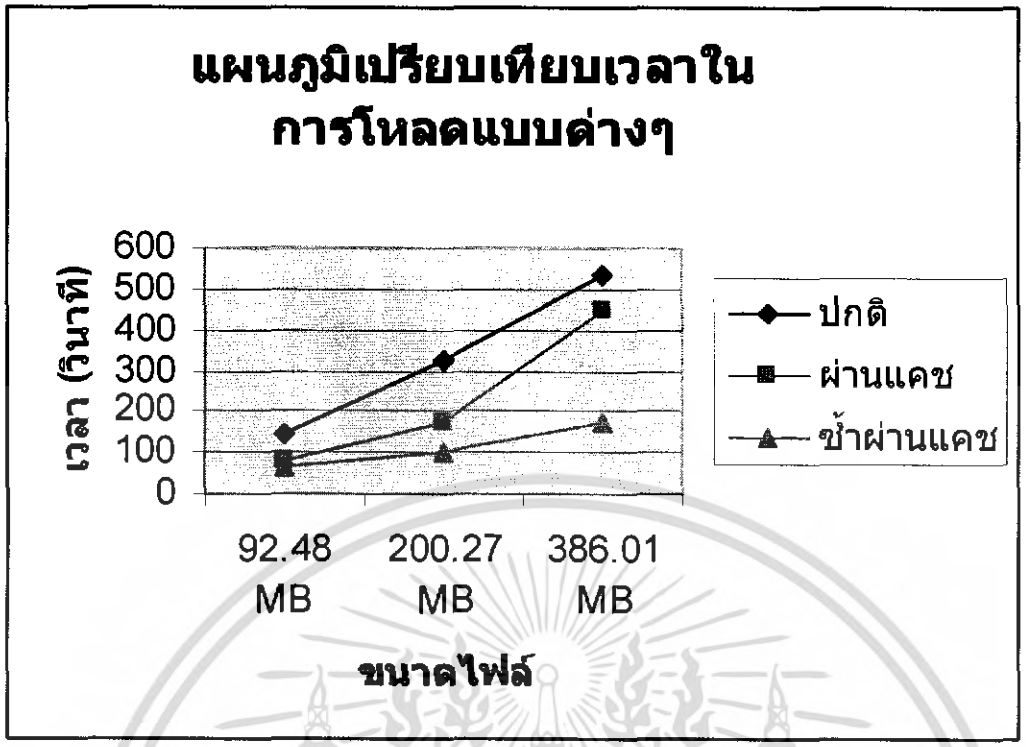
ตารางที่ 4.8 Client BitComet โหลดไฟล์จากแคช

File Size	Max DL Rate	Avg DL Rate	Elapse Time
92.48MB	2715 KB/s	1504 KB/s	1 min 3 s
200.27 MB	2937 KB/s	1973 KB/s	1 min 44 s
386.01 MB	3027 KB/s	2286 KB/s	2 min 53 s

สิ่งที่น่าสนใจ ตัวแคชทำงานตามที่ออกแบบไว้ คือไม่ทำการสร้าง Peer ใหม่ในฐานข้อมูลของแทรคเกอร์ภายในและไม่เรียกงานซ้ำซ้อนขึ้นมาทำ ความเร็วในการโหลดจากแคชนั้นเป็นไปเร็วกว่าการโหลดโดยตรง หรือเริ่มโหลดพร้อมแคชมาก

จากตารางที่ 4.6 และ 4.9 จะเห็นความแตกต่างของความเร็วในการโหลดได้ชัดเจน ในกรณีที่มีระบบแคช และมีไฟล์อยู่แล้ว การโหลดไฟล์จะทำได้อย่างรวดเร็วมาก เนื่องจากไฟล์นั้นอยู่บนเครื่องในเครือข่าย ทำให้สามารถส่งไฟล์ได้โดยตรง ขณะที่การทำงานแบบปกติ ที่ไม่มีระบบแคชจะต้องโหลดจากภายนอกทุกครั้ง แม้ว่าไฟล์จะซ้ำกันก็ตาม ส่งผลให้ความแตกต่างของทั้งสองตาราง มีค่าอย่างมาก

เมื่อเปรียบเทียบกรณีต่างๆ เฉพาะการทำงานของเครื่องภายในจะพบว่า ผลกระทบที่เกิดจากการนำระบบแคชมาใช้ มีผลเล็กน้อยมาก ซึ่งมาจากการติดต่อทั้งโหลดจากเครื่องภายนอก และการส่งให้เครื่องภายใน ทำให้การโหลดช้าในการโหลดพร้อมกับเครื่องภายใน แต่การโหลดช้านั้นแทบจะไม่ส่งผลใดๆเลย สำหรับวิธีลดปัญหาการโหลดช้าในการโหลดพร้อมกันกับเครื่องภายในสามารถทำได้โดยเพิ่มจำนวนเครื่องที่เป็นแคช เพื่อลดภาระในการโหลดงานของเครื่องแคช



รูปที่ 4.8 แผนภูมิเปรียบเทียบเวลาการโหลดของเครื่องภายในกรณีต่างๆ

4.4 ปัญหาที่พบระหว่างการทดลองและแนวทางแก้ไข

ปัญหา DHT Function ที่ทำให้ Peer สามารถค้นพบ Peer ตัวอื่นได้โดยไม่ต้องพึ่งแทรคเกอร์ เนื่องจากสภาพการทดสอบแตกต่างจากสภาพจริงคือตำแหน่งการวางเครื่องในเครือข่ายในสภาพการทำงานจริง จะมี Firewall และ Gateway อยู่ ซึ่งสามารถหยุดการค้นหา DHT ได้โดนการปิด UDP Packet ที่ส่งออกนอกเครือข่าย ดังนั้นปัญหานี้ในสภาพการทดสอบ แก้ไขโดยการปิด DHT Function ของตัวเครื่องภายนอกออก รวมถึงสร้างไฟล์ Metadata ที่ไม่รองรับ DHT ข้อแตกต่างระหว่าง Metadata ที่รองรับ DHT และไม่รองรับ DHT คือ แบบที่รองรับ DHT จะมีไอพีแอดเดรสของ DHT Nodes พ่วงที่ท้ายไฟล์หลัง Info field ซึ่งสามารถให้ระบบแคช จัดการทิ้งไปได้

ปัญหา Endgame Download Speed ที่ทำให้การโหลดชะงักที่ 99% พบในโปรแกรมตัวที่ใช้เป็นแคช ทำให้ผลการทดสอบช่วงแรกๆ มีความคลาดเคลื่อน สภาพการทดสอบนั่น มีข้อจำกัดก็คือมี Seeder แค่ตัวเดียว ทำให้ปัญหานี้เกิดขึ้นเป็นประจำ เพราะการทำงาน Endgame คือ การชะลอการโหลด แล้วส่ง Request ของชิ้นส่วนสุดท้ายไปยังทุกๆ Peer ที่มีชิ้นชิ้นนี้ แล้วเมื่อได้รับชิ้นส่วนมาจาก Peer ใดๆ ก็จะส่ง Request_Cancel ไปยัง Peer ตัวอื่นๆ การที่มี Seeder ตัวเดียว ทำให้มีโอกาสที่ Seeder จะรอให้ติดต่อขอจาก Peer ตัวอื่นแทน แต่ในสภาพจริง โอกาสที่งานโหลดบางงานจะมี Seeder แค่ตัวเดียวก็เกิดขึ้นได้ จึงต้องแก้ไขที่โค้ดของโปรแกรมให้ทำงาน Endgame เมื่อเลยจุดที่ควรจะเริ่มไปสักพัก แล้วนำมาทดสอบใหม่

สาเหตุที่เลือกใช้ Clients สำหรับโหลด BitTorrent แต่ละเครื่องไม่เหมือนกัน

BitComet เป็นที่โจษขานกันว่า ละเมิดข้อตกลงของผู้พัฒนาโปรแกรมประเภทนี้ ซึ่งก็คือการละเลย Private Flag ของไฟล์ Torrent และการให้ DHT Trackerless mode ทำงานแม้จะหยุดในเมนูการปรับแต่งไปแล้วจึงไม่เลือกมาใช้เป็นแคช แต่เนื่องจากความเป็นที่นิยมจึงนำมาใช้เป็นตัวโหลดภายในเครือข่ายและภายนอก เนื่องจากในสถานการณ์จริงเป็นไปได้ยากที่เราจะไปจำกัดเครื่องในเครือข่ายเราไม่ให้ใช้โปรแกรมใดๆ ที่ไม่ต้องการได้

BitTorrent บน Linux เป็นโค้ดไพตอน ซึ่งสามารถปรับแต่งและแก้ไขโค้ดให้ทำงานตามที่ได้ออกแบบไว้ได้ ซึ่งจำเป็นอย่างมากในการทำเป็น Internal Peer เพราะจำเป็นต้องตั้ง Peer ID ที่ตายตัว และแก้ไขปัญหา Endgame Speed ที่อาจพบได้ใน BitTorrent Client บางตัว

4.5 เปรียบเทียบสภาพการทดลองและสภาพการทำงานจริง

เนื่องจากมีความแตกต่างของสภาพที่ใช้ทดลองกับสภาพการทำงานจริง จำเป็นต้องนำผลการทดลองและข้อที่ต้องคำนึงถึงในสภาพที่ต่างกันมาใช้เพื่อนำไปประยุกต์ เพื่อให้รู้ว่า ในการใช้งานจริง ระบบนี้จะมีประสิทธิภาพเช่นใด จะพบปัญหาใดได้บ้าง และระบบเก่ากับระบบแคชจะจัดการกับปัญหาได้อย่างไร

4.5.1 ปัญหาที่ไม่พบในสภาพการทดลอง

การล่มของแทรคเกอร์

ปัญหาการล่มของแทรคเกอร์นั้น พบได้บ่อยมากในสภาพการทำงานจริง ซึ่งในสภาพจริงนั้น ในระบบปัจจุบันมีการใช้งาน DHT Trackerless Torrent ซึ่งใช้ UDP Port ค้นหา Node และ Peer ตัวอื่นๆ แต่บางครั้ง DHT ก็ไม่ได้รับการสนับสนุนในโปรแกรมที่ต่างกันไป เช่น Azureus ใช้ DHT ที่ไม่เหมือนกับ BitComet และทั้งสองโปรแกรมก็เป็นที่นิยมใช้เช่นกัน ส่วนระบบแคชนั้นปัญหาการล่มของแทรคเกอร์ภายใน ขึ้นอยู่กับเครื่องที่ทำงานเป็นแทรคเกอร์ ซึ่งหากระบบเครือข่ายภายในไม่ล่ม และไม่มีปัญหาด้านไฟฟ้าหรืออุปกรณ์แทรคเกอร์ภายในก็มีโอกาสล่มน้อยมาก ส่วนการติดต่อกับแทรคเกอร์ภายนอกของ Internal Peer จะเป็นปัญหาที่แก้ไขได้ยาก เนื่องจาก Internal Peer จะถูกปิดการติดต่อ DHT ไว้เพื่อไม่ให้ส่ง Peers Table ไปให้เครื่องในเครือข่าย แต่ทั้งนี้ ระบบแคชยังสามารถปล่อยไฟล์ที่เครื่อง Internal Peer โหลดมาไว้ได้แม้จะไม่สามารถโหลดต่อได้ขณะที่แทรคเกอร์ภายนอกล่ม ไม่มี Seeder ที่สมบูรณ์

สภาพไม่มีSeeder ที่สมบูรณ์

ปัญหาสภาพไม่มี Seeder ที่สมบูรณ์ในสภาพจริงนั้นเกิดขึ้นบ่อยมาก โดยเฉพาะในไฟล์ขนาดใหญ่ หรือไฟล์ที่ไม่เป็นที่นิยม ซึ่งเครื่องที่ปล่อยไฟล์เป็นเครื่องแรกอาจปล่อยไม่สมบูรณ์ หรือไม่มีเครื่องอื่นที่โหลดโดยสมบูรณ์ก่อนที่เครื่องเครื่องนั้นจะหยุดแจกจ่ายไฟล์ นอกจากนี้กรณีที่เครื่องที่โหลดจนเกือบสมบูรณ์เกิดปัญหา Endgame ก็จะชะงักที่ 99% จึงไม่สามารถทำให้การโหลดเสร็จสิ้นได้ รวมถึงทำให้เครื่องอื่นๆเจอปัญหาเดียวกัน ทางแก้ของการทำงานปกติ คือพยายามเปิดเครื่องไว้จนกว่าไฟล์จะโหลดเสร็จ ซึ่งสิ้นเปลืองพลังงาน ระบบแคชที่ออกแบบมาไม่สามารถแก้ไขปัญหาที่เครื่องทั่วไปอาจพบเจอได้ แต่ในทางกลับกับ ระบบสามารถทำงานได้ตลอดเวลา (เนื่องจากถือเป็นส่วนหนึ่งของระบบเครือข่าย) ทำให้สามารถโหลดได้ทันทีที่มี Seeder ใหม่เข้ามาแจกจ่ายไฟล์ นอกจากนี้เครื่องภายในเครือข่ายไม่จำเป็นต้องเปิดไว้ตลอดเวลาเพื่อหา Seeder เนื่องจากระบบจะหาและโหลดมารอไว้ให้ ทำให้สามารถประหยัดการใช้งานได้ด้วย และเมื่อไฟล์ในระบบแคชสมบูรณ์ เครื่องภายในเครือข่ายจะไม่พบสภาพไม่มี Seeder สมบูรณ์สำหรับไฟล์ไฟล์ดังกล่าวเลย

บทที่ 5
บทวิจารณ์และสรุป

5.1 บทสรุป

ในทุกวันนี้การใช้งานโปรแกรมบิตทอเรนต์เพื่อดาวน์โหลดข้อมูลระหว่างกันแบบ Peer-to-Peer เป็นที่นิยมใช้กันอย่างแพร่หลาย โดยสังเกตได้จากจำนวนเว็บไซต์ที่เป็นแทรคเกอร์ที่มีอยู่เป็นจำนวนมากมาย และมีการเปิดขึ้นอย่างต่อเนื่อง รวมไปถึงตัวโปรแกรม Bittorrent Client ที่เป็นโปรแกรมสำหรับการใช้งานบิตทอเรนต์นั้นก็มีผู้ผลิตออกมาจากหลายค่าย หลากหลายรูปแบบให้ได้ใช้งานกัน ซึ่งกลุ่มผู้ใช้งานโปรแกรมบิตทอเรนต์นั้นก็มีแทบทุกระดับ ตั้งแต่นักเรียน นักศึกษา รวมไปถึงคนวัยทำงาน ซึ่งเหตุผลในการใช้งานก็แตกต่างกันไป จึงสามารถกล่าวได้ว่าแทบทุกหน่วยงานจะต้องมีคนใช้งานโปรแกรมบิดทอเรนต์

ดังที่ได้กล่าวมาแล้วว่าการเชื่อมต่อ Peer-to-Peer ด้วยโปรโตคอลบิตทอเรนต์นั้น จะเป็นการเชื่อมต่อแบบหนึ่งต่อหนึ่ง คือ หนึ่งงานที่รันอยู่ต่อหนึ่ง Peer ที่เชื่อมต่อด้วย ยิ่งไฟล์ๆหนึ่งมี Peer มากๆ ก็จะเกิดการเชื่อมต่อมากมาย แล้วยิ่งมีผู้ใช้งานโปรแกรมบิตทอเรนต์พร้อมๆกันทีละหลายๆ คน ก็จะทำให้เกิดการเชื่อมต่อของบิตทอเรนต์อันมากมายมหาศาล

ถ้ามีการนำระบบแคชสำหรับโปรแกรมบิดทอเรนต์มาใช้งานนั้น ก็จะทำให้การเชื่อมต่อของการใช้โปรแกรมบิตทอเรนต์จากภายในเครือข่ายที่ออกไปลดน้อยลง กล่าวคือจะเป็นการเชื่อมต่อของตัว Internal Peer ของเราเท่านั้น นอกจากนี้เมื่อมีการดาวน์โหลดไฟล์ที่ซ้ำกับไฟล์ที่มีอยู่ในระบบแคชก็จะทำให้ผู้ใช้งานดาวน์โหลดไฟล์โดยตรงเลย ช่วยลดเวลาในการดาวน์โหลดลงได้

5.2 วิจารณ์สิ่งที่ได้จากโครงงาน

โปรแกรมบิตทอร์เรนท์ที่ใช้กันอยู่ทั่วไป อย่าง BitComet มีการใช้งาน Distributed Hash Table ในการค้นหา Peer นอกเหนือจากการติดต่อกับแทรคเกอร์ ซึ่งจะใช้ UDP port ที่ตรงกับ TCP Listening Port ของโปรแกรม ซึ่งทำให้สามารถเอาข้อมูล Peers Table มาจากภายนอกเองได้โดยไม่ต้องติดต่อกับแทรคเกอร์ปกตินั้นการทำงาน DHT สามารถปิดได้โดยการเลือกไม่ใช้งาน และตั้ง Private Flag ที่ท้าย metadata ของไฟล์ .torrent แต่ BitComet ที่ใช้กันแพร่หลายนั้น จะไม่ใส่ใจกับ Private Flag และเมื่อปิด DHT function ไปแล้ว ก็จะกลับมาเปิดเองเป็นบางครั้ง ซึ่งถือว่าละเมิดข้อตกลงระหว่างผู้พัฒนา BitTorrent Client ด้วยกัน แต่ด้วยความสะดวกในการโหลดไฟล์โปรแกรมนี้จึงยังเป็นที่แพร่หลายในกลุ่มผู้ชอบดาวน์โหลดเป็นส่วนใหญ่มาก

โปรแกรมบิตทอร์เรนท์บางตัว มีการจำกัดการอัตราอัพโหลดไว้ ส่งผลให้การแลกเปลี่ยนข้อมูลได้รับผลกระทบเมื่อนำมาจำกัดเข้าแค่ในเครือข่าย เพราะจำเป็นต้องพึ่งการอัพโหลดภายในเครือข่ายมากขึ้นเมื่อจำกัดการดาวน์โหลดจากภายนอก

โปรแกรมบิตทอร์เรนท์บางตัว มีปัญหา Endgame Download Speed ซึ่งทำให้การโหลดไฟล์ ชะงักที่ 99% ไปเรื่อยๆ มักจะเกิดขึ้นกรณีที่ไฟล์ที่แลกเปลี่ยนนั้น มี Seeder ไม่กี่เครื่อง ซึ่งไม่พอเพียงที่จะทำงานแบบ Endgame ได้จนกว่าจะมีการค้นพบ Peer ตัวอื่นที่สมบูรณ์แล้วเช่นกัน แล้วติดต่อขอชิ้นส่วนสุดท้ายมาได้ นอกจากนี้ยังสามารถแก้ไขได้โดยการติดต่องานนี้ใหม่อีกครั้ง Endgame Download โดยปกติ คือการชะลอการโหลด แล้วเรียก Request ชิ้นสุดท้ายไปยัง Peer ทุก Peer ที่มีชิ้นนี้ แล้วส่งยกเลิกตามไปที่ peer อื่นๆเมื่อได้รับชิ้นส่วนนี้แล้วจาก Peer ตัวใดๆ

5.3 ปัญหาอุปสรรคและแนวทางการแก้ไข

ในช่วงระหว่างการดำเนินโครงงานระบบแคชสำหรับโปรแกรมบิดทอเรนต์นั้น ได้ประสบปัญหาต่างๆ ซึ่งได้รวบรวมมาเป็นข้อๆ ดังนี้

1. การใช้ระบบปฏิบัติการลีนุกซ์ในการทำโครงงานนั้น ประสบปัญหาที่สำคัญคือผู้ใช้ไม่มีความรู้ ความชำนาญในการติดตั้งโปรแกรมบนลีนุกซ์ รวมไปถึงการเลือกว่าจะใช้ลีนุกซ์ตระกูลไหน รุ่นอะไร Kernel อะไรมาใช้สำหรับการทำโครงงานนี้ เพราะการใช้ลีนุกส์เวอร์ชั่นเก่าทำให้บางครั้งการลงโปรแกรมใหม่ๆจะเกิดปัญหาเรื่อง library ซึ่งแนวทางการแก้ไขปัญหานี้ก็คือ หาคู่มือการลงโปรแกรมแต่ละโปรแกรม และลงลีนุกซ์เวอร์ชั่นใหม่ๆหรืออัพเดต Kernel เสมอๆ
2. ในการเลือกใช้โปรแกรมบิตทอเรนต์ พบว่าโปรแกรมบิดทอเรนต์ส่วนใหญ่จะเหมาะกับการทำงานบนระบบปฏิบัติการ Windows ส่วนโปรแกรมบิตทอเรนต์ที่สามารถทำงานบนระบบปฏิบัติการลีนุกซ์ได้นั้น บางโปรแกรมก็เป็นเวอร์ชั่นที่อยู่ระหว่างพัฒนา หรือไม่ก็ยังติดปัญหาอยู่ ซึ่งแนวทางการแก้ไขปัญหานี้คือ อาจจะต้องพัฒนาโปรแกรมบิตทอเรนต์ขึ้นมาให้งานเอง เพื่อที่จะได้ตรงกับความต้องการของระบบที่สุด
3. ในการพัฒนาระบบที่ได้ออกแบบไว้นั้น ในช่วงแรกได้เลือกใช้โปรแกรมที่จะมาพัฒนาระบบผิด ทำให้เสียเวลาในการแก้ไข ซึ่งแนวทางการแก้ไขปัญหานี้คือ การปรึกษาสอบถาม หาข้อมูลจากผู้ที่มีความรู้มีประสบการณ์
4. ในการเขียนเว็บเซอร์วิสนั้น เนื่องจากผู้เขียนขาดความรู้ และประสบการณ์ในการเขียนด้วย PHP ทำให้บางครั้งมีปัญหาในการเขียน เช่น การสั่งให้รันโปรแกรมบิตทอเรนต์ผ่านทางเว็บ หรือการเขียนเว็บเซอร์วิสจัดการข้อมูลที่อยู่ในแทรคเกอร์นั้น จำเป็นที่จะต้องเข้าใจ

โครงสร้างของฐานข้อมูลของแทรคเกอร์ที่ใช้ด้วย ซึ่งแนวทางการแก้ปัญหานี้คือ การศึกษาหาความรู้จากหนังสือ หรือเว็บไซต์ หรืออาจจะสอบถามจากผู้มีประสบการณ์ในด้านนั้นๆ

5. ในการติดตั้งโปรแกรมเพื่อมาใช้งานในระบบนั้น มีหลายครั้งที่วิธีการติดตั้งที่อยู่ในเว็บไซต์นั้นไม่ละเอียด หรือติดตั้งไปแล้วเกิดความผิดพลาดโดยที่ไม่มีวิธีแก้ไขบอกเอาไว้ ซึ่งแนวทางการแก้ปัญหานี้คือ การหาข้อมูลจากหลายๆแห่งมาเปรียบเทียบกัน หรือลองดูในกระทู้ตอบปัญหาของทางเว็บไซต์ของตัวโปรแกรม

5.4 แนวทางการพัฒนาต่อ

สำหรับแนวทางในการพัฒนาต่อของโครงงานระบบแคชสำหรับโปรแกรมบิตทอเรนต์นั้น ขอเสนอแนะแนวทางเป็นข้อๆดังนี้

1. พัฒนาในส่วนของโปรแกรมบิตทอเรนต์ให้มีความสามารถมากขึ้น ซึ่งอาจจะพัฒนาโดยการเขียนขึ้นเองทั้งหมด หรือการนำโค้ดที่มีอยู่ตามเว็บไซต์มาพัฒนาต่อให้มีความเหมาะสมกับตัวของระบบ
2. พัฒนาระบบให้สามารถทำงานได้ในเครือข่ายที่ใช้ Private IP หรือ Dynamic IP
3. พัฒนาให้ระบบสามารถทำการ filter ในขั้นของ attachment ได้ ซึ่งจะทำให้สามารถใช้งานกับแทรคเกอร์แบบที่เป็น Forum ได้
4. พัฒนาให้ระบบมีกระบวนการเคลียร์ค่าข้อมูลที่ไม่ได้ใช้งานบ่อยๆ หรือหมดอายุการใช้งานที่อยู่ในแคชของระบบ เพื่อให้ระบบไม่เต็มไปด้วยข้อมูลที่ไม่ได้ใช้
5. พัฒนาในด้านความเร็วในการทำงานของระบบ โดยอาจจะเพิ่มจำนวน Internal Peer ให้มีจำนวนมากขึ้น เพื่อแบ่งเบาภาระในการทำงานของแต่ละตัว
6. พัฒนาในด้านการปลอกการเชื่อมต่อของ Internal Peer ออกไปยังภายนอก เมื่อทำการดาวน์โหลดข้อมูลที่ต้องการได้แล้ว
7. พัฒนาในด้านการรักษาความปลอดภัยให้กับระบบ เพราะระบบที่ออกแบบมานี้ไม่ได้คำนึงถึงในด้านนี้ ซึ่งถ้าจะนำไปใช้งานจริงก็จำเป็นที่จะต้องคำนึงถึงด้านความปลอดภัยของทั้งตัวระบบ รวมไปถึงเครือข่ายภายใน

บรรณานุกรม

[1] BitTorrent – Protocol,[Online] URL : http://www.bittorrent.com/protocol.html

[2] Yahoo! Groups BitTorrent,[Online] URL :http://groups.yahoo.com/group/BitTorrent/

[3] P2P Forums Index,[Online] URL : http://www.p2pforums.com/index.php

[4] เลือกใช้ Proxy อย่างไรให้มีประสิทธิภาพ,[Online] URL : http://www.sut.ac.th/ccs/news/tip_tech/tip002.asp

[5] TransparentProxy,[Online] URL : http://phst.ph.mahidol.ac.th/Linux/html/transparentproxy.html

[6] Squid Proxy Caching Server,[Online] URL : http://micro.se-ed.com/content/mc205/MC205_181.asp

[7] Squid Web Proxy Cache,[Online] URL : http://www.squid-cache.org

[8] Squidguard – An ultrafast and free filter/redirector/access controller for Squid,[Online] URL : http://www.squidguard.org/

[9] Wiki.Thoery.Org - Bit Torrent Specification,[Online] URL : http://wiki.theory.org/BitTorrentSpecification

[10]Python Programming Language,[Online] URL : http://www.python.org

[11]Fedora Project,sponsored by Red Hat,[Online] URL : http://fedora.redhat.com

[12]เว็บเซอร์วิส คือ อะไร?,[Online] URL : http://chatpong.exteen.com/20050814/entry-1

[13]Web Services เครื่องมือธุรกิจยุคใหม่,[Online] URL : http://www.wsiam.com/document/WebServices_BusinessTool.pdf

[14]สุวัฒน์ ปุณณชัยยะ, ต้น ตัณฑ์สุทธิวงศ์ และ สุพจน์ ปุณณชัยยะ 2545 เปิดโลก TCP/IP และโปรโตคอลของอินเตอร์เน็ต Second Edition สำนักพิมพ์ Provision

[15]ก่อกิจ วีระอาชากุล 2545 ติดตั้งและปรับแต่งเซิร์ฟเวอร์ Linux สำหรับ Admin Linux โดยเฉพาะ Third Edition สำนักพิมพ์ Infopress Developer Book

[16]กิตติภูมิ วรฉัตร 2543 php เปลี่ยนวิถีสู่การสร้างโฮมเพ็จอย่างมืออาชีพ พิมพ์ครั้งที่ 1 สำนักพิมพ์ บริษัท วิตตี้ กรุ๊ป จำกัด

[17]นราวุธ พลับประสิทธิ์ 2546 php เปลี่ยนวิถีสู่การสร้างโฮมเพ็จอย่างมืออาชีพ ขั้นที่2 พิมพ์ครั้งที่ 1 สำนักพิมพ์ บริษัท วิตตี้ กรุ๊ป จำกัด

[18]กอบเกียรติ สระอุบล 2549 สร้างสรรค์ ปรุงแต่งเว็บไซต์ด้วย PHP พิมพ์ครั้งที่ 1 สำนักพิมพ์ มีเดีย เน็ตเวิร์ค