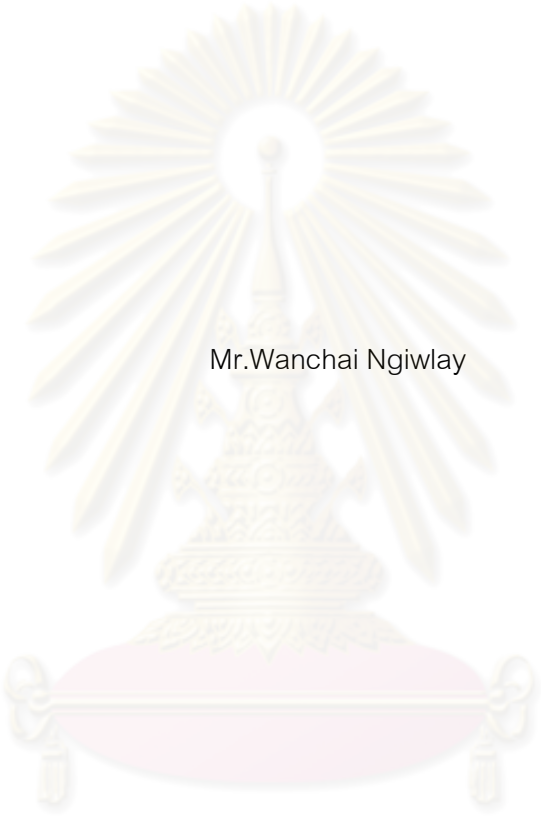
การระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ด้วยพฤติกรรมของขั้นตอนวิธีการเค้น

นายวันชัย งิ้วลาย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2551

BITTORRENT PEER IDENTIFICATION BASED ON BEHAVIORS OF A CHOKE ALGORITHM

Mr.Wanchai Ngiwlay



A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computer Science

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2008

หัวข้อวิทยานิพนธ์	การระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ด้วยพฤติกรรมของขั้นตอนวิธีการเค้น
โดย	นายวันชัย งิ้วลาย
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	อาจารย์ ดร.ยรรยง เต็งอำนวย
อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม	ผู้ช่วยศาสตราจารย์ ดร.เฉลิมเอก อินทนากรวิวัฒน์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(อาจารย์ ดร.เกริก ภิรมย์โสภา)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(อาจารย์ ดร.ยรรยง เต็งอำนวย)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม
(ผู้ช่วยศาสตราจารย์ ดร.เฉลิมเอก อินทนากรวิวัฒน์)

..... กรรมการภายนอกมหาวิทยาลัย
(รองศาสตราจารย์ ดร.อนันต์ ผลเพิ่ม)

วันชัย งิ้วลาย : การระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ด้วยพฤติกรรมของขั้นตอนวิธีการเค้น. (BITTORRENT PEER IDENTIFICATION BASED ON BEHAVIORS OF A CHOKE ALGORITHM) อ. ที่ปรึกษาวิทยานิพนธ์หลัก: อาจารย์ ดร.ยรรยง เต็งอำนวย, อ. ที่ปรึกษาวิทยานิพนธ์ร่วม: ผศ. ดร.เฉลิมเอก อินทนากรวิวัฒน์, 98 หน้า.

บิตทอร์เรนต์เป็นโพรโทคอลสำหรับแลกเปลี่ยนไฟล์แบบเพียร์ทูเพียร์ที่กำลังได้รับความนิยมอย่างมากในปัจจุบัน ส่งผลให้มีกระแสข้อมูลบิตทอร์เรนต์อยู่ในระบบเครือข่ายค่อนข้างมาก ซึ่งอาจส่งผลกระทบกับการใช้งานแอพลิเคชันอื่นบนระบบเครือข่าย การจะควบคุมการใช้งานบิตทอร์เรนต์ได้จำเป็นต้องมีวิธีการระบุเพียร์ที่มีประสิทธิภาพ ในงานวิจัยนี้นำเสนอการระบุเพียร์ของบิตทอร์เรนต์โดยอาศัยพฤติกรรมของกระแสข้อมูลที่เกิดจากขั้นตอนวิธีการเค้น ซึ่งเป็นขั้นตอนวิธีหลักในการควบคุมการแลกเปลี่ยนข้อมูลระหว่างเพียร์ ทำให้สามารถระบุหาเพียร์ได้แม้ว่าเพียร์จะมีการปรับเปลี่ยนรูปแบบการเชื่อมต่อในระดับโฟลเป็นแบบใดก็ตาม และในขั้นตอนการตรวจสอบได้ใช้ข้อมูลจากส่วนหัวของแพ็กเก็ตถึงระดับชั้นเครือข่าย จึงสามารถตรวจหาเพียร์ที่ทำการเข้ารหัสกระแสข้อมูลได้ มีสถานะที่ต้องจำขณะทำงานน้อยกว่างานวิจัยที่ผ่านมาที่ทำการตรวจสอบในระดับโฟล และไม่มีผลกระทบจากการเปลี่ยนแปลงในระดับชั้นขนส่ง จากผลการทดลองกับชุดข้อมูลควบคุมและชุดข้อมูลปกติพบว่า วิธีการที่นำเสนอสามารถตรวจจับเพียร์ที่มีการรับส่งข้อมูลจำนวนมากได้เป็นอย่างดี และยังสามารถตรวจจับได้อย่างรวดเร็ว ซึ่งจะช่วยให้สามารถควบคุมเพียร์ได้ก่อนที่เพียร์จะส่งข้อมูลได้เป็นจำนวนมาก นอกจากนี้ยังมีอัตราการตรวจจับผิดพลาดน้อย ทำให้ไม่เกิดผลกระทบกับการใช้งานแอพลิเคชันทั่วไป

ภาควิชา...วิศวกรรมคอมพิวเตอร์... ลายมือชื่อนิสิต...วันชัย งิ้วลาย...

สาขาวิชา...วิทยาศาสตร์คอมพิวเตอร์.. ลายมือชื่อ อ. ที่ปรึกษาวิทยานิพนธ์หลัก...

ปีการศึกษา...2551... ลายมือชื่อ อ. ที่ปรึกษาวิทยานิพนธ์ร่วม...

4970562821 : MAJOR COMPUTER SCIENCE

KEYWORDS: PEER-TO-PEER / BITTORRENT / PEER IDENTIFICATION / BEHAVIOR / CHOKE ALGORITHM

WANCHAI NGIWLAY: BITTORRENT PEER IDENTIFICATION BASED ON BEHAVIORS OF A CHOKE ALGORITHM, ADVISOR: YUNYONG TENG-AMNUAY, Ph.D., CO-ADVISOR: ASST. PROF. CHALERMEK INTANAGONWIWAT, Ph.D., 98 pp.

Bittorrent is currently one of the most popular peer-to-peer file sharing protocols. However, it incurs such excessive amount of traffic that it may adversely affect the performance of legacy internet applications. To limit this adverse impact, an efficient methodology for bittorrent identification may be needed. This work proposes a novel approach to identify bittorrent peers based on behaviors of the choke algorithm (the main algorithm used for controlling data exchanges among bittorrent peers) instead of using the transport-layer information. Therefore, this work can identify even the peers that attempt to avoid being detected by altering their flow connection at the transport layer. Given that this work relies on only the network-layer information, our approach is capable of achieving robustness to changes in the transport layer, maintaining fewer states, and identifying peers that encrypt their traffic. The experimental results on the controlled traffic and normal traffic indicate that this approach can efficiently and quickly identify most of excessive bandwidth-consuming peers. Hence, it is possible to manage, stop, or control bittorrent peers before they can transfer a large amount of data. Furthermore, with our low false-positive rate, the approach will rarely misidentify peers and will unlikely worsen the performance of legacy internet applications if not improving it.

Department: ..Computer Engineering... Student's Signature.....

Field of Study: ..Computer Science..... Advisor's Signature.....

Academic Year: ..2008..... Co-Advisor's Signature.....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดีด้วยความอนุเคราะห์ และความช่วยเหลืออย่างยิ่งจากอาจารย์ ดร.ยรรยง เต็งอำนวย และผู้ช่วยศาสตราจารย์ ดร.เฉลิมเอก อินทนากรวิวัฒน์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่กรุณาให้ข้อคิด แนวทางและคำปรึกษาแก่ผู้วิจัยตลอดการดำเนินการวิจัย รวมถึงตรวจทานแก้ไขให้วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วง ผู้วิจัยขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้ กราบขอบพระคุณอาจารย์ ดร.เกริก ภิรมย์โสภา และรองศาสตราจารย์ ดร.อนันต์ ผลเพิ่ม คณะกรรมการสอบวิทยานิพนธ์ที่ให้ข้อคิดและข้อเสนอแนะเพื่อให้วิทยานิพนธ์ฉบับนี้สมบูรณ์ยิ่งขึ้น และขอขอบพระคุณคณาจารย์ทุกท่านที่กรุณาประสิทธิประสาทวิชาความรู้อันมีค่ายิ่งให้แก่ผู้วิจัย

ขอขอบคุณเจ้าหน้าที่ศูนย์คอมพิวเตอร์ของคณะวิศวกรรมศาสตร์ และสำนักงานเทคโนโลยีสารสนเทศ จุฬาลงกรณ์มหาวิทยาลัย ที่ให้ความช่วยเหลือและอำนวยความสะดวกในการเก็บข้อมูลเพื่อใช้ในการวิจัยชิ้นนี้

ขอบคุณเพื่อนๆ พี่ๆ และน้องๆ ทุกคนที่ให้กำลังใจ ข้อคิดเห็น และให้ความช่วยเหลือในทุกๆ ด้าน ตลอดจนช่วยสร้างบรรยากาศในการวิจัยที่มีค่ายิ่งให้แก่ผู้วิจัย ทำให้งานวิจัยชิ้นนี้สำเร็จลุล่วงไปได้ด้วยดี

ท้ายนี้ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ ที่ให้การสนับสนุนและดูแลเอาใจใส่ผู้วิจัยด้วยความรักและเมตตา ตลอดจนเป็นกำลังใจให้แก่ผู้วิจัยจนสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี

สารบัญ

หน้า

หน้า

สารบัญภาพ

หน้า

หน้า

สารบัญตาราง

หน้า

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของปัญหา

โพรโทคอลบิตทอร์เรนต์ (BitTorrent Protocol) [1] เป็นโพรโทคอลสำหรับแลกเปลี่ยนข้อมูลระหว่างเครื่องบนระบบเครือข่ายที่กำลังได้รับความนิยมอย่างมากในปัจจุบัน ข้อมูลที่แลกเปลี่ยนกันด้วยโพรโทคอลนี้จะถูกสำเนาไปยังเครื่องต่างๆ ได้อย่างรวดเร็ว ยิ่งมีเครื่องที่ร่วมทำการแลกเปลี่ยนข้อมูลเดียวกันอยู่มากเท่าไร ก็ยิ่งทำให้การแลกเปลี่ยนข้อมูลนั้นของระบบโดยรวมมีประสิทธิภาพมากขึ้นเท่านั้น ซึ่งเอาชนะข้อจำกัดของระบบแลกเปลี่ยนไฟล์รูปแบบเดิมในลักษณะของเครื่องให้บริการ (Server) กับเครื่องลูกข่าย (Client) ที่มีปัญหาคอขวด (Bottleneck) ที่เครื่องให้บริการในเวลาที่มีผู้เข้ามาดาวน์โหลดไฟล์พร้อมกันจำนวนมาก

ด้วยประสิทธิภาพการทำงานของบิตทอร์เรนต์ที่สามารถดาวน์โหลดและอัพโหลดได้ในเวลาเดียวกัน ทำให้สามารถใช้ประโยชน์จากทั้งช่องทางการสื่อสารขาเข้า (Downstream) และช่องทางสื่อสารขาออก (Upstream) ได้อย่างเต็มประสิทธิภาพตลอดเวลา ประกอบกับจำนวนผู้ใช้งานที่เพิ่มมากขึ้นเรื่อยๆ ส่งผลให้มีกระแสข้อมูล (Data Traffic) ที่เกิดจากแอพลิเคชันของบิตทอร์เรนต์อยู่ในระบบเครือข่ายค่อนข้างมาก เมื่อเทียบกับกระแสข้อมูลที่เกิดจากการใช้งานอินเทอร์เน็ตแอพลิเคชันทั่วไป ซึ่งอาจส่งผลกระทบกับการใช้งานระบบเครือข่ายตามปกติ เช่น ทำให้การเปิดหน้าเว็บเพจของบริการเวิลด์ไวด์เว็บ (WWW) ช้ากว่าปกติ การจะควบคุมการใช้งานบิตทอร์เรนต์ได้ จำเป็นต้องมีวิธีการระบุหาเพียร์ที่ใช้งานบิตทอร์เรนต์ที่มีประสิทธิภาพ

การระบุหาเพียร์ (Peer Identification) ที่ใช้งานบิตทอร์เรนต์และเพียร์ทูเพียร์ตัวอื่นๆ มีการปรับเปลี่ยนและพัฒนามาอย่างต่อเนื่อง สาเหตุมาจากการที่แอพลิเคชันบนระบบเครือข่ายมีการพัฒนาอยู่ตลอดเวลา โดยในช่วงเริ่มแรกใช้การตรวจสอบจากหมายเลขพอร์ต (Port Number) [2, 3] เนื่องจากแอพลิเคชันจะใช้หมายเลขพอร์ตโดยปริยาย (Default Port Number) เช่น บิตทอร์เรนต์ใช้หมายเลขพอร์ตช่วง 6881-6889 ในการรอรับการเชื่อมต่อ แต่เมื่อแอพลิเคชันเริ่มมีการสุ่มหมายเลขพอร์ตเพื่อหลีกเลี่ยงการควบคุมการใช้งานมากขึ้น [4] การดูจากหมายเลขพอร์ตเพียงอย่างเดียวจึงไม่สามารถแยกแยะได้อย่างถูกต้องอีกต่อไป จึงได้มีการพัฒนาวิธีระบุกระแสข้อมูลด้วยการตรวจสอบจากเนื้อหาในแพ็กเก็ต (Packet) โดยใช้การตรวจหาลายเซ็นแอพลิเคชัน (Application Signature) [4, 5, 6] ที่ปรากฏอยู่ในแพ็กเก็ต ซึ่งวิธีนี้สามารถระบุหาเพียร์ได้อย่าง

แม่นยำ แต่มีข้อจำกัดอยู่หลายประการ เช่น ต้องการทรัพยากรในการประมวลผลค่อนข้างมาก ไม่สามารถระบุหาเพียร์ที่เข้ารหัส (Encryption) เนื้อหาในแพ็กเก็ต [7] และละเมิดความเป็นส่วนตัว (Privacy)

ในเวลาต่อมาจึงมีงานวิจัยที่ศึกษาและนำเสนอวิธีการระบุหาเพียร์และกระแสข้อมูลเพียร์ทูเพียร์ด้วยการวิเคราะห์จากพฤติกรรม (Behaviors) ของกระแสข้อมูล โดยอาศัยข้อมูลจากส่วนหัวของแพ็กเก็ตในระดับชั้นขนส่ง (Transport Layer) มาใช้ในการวิเคราะห์ ทำให้ไม่มีข้อจำกัดจากการตรวจสอบจากเนื้อหาในแพ็กเก็ต ซึ่งจะได้กล่าวเพิ่มเติมในส่วนของงานวิจัยที่เกี่ยวข้องต่อไป แต่โดยภาพรวมแล้วเป็นงานวิจัยที่ทำการตรวจสอบโดยใช้พฤติกรรมทั่วไปของกระแสข้อมูลเพียร์ทูเพียร์ ซึ่งพฤติกรรมบางอย่างไม่สามารถนำมาใช้กับโพรโทคอลบิตทอร์เรนต์ได้ บางแนวคิดยังมีข้อจำกัดจากสภาพแวดล้อมของระบบเครือข่าย และบางงานวิจัยอาจได้รับผลกระทบจากพฤติกรรมของผู้ใช้ เช่น ผู้ใช้ส่วนใหญ่อาจหันมาใช้หมายเลขพอร์ตที่ได้รับอนุญาติจากผู้ดูแลระบบ (Privileged port) เช่น หมายเลขพอร์ต 80 ในการทำงาน ซึ่งทำให้ความหลากหลายของหมายเลขพอร์ตลดลง เป็นต้น

จากปัญหาที่กล่าวมา งานวิจัยนี้จึงนำเสนอแนวคิดในการระบุหาเพียร์ของบิตทอร์เรนต์โดยอาศัยพฤติกรรมของกระแสข้อมูลที่เกิดจากขั้นตอนวิธีการเค้น (Choke Algorithm) [8, 9] ซึ่งเป็นขั้นตอนวิธีหลักของโพรโทคอลบิตทอร์เรนต์ โดยใช้ข้อมูลจากส่วนหัวของแพ็กเก็ตในระดับชั้นเครือข่าย (Network Layer) ในกระบวนการตรวจสอบ ทำให้ไม่มีข้อจำกัดที่เกิดขึ้นจากการตรวจสอบส่วนของเนื้อหาในแพ็กเก็ต และไม่ได้รับผลกระทบจากการเปลี่ยนแปลงองค์ประกอบที่เกี่ยวข้องในระดับชั้นขนส่ง เช่น การเปลี่ยนแปลงช่วงของหมายเลขพอร์ตที่ใช้งาน รวมถึงระบบเครือข่ายที่มีไฟร์วอลล์ (Firewall) หรือมีการแปลงเลขที่อยู่เครือข่าย (Network Address Translation) ซึ่งทำให้มีปัญหาในการตรวจสอบจากแพ็กเก็ต SYN เป็นต้น นอกจากนี้ การไม่นำหมายเลขพอร์ตเข้ามาเกี่ยวข้องในกระบวนการตรวจสอบ ทำให้สามารถลดสถานะที่ต้องจำระหว่างการทำงานลงไปได้จำนวนมาก (แต่ละหมายเลขไอพีมีจำนวนพอร์ต 65,535 พอร์ต)

1.2 วัตถุประสงค์ของการวิจัย

เพื่อสังเคราะห์วิธีการระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ โดยอาศัยพฤติกรรมของขั้นตอนวิธีการเค้น ซึ่งก่อให้เกิดพฤติกรรมของการจำกัดปริมาณการส่งข้อมูล

1.3 ขอบเขตของการวิจัย

1. ใช้ข้อมูลจากส่วนหัวของแพ็กเก็ตถึงระดับชั้นเครือข่ายในการวิเคราะห์
2. กระแสข้อมูลต้องไม่มีการเข้ารหัสส่วนหัวของโพรโทคอลไอพี
3. กระแสข้อมูลต้องไม่ถูกแปลงเลขที่อยู่เครือข่ายมาก่อนที่จะมาถึงตัวระบุเพียร์
4. นำเสนอวิธีการระบุเพียร์ที่ใช้งานบิตทอร์เรนต์เท่านั้น ไม่รวมถึงวิธีจัดการกับหมายเลขไอพีที่ระบุได้
5. สำรวจบิตทอร์เรนต์ไคลเอนต์ที่ได้รับความนิยม เพื่อดูว่ากระแสข้อมูลที่เกิดขึ้นมีลักษณะตรงกับแนวคิดที่ใช้ในงานวิจัยหรือไม่
6. สำรวจแอพลิเคชันที่มีพฤติกรรมการติดต่อกับหมายเลขไอพีจำนวนมาก เพื่อเปรียบเทียบกับบิตทอร์เรนต์
7. ค่าต่างๆ ที่ใช้ในกระบวนการตรวจสอบ ทำเป็นลักษณะของพารามิเตอร์ (Parameter) ที่สามารถปรับค่าได้
8. พัฒนาโปรแกรมตัวอย่างสำหรับระบุเพียร์ที่ใช้งานบิตทอร์เรนต์จากแนวคิดที่นำเสนอ

1.4 ขั้นตอนของการวิจัย

1. ศึกษาลักษณะของโพรโทคอลแบบเพียร์ทูเพียร์
2. ศึกษารายละเอียดและงานวิจัยที่เกี่ยวข้องกับโพรโทคอลบิตทอร์เรนต์
3. ศึกษาลักษณะของกระแสข้อมูลที่เกิดจากแอพลิเคชันต่างๆ บนระบบเครือข่าย
4. ศึกษางานวิจัยที่เกี่ยวกับการระบุหาเพียร์และกระแสข้อมูลบิตทอร์เรนต์ รวมถึงเพียร์ทูเพียร์ตัวอื่น
5. ออกแบบวิธีการระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ โดยใช้ข้อมูลถึงระดับชั้นเครือข่าย
6. ทดสอบประสิทธิภาพของวิธีการระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ที่ได้
7. ปรับปรุงวิธีการระบุกระแสข้อมูลบิตทอร์เรนต์ที่นำเสนอ
8. สรุปผลและเรียบเรียงวิทยานิพนธ์

1.5 ประโยชน์ที่ได้รับ

เพิ่มความสามารถให้กับกระบวนการระบุหาเพียร์ของบิตทอร์เรนต์ในระบบเครือข่าย โดยเฉพาะเพียร์ที่ทำการเข้ารหัสเพื่อเจตนาหลีกเลี่ยงการควบคุมของผู้ดูแลระบบ ซึ่งผลที่ได้จะช่วยให้สามารถควบคุมการใช้งานบิตทอร์เรนต์ให้เป็นไปตามนโยบายที่กำหนดไว้ขององค์กร และไม่สูญเสียทรัพยากรของระบบเครือข่ายโดยไม่จำเป็น

1.6 โครงสร้างของวิทยานิพนธ์

เนื้อหาของวิทยานิพนธ์ฉบับนี้แบ่งออกเป็น 5 บทดังนี้ คือ บทที่ 1 เป็นบทนำของงานวิจัย บทที่ 2 กล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้องกับงานวิจัยชิ้นนี้ บทที่ 3 กล่าวถึงวิธีการดำเนินงานวิจัยในแต่ละขั้นตอนอย่างละเอียด บทที่ 4 เป็นการทดลองและอธิบายผลการทดลองในประเด็นต่างๆ และบทที่ 5 เป็นการสรุปผลการทดลองและข้อเสนอแนะจากงานวิจัย ซึ่งอาจเป็นประโยชน์กับการวิจัยเพิ่มเติมในอนาคต

1.7 ผลงานตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของงานวิทยานิพนธ์นี้ได้รับการตีพิมพ์เป็นบทความวิชาการในหัวเรื่อง “Bittorrent Peer Identification based on Behaviors of a Choke Algorithm” โดยวันชัย งิ้วลาย เฉลิมเอก อินทนากรวิวัฒน์ และยรรยง เต็งอำนวย ในงานประชุมวิชาการ “The 4th Asian Internet Engineering Conference (AINTEC 2008)” ซึ่งจัดขึ้น ณ โรงแรมพูลแมน บางกอก คิง พาวเวอร์ กรุงเทพมหานคร ประเทศไทย ระหว่างวันที่ 18-20 พฤศจิกายน 2551 ดังภาคผนวก ก หน้า 81 และได้รับการตีพิมพ์เป็นวารสารเนคเทค (NECTEC Technical Journal) ในหัวเรื่อง “โปรแกรมตรวจหาและยับยั้งการใช้งานบิตทอร์เรนต์ในระบบเครือข่ายแบบเรียลไทม์โดยอาศัยพฤติกรรมของอัลกอริทึมการเค้น” โดยวันชัย งิ้วลาย ยรรยง เต็งอำนวย และเฉลิมเอก อินทนากรวิวัฒน์ ในงานประชุมวิชาการ “NECTEC Annual Conference and Exhibition 2008” ซึ่งจัดขึ้น ณ โรงแรมโซฟิเทลเซ็นทาราแกรนด์ กรุงเทพมหานคร ประเทศไทย ระหว่างวันที่ 24-25 กันยายน 2551 ดังภาคผนวก ก หน้า 91

บทที่ 2

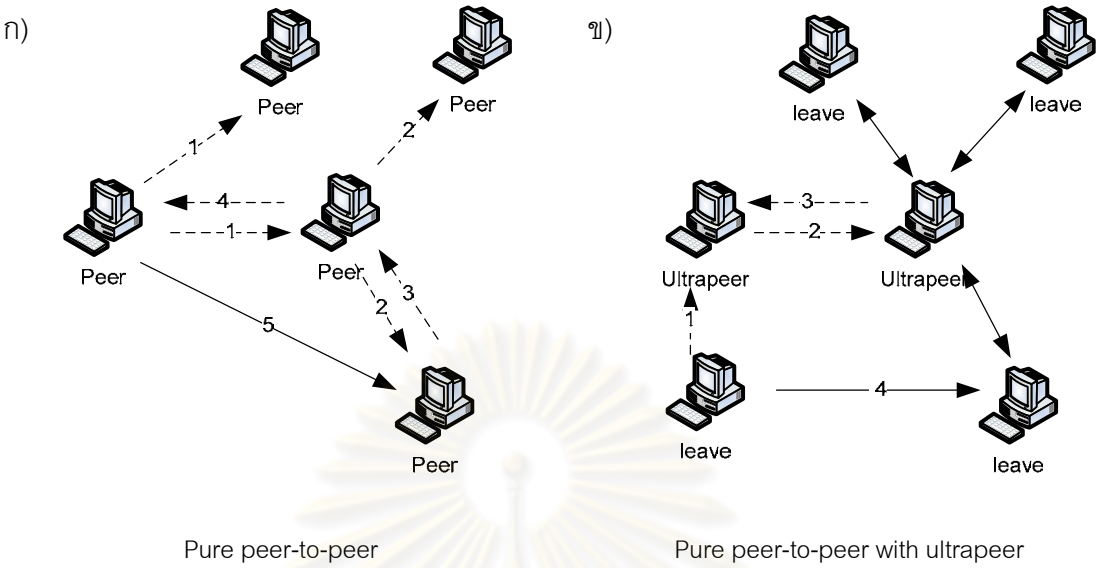
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 เพียร์ทูเพียร์

เพียร์ทูเพียร์ (Peer-to-peer) [10] คือ กลุ่มของระบบและแอพลิเคชันที่ใช้ทรัพยากรที่กระจาย (Distributed) กันอยู่เพื่อทำงานในลักษณะแยกจากศูนย์ (Decentralized) ซึ่งทรัพยากรต่างๆ นั้นได้แก่ ความสามารถในการคำนวณ ข้อมูล แบนด์วิดท์ของระบบเครือข่าย (Network Bandwidth) เป็นต้น คอมพิวเตอร์ในระบบเพียร์ทูเพียร์จะเชื่อมต่อกันโดยตรง ไม่มีการนำหลักการเกี่ยวกับเครื่องให้บริการเข้ามาเกี่ยวข้อง ทำให้แต่ละเครื่องในระบบมีบทบาทเป็นทั้งผู้ให้บริการและผู้ใช้บริการในเวลาเดียวกัน โครงสร้างของเพียร์ทูเพียร์เมื่อแบ่งตามระดับของการแยกจากศูนย์ (Levels of Decentralization) จะแบ่งได้เป็น 2 ประเภทหลักๆ คือ

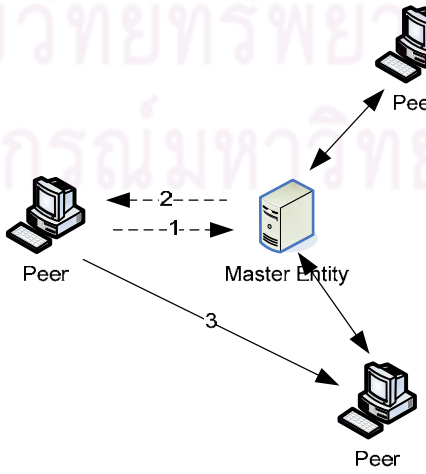
1. เพียร์ทูเพียร์แท้ (Pure Peer-to-Peer) เป็นโครงสร้างของเพียร์ทูเพียร์ที่แต่ละเอนทิตี (Entity) หรือเพียร์ (Peer) มีระดับความสำคัญเท่ากัน ไม่มีศูนย์กลางของระบบ ดังแสดงในรูปที่ 2.1 ก) ดังนั้นไม่ว่าเพียร์ใดเกิดความขัดข้อง (Failure) ระบบก็ยังสามารถทำงานต่อไปได้ ส่วนการทำงานนั้นจะต้องมีกระบวนการค้นหา (Discovery) ทรัพยากรหรือเพียร์ที่ต้องการติดต่อด้วย เช่น การส่งการร้องขอกระจายทุกทิศทาง (Flooding Request) จากนั้นจึงติดต่อกันโดยตรงเพื่อทำงานร่วมกันต่อไป โพรโทคอลเพียร์ทูเพียร์ที่มีโครงสร้างลักษณะนี้ได้แก่ นูเทลล่า (Gnutella) กาซ่า (Kazaa) และฟรีเน็ต (Freenet) เป็นต้น

อย่างไรก็ตาม การส่งการร้องขอกระจายทุกทิศทางไปยังเพียร์ทั้งหมดในระบบนั้นทำให้เกิดข้อความ (Message) ขึ้นในระบบเครือข่ายจำนวนมาก อีกทั้งยังส่งผลต่อความสามารถในการปรับขนาดได้ (Scalable) ของระบบ ดังนั้นในปัจจุบันบางโพรโทคอล เช่น นูเทลล่า จึงได้นำแนวคิดในการแบ่งระดับชั้น (Tier) ของเพียร์ออกเป็นอัลตราเพียร์ (Ultrapeer) และลีฟ (leave) เพื่อลดจำนวนข้อความที่ต้องมีการรับส่งกันในกระบวนการค้นหา รวมทั้งรองรับการขยายขนาดของระบบดีกว่าระบบแบบเดิม โดยอัลตราเพียร์ คือ เพียร์ที่ได้รับคัดเลือกจากเพียร์อื่นในกลุ่ม (หรือลีฟ) ให้เป็นตัวกลางในการติดต่อสื่อสารกับเพียร์ในกลุ่มอื่น รวมทั้งเป็นที่เก็บดัชนี (Index) ของทรัพยากรต่างๆ ของเพียร์ในกลุ่ม เมื่อเพียร์ในกลุ่มที่ต้องการค้นหาข้อมูลที่อยู่ภายนอก จะทำการร้องขอไปยังอัลตราเพียร์ จากนั้นอัลตราเพียร์จะส่งการร้องขอไปยังเพียร์อื่นๆ ผ่านทางอัลตราเพียร์ของแต่ละกลุ่มต่อไป ดังแสดงในรูปที่ 2.1 ข)



รูปที่ 2.1 ลักษณะโครงสร้างแบบเพียร์ทูเพียร์แท้และขั้นตอนการค้นหาข้อมูล

2. เพียร์ทูเพียร์ลูกผสม (Hybrid Peer-to-Peer) เป็นโครงสร้างของเพียร์ทูเพียร์ที่มีเอนทิตีหลัก (Master Entity) เป็นศูนย์กลางในการติดต่อหรือเป็นแหล่งเก็บรวบรวมที่อยู่ของทรัพยากร ดังแสดงในรูปที่ 2.2 ดังนั้นหากเอนทิตีหลักนี้เกิดความขัดข้องระบบก็อาจจะไม่สามารถทำงานต่อไปได้ ในการทำงานหากเอนทิตีหรือเพียร์ใดต้องการติดต่อกัน หรือต้องการทราบว่าทรัพยากรที่ต้องการอยู่ที่ใด ต้องทำการสอบถามที่อยู่จากเอนทิตีหลักนี้ เมื่อได้ที่อยู่มาแล้วจึงติดต่อกันเองโดยตรงต่อไป โพรโทคอลเพียร์ทูเพียร์ที่มีโครงสร้างลักษณะนี้ได้แก่ แนปสเตอร์ (Napster) และบิตทอร์เรนต์ (Bittorrent) เป็นต้น



รูปที่ 2.2 ลักษณะโครงสร้างแบบเพียร์ทูเพียร์ลูกผสมและขั้นตอนการค้นหาข้อมูล

2.2 โพรโทคอลบิตทอร์เรนต์

โพรโทคอลบิตทอร์เรนต์ (BitTorrent Protocol) [5, 8, 9] เป็นโพรโทคอลแบบเพียร์ทูเพียร์รูปแบบหนึ่ง คิดค้นโดยนายแบรม โคเฮน (Bram Cohen) ในปี ค.ศ. 2001 ถูกออกแบบมาสำหรับการแลกเปลี่ยนไฟล์บนระบบเครือข่าย ไฟล์ต้นฉบับจะถูกตัดแบ่งออกเป็นชิ้นส่วน (Pieces) ที่เพียร์เริ่มต้น จากนั้นเพียร์อื่นที่ต้องการไฟล์นี้จะทำการดาวน์โหลดชิ้นส่วนของไฟล์ที่ถูกตัดแบ่งเหล่านี้ไปทีละชิ้น โดยการคำนวณจากขั้นตอนวิธีในการเลือกชิ้นส่วนของโพรโทคอลบิตทอร์เรนต์ เรียกว่า Rarest First Algorithm [1, 3] จุดประสงค์คือเพื่อให้แต่ละเพียร์มีชิ้นส่วนของไฟล์แตกต่างกันมากที่สุด โพรโทคอลบิตทอร์เรนต์มีองค์ประกอบหลักที่สำคัญดังนี้

1. ไฟล์ทอร์เรนต์ (Torrent file) เป็นไฟล์ที่มีนามสกุล .torrent ซึ่งเก็บรายละเอียดที่สำคัญของไฟล์ที่จะให้ดาวน์โหลด เช่น ขนาดของไฟล์ จำนวนชิ้นส่วน ยูอาร์แอล (URL) ของแทรกเกอร์ ค่าแฮชของไฟล์ทอร์เรนต์ (Info hash) เป็นต้น

2. แทรกเกอร์ (Tracker) เป็นเครื่องให้บริการ (เอนทิตีหลัก) ที่คอยจัดเก็บสถานะและข้อมูลต่างๆ ของเพียร์ รวมถึงการส่งรายชื่อเพียร์ที่มีการติดต่อกับแทรกเกอร์อยู่ให้แก่เพียร์ที่ร้องขอ

3. เพียร์ (Peer) คือ เครื่องคอมพิวเตอร์แต่ละเครื่องที่กำลังเชื่อมต่อกันอยู่ เพื่อดาวน์โหลดหรืออัพโหลดไฟล์ในขณะนั้น กลุ่มของเพียร์ที่กำลังดาวน์โหลดหรืออัพโหลดข้อมูลของไฟล์ทอร์เรนต์เดียวกันเรียกว่าสวอร์ม (Swarm) เพียร์ที่เข้าร่วมอยู่ในสวอร์มนี้สามารถแบ่งออกได้เป็น 2 ประเภท ได้แก่

3.1 ซีด (Seed) คือ เพียร์ที่มีชิ้นส่วนของไฟล์ครบทุกชิ้น และคอยอัพโหลดชิ้นส่วนของไฟล์เหล่านี้ให้กับเพียร์อื่นในสวอร์ม โดยในแต่ละสวอร์มต้องมีเพียร์ที่เป็นซีดอย่างน้อย 1 เพียร์ การดาวน์โหลดจึงสามารถเสร็จสมบูรณ์ได้

3.2 ลีช (Leech) คือ เพียร์ที่ยังมีชิ้นส่วนของไฟล์ไม่ครบ และทำการดาวน์โหลดชิ้นส่วนของไฟล์จากเพียร์อื่นในสวอร์ม ในเวลาเดียวกันก็ช่วยอัพโหลดชิ้นส่วนที่ตนเองมีให้แก่เพียร์อื่นที่ต้องการอีกด้วย

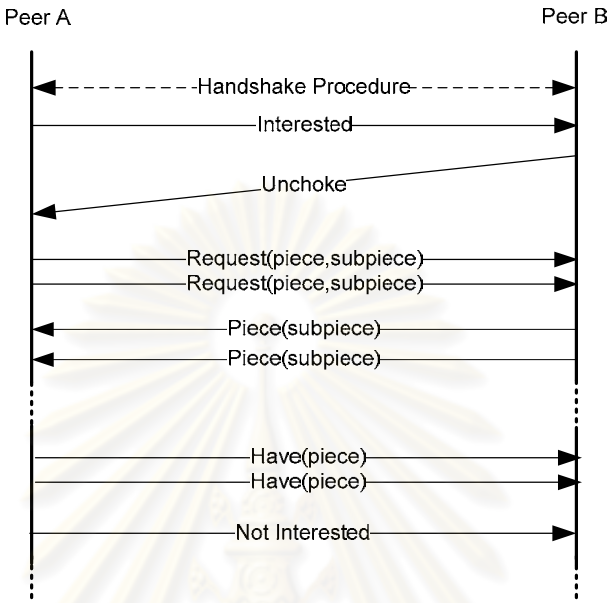
การใช้งานโพรโทคอลบิตทอร์เรนต์ เริ่มจากการที่ผู้ใช้ดาวน์โหลดไฟล์ทอร์เรนต์จากเว็บไซต์ หรืออาจได้มาด้วยวิธีการอื่น จากนั้นเปิดไฟล์ทอร์เรนต์นี้ด้วยโปรแกรมบิตทอร์เรนต์ไคลเอนต์ เช่น BitComet, Azureus เป็นต้น บิตทอร์เรนต์ไคลเอนต์จะทำการติดต่อไปยังแทรกเกอร์

ด้วยโพรโทคอลทีซีพี (TCP) ตามยูอาร์แอลที่ระบุไว้ในไฟล์ทอร์เรนต์ พร้อมทั้งส่งข้อมูลของตนเอง เช่น รหัสเพียร์ (Peer ID) หมายเลขไอพี หมายเลขพอร์ตที่เปิดรอ (Listening Port) รวมทั้งค่าแฮช ของไฟล์ทอร์เรนต์ไปยังแทรกเกอร์ แทรกเกอร์จะทำการสุ่มรายชื่อเพียร์ในสวอร์มจำนวน 50 เพียร์ (ค่าโดยปริยาย) ส่งมาให้บิตทอร์เรนต์ไคลเอนต์ เมื่อได้รับรายชื่อเพียร์จากแทรกเกอร์แล้ว บิตทอร์เรนต์ไคลเอนต์จะทำการเปิดการเชื่อมต่อด้วยโพรโทคอลทีซีพีไปยังเพียร์เหล่านั้นโดยตรง เรียกกลุ่มของเพียร์นี้ว่า เซตของเพียร์ (Peer Set)

โพรโทคอลบิตทอร์เรนต์จะใช้ข้อความในการควบคุมและประสานจังหวะการทำงานของแต่ละเพียร์ให้สอดคล้องกัน แต่ละข้อความมีขนาดคงที่ (ยกเว้นข้อความ Bitfield) ยกตัวอย่างการทำงาน เช่น เมื่อเพียร์ A ได้รับรายชื่อเพียร์มาจากแทรกเกอร์ ซึ่งมีเพียร์ B รวมอยู่ด้วย หากเพียร์ A ต้องการติดต่อแลกเปลี่ยนไฟล์กับเพียร์ B เพียร์ A จะทำการเปิดการเชื่อมต่อด้วยโพรโทคอลทีซีพีไปยังเพียร์ B จากนั้นเพียร์ A จะส่งข้อความ Handshake ไปยังเพียร์ B โดยในข้อความ Handshake นั้นมีรหัสเพียร์ของเพียร์ A และค่าแฮชของไฟล์ทอร์เรนต์ เมื่อเพียร์ B ได้รับข้อความ Handshake จากเพียร์ A เพียร์ B ก็จะส่งข้อความ Handshake กลับไปยังเพียร์ A และเมื่อเพียร์ A ได้รับข้อความ Handshake ตอบกลับมาจากเพียร์ B ก็จะตรวจสอบว่าค่ารหัสเพียร์ตรงกันกับที่ได้รับมาจากแทรกเกอร์หรือไม่ และค่าแฮชของไฟล์ทอร์เรนต์ตรงกับในไฟล์ทอร์เรนต์หรือไม่ หากไม่ตรงกันเพียร์ A จะตัดการเชื่อมต่อกับเพียร์ B ทันที

หลังจากผ่านกระบวนการแฮนด์เชคแล้ว เพียร์แต่ละฝ่ายจะถูกกำหนดสถานะเริ่มต้นเป็น Not Interested และ Choked จากอีกฝ่าย เช่น เพียร์ A กำหนดสถานะเริ่มต้นของเพียร์ B เป็น Not Interested และ Choked ซึ่งหมายความว่า เพียร์ B ไม่ต้องการชิ้นส่วนจากเพียร์ A และเพียร์ A ไม่ยอมอัพโหลดข้อมูลให้กับเพียร์ B ตามลำดับ จากนั้นข้อความแรกสุดที่ทั้งสองฝ่ายจะส่งหากันคือข้อความ Bitfield เป็นข้อความที่มีการส่งครั้งเดียวในตอนเริ่มแรกเพื่อแสดงให้เพียร์อีกฝ่ายรับรู้ว่าตนเองมีชิ้นส่วนใดของไฟล์อยู่บ้าง แต่ละบิตในข้อความ Bitfield เป็นดัชนีของแต่ละชิ้นส่วนในไฟล์ที่จะดาวน์โหลด ทำให้ข้อความชนิดนี้มีขนาดไม่คงที่ ขึ้นอยู่กับขนาดของไฟล์ที่จะดาวน์โหลด เมื่อเพียร์แต่ละฝ่ายได้รับข้อความ Bitfield และตรวจพบว่าเพียร์อีกฝ่ายมีชิ้นส่วนที่ตนเองต้องการ จะทำการส่งข้อความ Interested ไปให้อีกฝ่าย เมื่อเพียร์ที่มีชิ้นส่วนได้รับข้อความ Interested จะรับรู้ว่าเพียร์อีกฝ่ายต้องการดาวน์โหลดชิ้นส่วนจากตนเอง จากนั้นจึงเป็นการทำงานของขั้นตอนวิธีการเค้นที่จะตัดสินใจว่ายอมอัพโหลดให้หรือไม่ ถ้ายินยอมจะส่งข้อความ Unchoked กลับไปให้ และเมื่อเพียร์ที่ต้องการชิ้นส่วนได้รับข้อความ Unchoked จะทำการส่งข้อความ Request เพื่อแจ้งว่าตนเองต้องการชิ้นส่วนใดและเริ่มจากตำแหน่งที่เท่าใดของชิ้นส่วนนี้ จากนั้นจึงเริ่มการรับส่ง

ข้อมูลกันจริง โดยทำการรับส่งกันทีละชิ้นส่วนย่อย (Sub-piece) ขั้นตอนการติดต่อและการแลกเปลี่ยนชิ้นส่วนของไฟล์ระหว่างเพียร์แสดงได้ดังรูปที่ 2.3



รูปที่ 2.3 ขั้นตอนการติดต่อและการแลกเปลี่ยนชิ้นส่วนของไฟล์ระหว่างเพียร์

ในระหว่างการดาวน์โหลด เมื่อเพียร์ได้รับชิ้นส่วนใดมาแล้ว จะส่งข้อความ Have ไปยังเพียร์อื่นทุกตัวในเซตของเพียร์เพื่อบอกให้ทราบว่าตนเองมีชิ้นส่วนใหม่นี้แล้ว และเมื่อเพียร์ได้รับชิ้นส่วนครบตามที่ร้องขอไปและไม่ต้องการชิ้นส่วนใดจากเพียร์อีกฝ่ายแล้ว จะทำการส่งข้อความ Not Interested เพื่อบอกให้เพียร์อีกฝ่ายรู้ว่าตนเองไม่ต้องการชิ้นส่วนใดอีก

นอกจากนี้ ระหว่างที่เพียร์เข้าร่วมอยู่ในสวอร์ม จะมีการส่งสถานะของตนเองไปยังแทรกเกอร์ เช่น ค่าดาวน์โหลด ค่าอัพโหลด เป็นต้น และยังรักษาจำนวนเพียร์ในเซตของเพียร์ไม่ให้ต่ำกว่า 20 (ค่าโดยปริยาย) โดยถ้ามีจำนวนในเซตของเพียร์ต่ำกว่านี้ จะทำการร้องขอรายชื่อเพียร์ชุดใหม่จากแทรกเกอร์

2.3 ขั้นตอนวิธีการเค้น

ขั้นตอนวิธีการเค้น (Choke Algorithm) [8, 9] เป็นขั้นตอนวิธีของโพรโทคอลบิตทอร์เรนต์เพื่อรักษาความสมดุลของการอัพโหลดและการดาวน์โหลดระหว่างเพียร์ เพียร์ที่ดาวน์โหลดอย่างเดียวแต่ไม่ยอมอัพโหลดให้เพียร์อื่นจะไม่สามารถดาวน์โหลดได้ หรือดาวน์โหลดได้ด้วยอัตราที่ต่ำ โดยจากมุมมองของเพียร์เฉพาะที่ (Local peer) เพียร์ทั้งหมดในเซตของเพียร์จะมีเพียง 4 ตัว (ค่า

โดยปริยาย) เท่านั้นที่มีสถานะเป็น Unchoked และ Interested นั่นคือจะมี 4 ตัวเท่านั้นที่เพียร์ยอมอัพโหลดชิ้นส่วนของไฟล์ให้ เรียกกลุ่มของเพียร์เหล่านี้ว่า เซตของแอ็กทิฟเพียร์ (Active Peer Set) การทำงานของขั้นตอนวิธีการเค้นแสดงได้ดังรูปที่ 2.4

ขั้นตอนวิธีการเค้น (Choke Algorithm)

1: ในแต่ละรอบของการคำนวณ (ทุก 10 วินาที) เพียร์ทั้งหมดในเซตของเพียร์จะถูกเรียงลำดับตามอัตราค่าดาวน์โหลดที่เพียร์เฉพาะที่สามารถดาวน์โหลดได้

2: ทำการ Unchoke เพียร์ที่ให้อัตราค่าดาวน์โหลดสูงที่สุด 4 ลำดับแรกที่อยู่ในสถานะ Interested

3: ทุกรอบที่ 3 (หรือ 30 วินาที) ของการคำนวณขั้นตอนวิธี จะมีการทำ Optimistic Unchoked

รูปที่ 2.4 ขั้นตอนวิธีการเค้น

จากการทำงานตามปกติของขั้นตอนวิธีการเค้นพบว่า เพียร์ที่มีโอกาสดาวน์โหลดชิ้นส่วนจากเพียร์อื่นคือเพียร์ที่มีชิ้นส่วนที่เพียร์อื่นๆ ยังไม่มี และอัพโหลดชิ้นส่วนเหล่านี้ให้ก่อนในรอบก่อนหน้าของการคำนวณขั้นตอนวิธี เพื่อให้มีโอกาสถูก Unchoke จากเพียร์เหล่านั้น และสามารถดาวน์โหลดชิ้นส่วนที่ตนเองต้องการได้ในรอบถัดไป ดังนั้นจึงมีขั้นตอนวิธี Optimistic Unchoke เพื่อเปิดโอกาสให้เพียร์ใหม่ ซึ่งยังไม่มีชิ้นส่วนใดอยู่สามารถเข้าร่วมในการดาวน์โหลดได้ การทำงานของขั้นตอนวิธี Optimistic Unchoke แสดงได้ดังรูปที่ 2.5

ขั้นตอนวิธี Optimistic Unchoked

1: ในทุกรอบที่ 3 (หรือ 30 วินาที) ของขั้นตอนวิธีการเค้น หลังจากจัดเรียงเพียร์ตามอัตราค่าดาวน์โหลดแล้ว เพียร์ 3 ลำดับแรกจะถูก Unchoke ตามปกติ

2: เลือก Unchoke เพียร์อีก 1 ตัวอย่างสุ่ม จากเพียร์ที่เหลือทั้งหมดในเซ็ตของเพียร์ โดยไม่พิจารณาจากอัตราค่าดาวน์โหลด

รูปที่ 2.5 ขั้นตอนวิธี Optimistic Unchoke

2.4 งานวิจัยที่เกี่ยวข้อง

งานวิจัยเกี่ยวกับการระบุเพียร์และกระแสข้อมูลบิตทอร์เรนต์ รวมถึงเพียร์ทูเพียร์ตัวอื่น มีการพัฒนาอย่างต่อเนื่อง เพื่อให้ตามทันกับวิธีการหลบเลี่ยงการควบคุมการใช้งานของแอพลิเคชันเหล่านี้ ซึ่งโดยรวมแล้วแบ่งได้เป็น 3 แนวทาง คือ

1. การตรวจสอบโดยใช้หมายเลขพอร์ตโดยปริยายของโพรโทคอล เซน และคณะ [10] ได้ทำการวิเคราะห์กระแสข้อมูลเพียร์ทูเพียร์ในระบบเครือข่ายขนาดใหญ่ โดยใช้การพิจารณาหมายเลขพอร์ตโดยปริยายของแต่ละโพรโทคอลในการแบ่งแยกกระแสข้อมูล เช่น โพรโทคอลบิตทอร์เรนต์จะมีหมายเลขพอร์ตโดยปริยายอยู่ในช่วง 6881-6889 รวมถึง ซารอย และคณะ [3] ที่ได้ทำการวิเคราะห์กระแสข้อมูลบนอินเทอร์เน็ตโดยการพิจารณาจากหมายเลขพอร์ตโดยปริยายของแต่ละแอพลิเคชันเช่นเดียวกัน อย่างไรก็ตาม การตรวจสอบจากหมายเลขพอร์ตโดยปริยายไม่สามารถนำมาใช้ระบุหาเพียร์ที่ใช้งานบิตทอร์เรนต์ในปัจจุบัน เนื่องจากบิตทอร์เรนต์ไคลเอนต์มีความสามารถในการสุ่มหมายเลขพอร์ตเพื่อหลีกเลี่ยงการตรวจจับดังกล่าว

2. การตรวจสอบจากส่วนของเนื้อหาในแพ็กเก็ต คาราเกียนนิส และคณะ [4] ได้แสดงให้เห็นว่าปริมาณการใช้งานเพียร์ทูเพียร์ไม่ได้ลดลงแต่อย่างใด แต่ได้เปลี่ยนรูปแบบจากเดิมที่ใช้หมายเลขพอร์ตโดยปริยายมาเป็นการสุ่มพอร์ตขึ้นมาใช้งานแทน โดยทั้ง คาราเกียนนิส [4] และ เซน [6] ได้นำเสนอการระบุกระแสข้อมูลของโพรโทคอลเพียร์ทูเพียร์ด้วยวิธีตรวจสอบจากเนื้อหาในแพ็กเก็ต โดยใช้การตรวจหาลายเซ็นแอพลิเคชันของเพียร์ทูเพียร์แต่ละตัว ในเวลาต่อมา ฮอง และคณะ [11] ได้นำเสนอการระบุกระแสข้อมูลของโพรโทคอลบิตทอร์เรนต์แบบออนไลน์ (online) ด้วยวิธีตรวจหาลายเซ็นแอพลิเคชันของบิตทอร์เรนต์ในข้อความ Handshake อย่างไรก็ตาม ถึงแม้วิธีการตรวจสอบจากเนื้อหาในแพ็กเก็ตจะมีความแม่นยำสูง แต่กลับมีข้อจำกัดในด้านอื่นๆ เช่น ต้องการทรัพยากรในการประมวลผลค่อนข้างมาก ไม่สามารถตรวจสอบแพ็กเก็ตของบิตทอร์เรนต์ที่มีการเข้ารหัสได้ และละเมิดความเป็นส่วนตัว

3. การวิเคราะห์จากพฤติกรรมของกระแสข้อมูล คาราเกียนนิส และคณะ [12] ได้นำเสนอแนวคิดในการระบุกระแสข้อมูลของโพรโทคอลเพียร์ทูเพียร์ โดยอาศัยข้อมูลจากส่วนหัวของแพ็กเก็ตถึงระดับชั้นขนส่ง โดยเสนอว่าเพียร์ทูเพียร์มีการใช้งานทั้งโพรโทคอลทีซีพีและยูดีพี (UDP) ในเวลาเดียวกัน และพิจารณาความสมดุลระหว่างจำนวนหมายเลขไอพีและหมายเลขพอร์ตที่แตกต่างกันที่เชื่อมต่อกับหมายเลขพอร์ตใดๆ แต่บิตทอร์เรนต์ใช้โพรโทคอลทีซีพีในการส่งข้อมูลและข้อความควบคุม ซึ่งไม่ตรงกับแนวคิดแรก

คอลลินส์ และคณะ [13] ได้นำเสนอแนวคิดในการระบุกระแสข้อมูลเพียร์ทูเพียร์ในระดับโฟล (Flow) และใช้บิตทอร์เรนต์เป็นกรณีศึกษา โดยได้ทำการจัดกลุ่มของโฟลออกเป็น Short Flow, Message และ File Transfers และมีการใช้ฟังก์ชันในการคำนวณพฤติกรรมของโฟลในแต่ละกลุ่ม สามารถระบุได้ถูกต้อง 72%

คาราเกียนนิส และคณะ [14] ได้พัฒนาตัวระบุกระแสข้อมูลโดยใช้รูปแบบการเชื่อมต่อกันของโฟลในการพิจารณาหลายระดับ (Multi-level) ได้แก่ ระดับทางสังคม (Social Level) ระดับหน้าที่การทำงาน (Functional Level) และระดับแอพลิเคชัน (Application Level) สามารถจำแนก (Classify) โฟลออกเป็นแอพลิเคชันได้ถูกต้องมากกว่า 80% อย่างไรก็ตาม แนวคิดในงานวิจัยนี้มีการคำนวณที่ซับซ้อน ทำให้ต้องใช้ทรัพยากรในการประมวลผลมาก จึงยังไม่เหมาะสมกับการนำมาใช้งานจริง

บาร์เล็ตต์ และคณะ [15] ระบุหาเพียร์โดยอาศัยพฤติกรรมภายใน (Inherent Behaviors) ของเพียร์ทูเพียร์ คือ มีจำนวนการเชื่อมต่อที่ไม่สำเร็จในอัตราที่สูงกว่าแอพลิเคชันทั่วไป มีจำนวนการเชื่อมต่อขาออกและการเชื่อมต่อขาเข้าในระดับที่สมดุลกัน โดยใช้การตรวจสอบจากแพ็กเก็ต SYN และ SYN/ACK ที่เข้าและออกจากเพียร์ในแต่ละโฟล ผลการทดลองแสดงให้เห็นว่า 75% ของเพียร์สามารถระบุได้ภายใน 10 นาที แต่แนวคิดเกี่ยวกับจำนวนการเชื่อมต่อขาออกและการเชื่อมต่อขาเข้าในระดับที่สมดุลกันยังมีข้อจำกัดอยู่ เช่น การนำไปใช้กับระบบที่มีการแปลงเลขที่อยู่เครือข่าย หรือมีการติดตั้งไฟร์วอลล์

งานวิจัยที่อาศัยพฤติกรรมของกระแสข้อมูลดังที่กล่าวมา ใช้การตรวจสอบในระดับโฟล ทำให้บางแนวคิดที่อาศัยความแตกต่างของหมายเลขพอร์ต [14, 15] เป็นส่วนหนึ่งในการวิเคราะห์อาจได้รับผลกระทบ หากเพียร์มีการเปลี่ยนพฤติกรรมมาใช้หมายเลขพอร์ตที่ระบบส่วนใหญ่กำหนดให้ใช้ (Well Known Ports) [16] เช่น หมายเลขพอร์ต 80 ที่กำหนดให้ใช้กับบริการเวิลด์ไวด์เว็บ ซึ่งจะทำให้ความหลากหลายของหมายเลขพอร์ตลดลง ดังนั้นการดูจากรูปแบบการเชื่อมต่อของหมายเลขพอร์ตจึงอาจเกิดข้อผิดพลาด

นอกจากนี้ บิตทอร์เรนต์สามารถใช้การเชื่อมต่อเพียง 1 โฟลระหว่างเพียร์ในการส่งข้อมูลและข้อความควบคุม ซึ่งจากการทำงานของขั้นตอนวิธีการเค้น ทำให้โฟลดูเหมือนมีข้อมูลรับส่งจำนวนมากในบางช่วงเวลา (ช่วง Unchoked) แต่ในเวลาต่อมาไม่นานอาจเหลือเพียงแค่การรับส่งข้อความควบคุม (ช่วง Choked) ทำให้การพิจารณาจากค่าสถิติของโฟลเกิดความผิดพลาดได้เช่นกัน

จากปัญหาที่กล่าวมา งานวิจัยนี้จึงนำเสนอวิธีการระบุเพียร์ที่ใช้งานบิตทอร์เรนต์โดยอาศัยพฤติกรรมของกระแสข้อมูล ซึ่งเกิดจากการทำงานของขั้นตอนวิธีการเค้นที่แต่ละเพียร์ เพื่อให้สามารถลดข้อจำกัดที่เกิดขึ้นในงานวิจัยที่ผ่านมา โดยสามารถสรุปและเปรียบเทียบกับงานวิจัยที่เกี่ยวข้องได้ดังตารางที่ 2.1

ตารางที่ 3.1 ตารางเปรียบเทียบงานวิจัยที่นำเสนอและงานวิจัยที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวข้อง	วิธีที่ใช้	ระดับชั้น	เกิดผลกระทบจาก				
			สุ่มพอร์ต	เข้ารหัส	การเปลี่ยนแปลงในชั้นขนส่ง	Restricted Network	อื่นๆ
Sen et al., 2002	Default Port	Transport	yes	no	yes	no	-
Saroiu et al., 2002	Default Port	Transport	yes	no	yes	no	-
Sen et al., 2004	Signature	Application	no	yes	no	no	Privacy
Karagiannis et al., 2004	Signature	Application	no	yes	no	no	Privacy
Karagiannis et al., 2004	Flow pattern	Transport	no	no	yes	yes	-
Karagiannis et al., 2005	Flow pattern	Transport	no	no	yes	yes	Complexity
Horng et al., 2006	Signature	Application	no	yes	no	no	Privacy
Collins et al., 2006	Flow statistics	Transport	no	no	yes	no	Choke Algorithm
Bartlett et al., 2007	Flow pattern	Transport	no	no	yes	yes	-
งานวิจัยที่นำเสนอ (2008)	Choke Algorithm	Network	no	no	no	no	-

บทที่ 3

วิธีดำเนินงานวิจัย

จากปัญหาที่เกิดขึ้นของการระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ในงานวิจัยที่เกี่ยวข้อง งานวิจัยนี้จึงมีจุดมุ่งหมายเพื่อนำเสนอวิธีการระบุเพียร์ที่อาศัยพฤติกรรมของกระแสข้อมูล โดยไม่มีข้อจำกัดจากการตรวจสอบเนื้อหาในแพ็กเก็ต การเปลี่ยนแปลงหมายเลขพอร์ตและปัจจัยต่างๆ ในระดับชั้นขนส่ง รวมถึงสภาพแวดล้อมของระบบเครือข่าย

ในงานวิจัยนี้ได้นำพฤติกรรมของกระแสข้อมูล 4 อย่างที่เกิดจากขั้นตอนวิธีการเค้น (Choke Algorithm) ซึ่งเป็นขั้นตอนวิธีหลักที่ใช้ในการทำงานของบิตทอร์เรนต์ มาสร้างเป็นเกณฑ์วัด (Metric) เพื่อใช้ทดสอบหมายเลขไอพีว่ามีการใช้งานบิตทอร์เรนต์หรือไม่ โดยข้อดีของการตรวจสอบจากขั้นตอนวิธีหลัก คือ สามารถตรวจจับเพียร์ได้แม้ว่าเพียร์จะหลบหลีกการตรวจจับด้วยการเข้ารหัสข้อมูล หรือใช้การปรับเปลี่ยนการเชื่อมต่อระหว่างเพียร์ให้เป็นรูปแบบต่างๆ ที่ใกล้เคียงกับแอพลิเคชันตามปกติ เพราะตราบเท่าที่บิตทอร์เรนต์ยังใช้ขั้นตอนวิธีการเค้นในการควบคุมการทำงานอยู่ พฤติกรรมต่างๆ ที่เกิดขึ้นในระหว่างการดาวน์โหลดจะถูกตรวจจับได้ด้วยเกณฑ์วัดที่นำเสนอ

การดำเนินงานวิจัยเพื่อสร้างตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ โดยอาศัยพฤติกรรมของกระแสข้อมูลที่เกิดจากขั้นตอนวิธีการเค้น เป็นไปดังนี้

3.1 การทดลองสร้างตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ด้วยวิธีต่าง ๆ

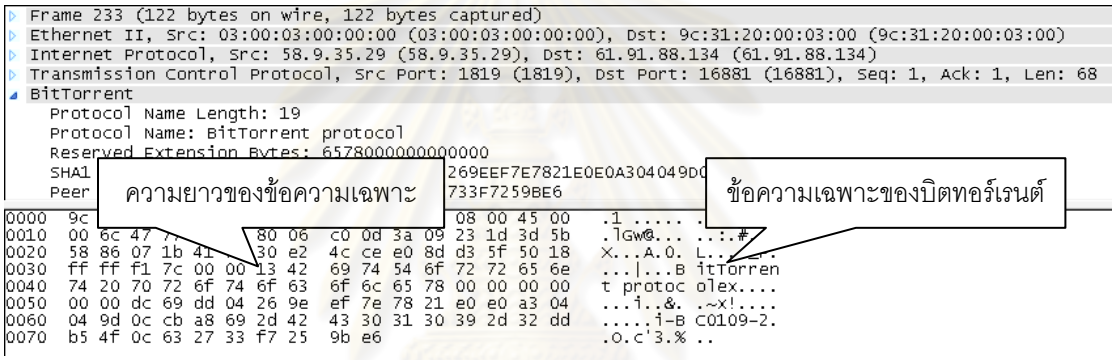
นอกจากตัวระบุเพียร์ที่อาศัยพฤติกรรมจากขั้นตอนวิธีการเค้นที่นำเสนอ ในงานวิจัยนี้ยังได้ศึกษาและทดลองสร้างตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ด้วยแนวคิดอื่นๆ เพื่อหาแนวทางที่เหมาะสม ดังนี้

3.1.1 การทดลองสร้างตัวระบุเพียร์ด้วยการตรวจหาลายเซ็นแอพลิเคชัน

เนื่องจากการระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ด้วยการตรวจหาลายเซ็นแอพลิเคชันเป็นวิธีที่ให้ความถูกต้องที่สุด อีกทั้งในช่วงเวลาที่ผู้วิจัยเริ่มศึกษา การเข้ารหัสแพ็กเก็ตยังไม่เป็นที่นิยมเท่าในปัจจุบัน ดังนั้น ในงานวิจัยนี้จึงได้มีการทดลองสร้างตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ด้วยการตรวจหาลายเซ็นแอพลิเคชัน โดยทำการทดลองกับข้อมูลปกติจากระบบเครือข่ายที่ได้เก็บรวบรวม

ในปี พ.ศ. 2549 ซึ่งจะอธิบายถึงขั้นตอนการเก็บและรายละเอียดของข้อมูลชุดนี้อีกครั้งในบทที่ 4 หัวข้อที่ 4.1

เครื่องมือที่ใช้ในการตรวจหาลายเซ็นในหัวข้อนี้ ผู้วิจัยได้พัฒนาขึ้นเองโดยใช้การตรวจหาข้อความเฉพาะของบิตทอร์เรนต์ [4, 6] ในทุกแพ็กเก็ตของชุดข้อมูลปกติ โดยข้อความเฉพาะดังกล่าวจะอยู่ถัดจากส่วนหัวของโพรโทคอลทีซีพี ซึ่งเป็นส่วนเริ่มของเนื้อหาในแพ็กเก็ตของข้อความ Handshake โดยไบต์แรกจะมีค่ารหัสแอสกี (ASCII) เท่ากับ 19 (0x13) ซึ่งเป็นความยาวของข้อความ “BitTorrent protocol” ที่ปรากฏตามมา โดยหากตรวจสอบแพ็กเก็ตดังกล่าวด้วยโปรแกรมวิเคราะห์แพ็กเก็ต เช่น โปรแกรม Ethereal จะมีลักษณะดังแสดงในรูปที่ 3.1



รูปที่ 3.1 ลักษณะแพ็กเก็ตของข้อความ Handshake

สาเหตุที่ต้องพัฒนาโปรแกรมดังกล่าวขึ้นเองเนื่องจากงานวิจัยนี้ต้องการเก็บข้อมูลของหมายเลขไอพีเพิ่มเติม เพื่อนำไปวิเคราะห์ในส่วนของความเร็วในการตรวจหาเพียร์ ปริมาณข้อมูลที่เพียร์มีการรับส่ง และประเด็นที่เกี่ยวข้องอื่นๆ นอกเหนือไปจากการตรวจสอบว่าหมายเลขไอพีใดมีการใช้งานบิตทอร์เรนต์บ้าง ซึ่งจะได้อธิบายโดยละเอียดต่อไปในบทที่ 4 ผลการทดลองตรวจหาเพียร์โดยใช้ลายเซ็นแอพลิเคชันด้วยโปรแกรมที่พัฒนาขึ้น แสดงได้ดังตารางที่ 3.1

ตารางที่ 3.1 การตรวจหาเพียร์ของข้อมูลปกติโดยใช้ลายเซ็น

	หมายเลขไอพี		จำนวนไบต์ที่รับส่ง	
	จำนวน	%	กิกะไบต์	%
หมายเลขไอพีทั้งหมด	1,519		405.54	
หมายเลขไอพีที่ใช้งานบิตทอร์เรนต์	106	6.98	374.85	92.43
หมายเลขไอพีที่คาดว่าไม่มีการใช้งานบิตทอร์เรนต์	1,413	93.02	30.69	7.57

จากผลการทดลอง พบว่ามีหมายเลขไอพีเพียง 106 หมายเลข หรือคิดเป็น 6.98% ของหมายเลขไอพีทั้งหมด ที่มีการรับส่งแพ็กเก็ตที่มีลายเซ็นแอพลิเคชันของบิตทอร์เรนต์ แต่เมื่อพิจารณาจากจำนวนไบต์ที่มีการรับส่ง พบว่าหมายเลขไอพีเหล่านี้มีการรับส่งข้อมูลรวมกันถึง 374 กิกะไบต์ หรือคิดเป็น 92.43% ของปริมาณข้อมูลที่มีการรับส่งทั้งหมด แสดงให้เห็นว่าวิธีการตรวจหาลายเซ็นแอพลิเคชัน สามารถตรวจหาหมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์ได้อย่างมีประสิทธิภาพ เพราะถึงแม้ว่าอาจมีบางเพียร์ที่ไม่สามารถตรวจหาได้พบ แต่เพียร์เหล่านั้นและหมายเลขไอพีที่ไม่ได้ใช้งานบิตทอร์เรนต์ก็มีการรับส่งข้อมูลรวมกันเพียง 7.57% ของปริมาณข้อมูลที่มีการรับส่งทั้งหมดเท่านั้น

อย่างไรก็ตาม การตรวจหาลายเซ็นแอพลิเคชันจากทุกแพ็กเก็ตต้องใช้ทรัพยากรในการประมวลผลเป็นอย่างมาก ผู้วิจัยจึงมีแนวความคิดที่จะใช้การสุ่ม (Sampling) เพื่อลดจำนวนของแพ็กเก็ตที่ต้องประมวลผล แต่เนื่องจากลายเซ็นแอพลิเคชันของบิตทอร์เรนต์ปรากฏอยู่เฉพาะในข้อความ Handshake ซึ่งเป็นข้อความที่มีการส่งกันตอนเริ่มต้นของการติดต่อกันระหว่างคู่เพียร์เท่านั้น ทำให้การระบุเพียร์ที่ใช้งานบิตทอร์เรนต์โดยใช้การสุ่มแพ็กเก็ตเพื่อตรวจหาลายเซ็นแอพลิเคชันมีผลการทดลองไม่ดีนัก อีกทั้งบิตทอร์เรนต์ไคลเอนต์เริ่มมีความสามารถในการเข้ารหัสข้อมูล ทำให้ไม่สามารถตรวจหาลายเซ็นแอพลิเคชันได้ดังเดิม ผู้วิจัยนี้จึงเริ่มมุ่งเน้นการสร้างตัวระบุเพียร์โดยใช้พฤติกรรมของกระแสข้อมูลแทนการตรวจสอบจากเนื้อหาในแพ็กเก็ต

3.1.2 การทดลองสร้างตัวระบุเพียร์โดยอาศัยพฤติกรรมการเปิดการเชื่อมต่อ

พฤติกรรมแรกของกระแสข้อมูลบิตทอร์เรนต์ที่งานวิจัยนี้นำมาทดลอง คือ พฤติกรรมการเปิดการเชื่อมต่อของเพียร์ เนื่องจากบิตทอร์เรนต์เป็นโพรโทคอลแบบเพียร์ทูเพียร์ ทำให้แต่ละเพียร์สามารถเป็นได้ทั้งเครื่องให้บริการรอรับการเชื่อมต่อขาเข้า (Incoming Connections) จากเครื่องลูกข่าย และสามารถเป็นเครื่องลูกข่ายที่เปิดการเชื่อมต่อขาออก (Outgoing Connections) ไปยังเครื่องให้บริการเพื่อขอใช้บริการ พฤติกรรมเช่นนี้ทำให้เพียร์มีจำนวนการเชื่อมต่อขาเข้าและจำนวนการเชื่อมต่อขาออกในระดับที่สมดุลกัน

พฤติกรรมการเปิดการเชื่อมต่อนี้ คล้ายกับพฤติกรรมของกระแสข้อมูลเพียร์ทูเพียร์ที่นำเสนอโดย บาร์เล็ตต์ และคณะ [15] แต่ต่างกันที่งานวิจัยของบาร์เล็ตต์ใช้การพิจารณาในระดับโฟล (Flow Level) ซึ่งคำนวณจากจำนวนโฟลที่เชื่อมต่อเข้ามายังเพียร์ และจำนวนโฟลที่เปิดการเชื่อมต่อออกไปจากเพียร์ ส่วนในงานวิจัยนี้ใช้การพิจาณาในระดับเพียร์ (Peer Level) ซึ่งคำนวณจากจำนวนหมายเลขไอพีที่เปิดการเชื่อมต่อเข้ามายังเพียร์ และจำนวนหมายเลขไอพีที่

เพียร์เปิดการเชื่อมต่อออกไปหา เนื่องจากต้องการดูความสัมพันธ์ระหว่างคู่หมายเลขไอพีเท่านั้น ว่าฝ่ายใดมีบทบาทเป็นผู้ให้บริการหรือผู้ใช้บริการ จึงไม่จำเป็นต้องพิจารณาจากทุกโฟลที่เชื่อมต่อกันอยู่ เพราะอาจถูกรบกวนจากแอพลิเคชันที่มีการสร้างโฟลจำนวนมากไปยังเครื่องให้บริการ เช่น การเปิดเว็บไซต์ เป็นต้น นอกจากนี้ การพิจารณาในระดับเพียร์ยังลดสถานะ (State) ที่ต้องจำในขณะทำงานลงอีกด้วย เพราะมีจำนวนสถานะที่ต้องจำระหว่างคู่หมายเลขไอพีเพียง 1 ตัว ต่างจากการพิจารณาในระดับโฟลที่จำนวนสถานะที่ต้องจำระหว่างคู่หมายเลขไอพีขึ้นอยู่กับจำนวนโฟลที่เชื่อมต่อกันอยู่ ซึ่งผลการทดสอบกับกระแสข้อมูลบิตทอร์เรนต์ที่เกิดจากการดาวน์โหลดผ่านทางโมเด็มเอดีเอสแอล (ADSL Modem) โดยไม่มีไฟร์วอลล์และไม่มีการแปลงเลขที่อยู่เครือข่าย พบว่าสามารถระบุเพียร์ได้เป็นอย่างดี

แต่เนื่องจากระบบเครือข่ายที่ใช้เก็บข้อมูลที่ใช้ทดลองมีอุปกรณ์ไฟร์วอลล์ติดตั้งอยู่ ซึ่งคอยบอกปัดการเปิดการเชื่อมต่อเข้ามายังหมายเลขพอร์ตที่ไม่ได้รับอนุญาตให้ใช้งาน และจุดที่เก็บข้อมูลอยู่หลังอุปกรณ์ไฟร์วอลล์ ดังนั้นผลการทดลองด้วยพฤติกรรมการเปิดการเชื่อมต่อกับทั้งสองชุดข้อมูลจึงไม่ดีนัก อีกทั้งระบบเครือข่ายโดยทั่วไปขององค์กรมักมีโครงสร้างในลักษณะเช่นนี้ ดังนั้นพฤติกรรมที่อาศัยความสมดุลระหว่างการเชื่อมต่อขาเข้าและการเชื่อมต่อขาออกจึงไม่เหมาะกับการนำมาสร้างเป็นตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์เพื่อใช้งานร่วมกับอุปกรณ์ที่มีอยู่ในระบบเครือข่ายดังกล่าว

3.2 การสร้างตัวระบุเพียร์โดยอาศัยพฤติกรรมของขั้นตอนวิธีการเค้น

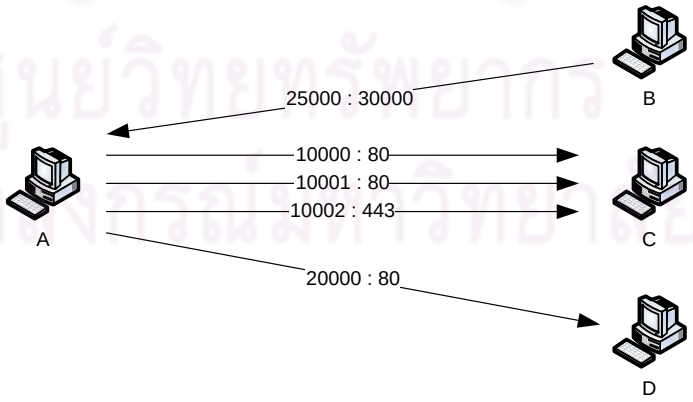
จากข้อจำกัดที่เกิดขึ้นในการทดลองสร้างตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ตามหัวข้อที่ 3.1.1 และ 3.1.2 งานวิจัยนี้จึงมีแนวคิดที่จะสร้างตัวระบุเพียร์ของบิตทอร์เรนต์โดยไม่ต้องตรวจสอบส่วนของเนื้อหาในแพ็กเก็ต และสามารถทำงานได้ในระบบเครือข่ายที่มีสภาพแวดล้อมแบบต่างๆ รวมทั้งสภาพแวดล้อมที่สามารถมองเห็นกระแสข้อมูลที่สมบูรณ์ได้เพียงด้านเดียว เช่น มีอุปกรณ์ไฟร์วอลล์ติดตั้ง หรือมีการทำการแปลงที่อยู่เครือข่าย เป็นต้น

ตามหลักการของการเค้น (Choke) ที่ได้กล่าวไว้ในบทที่ 2 หัวข้อที่ 2.3 โพรโทคอลบิตทอร์เรนต์มีขั้นตอนวิธีการเค้น (Choke algorithm) เป็นองค์ประกอบหลักในการควบคุมการดาวน์โหลดและอัพโหลดที่แต่ละเพียร์ ทำให้กระแสข้อมูลที่เกิดขึ้นจากแต่ละเพียร์มีความแตกต่างจากกระแสข้อมูลของแอพลิเคชันทั่วไป งานวิจัยนี้จึงอาศัยพฤติกรรมของกระแสข้อมูลที่เกิดขึ้นจากขั้นตอนวิธีการเค้นมาสร้างเป็นตัวระบุเพียร์ โดยมีรายละเอียดดังนี้

3.2.1 **พฤติกรรมของกระแสข้อมูลที่เกิดจากขั้นตอนวิธีการเค้น**

พฤติกรรมของกระแสข้อมูลที่เกิดจากขั้นตอนวิธีการเค้นในหัวข้อนี้ เป็นพฤติกรรมที่อาศัยมุมมองของเพียร์เฉพาะที่ (Local Peer) ซึ่งจะพิจารณาการเชื่อมต่อที่เพียร์เฉพาะที่เป็นผู้ทำการเชื่อมต่อออกไปเท่านั้น ทำให้ไม่มีข้อจำกัดในการทำงานกับเครือข่ายที่มีการแปลงเลขที่อยู่เครือข่ายหรือมีการติดตั้งไฟร์วอลล์ การตรวจสอบจะใช้ข้อมูลถึงระดับชั้นขนส่ง แต่ไม่มีการนำหมายเลขพอร์ตเข้ามาเกี่ยวข้องในกระบวนการตรวจสอบ เพราะงานวิจัยนี้อาศัยพฤติกรรมจากขั้นตอนวิธีการเค้น ซึ่งก่อให้เกิดแบบแผนการตอบแทนกัน (Reciprocation scheme) ระหว่างเพียร์ในสวอร์ม (Swarm) และพฤติกรรมการเปลี่ยนคู่ส่งข้อมูลของเพียร์ จึงไม่มีการพิจารณาความสัมพันธ์ในระดับโฟล

เนื่องจากงานวิจัยนี้ทำการพิจารณาในมุมมองของเพียร์เฉพาะที่ ดังนั้นการเชื่อมต่อระหว่างหมายเลขไอพีหรือระหว่างเพียร์ (IP Connection) ในงานวิจัยนี้ จึงหมายถึงการที่หมายเลขไอพี 2 หมายเลข มีการเปิดการเชื่อมต่อถึงกันด้วยโพรโทคอลทีซีพีอย่างน้อย 1 การเชื่อมต่อโดยไม่พิจารณาจากจำนวนโฟล จากตัวอย่างในรูปที่ 3.2 เครื่อง A ถือว่ามีการเชื่อมต่อกับเครื่อง C และ D เนื่องจากเครื่อง A เป็นฝ่ายเปิดการเชื่อมต่อไปยังเครื่อง C และ D แต่เครื่อง A ไม่มีการเปิดการเชื่อมต่อไปยังเครื่อง B ดังนั้นถือว่าเครื่อง A ไม่มีการเชื่อมต่อกับเครื่อง B จากหลักการดังกล่าวแนวคิดที่นำเสนอจึงไม่มีผลกระทบจากอุปกรณ์ไฟร์วอลล์ที่คอยบอกปัดการเชื่อมต่อจากหมายเลขไอพีภายนอก



รูปที่ 3.2 ตัวอย่างของการเชื่อมต่อระหว่างหมายเลขไอพี

ตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ที่ได้ออกแบบในหัวข้อนี้ แบ่งออกเป็น 2 ขั้นตอน คือ ขั้นตอนของการระบุหาเพียร์ที่คาดว่ามีการใช้งานบิตทอร์เรนต์ เพื่อตรวจหาหมายเลขไอพีที่มีพฤติกรรมเหมือนกับแอพลิเคชันของบิตทอร์เรนต์ และขั้นตอนของการติดตามหมายเลขไอพีที่คาดว่ามีการใช้งานบิตทอร์เรนต์ เพื่อตรวจสอบให้แน่ชัดว่าหมายเลขไอพีที่ได้จากขั้นตอนแรกมีการใช้งานโพรโทคอลบิตทอร์เรนต์จริง โดยขั้นตอนทั้ง 2 เป็นไปตามนี้

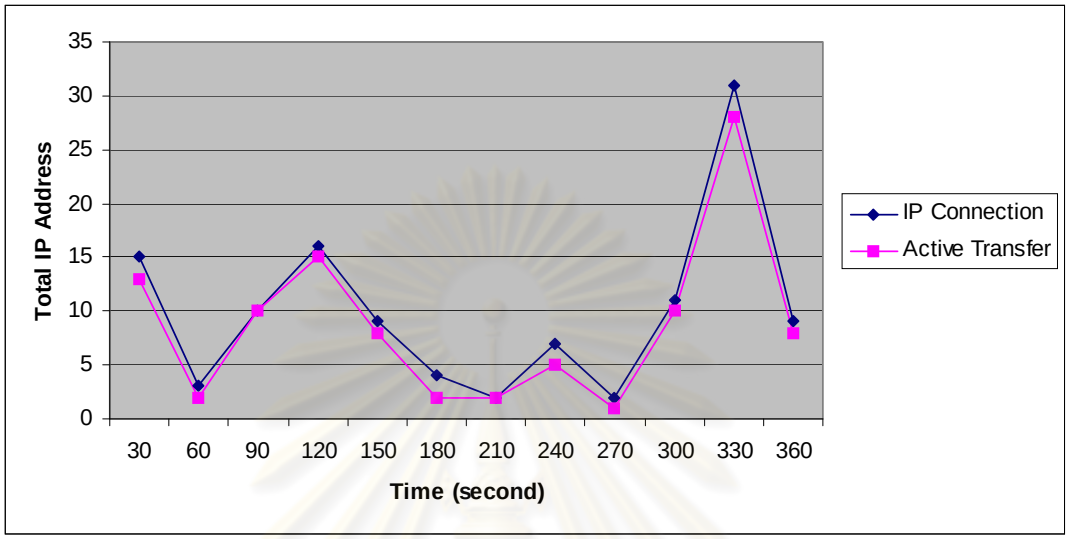
ขั้นตอนที่ 1 การระบุหาเพียร์ที่คาดว่ามีการใช้งานบิตทอร์เรนต์ โดยอาศัยลักษณะเฉพาะของกระแสข้อมูลที่เกิดจากการทำงานของขั้นตอนวิธีการเค้น ซึ่งทำให้กระแสข้อมูลของบิตทอร์เรนต์เกิดความแตกต่างจากกระแสข้อมูลของแอพลิเคชันทั่วไป โดยมีแนวคิดดังต่อไปนี้

1. แอพลิเคชันทั่วไปบนอินเทอร์เน็ต จะมีการเปิดการเชื่อมต่อไปยังหมายเลขไอพีต่างๆ เท่าที่จำเป็นต้องใช้ในเวลานั้น และจำนวนหมายเลขไอพีที่เปิดออกไปก็มีค่อนข้างน้อย เช่น การใช้งานเอฟทีพี จะมีการเปิดการเชื่อมต่อไปยังเครื่องให้บริการเอฟทีพีเพียงหมายเลขไอพีเดียว แต่อาจมีบางแอพลิเคชัน เช่น เวิลด์ไวด์เว็บ ที่ไม่เพียงเปิดการเชื่อมต่อออกไปยังหมายเลขไอพีของเครื่องให้บริการเวิลด์ไวด์เว็บเท่านั้น แต่อาจมีส่วนของการเชื่อมโยง (Link) เพื่อไปดึงข้อมูลจากเว็บไซต์อื่น มาแสดงที่หน้าเว็บของตนเอง [17] เช่น แถบประกาศ (Banner) หรือเว็บเพจที่มีลักษณะการเชื่อมโยงข้ามโดเมน (Domain) ซึ่งหากมีการเปิดเว็บไซต์ที่มีลักษณะนี้พร้อมกันหลายหน้า อาจทำให้มีการเปิดการเชื่อมต่อออกไปยังหมายเลขไอพีที่แตกต่างกันจำนวนมาก

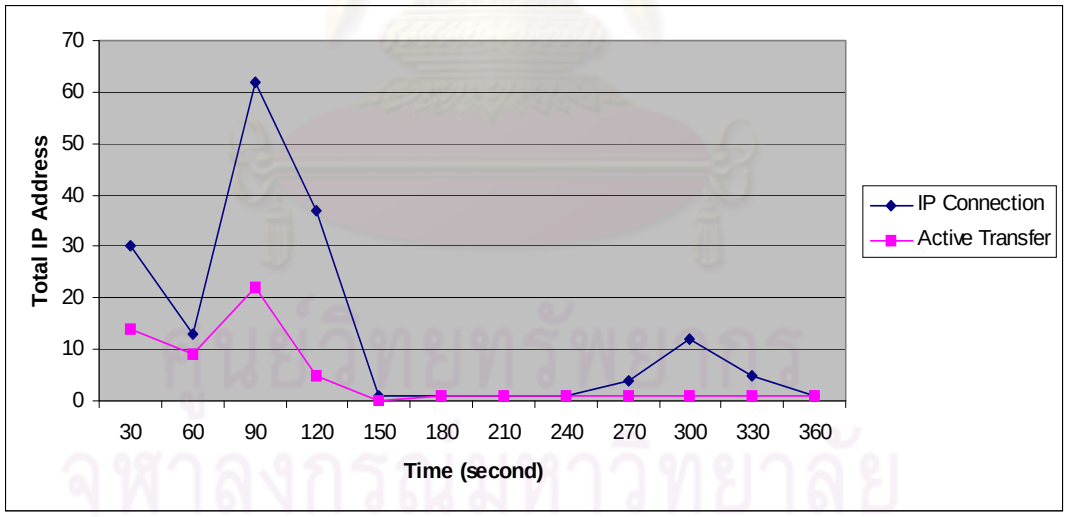
2. แอพลิเคชันทั่วไปบนอินเทอร์เน็ต เมื่อสร้างการเชื่อมต่อสำเร็จจะมีการรับส่งข้อมูลกันทันที เช่น เมื่อมีการเปิดการเชื่อมต่อเพื่อขอใช้บริการเวิลด์ไวด์เว็บ จะมีกระบวนการตามโพรโทคอลเอชทีทีพี (HTTP) ซึ่งมีการร้องขอและการส่งข้อมูลกันเกิดขึ้นทันทีหลังการเปิดการเชื่อมต่อสำเร็จ ดังแสดงในรูปที่ 3.3 ต่างจากบิตทอร์เรนต์ที่ต้องมีการคำนวณตามขั้นตอนวิธีการเค้นที่แต่ละเพียร์ก่อนจึงสามารถรับส่งข้อมูลได้ การเชื่อมต่อระหว่างเพียร์จึงมีการรับส่งเฉพาะข้อความควบคุมเท่านั้น ดังแสดงในรูปที่ 3.4 โดยการเชื่อมต่อที่มีการรับส่งข้อมูลในขั้นตอนที่ 1 หมายถึงการเชื่อมต่อที่มีการรับส่งแพ็กเก็ตขนาดตั้งแต่ 200 กิโลไบต์ (KB) ขึ้นไป [17] จำนวนหนึ่งจึงครอบคลุมถึงการเชื่อมต่อกันเพื่อรับส่งข้อมูลของบริการเวิลด์ไวด์เว็บและแอพลิเคชันอื่นด้วย

จากแนวคิดที่กล่าวมา สามารถนำมาสร้างเป็นวิธีการตรวจหาหมายเลขไอพีที่คาดว่ามีการใช้งานบิตทอร์เรนต์ได้ โดยหากในช่วงเวลา 30 วินาทีใดๆ ของแต่ละหมายเลขไอพี มีอัตราส่วนระหว่างจำนวนการเชื่อมต่อที่มีการรับส่งข้อมูล (Active Transfer) ต่อจำนวนการเชื่อมต่อทั้งหมด (Connected IPs) ไม่ถึงค่าขีดเริ่มเปลี่ยน (Threshold) ของแอพลิเคชันทั่วไปบนอินเทอร์เน็ต ถือว่า

หมายเลขไอพีนั้นเป็นหมายเลขไอพีที่คาดว่ามีการใช้งานบิตทอร์เรนต์ (หรือเป็นเพียร์) จากนั้นจึงติดตามดูพฤติกรรมของหมายเลขไอพีเหล่านี้ เพื่อให้แน่ใจว่ามีการใช้งานบิตทอร์เรนต์จริง



รูปที่ 3.3 เปรียบเทียบการรับส่งข้อมูลกับการเชื่อมต่อทั้งหมดของบริการเวิลด์ไวด์เว็บ



รูปที่ 3.4 เปรียบเทียบการรับส่งข้อมูลกับการเชื่อมต่อทั้งหมดของเพียร์ในบิตทอร์เรนต์

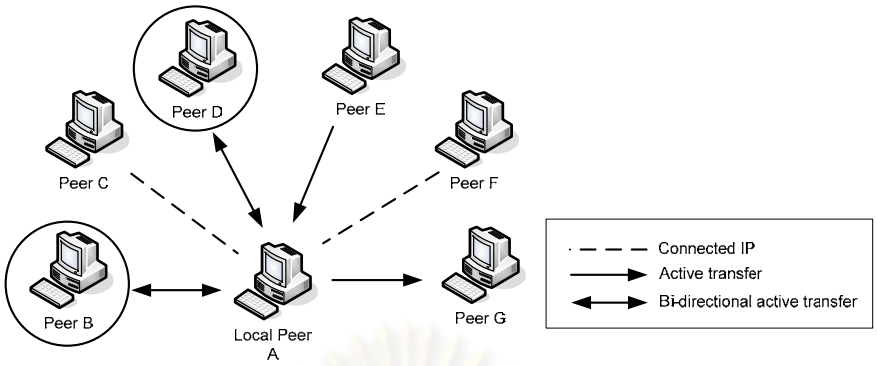
หมายเลขไอพีที่ตรวจจับได้จากขั้นตอนที่ 1 จะถูกกำหนดช่วงของการหมดเวลารอ (Timeout) สำหรับการตรวจสอบในขั้นตอนที่ 2 เพื่อให้มีความต่อเนื่องกับการตรวจสอบจากขั้นตอนแรก และเพื่อให้แน่ใจว่าพฤติกรรมที่ใช้ตรวจจับในทั้งสองขั้นตอนมาจากแอพลิเคชันเดียวกันในช่วงเวลาดังกล่าว ดังนั้นหากไม่สามารถตรวจจับหมายเลขไอพีนี้ด้วยขั้นตอนที่ 2

ภายในเวลาที่กำหนด หมายเลขไอพีดังกล่าวจะถูกยกเลิกการตรวจสอบในขั้นตอนที่ 2 และกลับไปเริ่มตรวจสอบใหม่ตั้งแต่ขั้นตอนแรก

ขั้นตอนที่ 2 การติดตามหมายเลขไอพีที่คาดว่ามีการใช้งานบิตทอร์เรนต์ เนื่องจากอาจมีบางหมายเลขไอพีที่ไม่ได้ใช้งานบิตทอร์เรนต์ แต่กระแสข้อมูลที่ออกมามีลักษณะคล้ายกับกระแสข้อมูลของบิตทอร์เรนต์ในช่วงเปิดการเชื่อมต่อ ในขั้นตอนนี้จึงอาศัยพฤติกรรมของกระแสข้อมูลที่เกิดจากขั้นตอนวิธีการเค้นระหว่างการดาวน์โหลดหรืออัพโหลดข้อมูล ซึ่งแตกต่างจากขั้นตอนแรกที่อาศัยพฤติกรรมช่วงเริ่มเปิดการเชื่อมต่อของเพียร์ โดยมีแนวคิดดังนี้

1. การเชื่อมต่อที่มีการรับส่งข้อมูล (Active Transfer) ของขั้นตอนนี้ คือการเชื่อมต่อที่มีการรับส่งแพ็กเก็ตจำนวนหนึ่งที่มีขนาดใกล้เคียงกับเอ็มทียู (MTU) ของระดับชั้นเชื่อมโยง (Link Layer) เนื่องจากขั้นตอนนี้ติดตามพฤติกรรมของการรับส่งข้อมูล และโพรโทคอลบิตทอร์เรนต์ถูกออกแบบมาสำหรับการแลกเปลี่ยนไฟล์บนระบบเครือข่าย ขนาดของแพ็กเก็ตที่ส่งในแต่ละครั้งจึงควรมีขนาดสูงสุดเพื่อให้การรับส่งมีประสิทธิภาพ ต่างจากแอพลิเคชันอื่น เช่น เกมส์ออนไลน์ (Game Online) ที่มีพฤติกรรมเปิดการเชื่อมต่อจำนวนมากเช่นเดียวกัน (ทำให้อาจตรวจสอบได้จากขั้นตอนแรก) แต่ข้อมูลที่ส่งเป็นเพียงสถานะต่างๆ ที่จำเป็นเท่านั้น ดังนั้นขนาดของแพ็กเก็ตจึงเล็กกว่าขนาดของเอ็มทียูมาก แนวคิดนี้จึงเป็นเหมือนการกรองแอพลิเคชันที่มีแต่การส่งข้อมูลสถานะออกจากกลุ่มของเพียร์ที่มีการรับส่งชิ้นส่วนของไฟล์ ดังนั้นการตรวจสอบในขั้นตอนที่ 2 แอพลิเคชันที่ไม่ได้มีการออกแบบมาสำหรับการรับส่งไฟล์บนระบบเครือข่าย จึงมีจำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูลน้อยกว่าจำนวนหมายเลขไอพีที่ติดต่อด้วยค่อนข้างมาก ซึ่งต่างกับเพียร์ของบิตทอร์เรนต์ที่มีการดาวน์โหลดชิ้นส่วนจากหลายเพียร์พร้อมกัน จึงมีการเชื่อมต่อที่มีการรับส่งข้อมูลจำนวนมาก

2. โพรโทคอลบิตทอร์เรนต์มีความสามารถในการดาวน์โหลดและอัพโหลดในเวลาเดียวกัน เพราะไฟล์ถูกตัดแบ่งออกเป็นชิ้นส่วนย่อย จึงไม่ต้องรอให้ดาวน์โหลดจนครบทั้งไฟล์ก่อน และการทำงานของขั้นตอนวิธีการเค้นทำให้เพียร์เลือกอัพโหลดให้แก่ผู้ที่ตนเองดาวน์โหลดได้เร็วที่สุด ทำให้การเชื่อมต่อระหว่างบางคู่เพียร์มีข้อมูลรับส่งกันทั้งสองทิศทาง (Bi-directional active transfer) ดังนั้นหากหมายเลขไอพีที่ระบุได้จากขั้นตอนแรกมีกลุ่มของการเชื่อมต่อที่มีลักษณะการถ่ายโอนข้อมูลทั้งสองทิศทางในเวลาเดียวกัน ถือว่าหมายเลขไอพีนั้นมีการใช้งานบิตทอร์เรนต์ ดังแสดงในรูปที่ 3.5 เพียร์ A มีการถ่ายโอนข้อมูลสองทิศทางกับเพียร์ B และเพียร์ D

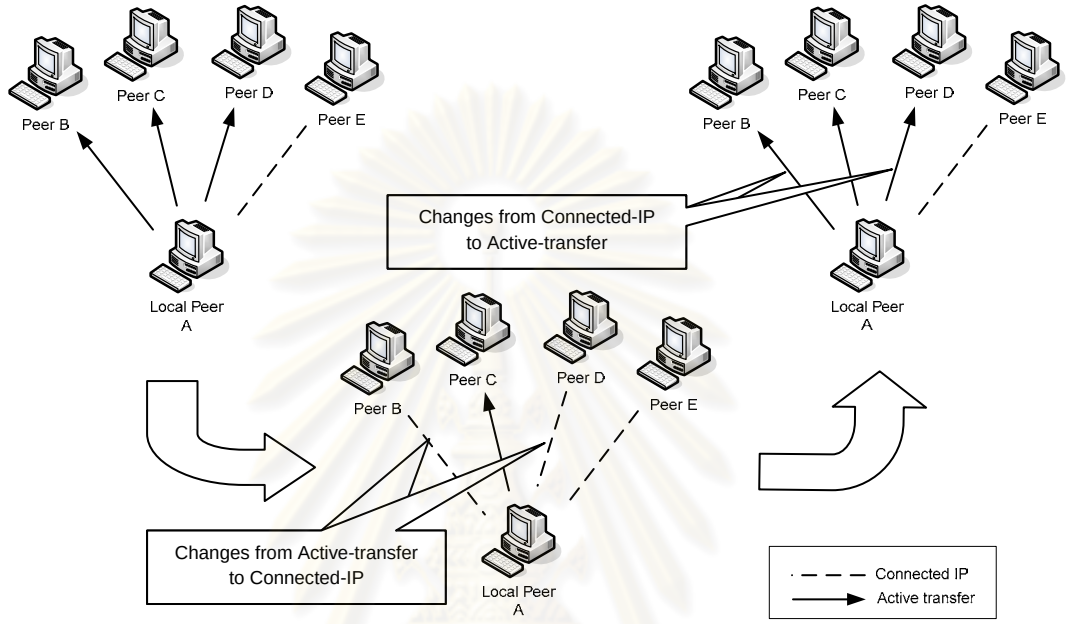


รูปที่ 3.5 การเชื่อมต่อที่มีลักษณะการถ่ายโอนข้อมูลทั้งสองทิศทางในเวลาเดียวกัน

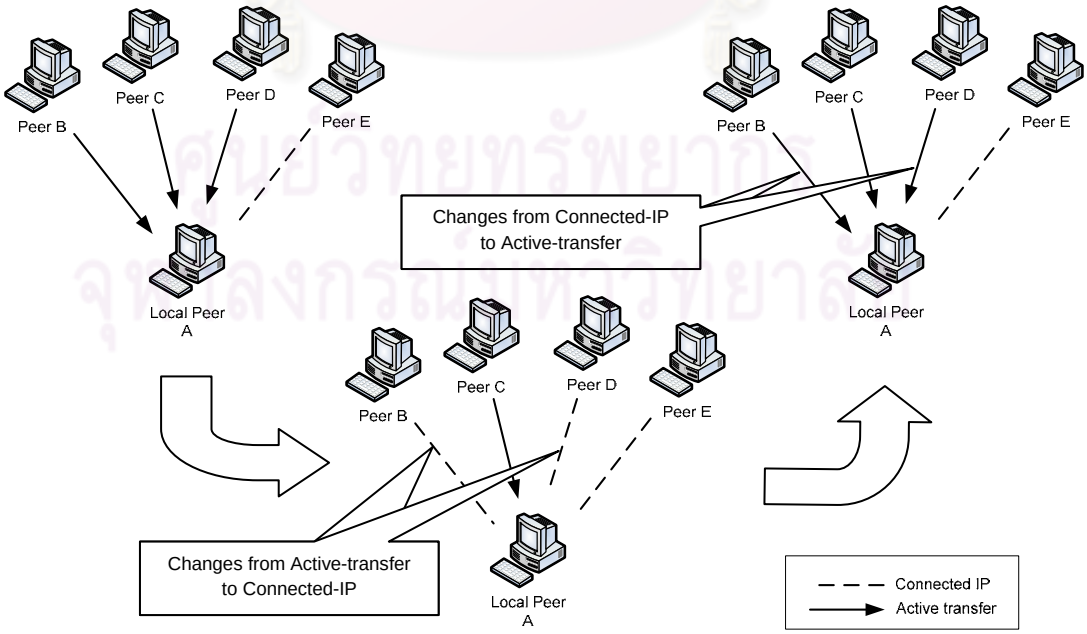
3. จากการทำงานของขั้นตอนวิธีการเค้น เพียร์เฉพาะที่เลือกอัพโหลดให้แก่เพียร์ที่ให้ค่าดาวน์โหลดสูงที่สุด 4 อันดับแรกในแต่ละรอบการคำนวณ แต่ความเร็วในการดาวน์โหลดมีโอกาสเปลี่ยนแปลงได้ตลอดเวลา ดังนั้นเพียร์ที่สามารถดาวน์โหลดข้อมูลจากเพียร์เฉพาะที่ได้ในรอบแรกอาจถูกระงับการดาวน์โหลดชั่วคราวในรอบถัดมา เนื่องจากเพียร์เฉพาะที่อาจพบเพียร์อื่นที่ให้ค่าดาวน์โหลดสูงกว่า จึงเปลี่ยนไปอัพโหลดให้กับเพียร์นั้นแทน แต่ในรอบถัดมาอาจตัดสินใจเลือกอัพโหลดให้เพียร์ในรอบแรกอีกครั้ง ส่งผลให้มีการเปลี่ยนสถานะของความสัมพันธ์ระหว่างคู่หมายเลขไอพี (IP-Relation Changes) ระหว่างความสัมพันธ์ที่มีข้อมูลส่งออกและความสัมพันธ์ที่ไม่มีข้อมูลส่งออก ดังแสดงในรูปที่ 3.6 โดยในรอบแรกของการทำงานของขั้นตอนวิธีการเค้นที่เพียร์ A เพียร์ A เลือกอัพโหลดให้แก่เพียร์ B เพียร์ C และเพียร์ D แต่ในรอบที่ 2 เพียร์ A พบเพียร์อื่นที่ให้ตนเองดาวน์โหลดได้เร็วกว่าเพียร์ B และเพียร์ D จึงระงับการอัพโหลดให้แก่เพียร์ B และเพียร์ D ชั่วคราว (Choke) เพื่อเลือกอัพโหลดให้แก่เพียร์เหล่านั้น และในรอบที่ 3 เพียร์ A พบว่าสามารถดาวน์โหลดจากเพียร์ B และเพียร์ D ได้เร็วกว่าเพียร์ที่กำลังอัพโหลดให้บางตัว หรือมีการทำ Optimistic Unchoke ที่เพียร์ A ซึ่งสุ่มเลือกอัพโหลดให้แก่เพียร์อื่นโดยไม่พิจารณาจากค่าดาวน์โหลดที่ตนเองได้รับ ดังที่ได้อธิบายไว้ในบทที่ 2 หัวข้อที่ 2.3 เพียร์ A จึงกลับมาเลือกอัพโหลดให้แก่เพียร์ B และเพียร์ D อีกครั้งในรอบที่ 3

4. หากมองพฤติกรรมการเปลี่ยนสถานะความสัมพันธ์ระหว่างคู่หมายเลขไอพี ซึ่งเกิดขึ้นกับเพียร์ทั้งหมดในเซตของเพียร์ ทำให้เพียร์เฉพาะที่มีโอกาสถูกปฏิบัติในลักษณะเช่นเดียวกัน นั่นคือมีการเปลี่ยนสถานะของความสัมพันธ์ระหว่างคู่หมายเลขไอพี (IP-Relation Changes) ระหว่างความสัมพันธ์ที่มีข้อมูลรับเข้าและความสัมพันธ์แบบไม่มีข้อมูลรับเข้า ดังแสดงในรูปที่ 3.7 โดยในรอบแรกของขั้นตอนวิธีการเค้น เพียร์ A สามารถดาวน์โหลดชิ้นส่วนของไฟล์ได้จากเพียร์ B เพียร์ C และเพียร์ D แต่ในรอบที่ 2 เพียร์ B และเพียร์ D พบเพียร์อื่นที่ให้ตนเองดาวน์

โหลดได้เร็วกว่าเพียร์ A จึงระงับการอัพโหลดให้แก่เพียร์ A ชั่วคราว เพื่ออัพโหลดให้แก่เพียร์ที่ค้นพบใหม่ และในรอบที่ 3 ทั้งเพียร์ B และเพียร์ D พบว่าเพียร์ A ให้ตนเองดาวน์โหลดได้เร็วกว่าเพียร์ที่กำลังอัพโหลดให้บางตัว หรืออาจเลือกเพียร์ A จากการทำ Optimistic Unchoke จึงยอมอัพโหลดให้แก่เพียร์ A อีกครั้ง



รูปที่ 3.6 การเปลี่ยนสถานะระหว่างความสัมพันธ์ที่มีข้อมูลส่งออกและไม่มีข้อมูลส่งออก



รูปที่ 3.7 การเปลี่ยนสถานะระหว่างความสัมพันธ์ที่มีข้อมูลรับเข้าและไม่มีข้อมูลรับเข้า

จากพฤติกรรมการเปลี่ยนสถานะของความสัมพันธ์ระหว่างคู่หมายเลขไอพี สามารถนำมาติดตามพฤติกรรมของหมายเลขไอพีที่คาดว่ามีการใช้งานบิตทอร์เรนต์ได้ โดยแต่ละหมายเลขไอพีที่ตรวจสอบได้จากขั้นตอนแรก ในทุกๆ 30 วินาที หากพบว่ามีการเชื่อมต่อจำนวนหนึ่งที่มีการเปลี่ยนสถานะของความสัมพันธ์ระหว่างคู่หมายเลขไอพี ถือว่าหมายเลขไอพีนั้นมีการใช้งานบิตทอร์เรนต์ เพราะถึงแม้จะมีแอพลิเคชันใดที่มีพฤติกรรมคล้ายกับบิตทอร์เรนต์ในช่วงของการเปิดการเชื่อมต่อ แต่แอพลิเคชันนั้นไม่มีพฤติกรรมการเปลี่ยนคู่ส่งเช่นเดียวกับบิตทอร์เรนต์ หากดาวน์โหลดหรืออัพโหลดกับหมายเลขไอพีใด ก็จะดาวน์โหลดหรืออัพโหลดกับหมายเลขไอพีนั้นจนสำเร็จ ทำให้ไม่มีพฤติกรรมของการเปลี่ยนสถานะของความสัมพันธ์ระหว่างคู่หมายเลขไอพีระหว่างความสัมพันธ์ที่มีข้อมูลรับส่งและความสัมพันธ์ที่ไม่มีข้อมูลรับส่ง

จากการตรวจสอบทั้งสองขั้นตอนดังกล่าว ทำให้สามารถสร้างตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์โดยใช้ข้อมูลถึงระดับชั้นขนส่ง แต่ไม่มีการนำหมายเลขพอร์ตเข้ามาเกี่ยวข้องในกระบวนการตรวจสอบ จึงสามารถลดสถานะที่ต้องจำลงได้จำนวนมาก อีกทั้งในการตรวจสอบยังพิจารณาเฉพาะการเชื่อมต่อที่เพียร์เฉพาะที่เป็นผู้ทำการเปิดการเชื่อมต่อออกไปเท่านั้น ทำให้สามารถทำงานได้ในสภาพแวดล้อมที่หลากหลาย

อย่างไรก็ตาม ตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ที่สร้างขึ้นในหัวข้อนี้ยังมีข้อจำกัดอยู่บางประการ ได้แก่

1. การพิจารณาเฉพาะการเชื่อมต่อที่เพียร์เฉพาะที่สร้างขึ้น ถึงแม้ว่าสามารถหลีกเลี่ยงข้อจำกัดที่เกิดขึ้นจากอุปกรณ์ไฟร์วอลล์หรือการแปลงเลขที่อยู่เครือข่าย แต่อาจทำให้ข้อมูลที่เกี่ยวข้องบางส่วนหายไปได้ ยกตัวอย่างเช่น การที่ผู้ใช้งานบิตทอร์เรนต์ใช้หมายเลขพอร์ตรอรับการเชื่อมต่อที่ได้รับอนุญาตให้ผ่านเข้ามาได้ เช่น หมายเลขพอร์ต 80 ทำให้บางการเชื่อมต่อระหว่างหมายเลขไอพีอาจเป็นการเชื่อมต่อที่สร้างขึ้นโดยหมายเลขไอพีที่อยู่ภายนอก แต่แนวคิดที่ใช้ในการพิจารณาในหัวข้อนี้จะไม่พิจารณาการเชื่อมต่อระหว่างหมายเลขไอพีเหล่านี้

2. การตรวจสอบในขั้นตอนแรกของตัวระบุเพียร์อาศัยพฤติกรรมการเปิดการเชื่อมต่อของเพียร์ ซึ่งปรากฏชัดเจนที่สุดในช่วงเริ่มต้น เนื่องจากมีการเปิดการเชื่อมต่อไปยังแต่ละหมายเลขไอพีที่ได้รับมาจากแทรกเกอร์ หลังจากนั้นอาจมีการเปิดการเชื่อมต่อไปยังเพียร์ใหม่เป็นระยะ แต่อาจไม่ชัดเจนจนสามารถสังเกตได้เหมือนในช่วงแรกดังแสดงในรูปที่ 3.4 ดังนั้นหากพฤติกรรมที่เกิดจากการรับส่งข้อมูล ซึ่งใช้ในการยืนยันหมายเลขไอพีที่คาดว่ามีการใช้งานบิตทอร์เรนต์จากขั้นตอน

แรก เกิดขึ้นหลังจากช่วงแรกของการเปิดการเชื่อมต่อเป็นเวลานาน อาจทำให้ไม่สามารถระบุเพียร์เหล่านี้ได้ เพราะหมายเลขไอพีที่ได้จากขั้นตอนแรกอาจหมดเวลารอไปก่อน

3.2.2 การปรับปรุงพฤติกรรมของกระแสข้อมูลที่เกิดจากขั้นตอนวิธีการเค้น

จากข้อจำกัดเกี่ยวกับพฤติกรรมที่ใช้ในการสร้างตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ในหัวข้อที่ 3.2.1 จึงได้ปรับปรุงแนวคิดบางอย่างเกี่ยวกับพฤติกรรมของกระแสข้อมูลบิตทอร์เรนต์ที่เกิดจากขั้นตอนวิธีการเค้น เพื่อให้กระบวนการตรวจสอบมีประสิทธิภาพมากขึ้น ดังนี้

1. พิจารณาทุกการเชื่อมต่อระหว่างเพียร์ โดยไม่แบ่งแยกว่าเป็นการเชื่อมต่อที่เพียร์เฉพาะที่เป็นผู้สร้างหรือเพียร์ที่อยู่ด้านนอกเป็นผู้สร้าง ทำให้ไม่สูญเสียข้อมูลที่เกี่ยวข้องบางส่วนที่เกิดจากเพียร์ด้านนอกเป็นฝ่ายทำการติดต่อเข้ามา

2. ไม่มีการนำ Flag เช่น SYN ในส่วนหัวของโพรโทคอลทีซีพีมาเกี่ยวข้องในการพิจารณา โดยแนวคิดที่ปรับปรุงใหม่นี้ใช้การพิจารณาจากขนาดของแพ็กเก็ตทีซีพีเพียงอย่างเดียว จึงสามารถทำงานในสภาพแวดล้อมต่างๆ เช่น มีการติดตั้งอุปกรณ์ไฟร์วอลล์ หรือมีการทำการแปลงเลขที่อยู่เครือข่ายได้ ดังนั้นตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ที่ได้จากงานวิจัยนี้จึงใช้ข้อมูลจากแพ็กเก็ตถึงระดับชั้นเครือข่ายเท่านั้น

3. ทำการพิจารณาในขั้นตอนเดียว โดยใช้เกณฑ์วัดของแต่ละพฤติกรรมเป็นตัวประเมินว่ากระแสข้อมูลของแต่ละหมายเลขไอพีมีพฤติกรรมของบิตทอร์เรนต์หรือไม่ จึงไม่มีปัญหาเรื่องช่วงเวลาการเกิดของแต่ละพฤติกรรม

3.2.3 การสร้างเกณฑ์วัดสำหรับแต่ละพฤติกรรม

โดยสรุปแล้ว ในงานวิจัยนี้นำเสนอพฤติกรรมของกระแสข้อมูลบิตทอร์เรนต์ 4 อย่าง เพื่อนำมาสร้างเป็นตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ ได้แก่

1. จำนวนของหมายเลขไอพีที่ติดต่อด้วย (Connected IPs: *C*) เนื่องจากเพียร์จะมีการติดต่อกับหมายเลขไอพีอื่นจำนวนมากอยู่ตลอดเวลา โดยความหมายของการติดต่อกันระหว่างหมายเลขไอพีในงานวิจัยนี้ คือ มีการส่งแพ็กเก็ตทีซีพีอย่างน้อย 1 แพ็กเก็ตทั้งสองทิศทาง เพราะระหว่างที่เพียร์เข้าร่วมการดาวน์โหลด จะมีการแลกเปลี่ยนข้อความกับเพียร์อื่นๆ ในเซ็ตของเพียร์ เช่น ข้อความ Have หรือข้อความ Interested อยู่เป็นระยะ อีกทั้งเพียร์จะพยายามรักษาจำนวน

เพียร์ในเซตของเพียร์ไม่ให้ต่ำกว่า 20 จึงเกิดพฤติกรรมการติดต่อกับหมายเลขไอพีจำนวนมากอยู่ตลอดเวลา

2. จำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูล (Active Transfers: *A*) เนื่องจากโพรโทคอลบิตทอร์เรนต์มีการตัดแบ่งไฟล์ออกเป็นชิ้นส่วนย่อย เพียร์จึงสามารถดาวน์โหลดแต่ละชิ้นส่วนจากเพียร์ที่ต่างกันได้ในเวลาเดียวกัน จึงทำให้มีการเชื่อมต่อที่มีการรับส่งข้อมูลเป็นอัตราส่วนโดยตรงกับจำนวนของหมายเลขไอพีที่ติดต่อด้วย โดยความหมายของการเชื่อมต่อที่มีการรับส่งข้อมูลในงานวิจัยนี้ คือ มีการส่งแพ็กเก็ตทีซีพีขนาดใกล้เคียงกับเอ็มทียูตั้งแต่ 5 แพ็กเก็ตขึ้นไป ต่างกับเพียร์ทูเพียร์ที่ไม่มีการแบ่งไฟล์ออกเป็นชิ้นส่วนย่อย เช่น นูเทลล่า หรือโซลซีค (Soulseek) ซึ่งต้องดาวน์โหลดทั้งไฟล์จากเพียร์ต้นทางเพียงเพียร์เดียว

3. จำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูลสองทิศทาง (Bi-directional Active Transfers: *B*) โพรโทคอลบิตทอร์เรนต์มีขั้นตอนวิธีการเค้นคอยควบคุมการดาวน์โหลดและอัพโหลด โดยพยายามเลือกอัพโหลดให้แก่เพียร์ที่ตนเองดาวน์โหลดได้เร็วที่สุด ทำให้การเชื่อมต่อระหว่างบางคู่เพียร์มีข้อมูลรับส่งกันทั้งสองทิศทาง ดังที่ได้อธิบายในรูปที่ 3.5

4. จำนวนของความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลง (IP-relation Changes: *R*) ขั้นตอนวิธีการเค้นในโพรโทคอลบิตทอร์เรนต์ก่อให้เกิดพฤติกรรมของการค้นหาเพียร์ที่ให้ค่าดาวน์โหลดที่ดีกว่า จึงมีการเปลี่ยนคู่ถ่ายโอนข้อมูลอย่างต่อเนื่อง ทำให้มีพฤติกรรมของการเปลี่ยนสถานะความสัมพันธ์ของคู่หมายเลขไอพี ระหว่างความสัมพันธ์ที่มีข้อมูลรับส่งและความสัมพันธ์ที่ไม่มีข้อมูลรับส่ง ดังที่ได้อธิบายในรูปที่ 3.6 และรูปที่ 3.7

พฤติกรรมของกระแสข้อมูลบิตทอร์เรนต์ทั้ง 4 อย่าง สามารถนำมาสร้างเป็นเกณฑ์วัดเพื่อนำไปใช้ในการระบุเพียร์ได้ดังนี้

<u>เกณฑ์ที่ 1</u> จำนวนของหมายเลขไอพีที่ติดต่อด้วย (Connected IPs: *C*) หมายเลขไอพีที่ใช้งานบิตทอร์เรนต์จะมีจำนวนของหมายเลขไอพีที่ติดต่อด้วยเท่ากับหรือมากกว่าค่าขีดเริ่มเปลี่ยนของเกณฑ์วัดนี้ ($C_{threshold}$) ดังสมการที่ 3.1

$$C \geq C_{threshold} \tag{3.1}$$

<u>เกณฑ์ที่ 2</u> อัตราส่วนของการเชื่อมต่อที่มีการรับส่งข้อมูล (Active Transfer ratio: r_A) หมายเลขไอพีที่ใช้งานบิตทอร์เรนต์จะมีจำนวนการเชื่อมต่อที่มีการรับส่งข้อมูลเป็นอัตราส่วน

โดยตรงกับหมายเลขไอพีที่ติดต่อด้วย ทำให้มีอัตราส่วน (ratio) ระหว่างพฤติกรรมทั้งสองเท่ากับหรือมากกว่าค่าขีดเริ่มเปลี่ยนของเกณฑ์วัดนี้ ($r_{Athreshold}$) ดังสมการที่ 3.2

$$r_A \geq r_{Athreshold} \tag{3.2}$$

โดยที่

$$r_A = \frac{A}{C} \tag{3.3}$$

<u>เกณฑ์ที่ 3</u> จำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูลสองทิศทาง (Bi-directional Active Transfers: *B*) หมายเลขไอพีที่ใช้งานบิตทอร์เรนต์จะมีจำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูลสองทิศทางเท่ากับหรือมากกว่าค่าขีดเริ่มเปลี่ยนของเกณฑ์วัดนี้ ($B_{threshold}$) ดังสมการที่ 3.4

$$B \geq B_{threshold} \tag{3.4}$$

<u>เกณฑ์ที่ 4</u> อัตราส่วนความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลง (IP-relation Change ratio: r_R) หมายเลขไอพีที่ใช้งานบิตทอร์เรนต์จะมีจำนวนของความสัมพันธ์ระหว่างคู่หมายเลขไอพีที่เปลี่ยนแปลง เป็นอัตราส่วนโดยตรงกับจำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูล ทำให้มีอัตราส่วนระหว่างพฤติกรรมทั้งสองเท่ากับหรือมากกว่าค่าขีดเริ่มเปลี่ยนของเกณฑ์วัดนี้ ($r_{Rthreshold}$) ดังสมการที่ 3.5

$$r_R \geq r_{Rthreshold} \tag{3.5}$$

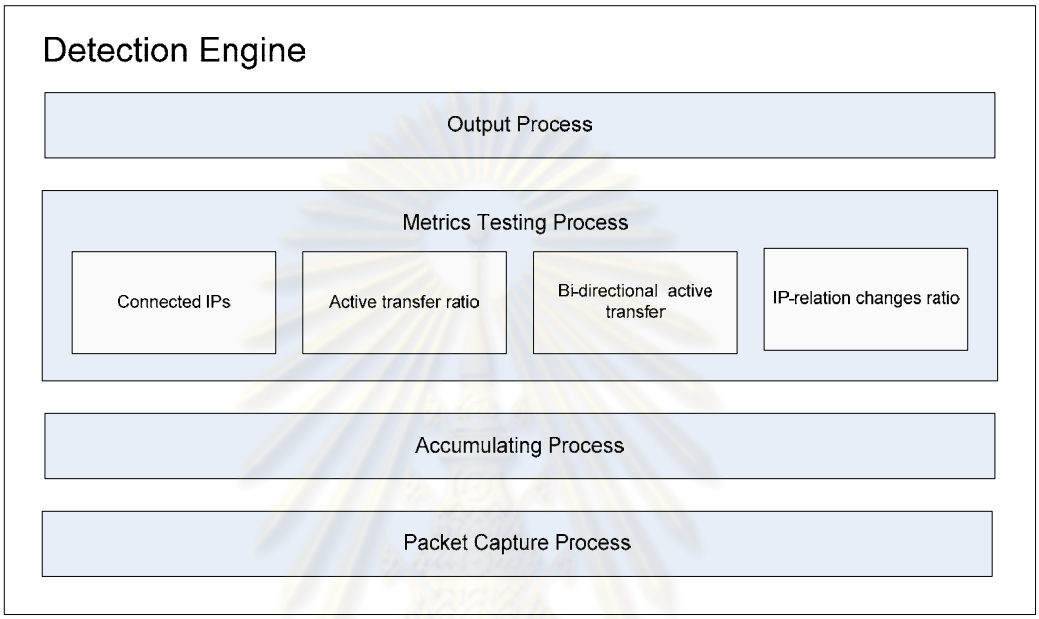
โดยที่

$$r_R = \frac{R}{A} \tag{3.6}$$

เกณฑ์วัดแต่ละตัวมีการทำงานเป็นอิสระจากกัน และมีผลการทดสอบเป็นจริง (เป็นเพียร์ของบิตทอร์เรนต์) หรือเท็จ (ไม่ใช่เพียร์ของบิตทอร์เรนต์) เท่านั้น โดยผลการทดสอบจากแต่ละเกณฑ์วัดมีค่าถ่วงน้ำหนัก (Weight) เท่ากันทั้งหมด การตรวจสอบหมายเลขไอพีในแต่ละรอบการคำนวณ จะมีการส่งข้อมูลที่เกี่ยวข้องไปยังทุกเกณฑ์วัดเพื่อใช้คำนวณค่าเกณฑ์วัดเพื่อเปรียบเทียบกับค่าขีดเริ่มเปลี่ยนที่ได้กำหนดไว้ หมายเลขไอพีที่จะถูกระบุว่ามีการใช้งานบิตทอร์เรนต์ต้องมีผลการทดสอบจากเกณฑ์วัดทุกตัวเป็นจริงเท่านั้น

3.2.4 การสร้างตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์จากเกณฑ์วัด

เกณฑ์วัดทั้ง 4 ที่ได้จากหัวข้อที่ 3.2.3 สามารถนำมาสร้างเป็นตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ได้ดังแสดงในรูปที่ 3.8 ซึ่งประกอบด้วยกระบวนการ (Process) ต่างๆ ดังนี้



รูปที่ 3.8 องค์ประกอบของตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ที่นำเสนอ

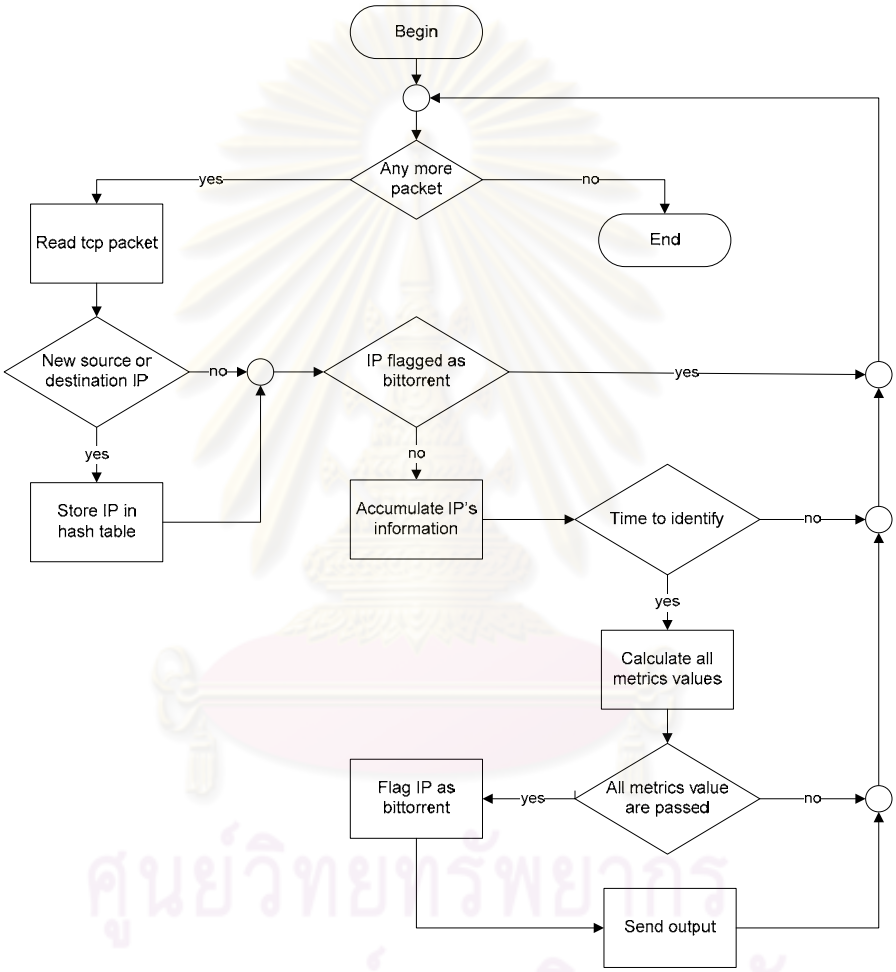
1. กระบวนการดักจับแพ็กเก็ต (Packet Capture Process) ทำหน้าที่ดักจับแพ็กเก็ตทีซีพีจากระบบเครือข่าย และทำการถอดรหัส (Decode) ส่วนหัวของแพ็กเก็ตถึงระดับชั้นเครือข่าย

2. กระบวนการเก็บรวบรวมข้อมูล (Accumulating Process) ทำหน้าที่เก็บรวบรวมข้อมูลที่เกี่ยวข้องของแต่ละเกณฑ์วัด

3. กระบวนการทดสอบค่าเกณฑ์วัด (Metrics Testing Process) ทำหน้าที่คำนวณค่าเกณฑ์วัดของแต่ละพฤติกรรม เพื่อนำไปเปรียบเทียบกับค่าขีดเริ่มเปลี่ยนของแต่ละเกณฑ์วัด โดยแต่ละเกณฑ์วัดเป็นอิสระจากกัน จึงสามารถนำแต่ละเกณฑ์วัดมาสร้างเป็นเกณฑ์วัดร่วม (Combined metric) เพื่อเพิ่มประสิทธิภาพและเหมาะสมกับสภาพแวดล้อมของระบบเครือข่ายที่ใช้งาน

4. กระบวนการส่งออก (Output Process) ทำหน้าที่ส่งหมายเลขไอพีที่ตรวจจับได้ว่ามีการใช้งานบิตทอร์เรนต์ไปยังไปยังกระบวนการที่เกี่ยวข้องต่อไป

ตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ที่สร้างขึ้น มีการคำนวณค่าเกณฑ์วัดของแต่ละพฤติกรรมทุก 30 วินาที เพื่อนำไปเทียบกับค่าขีดเริ่มเปลี่ยนของแต่ละเกณฑ์วัด โดยค่าขีดเริ่มเปลี่ยนของแต่ละเกณฑ์วัดสามารถปรับเปลี่ยนได้ เพื่อให้เหมาะสมกับสภาพแวดล้อมของระบบเครือข่ายแต่ละแห่ง ผังงาน (Flowchart) การทำงานของตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ในงานวิจัยนี้แสดงได้ดังรูปที่ 3.9



รูปที่ 3.9 ผังงานของตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์

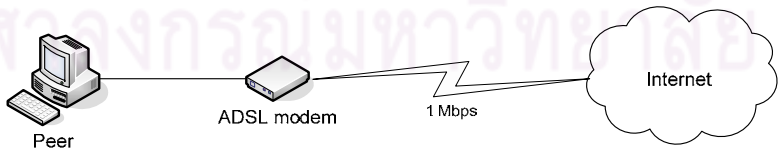
จากผังงานในรูปที่ 3.9 ตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์จะทำการประมวลผลแพ็กเก็ตทีซีพีที่เข้ามาในระบบทีละแพ็กเก็ต หมายเลขไอพีต้นทาง (Source IP address) และหมายเลขไอพีปลายทาง (Destination IP address) จะถูกจัดเก็บในตารางแฮช (Hash table) จากนั้น หมายเลขไอพีต้นทางและหมายเลขไอพีปลายทางที่ยังไม่ถูกระบุว่ามีการใช้งานบิตทอร์เรนต์ จะเข้าสู่กระบวนการเก็บรวบรวมข้อมูลที่เกี่ยวข้องของแต่ละพฤติกรรม และหากถึงกำหนดเวลาการตรวจสอบของหมายเลขไอพี จะมีการคำนวณค่าของแต่ละเกณฑ์วัดจากข้อมูลที่ได้เก็บรวบรวมไว้

จากนั้นนำไปเปรียบเทียบกับค่าขีดเริ่มเปลี่ยนของแต่ละเกณฑ์วัด หากหมายเลขไอพีใดมีค่าของเกณฑ์วัดมากกว่าหรือเท่ากับค่าขีดเริ่มเปลี่ยนในทุกเกณฑ์วัด ถือว่าหมายเลขไอพีนั้นมีการใช้งานบิตทอร์เรนต์ โดยมีค่าตัวบ่งชี้กำกับที่หมายเลขไอพี จากนั้นจึงส่งออกไปยังกระบวนการที่เกี่ยวข้อง เช่น นำไปสร้างเป็นกฎ (Rule) เพื่อเพิ่มเข้าไปในไฟร์วอลล์สำหรับควบคุมการรับส่งข้อมูลของหมายเลขไอพีนั้น ซึ่งอยู่นอกเหนือขอบเขตของงานวิจัยนี้

3.2.5 การเปรียบเทียบพฤติกรรมของกระแสข้อมูลที่เกิดจากการเค้นของบิตทอร์เรนต์ไคลเอนต์และแอพลิเคชันอื่น

หัวข้อนี้เป็นการเปรียบเทียบพฤติกรรมของกระแสข้อมูลที่เกิดจากการเค้นในบิตทอร์เรนต์ไคลเอนต์แต่ละตัว เพื่อตรวจสอบว่ามีพฤติกรรมทั้ง 4 อย่างที่นำเสนอในหัวข้อที่ 3.2.3 เหมือนหรือแตกต่างกันอย่างไร และเกณฑ์วัดที่สร้างขึ้นจะสามารถนำไปตรวจจับบิตทอร์เรนต์ไคลเอนต์ที่มีอยู่ในปัจจุบันได้หรือไม่ รวมถึงการเปรียบเทียบกับพฤติกรรมของกระแสข้อมูลที่เกิดจากแอพลิเคชันอื่น เพื่อตรวจสอบว่ามีความเหมือนหรือแตกต่างจากบิตทอร์เรนต์ไคลเอนต์อย่างไร และเกณฑ์วัดที่นำเสนอจะสามารถแยกแยะความแตกต่างได้หรือไม่

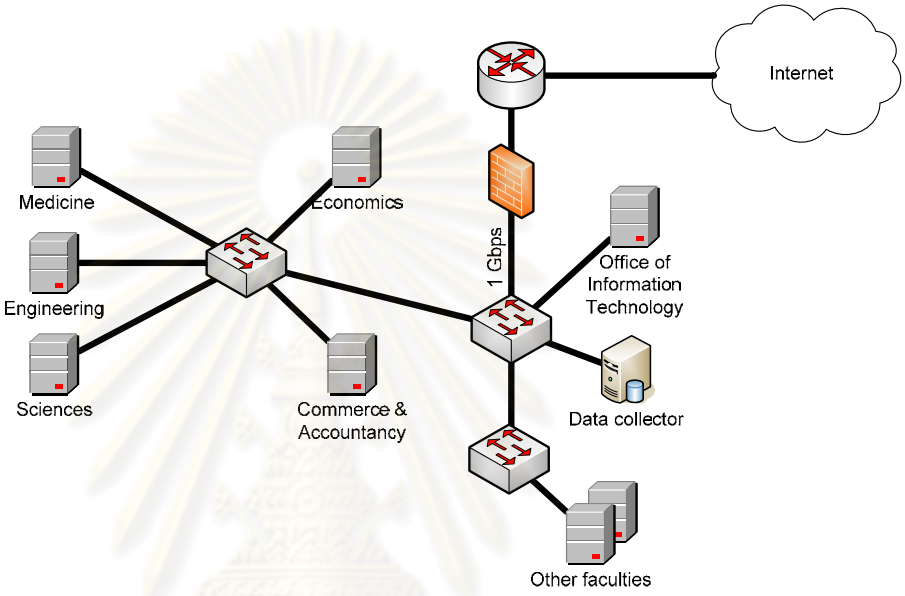
บิตทอร์เรนต์ไคลเอนต์ที่นำมาเปรียบเทียบในหัวข้อนี้ได้แก่ BitComet, BitTorrent, BitTornado, uTorrent และ Vuze (ชื่อเดิมคือ Azureus) เนื่องจากเป็นบิตทอร์เรนต์ไคลเอนต์ที่ได้รับความนิยม มีความสามารถในการเข้ารหัส และมีการนำไปพัฒนาต่อเป็นบิตทอร์เรนต์ไคลเอนต์ตัวอื่น โดยกระแสข้อมูลที่นำมาใช้ในการเปรียบเทียบได้จากการดาวน์โหลดไฟล์ทอร์เรนต์ของระบบปฏิบัติการลีนุกส์เฟโดราคอร์ 9 (Fedora Core 9) ขนาดไฟล์เท่ากับ 3 กิกะไบต์ ผ่านทางอินเทอร์เน็ตความเร็ว 1 เมกกะบิตต่อวินาที ที่ต่อโดยตรงกับโมเด็มเอดีเอสแอลโดยไม่มีการแปลงเลขที่อยู่เครือข่าย และไม่มีอุปกรณ์หรือโปรแกรมไฟร์วอลล์ติดตั้งอยู่ ดังในรูปที่ 3.10



รูปที่ 3.10 แผนภาพแสดงการเก็บกระแสข้อมูลบิตทอร์เรนต์จากไคลเอนต์

ส่วนกระแสข้อมูลของแอพลิเคชันทั่วไป ได้จากการดักจับแพ็กเก็ตจากอุปกรณ์เครือข่ายหลักที่สำนักเทคโนโลยีสารสนเทศ จุฬาลงกรณ์มหาวิทยาลัย โดยจุดที่เก็บข้อมูลเป็นจุดเชื่อมต่อ

ระหว่างเครือข่ายภายนอกและเครือข่ายภายในมหาวิทยาลัย และมีความเร็วเท่ากับ 1 กิกะบิตต่อวินาที ดังแสดงในรูปที่ 3.11 กระแสข้อมูลปกติที่นำมาเปรียบเทียบเป็นกระแสข้อมูลที่เกิดจากเครื่องให้บริการต่างๆ ของคณะวิศวกรรมศาสตร์ (Engineering) และสำนักเทคโนโลยีสารสนเทศ (Office of Information Technology: IT) ซึ่งแต่ละเครื่องอาจมีมากกว่า 1 แอพลิเคชันทำงานอยู่ เช่น บริการเว็บไซต์ บริการเอฟทีพี และบริการโดเมนเนม อยู่บนเครื่องเดียวกัน (Web+FTP+DNS)



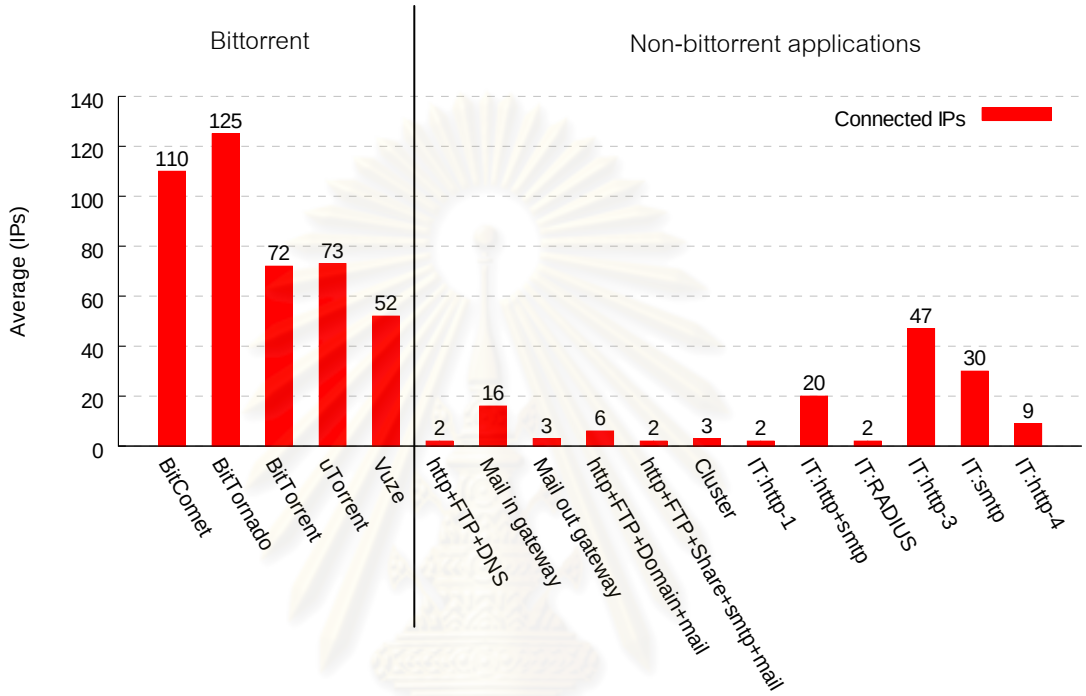
รูปที่ 3.11 แผนภาพแสดงการเก็บกระแสข้อมูลของแอพลิเคชันทั่วไป

เนื่องจากการเปรียบเทียบการเค้นในหัวข้อนี้ต้องการสำรวจพฤติกรรมของแอพลิเคชันปกติในช่วงที่มีการใช้งานสูงสุด เพราะมีโอกาสที่กระแสข้อมูลที่เกิดขึ้นจะมีลักษณะใกล้เคียงกับของบิตทอร์เรนต์ จึงทำการตรวจสอบกระแสข้อมูลที่เกิดขึ้นในช่วง 11:00-17:00 น. เท่านั้น เพราะเป็นช่วงที่มีการใช้งานแอพลิเคชันเหล่านี้มากที่สุด

ในการเปรียบเทียบจะใช้ค่าเฉลี่ยของแต่ละเกณฑ์วัดที่คำนวณได้จากแต่ละไคลเอนต์และแอพลิเคชัน ซึ่งได้แก่ จำนวนของหมายเลขไอพีที่ติดต่อด้วย อัตราส่วนของการเชื่อมต่อที่มีการรับส่งข้อมูล จำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูลสองทิศทาง และอัตราส่วนความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลง โดยมีผลการเปรียบเทียบของแต่ละเกณฑ์วัดดังนี้

1. เปรียบเทียบจำนวนของหมายเลขไอพีที่ติดต่อด้วย (Connected IPs: *C*) เนื่องจากเพียร์ของบิตทอร์เรนต์จะพยายามรักษาการเชื่อมต่อกับเพียร์อื่นจำนวนมากตลอดเวลา จึงมีจำนวนหมายเลขไอพีที่ติดต่อด้วยโดยเฉลี่ยมากกว่า 60 หมายเลข ดังแสดงในรูปที่ 3.12 โดย

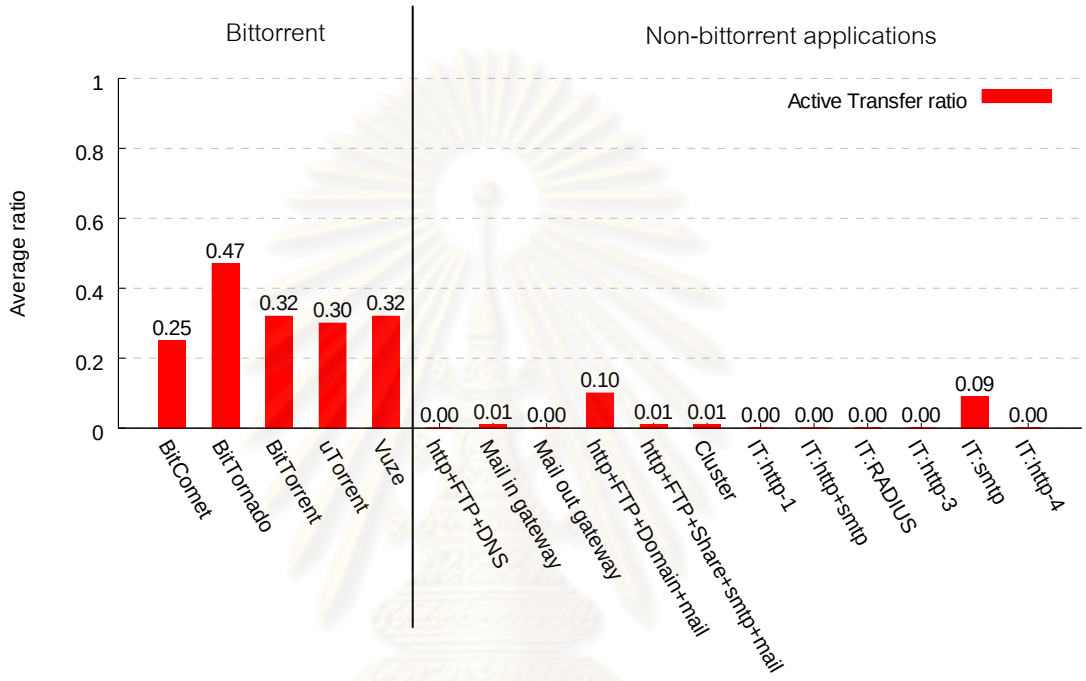
สาเหตุที่แต่ละไคลเอนต์มีจำนวนเฉลี่ยแตกต่างกันเป็นเพราะมีผู้พัฒนาต่างทีมกัน ทำให้มีคุณสมบัติในการรักษาการเชื่อมต่อกับเพียร์อื่นแตกต่างกัน แต่โดยภาพรวมแล้วบิตทอร์เรนต์ไคลเอนต์ทุกตัวมีความแตกต่างจากแอพลิเคชันอื่นอย่างเห็นได้ชัด



รูปที่ 3.12 เปรียบเทียบจำนวนของหมายเลขไอพีที่ติดต่อด้วย (C)

สำหรับแอพลิเคชันทั่วไปส่วนใหญ่ แม้จะมีบางช่วงเวลาที่มีการติดต่อกับหมายเลขไอพีจำนวนมาก แต่แอพลิเคชันเหล่านี้ไม่ได้รักษาการเชื่อมต่อกับหมายเลขไอพีเหล่านั้นไว้ จำนวนหมายเลขไอพีที่ติดต่อด้วยโดยเฉลี่ยจึงน้อยกว่าของบิตทอร์เรนต์ไคลเอนต์ แต่ยังมีบางเครื่องที่มีจำนวนหมายเลขไอพีที่ติดต่อด้วยเฉลี่ยจำนวนมาก ได้แก่ เครื่องให้บริการเอชทีทีพีและเอสเอ็มทีพีของสำนักเทคโนโลยีสารสนเทศ (IT: http-3 และ IT: smtp ตามลำดับ) เนื่องจากมีผู้เข้าชมเว็บไซต์จำนวนมากในช่วง 11:00-17:00 น. จึงทำให้เครื่องให้บริการเอชทีทีพีมีการติดต่อกับหมายเลขไอพีอื่นจำนวนมาก ส่วนบริการเอสเอ็มทีพีเป็นบริการที่ไม่ขึ้นอยู่กับผู้ใช้ว่ากำลังใช้งานอยู่หรือไม่ แต่ขึ้นอยู่กับจำนวนของบัญชีผู้ใช้ (Account) จึงมีโอกาสที่จะมีการติดต่อกับเครื่องให้บริการอีเมล์ภายนอกได้ตลอดเวลา โดยจากผลการเปรียบเทียบสามารถสรุปได้ว่าเกณฑ์วัด C สามารถแยกแยะกระแสข้อมูลของบิตทอร์เรนต์และกระแสข้อมูลของแอพลิเคชันปกติได้ดีในระดับหนึ่ง แต่ยังไม่สามารถแยกแยะความแตกต่างกับเครื่องให้บริการที่มีผู้ใช้งานจำนวนมากได้

2. เปรียบเทียบอัตราส่วนของการเชื่อมต่อที่มีการรับส่งข้อมูล (Active Transfer ratio: r_A) เนื่องจากเพียร์ของบิตทอร์เรนต์สามารถดาวน์โหลดชิ้นส่วนของไฟล์จากหลายเพียร์ได้พร้อมกัน จึงมีอัตราส่วนของการเชื่อมต่อที่มีการรับส่งข้อมูลโดยเฉลี่ยในระดับที่แตกต่างจากแอพลิเคชันปกติ ดังแสดงในรูปที่ 3.13



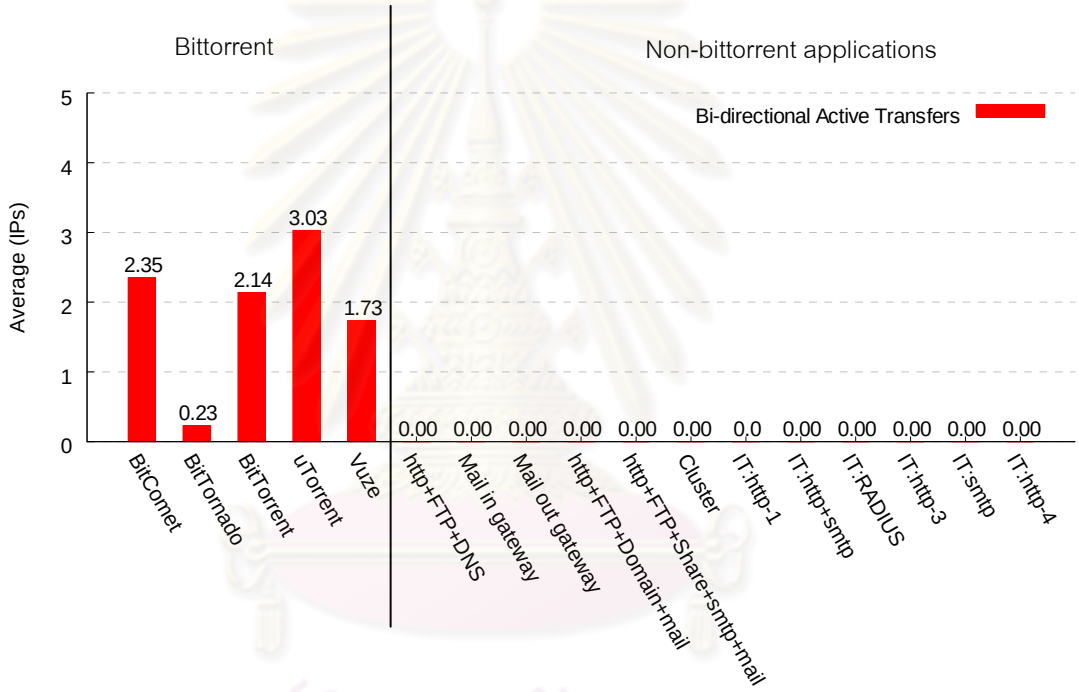
รูปที่ 3.13 เปรียบเทียบอัตราส่วนของการเชื่อมต่อที่มีการรับส่งข้อมูล (r_A)

จากรูปที่ 3.13 บิตทอร์เรนต์ไคลเอนต์ที่มีอัตราส่วนโดยเฉลี่ยดังกล่าวสูงสุดในการทดลองนี้คือ BitTornado ซึ่งมีค่าเฉลี่ยถึง 0.47 แสดงให้เห็นว่า BitTornado สามารถใช้ประโยชน์จากการเชื่อมต่อกับเพียร์อื่นๆ ได้ดีที่สุด เพราะถึงแม้จะมีจำนวนหมายเลขไอพีที่ติดต่อด้วยโดยเฉลี่ยมากกว่าไคลเอนต์ตัวอื่น แต่กลับมีอัตราส่วนของการเชื่อมต่อที่มีการรับส่งข้อมูลมากกว่าไคลเอนต์ตัวอื่น

สำหรับแอพลิเคชันทั่วไป มีอัตราส่วนของการเชื่อมต่อที่มีการรับส่งข้อมูลค่อนข้างน้อย เนื่องจากแอพลิเคชันส่วนใหญ่ไม่ได้ถูกออกแบบมาสำหรับแลกเปลี่ยนไฟล์ การเชื่อมต่อที่เกิดขึ้นจึงมีการถ่ายโอนข้อมูลด้วยขนาดเล็ก จึงไม่ถูกจัดเป็นการเชื่อมต่อที่มีการรับส่งข้อมูลในงานวิจัยนี้ โดยสังเกตได้จากเครื่องให้บริการเอชทีทีพีและเอสเอ็มทีพีของสำนักเทคโนโลยีสารสนเทศ (IT: http-3 และ IT: smtp ตามลำดับ) ที่มีจำนวนหมายเลขไอพีที่ติดต่อด้วยโดยเฉลี่ยถึง 47 และ 30 หมายเลขตามลำดับ แต่อัตราส่วนของการเชื่อมต่อที่มีการรับส่งข้อมูลโดยเฉลี่ยกลับมีเพียง 0.00

และ 0.09 เท่านั้น โดยจากผลการเปรียบเทียบสามารถสรุปได้ว่าเกณฑ์วัด r_A สามารถแยกแยะกระแสข้อมูลของบิตทอร์เรนต์และกระแสข้อมูลของแอพลิเคชันปกติได้เป็นอย่างดี

3. เปรียบเทียบจำนวนของการเชื่อมต่อที่มีการถ่ายโอนข้อมูลสองทิศทาง (Bi-directional Active Transfers: *B*) เนื่องจากกระแสข้อมูลบิตทอร์เรนต์ที่สร้างขึ้น ได้จากการดาวน์โหลดไฟล์ทอร์เรนต์เพียง 1 ไฟล์ จึงมีโอกาสเกิดการถ่ายโอนข้อมูลสองทิศทางในแต่ละช่วงเวลาเท่ากับ 4 เท่านั้น อีกทั้งพฤติกรรมการถ่ายโอนข้อมูลสองทิศทางไม่ได้เกิดขึ้นอย่างต่อเนื่องเหมือนกับสองพฤติกรรมแรก ทำให้มีค่าเฉลี่ยจากช่วงเวลาทั้งหมดค่อนข้างน้อย ดังแสดงในรูปที่ 3.14

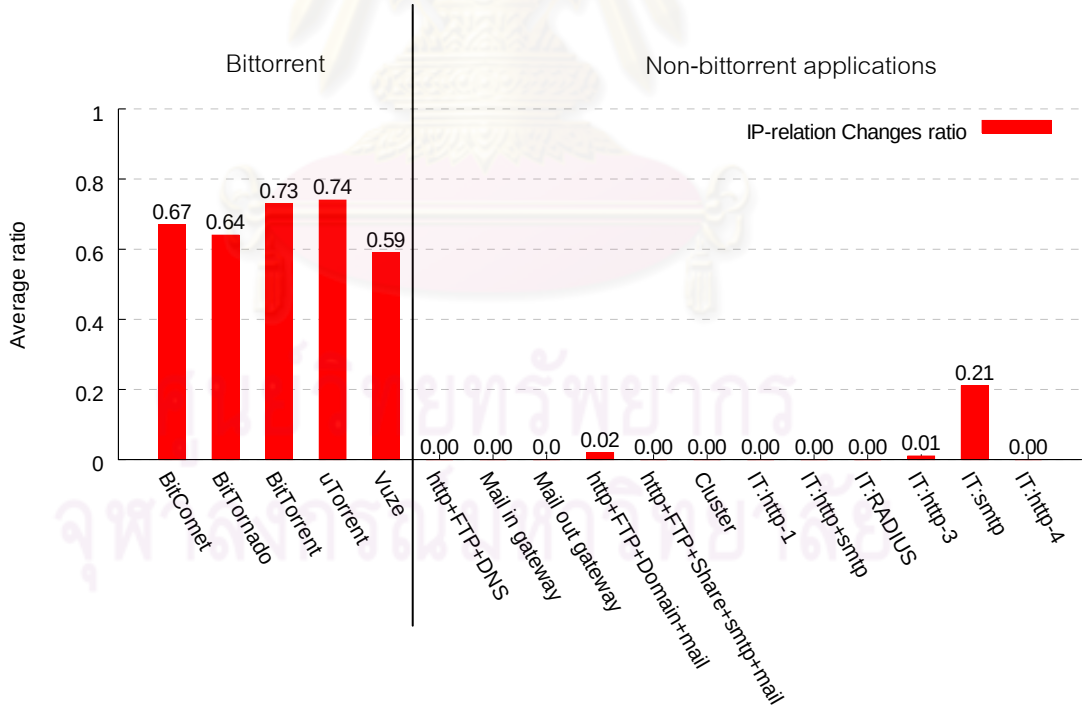


รูปที่ 3.14 เปรียบเทียบจำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูลสองทิศทาง (B)

สาเหตุที่พฤติกรรมการถ่ายโอนข้อมูลสองทิศทางไม่ได้เกิดขึ้นอย่างต่อเนื่อง เป็นเพราะบิตทอร์เรนต์ไคลเอนต์มีการถ่ายโอนข้อมูลสองทิศทางในช่วงแรกของการดาวน์โหลด เมื่อผ่านไประยะหนึ่งพบว่าตัวไคลเอนต์มีการดาวน์โหลดข้อมูลจากซีด (Seed) เกือบทั้งหมด จึงมีการถ่ายโอนสองทิศทางเกิดขึ้นน้อยมาก กรณีดังกล่าวอาจเกิดจากการปรับแต่งตัวไคลเอนต์ให้เลือกติดต่อและขอดาวน์โหลดชิ้นส่วนของไฟล์จากซีดมากกว่าลีช (Leech) เพื่อให้ตนเองสามารถดาวน์โหลดได้โดยไม่ต้องอัพโหลดตอบแทน

อย่างไรก็ตาม จำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูลสองทิศทางโดยเฉลี่ยของบิตทอร์เรนต์ไคลเอนต์ยังคงมีความแตกต่างจากแอพลิเคชันทั่วไปอยู่ เพราะแอพลิเคชันทั่วไปมีการถ่ายโอนข้อมูลจากไคลเอนต์ไปยังเครื่องให้บริการ หรือจากเครื่องให้บริการไปยังไคลเอนต์ทิศทางใดทิศทางหนึ่งในเวลาเดียวกัน จึงไม่มีการถ่ายโอนข้อมูลสองทิศทาง ส่งผลให้จำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูลสองทิศทางโดยเฉลี่ยของแอพลิเคชันทั่วไปมีค่าเป็น 0 ดังนั้นเกณฑ์วัด B จะสามารถแยกแยะความแตกต่างของกระแสข้อมูลบิตทอร์เรนต์และกระแสข้อมูลของแอพลิเคชันทั่วไปได้เป็นอย่างดี หากมีการดาวน์โหลดจากหลายไฟล์ทอร์เรนต์พร้อมกัน เพราะจะทำให้พฤติกรรมของการถ่ายโอนข้อมูลสองทิศทางเด่นชัดมากขึ้น ซึ่งทำให้แตกต่างจากแอพลิเคชันทั่วไปมากขึ้นตามไปด้วย

4. อัตราส่วนความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลง (IP-relation Change ratio: r_R) เพียร์ของบิตทอร์เรนต์มีพฤติกรรมของการเปลี่ยนคู่ถ่ายโอนข้อมูลอย่างต่อเนื่อง ทำให้มีอัตราส่วนความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลงโดยเฉลี่ยแตกต่างจากแอพลิเคชันปกติ ดังแสดงในรูปที่ 3.15



รูปที่ 3.15 เปรียบเทียบอัตราส่วนความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลง (r_R)

จากรูปที่ 3.15 บิตทอร์เรนต์ไคลเอนต์ทุกตัวมีอัตราส่วนความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลงในระดับที่ใกล้เคียงกัน แสดงให้เห็นว่าแม้บิตทอร์เรนต์ไคลเอนต์จะถูกพัฒนาจาก

ทีมงานต่างกัน แต่พฤติกรรมการเปลี่ยนคู่ถ่ายโอนข้อมูลซึ่งเกิดจากขั้นตอนวิธีการเค้นยังคงเด่นชัดและแตกต่างจากแอพลิเคชันทั่วไปอย่างมาก

สำหรับแอพลิเคชันทั่วไปซึ่งมีจำนวนการเชื่อมต่อที่มีการรับส่งข้อมูลค่อนข้างน้อย และไม่มีพฤติกรรมของการเปลี่ยนคู่ถ่ายโอนข้อมูล อัตราส่วนความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลงจึงมีค่าน้อยมาก ส่วนสาเหตุที่เครื่องให้บริการเอสเอ็มทีพี (IT: smtp) มีอัตราส่วนดังกล่าวถึง 0.21 เป็นเพราะมีการเชื่อมต่อที่มีการรับส่งข้อมูลอยู่จำนวนหนึ่งดังแสดงในรูปที่ 3.13 และเมื่อมีบางการเชื่อมต่อเกิดการเปลี่ยนแปลงความสัมพันธ์ อัตราส่วนที่เกิดขึ้นจึงมีค่ามาก แต่ยังคงแตกต่างกับบิตทอร์เรนต์ไคลเอนต์อย่างชัดเจน ผลการเปรียบเทียบจึงสามารถสรุปได้ว่าเกณฑ์วัด r_R สามารถแยกแยะกระแสข้อมูลของบิตทอร์เรนต์และแอพลิเคชันปกติได้เป็นอย่างดี และพฤติกรรมการเปลี่ยนคู่ถ่ายโอนข้อมูล ซึ่งถูกนำมาสร้างเป็นเกณฑ์วัด r_R เป็นพฤติกรรมที่มีความคล้ายกันมากที่สุดในบิตทอร์เรนต์ไคลเอนต์แต่ละตัวที่นำมาเปรียบเทียบ

จากผลการเปรียบเทียบพฤติกรรมของกระแสข้อมูลโดยใช้เกณฑ์วัดทั้ง 4 ที่นำเสนอพบว่ากระแสข้อมูลที่เกิดจากการเค้นของบิตทอร์เรนต์ไคลเอนต์มีความแตกต่างกับกระแสข้อมูลที่เกิดจากแอพลิเคชันอื่นอย่างเห็นได้ชัดเมื่อนำพฤติกรรมทั้ง 4 มาพิจารณาร่วมกัน เนื่องจากขั้นตอนวิธีการเค้นออกแบบมาโดยเฉพาะสำหรับควบคุมการดาวน์โหลดและอัพโหลดระหว่างเพียร์ในบิตทอร์เรนต์ จึงมีการใช้งานในบิตทอร์เรนต์ไคลเอนต์เท่านั้น กระแสข้อมูลที่เกิดจากแอพลิเคชันทั่วไปจึงมีลักษณะของกระแสข้อมูลที่เกิดจากการเค้นน้อยมาก

บทที่ 4

การทดลองและผลการทดลอง

4.1 การเก็บรวบรวมข้อมูลที่ใช้ในงานวิจัย

งานวิจัยนี้ได้ออกแบบการทดลองเป็นสองส่วน คือ การทดลองกับข้อมูลที่สร้างขึ้นจากสภาพแวดล้อมควบคุม และการทดลองกับข้อมูลปกติที่ได้จากระบบเครือข่าย โดยจะอธิบายถึงการออกแบบการทดลองกับข้อมูลทั้งสองชุดอย่างละเอียดในหัวข้อที่ 4.2

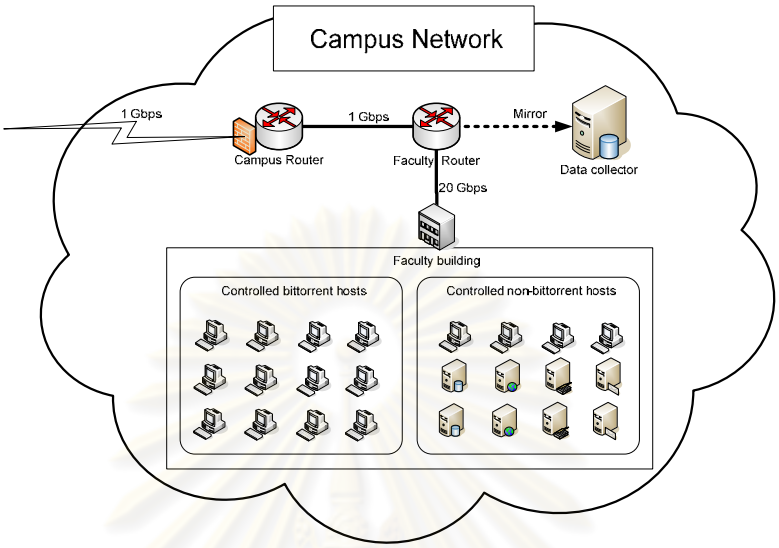
ข้อมูลทั้งสองชุดได้เก็บรวบรวมจากระบบเครือข่ายของคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ซึ่งเครื่องคอมพิวเตอร์มีการเชื่อมต่อกันด้วยระบบเครือข่ายอีเทอร์เน็ต (Ethernet) ความเร็ว 100 เมกะบิตต่อวินาที (Mbps) โดยที่เราท์เตอร์หลักของคณะเชื่อมต่อกับเราท์เตอร์หลักของมหาวิทยาลัยด้วยเครือข่ายเส้นใยแก้วนำแสง (Fiber Optic) ความเร็ว 1 กิกะบิตต่อวินาที (Gbps) จุดที่ทำการเก็บข้อมูลอยู่ที่เราท์เตอร์หลักของคณะวิศวกรรมศาสตร์ โดยใช้วิธีสำเนาทุกแพ็กเก็ตจากเราท์เตอร์หลักมายังเครื่องเก็บข้อมูล

4.1.1 ข้อมูลที่สร้างขึ้นจากสภาพแวดล้อมควบคุม

ข้อมูลชุดนี้ได้ทำการเก็บรวบรวมจากสภาพแวดล้อมควบคุม โดยกำหนดให้หมายเลขไอพีกลุ่มที่หนึ่งมีการใช้งานบิตทอร์เรนต์ หมายเลขไอพีกลุ่มที่สองไม่มีการใช้งานบิตทอร์เรนต์ โดยที่หมายเลขไอพีทั้งหมดสามารถติดต่อกับหมายเลขไอพีภายนอกได้ตามปกติ ดังแสดงในรูปที่ 4.1

หมายเลขไอพีกลุ่มที่หนึ่งประกอบด้วยเครื่องที่มีการใช้งานบิตทอร์เรนต์เพียงอย่างเดียว จำนวน 18 เครื่อง แต่ละเครื่องมีการเข้ารหัสกระแสข้อมูลและดาวน์โหลดไฟล์ทอร์เรนต์ที่แตกต่างกัน โดยไฟล์ทอร์เรนต์ทั้งหมดได้นำมาจากเว็บไซต์บนอินเทอร์เน็ตและมีผู้ร่วมดาวน์โหลดอยู่จริงในช่วงเวลาที่ทำการทดลอง ดังนั้นหมายเลขไอพีที่เปิดใช้งานบิตทอร์เรนต์จึงมีการรับส่งข้อมูลกับเพียร์ภายนอกโดยไม่มีการดาวน์โหลดกันเอง ข้อมูลของแต่ละไฟล์ทอร์เรนต์มีขนาดตั้งแต่ 700 เมกะไบต์ (MB) ถึง 4.3 กิกะไบต์ ซึ่งจากการดาวน์โหลดในช่วงระยะเวลาของการทดลองปรากฏว่าสามารถดาวน์โหลดได้สำเร็จ 16 เครื่อง ด้วยความเร็วเฉลี่ย 200 กิโลไบต์ (KB) ต่อวินาที ถึง 1 เมกะไบต์ต่อวินาที ส่วนอีก 2 เครื่องดาวน์โหลดไม่สำเร็จ เนื่องจากมีความเร็วในการดาวน์โหลดค่อนข้างน้อย โดยมีความเร็วอยู่ในช่วง 5-30 กิโลไบต์ต่อวินาที

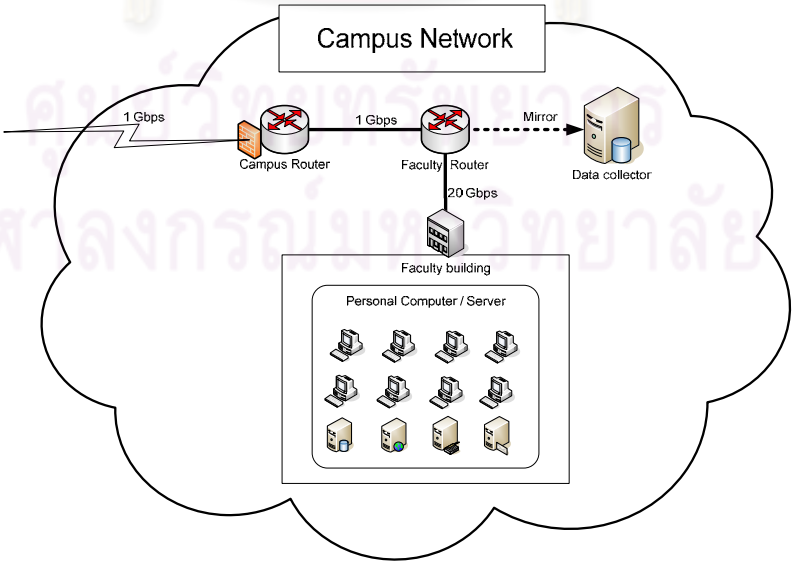
ส่วนหมายเลขไอพีกลุ่มที่สองมีการใช้งานแอพลิเคชันปกติ ไม่มีการใช้งานบิตทอร์เรนต์ โดยมีทั้งหมายเลขไอพีของเครื่องให้บริการและเครื่องคอมพิวเตอร์ของผู้ใช้รวมทั้งหมด 30 เครื่อง



รูปที่ 4.1 แผนภาพแสดงการเก็บข้อมูลที่สร้างขึ้นจากสภาพแวดล้อมควบคุม

4.1.2 ข้อมูลปกติจากระบบเครือข่าย

ข้อมูลชุดนี้ได้ทำการเก็บรวบรวมจากสภาพแวดล้อมที่มีการใช้งานตามปกติของระบบเครือข่ายในปี พ.ศ. 2549 ซึ่งมีแผนภาพแสดงการเก็บข้อมูลดังรูปที่ 4.2 ข้อมูลชุดนี้มีขนาดประมาณ 405.54 กิกะไบต์ (GB) มีจำนวนแพ็กเก็ตเท่ากับ 610,310,682 แพ็กเก็ต

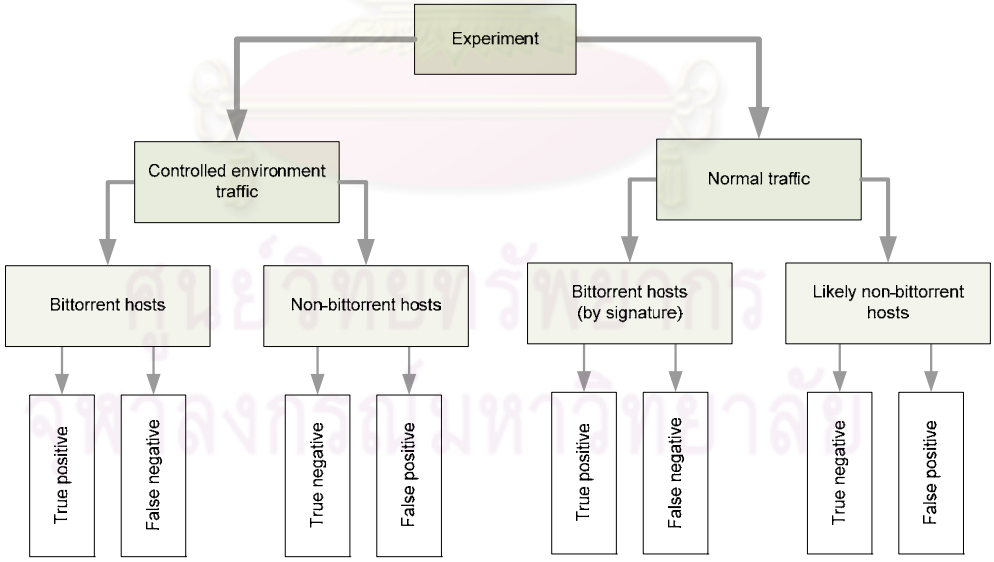


รูปที่ 4.2 แผนภาพแสดงการเก็บข้อมูลปกติจากระบบเครือข่าย

4.2 การออกแบบการทดลอง

งานวิจัยนี้ได้ออกแบบการทดลองเป็นสองส่วน คือ การทดลองกับข้อมูลที่สร้างขึ้นจากสภาพแวดล้อมควบคุม และการทดลองกับข้อมูลปกติที่ได้จากระบบเครือข่าย เพราะหากนำตัวระบุเพียร์ที่นำเสนอไปทดสอบกับข้อมูลปกติที่ได้จากระบบเครือข่ายเพียงอย่างเดียว ผลการทดสอบอาจไม่สามารถบอกได้อย่างชัดเจน เนื่องจากในข้อมูลเหล่านั้นอาจมีหมายเลขไอพีที่ใช้งานบิตทอร์เรนต์ที่มีการเข้ารหัสอยู่ ทำให้ไม่สามารถตรวจสอบหมายเลขไอพีเหล่านี้ได้ ซึ่งส่งผลกระทบกับความถูกต้องของการทดลอง ดังนั้นการทดสอบกับข้อมูลที่สร้างขึ้นจากสภาพแวดล้อมควบคุมจึงเป็นการทดลอบที่ช่วยยืนยันความสามารถของตัวระบุเพียร์ที่นำเสนอ ว่ามีความสามารถในการตรวจจับ (Detect) หมายเลขไอพีที่เปิดใช้งานบิตทอร์เรนต์แบบเข้ารหัสได้หรือไม่ และเนื่องจากวิธีการที่นำเสนอใช้พฤติกรรมที่เกิดขึ้นในมุมมองของเพียร์เฉพาะที่ จึงจำเป็นต้องรู้ว่าหมายเลขไอพีนั้นมีการติดต่อกับหมายเลขไอพีภายนอกใดบ้าง ดังนั้นการทดลองในงานวิจัยนี้จึงสนใจหมายเลขไอพีที่อยู่ในช่วงที่กำหนดให้กับคณะวิศวกรรมศาสตร์เท่านั้น

ข้อมูลทั้งสองชุดที่ได้ทำการเก็บรวบรวมตามขั้นตอนในหัวข้อที่ 4.1 จะถูกแบ่งออกเป็น 2 กลุ่ม เพื่อประเมินความถูกต้องในแต่ละด้าน ดังแสดงในรูปที่ 4.3 ดังนี้



รูปที่ 4.3 การแบ่งกลุ่มของข้อมูลทั้งสองชุด

1. การแบ่งกลุ่มข้อมูลที่สร้างขึ้นจากสภาพแวดล้อมควบคุม แต่ละเครื่องในกลุ่มนี้ถูกกำหนดอย่างชัดเจนว่ามีการใช้งานบิตทอร์เรนต์หรือไม่ ดังรายละเอียดที่ได้กล่าวไว้ในหัวข้อที่ 4.1.1 โดยหมายเลขไอพีในข้อมูลชุดนี้ถูกแบ่งเป็น 2 กลุ่ม ได้แก่

1.1 กลุ่มของหมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์ (Bittorrent hosts) คือ กลุ่มของหมายเลขไอพีที่มีการเปิดใช้งานบิตทอร์เรนต์แบบเข้ารหัส ใช้สำหรับประเมินความสามารถในการตรวจจับเพียร์หรือผลบวกแท้ (True positive) และความผิดพลาดที่ไม่สามารถตรวจจับเพียร์ได้หรือผลลบลวง (False negative)

1.2 กลุ่มของหมายเลขไอพีที่ไม่มีการใช้งานบิตทอร์เรนต์ (Non-bittorrent hosts) คือ กลุ่มของหมายเลขไอพีที่ไม่มีการใช้งานบิตทอร์เรนต์ ซึ่งมีทั้งหมายเลขไอพีของเครื่องที่ใช้งานปกติและหมายเลขไอพีของเครื่องให้บริการต่างๆ เช่น เครื่องให้บริการเว็บไซต์ หมายเลขไอพีในกลุ่มนี้ใช้สำหรับประเมินความสามารถในการเพิกเฉยต่อหมายเลขไอพีที่ไม่มีการใช้งานบิตทอร์เรนต์หรือผลลบแท้ (True negative) และความผิดพลาดในการตรวจจับหมายเลขไอพี โดยระบุหมายเลขไอพีที่ไม่มีการใช้งานบิตทอร์เรนต์ว่ามีการใช้งานบิตทอร์เรนต์ หรือผลบวกลวง (False positive) ซึ่งเป็นความผิดพลาดที่ส่งผลเสียอย่างมาก เนื่องจากหมายเลขไอพีที่ไม่ได้ใช้งานบิตทอร์เรนต์ที่ถูกตรวจจับเหล่านี้อาจถูกยับยั้ง (Block) การรับส่งข้อมูล หรือถูกควบคุมการใช้งานตามนโยบายขององค์กรที่กำหนดไว้สำหรับการใช้งานบิตทอร์เรนต์ ทำให้ส่งผลกระทบต่อการทำงานบนระบบเครือข่ายของหมายเลขไอพีเหล่านั้น

2. การแบ่งกลุ่มข้อมูลปกติจากระบบเครือข่าย ทำโดยใช้วิธีตรวจสอบเนื้อหาในแพ็กเก็ตโดยใช้โปรแกรมที่พัฒนาขึ้นเองเช่นเดียวกับการทดลองในบทที่ 3 หัวข้อที่ 3.1.1 ดังนั้นหมายเลขไอพีทั้งหมดในข้อมูลชุดนี้จึงถูกแบ่งออกเป็น 2 กลุ่ม ได้แก่

2.1 กลุ่มของหมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์ คือ กลุ่มของหมายเลขไอพีที่มีการรับส่งแพ็กเก็ตที่มีลายเซ็นแอพลิเคชันของบิตทอร์เรนต์ปรากฏอยู่ ซึ่งหมายเลขไอพีที่ตรวจพบลายเซ็นแอพลิเคชันเหล่านี้ ถือเป็นตัวแทนเพียร์ของบิตทอร์เรนต์ในชุดข้อมูลปกติจากระบบเครือข่ายในการทดลองที่จะกล่าวถึงต่อไปในบทนี้ โดยหมายเลขไอพีในกลุ่มนี้ใช้สำหรับประเมินผลบวกแท้และผลลบลวง

<u>หมายเหตุ</u>: สังเกตว่าจำนวนเพียร์ที่ตรวจจับได้ด้วยลายเซ็นแอพลิเคชันอาจไม่ตรงกับความเป็นจริง เนื่องจากอาจมีเพียร์บางตัวที่ใช้งานบิตทอร์เรนต์แบบเข้ารหัส ทำให้ไม่สามารถตรวจสอบเนื้อหาในแพ็กเก็ตได้ เพียร์เหล่านี้จึงถูกจัดอยู่ในกลุ่มหมายเลขไอพีที่คาดว่าไม่มีการใช้งานบิตทอร์เรนต์ กรณีเช่นนี้ส่งผลให้ค่าของผลบวกลวงสูงกว่าความเป็นจริง หากตัวระบุเพียร์สามารถตรวจจับเพียร์ที่ใช้งานบิตทอร์เรนต์แบบเข้ารหัสได้ เพราะถูกมองว่าเป็นการตรวจจับหมายเลขไอพีที่ไม่มีการใช้งานบิตทอร์เรนต์ว่ามีการใช้งานบิตทอร์เรนต์ งานวิจัยนี้จึงได้วิเคราะห์

พฤติกรรมของกระแสข้อมูลจากผลบวกลวงเหล่านี้เพิ่มเติม เพื่อหาสาเหตุที่ทำให้มีพฤติกรรมคล้ายกับบิตทอร์เรนต์ โดยจะได้กล่าวถึงต่อไปส่วนของผลการทดลอง

นอกจากนี้ ปัจจัยดังกล่าวอาจทำให้ค่าของผลบวกแท้สูงกว่าความเป็นจริง หากตัวระบุเพียร์ไม่มีการตรวจจับเพียร์ที่เข้ารหัสเหล่านี้ว่ามีการใช้งานบิตทอร์เรนต์ แต่จากผลการทดลองในงานวิจัยนี้พบว่าปริมาณข้อมูลที่เกิดจากเพียร์ (โดยใช้ลายเซ็น) มีสัดส่วนถึง 92.43% ดังนั้น แม้จะมีบางเพียร์ที่ไม่สามารถแบ่งกลุ่มได้ด้วยลายเซ็น แต่ปริมาณข้อมูลที่เกิดจากเพียร์เหล่านี้มีน้อยมากในข้อมูลชุดนี้ ความผิดพลาดของผลบวกแท้ที่เกิดขึ้นจึงแตกต่างกับผลลัพธ์ที่แท้จริงน้อยมากเช่นกัน

2.2 กลุ่มของหมายเลขไอพีที่คาดว่าไม่มีการใช้งานบิตทอร์เรนต์ (Likely non-bittorrent hosts) คือ กลุ่มของหมายเลขไอพีที่ไม่มีการรับส่งแพ็กเก็ตที่มีลายเซ็นแอพลิเคชันของบิตทอร์เรนต์ปรากฏอยู่ ใช้สำหรับประเมินผลลบแท้และผลบวกลวง

จากที่กล่าวมาเกี่ยวกับการประเมินความถูกต้องทางด้านผลบวกแท้ ผลลบลวง ผลลบแท้และผลบวกลวง ตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ที่นำเสนอจึงมีเป้าหมายในการทำให้ผลบวกแท้มีค่ามาก และทำให้ผลบวกลวงมีค่าน้อยที่สุด

4.3 การวัดความถูกต้องของตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์

4.3.1 ความถูกต้องในการตรวจจับ

ในหัวข้อนี้เป็นการประเมินความถูกต้องของตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ที่นำเสนอกับข้อมูลทั้งสองชุดในด้านต่างๆ ดังที่ได้อธิบายไว้ในหัวข้อที่ 4.2 โดยตัวระบุเพียร์ที่นำมาใช้ทดสอบสร้างขึ้นจากพฤติกรรมของกระแสข้อมูลบิตทอร์เรนต์ 4 อย่าง ซึ่งแต่ละพฤติกรรมมีเกณฑ์วัดที่ใช้ทดสอบแยกจากกัน จึงมีเกณฑ์วัดที่ใช้ทดสอบ 4 เกณฑ์ ดังนี้

1. จำนวนของหมายเลขไอพีที่ติดต่อด้วย (Connected IPs: *C*) เพียร์จะมีการติดต่อกับหมายเลขไอพีอื่นๆ จำนวนมากอยู่ตลอดเวลา เพราะพยายามรักษาจำนวนเพียร์ในเซตของเพียร์ไม่ให้ต่ำกว่า 20 และมีการแลกเปลี่ยนข้อความกันเป็นระยะ

2. อัตราส่วนของการเชื่อมต่อที่มีการรับส่งข้อมูล (Active Transfer ratio: r_A) เพียร์สามารถดาวน์โหลดแต่ละชิ้นส่วนจากเพียร์ที่ต่างกันได้ในเวลาเดียวกัน จึงทำให้มีจำนวนการเชื่อมต่อที่มีการรับส่งข้อมูลเป็นอัตราส่วนโดยตรงกับจำนวนของหมายเลขไอพีที่ติดต่อด้วย

3. จำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูลสองทิศทาง (Bi-directional Active Transfers: *B*) แต่ละเพียร์พยายามเลือกอัพโหลดให้แก่เพียร์ที่ตนเองดาวน์โหลดได้เร็วที่สุด ทำให้การเชื่อมต่อระหว่างบางคู่เพียร์มีข้อมูลรับส่งกันทั้งสองทิศทาง

4. อัตราส่วนความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลง (IP-relation Change ratio: r_R) ขั้นตอนวิธีการเค้นในโพรโทคอลบิตทอร์เรนต์ก่อให้เกิดพฤติกรรมของการค้นหาเพียร์ที่ให้ค่าดาวน์โหลดที่ดีกว่า ทำให้มีการเปลี่ยนคู่ถ่ายโอนข้อมูลอย่างต่อเนื่อง จึงมีจำนวนของความสัมพันธ์ระหว่างหมายเลขไอพีที่เปลี่ยนแปลงเป็นอัตราส่วนโดยตรงกับจำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูล

หมายเลขไอพีที่จะถูกระบุว่ามีการใช้งานบิตทอร์เรนต์ ต้องมีผลการทดสอบจากทุกเกณฑ์วัดเป็นจริง (เป็นหมายเลขไอพีที่ใช้งานบิตทอร์เรนต์) โดยเกณฑ์วัดที่ไม่ได้นำมาใช้ทดสอบจะถูกกำหนดค่าขีดเริ่มเปลี่ยนเท่ากับ 0 และเพื่อเป็นการลดจำนวนของผลบวกลวงให้เหลือน้อยที่สุด การทดลองนี้จึงกำหนดให้หมายเลขไอพีต้องผ่านการทดสอบจากทุกเกณฑ์วัดติดต่อกัน 3 รอบการคำนวณ จึงจะถูกระบุว่ามีการใช้งานบิตทอร์เรนต์ ซึ่งผลที่ได้สามารถลดจำนวนของผลบวกลวงลงได้อย่างมาก

สำหรับข้อมูลทดสอบทั้ง 2 ชุด เกณฑ์วัดร่วม (Combined metrics) $C + r_R$ และเกณฑ์วัดร่วม $C + r_A + r_R$ ให้ผลการทดลองที่ดีที่สุด โดยคุณสมบัติและผลการทดลองในชุดข้อมูลควบคุมแสดงได้ดังตารางที่ 4.1 และตารางที่ 4.2 ตามลำดับ ส่วนคุณสมบัติและผลการทดลองในชุดข้อมูลปกติจากระบบเครือข่ายแสดงได้ดังตารางที่ 4.3 และตารางที่ 4.4 ตามลำดับ

จากผลการทดลองดังตารางที่ 4.2 และ 4.4 พบว่าเกณฑ์วัดร่วม $C + r_R$ มีประสิทธิภาพดีที่สุด โดยสามารถตรวจจับหมายเลขไอพีที่ใช้งานบิตทอร์เรนต์หรือเพียร์ (ผลบวกแท้) ในชุดข้อมูลที่สร้างจากสภาพแวดล้อมควบคุมได้ทั้งหมด และตรวจจับเพียร์ในชุดข้อมูลปกติจากระบบเครือข่ายได้ 83.02% และเมื่อพิจารณาจากปริมาณข้อมูลที่เกิดจากเพียร์ที่ตรวจจับได้เหล่านี้ พบว่ามีสัดส่วนถึง 100% และ 98.75% ของปริมาณข้อมูลที่เกิดจากเพียร์ทั้งหมดในข้อมูลทั้ง 2 ชุด ตามลำดับ แสดงให้เห็นว่าเกณฑ์วัดร่วม $C + r_R$ สามารถตรวจจับเพียร์ที่มีการรับส่งข้อมูลจำนวนมากได้อย่างมีประสิทธิภาพ

ตารางที่ 4.1 คุณสมบัติของชุดข้อมูลในสภาพแวดล้อมควบคุม

	หมายเลขไอพี		จำนวนไบต์ที่รับส่ง	
	จำนวน	%	กิกะไบต์	%
หมายเลขไอพีทั้งหมด	48		39.48	
หมายเลขไอพีที่ใช้งานบิตทอร์เรนต์	18	37.5	25.02	63.37
หมายเลขไอพีที่ไม่มีการใช้งานบิตทอร์เรนต์	30	62.5	14.46	36.63

ตารางที่ 4.2 ผลการทดลองในชุดข้อมูลในสภาพแวดล้อมควบคุม

	$C + r_R$		$C + r_A + r_R$	
	จำนวนหมายเลขไอพี	จำนวนไบต์ที่รับส่ง (GB)	จำนวนหมายเลขไอพี	จำนวนไบต์ที่รับส่ง (GB)
ผลบวกแท้ (True positive)	**18 (100%)**	**25.02 (100%)**	**17 (94.44%)**	**24.71 (98.76%)**
ผลลบลวง (False negative)	0 (0%)	0 (0%)	1 (5.56%)	0.31 (1.24%)
ผลลบแท้ (True negative)	30 (100%)	14.46 (100%)	30 (100%)	14.46 (100%)
ผลบวกลวง (False positive)	**0 (0%)**	0 (0%)	**0 (0%)**	0 (0%)

ตารางที่ 4.3 คุณสมบัติของชุดข้อมูลปกติ

	หมายเลขไอพี		จำนวนไบต์ที่รับส่ง	
	จำนวน	%	กิกะไบต์	%
หมายเลขไอพีทั้งหมด	1,519		405.54	
หมายเลขไอพีที่ใช้งานบิตทอร์เรนต์	106	6.98	374.85	92.43
หมายเลขไอพีที่ไม่มีการใช้งานบิตทอร์เรนต์	1,413	93.02	30.69	7.57

ตารางที่ 4.4 ผลการทดลองในชุดข้อมูลปกติ

	$C + r_R$		$C + r_A + r_R$	
	จำนวนหมายเลขไอพี	จำนวนไบต์ที่รับส่ง (GB)	จำนวนหมายเลขไอพี	จำนวนไบต์ที่รับส่ง (GB)
ผลบวกแท้ (True positive)	88 (83.02%)	**370.16 (98.75%)**	63 (59.43%)	**340.11 (90.73%)**
ผลลบลวง (False negative)	18 (16.98%)	4.69 (1.25%)	43 (40.57%)	34.74 (9.27%)
ผลลบแท้ (True negative)	1,402 (99.22%)	24.60 (80.16%)	1,410 (99.79%)	25.79 (84.07%)
ผลบวกลวง (False positive)	**11 (0.78%)**	6.08 (19.84%)	**3 (0.21%)**	4.89 (15.93%)

ตัวระบุเพียร์ที่นำเสนอไม่สามารถตรวจจับเพียร์ที่มีการติดต่อกับหมายเลขไอพีอื่นน้อย และในจำนวนนี้มีบางการเชื่อมต่อเท่านั้นที่มีการรับส่งข้อมูล เนื่องจากกระแสข้อมูลที่เกิดขึ้นมีความคล้ายคลึงกับกระแสข้อมูลของอินเทอร์เน็ตแอพลิเคชันทั่วไป เช่น เอฟทีพี (FTP) เป็นต้น โดยจากตารางที่ 4.2 พบว่าเกณฑ์วัดร่วม $C + r_A + r_R$ ไม่สามารถตรวจจับเพียร์ได้ (ผลลบลวง) 1 ตัว แต่ปริมาณข้อมูลที่เกิดจากเพียร์นี้มีเพียง 1.24% ของปริมาณข้อมูลที่เกิดจากเพียร์ทั้งหมดในข้อมูลชุดนี้ ดังนั้นการควบคุมเพียร์ที่ตรวจจับได้ จึงเท่ากับสามารถรักษาแบนด์วิดท์เกือบทั้งหมดเพื่อการใช้งานแอพลิเคชันอื่นบนระบบเครือข่ายได้

เกณฑ์วัดร่วม $C + r_A + r_R$ มีประสิทธิภาพด้อยกว่าเกณฑ์วัดร่วม $C + r_R$ ในด้านของผลบวกแท้ เนื่องจากมีเกณฑ์วัดที่ต้องทดสอบมากกว่า โดยเกณฑ์วัดที่เพิ่มขึ้นมาคือเกณฑ์วัด r_A ซึ่งตรวจจับพฤติกรรมการรับส่งข้อมูลกับหลายเพียร์ในเวลาเดียวกัน ทำให้เกณฑ์วัดร่วม $C + r_A + r_R$ ไม่สามารถตรวจหาบางเพียร์ที่ไม่ผ่านการทดสอบเกณฑ์วัด r_A ได้ เพราะมีจำนวนของเพียร์ที่กำลังรับส่งข้อมูลด้วยน้อย โดยจากตารางที่ 4.2 และตารางที่ 4.4 พบว่าเกณฑ์วัดร่วม $C + r_A + r_R$ สามารถตรวจจับเพียร์ในชุดข้อมูลควบคุมและชุดข้อมูลปกติได้ 94.44% และ 59.43% ตามลำดับ แต่เมื่อพิจารณาจากปริมาณข้อมูลที่เกิดจากเพียร์ที่ตรวจจับได้เหล่านี้ พบว่ายังคงมีอัตราส่วนที่ค่อนข้างสูง โดยคิดเป็น 98.76% และ 90.73% ในชุดข้อมูลควบคุม และชุดข้อมูลปกติตามลำดับ แสดงให้เห็นว่าเกณฑ์วัดร่วม $C + r_A + r_R$ สามารถตรวจจับเพียร์ที่มีการรับส่งข้อมูลจำนวนมากได้เป็นอย่างดีเช่นเดียวกัน

อย่างไรก็ตาม เกณฑ์วัดร่วม $C + r_A + r_R$ มีประสิทธิภาพดีกว่าเกณฑ์วัดร่วม $C + r_R$ ในด้านของผลบวกลวงในชุดข้อมูลปกติ เนื่องจากมีเกณฑ์วัดที่ใช้ทดสอบมากกว่า หมายเลขไอพีที่ไม่ได้ใช้งานบิตทอร์เรนต์ที่มีการติดต่อกับหมายเลขไอพีอื่นจำนวนมาก เช่น เครื่องให้บริการเวิลด์ไวด์เว็บ แต่มีการรับส่งข้อมูลกับหมายเลขไอพีอื่นน้อย และมีเพียงบางการเชื่อมต่อเท่านั้นที่มีการเปลี่ยนแปลงความสัมพันธ์ของคู่หมายเลขไอพี (เมื่อค่าของ A และ R น้อยใกล้เคียงกัน จะส่งผลให้เกณฑ์วัด r_R มีค่ามาก) จึงถูกตรวจจับได้ด้วยเกณฑ์วัดร่วม $C + r_R$ แต่เมื่อนำเกณฑ์วัด r_A มาช่วยตรวจสอบ หมายเลขไอพีเหล่านี้จึงไม่ถูกตรวจจับ เพราะมีการเชื่อมต่อที่มีการรับส่งข้อมูลน้อยกว่าจำนวนไอพีที่ติดต่อด้วยอย่างมาก ตัวระบุเพียร์ที่ได้จากเกณฑ์วัดร่วม $C + r_A + r_R$ จึงมีความเฉพาะเจาะจงกับพฤติกรรมของเพียร์ในบิตทอร์เรนต์มากกว่า แต่โดยภาพรวมแล้ว เกณฑ์วัดร่วมทั้งสองให้ค่าของผลบวกลวงไม่ถึง 1% แสดงให้เห็นว่าพฤติกรรมของกระแสข้อมูลบิตทอร์เรนต์ที่นำมาสร้างเป็นเกณฑ์วัด มีความแตกต่างจากพฤติกรรมของกระแสข้อมูลที่เกิดจากแอพลิเคชันที่ใช้งานทั่วไป

4.3.2 การวิเคราะห์ผลบวกลวง

เนื่องจากการตรวจหาลายเซ็นแอพลิเคชันไม่สามารถตรวจจับเพียร์ที่มีการเข้ารหัสแพ็กเก็ตได้ ซึ่งอาจทำให้มีเพียร์ที่เข้ารหัสบางตัวถูกจัดอยู่ในกลุ่มของหมายเลขไอพีที่คาดว่าไม่มีการใช้งานบิตทอร์เรนต์ ดังนั้นหมายเลขไอพีที่เป็นผลบวกลวงจากการทดลองนี้จึงอาจเป็นเพียร์ที่มีการเข้ารหัสได้ งานวิจัยนี้จึงทำการวิเคราะห์ผลบวกลวงเหล่านี้โดยละเอียดว่ามีความเป็นไปได้ที่จะเป็นเพียร์ของบิตทอร์เรนต์เพียงใด รวมถึงการตรวจสอบว่าหมายเลขไอพีดังกล่าวมีการใช้งานแอพลิเคชันใด จึงทำให้กระแสข้อมูลมีลักษณะคล้ายกับบิตทอร์เรนต์จนทำให้กลายเป็นผลบวกลวง (ไม่ใช่บิตทอร์เรนต์แต่ระบุว่าเป็น)

ผลจากการวิเคราะห์เบื้องต้นของแต่ละหมายเลขไอพีแสดงดังตารางที่ 4.5 โดยผลบวกลวงที่เกิดจากเกณฑ์วัดร่วม $C + r_A + r_R$ ทั้งสามตัวมีการรับส่งข้อมูลโดยเฉลี่ยถึง 1.6 กิกะไบต์ ซึ่งค่อนข้างมากเมื่อเทียบกับปริมาณข้อมูลโดยเฉลี่ยที่เกิดจากหมายเลขไอพีที่คาดว่าไม่มีการใช้งานบิตทอร์เรนต์ทั้งหมด แสดงให้เห็นว่าหมายเลขไอพีที่ถูกตรวจจับด้วยเกณฑ์วัดร่วม $C + r_A + r_R$ มักเป็นหมายเลขไอพีที่รับส่งข้อมูลจำนวนมาก ซึ่งมีแนวโน้มที่จะทำให้เกิดผลเสียต่อระบบเครือข่าย

ตารางที่ 4.5 ผลการวิเคราะห์ผลบวกลวงเบื้องต้น

หมายเลขไอพี	จำนวนไบต์ที่รับส่ง (MB)	เป็นผลบวกลวงของ		มีการใช้หมายเลขพอร์ตของบิตทอร์เรนต์	หมายเหตุ
		$C + r_R$	$C + r_A + r_R$		
161.200.AA.กก	33.63	✓		✓	
161.200.BB.ขข	186.26	✓		✓	
161.200.CC.คค	104.87	✓		✓	
161.200.DD.งง	82.86	✓			
161.200.EE.จจ	1,783.77	✓	✓	✓	
161.200.AA.ฉฉ	1,472.37	✓	✓		เป็น NAT
161.200.FF.ชช	305.67	✓			
161.200.FF.ซซ	45.35	✓			
161.200.FF.ฌฌ	51.06	✓		✓	
161.200.GG.ญญ	541.02	✓			
161.200.HH.ฏฏ	1,641.03	✓	✓		

อย่างไรก็ตาม จากการตรวจสอบหมายเลขไอพีของเครื่องให้บริการในคณะวิศวกรรมศาสตร์พบว่า หมายเลขไอพี 161.200.AA.ฉฉ เป็นหมายเลขไอพีของเครื่องให้บริการแปลง

ที่อยู่เครือข่าย (NAT) จึงไม่ถือว่าหมายเลขไอพีนี้เป็นผลบวกลวง เนื่องจากงานวิจัยนี้มีข้อจำกัดว่ากระแสข้อมูลต้องไม่ถูกแปลงเลขที่อยู่เครือข่ายมาก่อน เพราะจะทำให้กระแสข้อมูลที่เกิดขึ้นมาจากหลายหมายเลขไอพี ส่งผลให้การตรวจสอบเกิดความผิดพลาด

จากตารางที่ 4.5 พบว่า ในจำนวนผลบวกลวง 11 ตัวของเกณฑ์วัดร่วม $C + r_R$ มี 5 ตัวที่มีการติดต่อกับหมายเลขไอพีอื่นด้วยหมายเลขพอร์ตโดยปริยายของบิตทอร์เรนต์ ส่วนผลบวกลวง 3 ตัวที่เกิดจากเกณฑ์วัดร่วม $C + r_A + r_R$ พบว่ามี 1 ตัวที่มีการติดต่อกับหมายเลขไอพีอื่นด้วยหมายเลขพอร์ตโดยปริยายของบิตทอร์เรนต์ แต่ไอพีเหล่านี้ไม่มีลายเซ็นแอพลิเคชันของบิตทอร์เรนต์ จึงกลายเป็นผลบวกลวง นอกจากนี้ หมายเลขไอพีเหล่านี้ยังมีการถ่ายโอนข้อมูลจำนวนมากทางหมายเลขพอร์ตที่ไม่มีการระบุว่าเป็นของแอพลิเคชันใด [18] ดังตัวอย่างในตารางที่ 4.6 ซึ่งเป็นหมายเลขพอร์ตปลายทางและปริมาณข้อมูลที่มีการรับส่งของหมายเลขไอพี 161.200.EE.จจ

ตารางที่ 4.6 ตัวอย่างหมายเลขพอร์ตปลายทางที่ 161.200.EE.จจ ติดต่อด้วย

Port number	# of flows	Transferred bytes (MB)	Application
443	2	0.08	Hypertext Transfer Protocol over TLS/SSL (HTTPS) Official
2369	3	0.01	Default for BMC Software CONTROL-M/Server Configuration Agent
2370	1	0.00	Default for BMC Software CONTROL-M/Server
[2700, 2800]	18	0.00	KnowShowGo P2P Official
3723	12	0.03	Used by many Battle.net Blizzard games (Diablo II, Warcraft II, Warcraft III, StarCraft) Unofficial
3785	2	0.00	Ventrilo VoIP program used by Ventrilo Unofficial
4100	2	0.00	WatchGuard Authentication Applet default Unofficial
6346	669	0.11	gnutella-svc, Gnutella (FrostWire, Limewire, Shareaza, etc.) Official
[6881, 6999]	11	0.00	BitTorrent part of full range of ports used most often Unofficial
other ports	6,926	1,762.31	Unknown

อย่างไรก็ตาม หมายเลขไอพีนี้มีการเปิดการเชื่อมต่อไปยังหมายเลขพอร์ตโดยปริยายของนูเทลล่า (Gnutella) จำนวนหนึ่ง จึงยังไม่สามารถบอกได้ว่าปริมาณข้อมูลที่เกิดจากหมายเลขพอร์ตที่ไม่มีการระบุแอพลิเคชันนั้น เกิดจากบิตทอร์เรนต์หรือนูเทลล่า

นอกจากหมายเลขไอพีนี้แล้ว หมายเลขไอพีที่มีการติดต่อกับหมายเลขพอร์ตโดยปริยายของบิตทอร์เรนต์อีก 4 ตัวที่เหลือมีการเชื่อมต่อไปยังหมายเลขพอร์ตที่ไม่มีการระบุว่าเป็นของแอพลิเคชันใด และมีการถ่ายโอนข้อมูลส่วนใหญ่ทางหมายเลขพอร์ตเหล่านี้เช่นกัน จึงมีความเป็นไปได้ว่าหมายเลขไอพีที่เป็นผลบวกลวงทั้ง 5 ตัวดังกล่าวอาจเป็นเพียร์ของบิตทอร์เรนต์ที่ทำการเข้ารหัสแพ็กเก็ตไว้ แต่ถึงแม้เพียร์เหล่านี้จะมีการติดต่อกับหมายเลขพอร์ตโดยปริยายของบิตทอร์เรนต์และมีพฤติกรรมคล้ายกับบิตทอร์เรนต์ แต่งานวิจัยนี้ยังไม่สามารถยืนยันได้อย่างชัดเจนว่าเป็นเพียร์ของบิตทอร์เรนต์จริง เพราะวิธีการตรวจหาบิตทอร์เรนต์ที่แม่นยำที่สุดในปัจจุบันคือการตรวจสอบลายเซ็นแอพลิเคชัน จึงจำเป็นต้องมีการศึกษาเพิ่มเติมในส่วนนี้ต่อไป

ในส่วนของหมายเลขไอพีอีก 5 ตัวที่ไม่มีการเปิดการเชื่อมต่อไปยังหมายเลขพอร์ตโดยปริยายของบิตทอร์เรนต์นั้น พบว่ามีบางตัวที่แม้จะมีการเชื่อมต่อไปยังหมายเลขพอร์ตที่ระบุแอพลิเคชันไว้ แต่กลับมีพฤติกรรมผิดปกติไปจากที่ควรจะเป็น เช่น หมายเลขไอพี 161.200.FF.ชช ที่คาดว่าเป็นการใช้งานเว็บของผู้ใช้ทั่วไป แต่กลับมีโฟลจำนวนมากที่มีหมายเลขพอร์ตปลายทางเป็น 445 ซึ่งเป็นของบริการแอ็กทีฟไดเร็คทอรี (Active Directory) ดังแสดงในตารางที่ 4.7

ตารางที่ 4.7 หมายเลขไอพีที่มีการเชื่อมต่อผิดปกติของการใช้งานแอ็กทีฟไดเร็คทอรี

Port number	# of flows	Transferred bytes (MB)	Application
80	837	6.09	Hypertext Transfer Protocol (HTTP) Official
81	116	2.37	Torpark Onion routing Unofficial
443	58	1.40	Hypertext Transfer Protocol over TLS/SSL (HTTPS) Official
445	795,849	295.63	Microsoft-DS Active Directory, Windows shares Official
8080	2	0.18	HTTP alternate (http_alt) commonly used for Web proxy
other ports	2	0.00	Unknown

และหมายเลขไอพี 161.200.DD.งง ที่มีโฟลจำนวนมากที่มีหมายเลขพอร์ตปลายทางเป็น 25 ซึ่งเป็นของบริการเอสเอ็มทีพี (SMTP) แต่ปริมาณข้อมูลที่เกิดจากโฟลเหล่านี้กลับมีน้อยมาก ดังแสดงในตารางที่ 4.8 กรณีเช่นนี้มีความผิดปกติไปจากการใช้งานจริง เพราะหากมีการใช้งานบริการอีเมลจริง ปริมาณข้อมูลที่เกิดจากโฟลถึง 5,424 โฟลควรมีมากกว่านี้

ตารางที่ 4.8 หมายเลขไอพีที่มีการเชื่อมต่อผิดปกติของการใช้งานอีเมล์

Port number	# of flows	Transferred bytes (MB)	Application
25	5,424	1.92	Simple Mail Transfer Protocol (SMTP) Official
80	147	80.53	Hypertext Transfer Protocol (HTTP) Official
443	19	0.12	Hypertext Transfer Protocol over TLS/SSL (HTTPS) Official
other ports	17	0.28	Unknown

จากการรายงานเกี่ยวกับความปลอดภัยบนอินเทอร์เน็ตในช่วงปี 2547-2549 พบว่าหมายเลขพอร์ต 445 และ 25 ถูกใช้เป็นช่องทางในการโจมตีระบบเครือข่ายอย่างแพร่หลาย เช่น หมายเลขพอร์ต 445 ถูกใช้เป็นช่องทางหนึ่งในการแพร่กระจายของหนอนอินเทอร์เน็ต (Internet worm) ชื่อ Nimda และ Sasser ส่วนหมายเลขพอร์ต 25 ถูกใช้เป็นช่องทางแพร่กระจายของ Comcast's Internet zombies เป็นต้น จึงมีความเป็นไปได้ที่หมายเลขไอพีทั้งสองดังกล่าวถูกใช้เป็นเครื่องมือในการสร้างความเสียหายให้แก่ระบบเครือข่าย ทำให้มีโฟลจำนวนมากเกิดขึ้นดังกล่าว

โดยสรุปแล้ว หมายเลขไอพีที่เป็นผลบวกลวงของตัวระบุเพียร์ที่นำเสนอส่วนใหญ่มีลักษณะใกล้เคียงกับเพียร์ของบิตทอร์เรนต์ โดย 50% ของผลบวกลวงทั้งหมดมีการติดต่อกับหมายเลขไอพีอื่นทางหมายเลขพอร์ตโดยปริยาย และถ่ายโอนข้อมูลส่วนใหญ่ทางหมายเลขพอร์ตที่ไม่มีการระบุแอพลิเคชัน เพียงแต่งานวิจัยนี้ยังไม่สามารถยืนยันได้อย่างชัดเจนว่าเป็นบิตทอร์เรนต์จริง นอกจากนี้ยังพบว่า 30% ของผลบวกลวงทั้งหมดเป็นหมายเลขไอพีที่มีพฤติกรรมผิดปกติไปจากการใช้งานของผู้ใช้ทั่วไป โดยอาจเป็นหมายเลขไอพีที่ถูกใช้ในการโจมตีระบบเครือข่าย แสดงให้เห็นว่าหมายเลขไอพีที่ถูกระบุว่าเป็นเพียร์ของบิตทอร์เรนต์ มักเป็นหมายเลขไอพีที่มีแนวโน้มในการรับส่งข้อมูลจำนวนมากในลักษณะเดียวกับบิตทอร์เรนต์ หรือมีพฤติกรรมการใช้งานที่ผิดปกติไปจากที่ควรจะเป็น และมีโอกาสสร้างความเสียหายแก่ระบบเครือข่าย ดังนั้น แนวคิดในงานวิจัยนี้จึงอาจนำไปประยุกต์ใช้สำหรับการตรวจหาหมายเลขไอพีที่มีพฤติกรรมผิดปกติในระบบเครือข่าย หรือนำไปประยุกต์ใช้ตรวจจับเพียร์ทูเพียร์ตัวอื่นที่มีโครงสร้างและการทำงานคล้ายกับบิตทอร์เรนต์ เพราะผู้วิจัยเชื่อว่าบิตทอร์เรนต์เป็นหนึ่งในโพรโทคอลสำหรับแลกเปลี่ยนไฟล์ที่มีประสิทธิภาพที่สุดในปัจจุบัน จึงมีความเป็นไปได้อย่างมากที่จะถูกนำไปดัดแปลงหรือทำเป็นโครงสร้างพื้นฐานของแอพลิเคชันที่จะเกิดขึ้นใหม่ในอนาคต

4.4 การวัดความเร็วในการตรวจจับของตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์

งานวิจัยนี้คาดหวังว่าสามารถนำตัวระบุเพียร์ที่นำเสนอไปประยุกต์ใช้ในแบบออนไลน์ (online) จึงมีการวัดความเร็วในการตรวจจับเพียร์ รวมถึงปริมาณข้อมูลที่เพียร์เหล่านี้สามารถรับส่งกันได้ก่อนถูกตรวจจับ เนื่องจากการควบคุมการใช้งานบิตทอร์เรนต์ที่มีประสิทธิภาพ ตัวระบุเพียร์ต้องสามารถตรวจจับเพียร์เหล่านี้ได้ก่อนที่จะมีการรับส่งข้อมูลไปเป็นจำนวนมาก โดยการวัดความเร็วในการตรวจจับเพียร์จะเริ่มจับเวลาเมื่อมีแพ็กเก็ตที่เป็นของแอพลิเคชันบิตทอร์เรนต์แพ็กเก็ตแรกปรากฏ สำหรับชุดข้อมูลปกติจากระบบเครือข่าย จะเริ่มจับเวลาเมื่อหมายเลขไอพีมีการรับส่งแพ็กเก็ตที่มีลายเซ็นแอพลิเคชันของบิตทอร์เรนต์แพ็กเก็ตแรก ส่วนในชุดข้อมูลที่สร้างขึ้นจากสภาพแวดล้อมควบคุม ซึ่งมีการเข้ารหัสแพ็กเก็ต ทำให้ไม่สามารถตรวจหาลายเซ็นแอพลิเคชันได้ จึงเริ่มจับเวลาเมื่อหมายเลขไอพีมีการรับส่งแพ็กเก็ตกับหมายเลขไอพีของแทรกเกอร์ของแต่ละไฟล์ทอร์เรนต์ที่ใช้ในการทดลอง

ความเร็วในการตรวจจับเพียร์ด้วยเกณฑ์วัดร่วม $C + r_R$ ในชุดข้อมูลที่สร้างขึ้นจากสภาพแวดล้อมควบคุม และชุดข้อมูลปกติจากระบบเครือข่ายแสดงได้ดังรูป 4.4 และรูปที่ 4.5 ตามลำดับ ส่วนความเร็วในการตรวจจับหมายเลขไอพีที่ใช้งานบิตทอร์เรนต์ด้วยเกณฑ์วัดร่วม $C + r_A + r_R$ แสดงดังรูปที่ 4.6 และรูปที่ 4.7 ตามลำดับ โดยมีค่าของแกนนอน (แกน X) ในแต่ละกราฟแทนระยะเวลาเป็นนาที ส่วนแกนตั้ง (แกน Y) เป็นจำนวนเปอร์เซ็นต์สะสม (Cumulative percentage) ของแต่ละเส้นกราฟ ซึ่งมีความหมายดังนี้

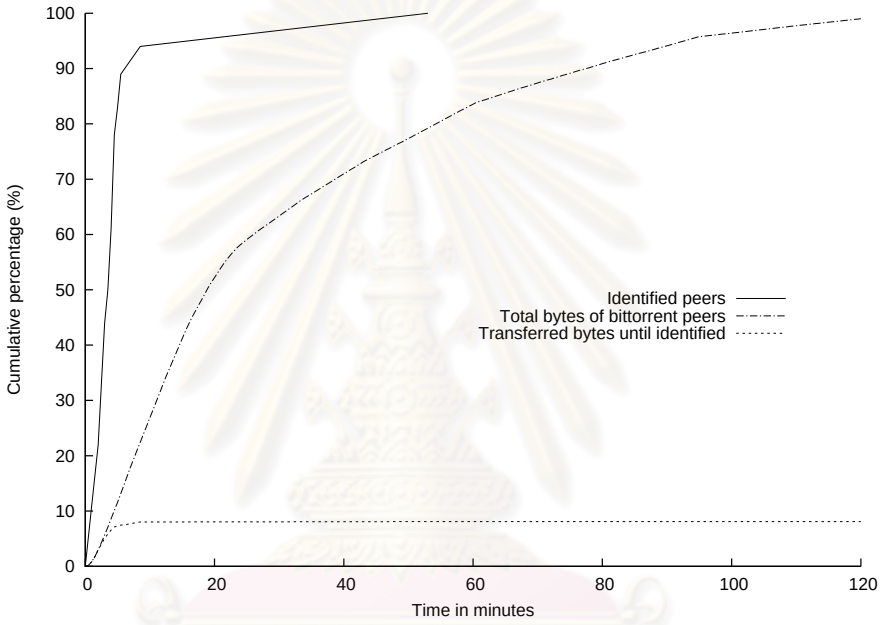
1. Identified peers คือ จำนวนเปอร์เซ็นต์สะสมของเพียร์ที่ตรวจจับได้ในช่วงเวลาต่างๆ ของการทดลอง โดยเทียบอัตราส่วนจากจำนวนเพียร์ที่ตรวจจับได้ทั้งหมดในแต่ละชุดข้อมูล

2. Total bytes of bittorrent peers คือ จำนวนเปอร์เซ็นต์สะสมของปริมาณข้อมูลที่เพียร์ทั้งหมดมีการรับส่งกันในช่วงเวลาต่างๆ ของการทดลอง โดยเทียบอัตราส่วนจากปริมาณข้อมูลที่เกิดจากเพียร์ทั้งหมดในแต่ละชุดข้อมูล

3. Transferred bytes until identified คือ จำนวนเปอร์เซ็นต์สะสมของปริมาณข้อมูลที่เพียร์ทั้งหมดมีการรับส่งกันก่อนถูกตรวจจับในช่วงเวลาต่างๆ ของการทดลอง โดยเทียบอัตราส่วนจากปริมาณข้อมูลที่เกิดจากเพียร์ทั้งหมดในแต่ละชุดข้อมูล

กราฟในรูปที่ 4.4 และ 4.5 แสดงความเร็วในการตรวจจับด้วยเกณฑ์วัดร่วม $C + r_R$ เมื่อพิจารณาจากจำนวนสะสมของเพียร์ที่ตรวจจับได้ (Identified peers) พบว่า 95% ของเพียร์ที่

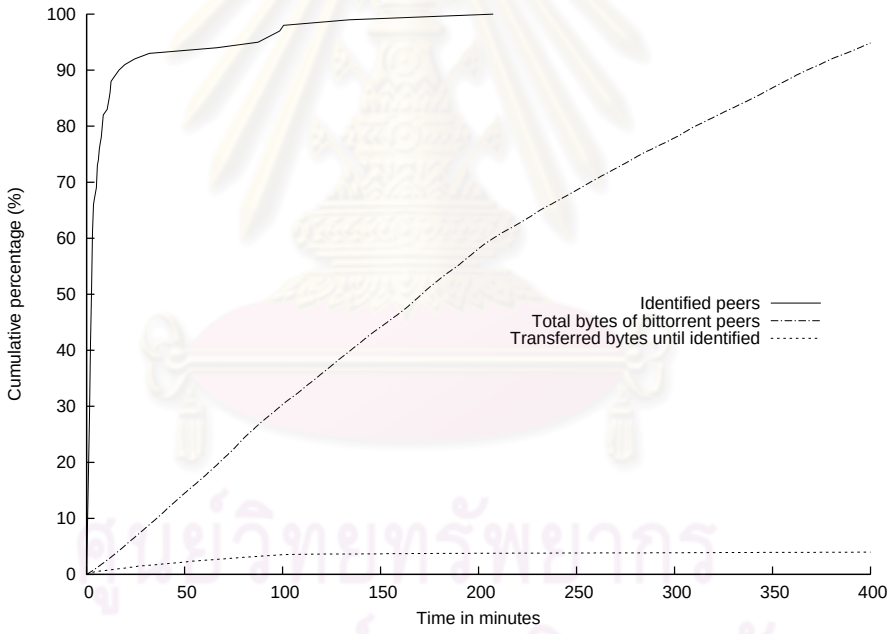
ตรวจจับได้ในชุดข้อมูลควบคุม ใช้เวลาไม่เกิน 6 นาที ส่วนในชุดข้อมูลปกติพบว่า 90% ของเพียร์ที่ตรวจจับได้ใช้เวลาไม่เกิน 10 นาทีแรก โดยสาเหตุที่เกณฑ์วัดร่วม $C + r_R$ ใช้เวลาตรวจจับเพียร์ในชุดข้อมูลปกตินานกว่าชุดข้อมูลควบคุม เป็นเพราะในชุดข้อมูลปกติมีเพียร์จำนวนหนึ่งที่ตรวจจับได้มีการอัตราการรับส่งข้อมูลค่อนข้างน้อย จึงต้องใช้เวลาในการนานกว่าที่เพียร์เหล่านี้จะถูกทดสอบผ่านทั้งเกณฑ์วัด C และ r_R ซึ่งต่างกับชุดข้อมูลควบคุมที่เพียร์เกือบทั้งหมดมีความเร็วในการดาวน์โหลดมากกว่า 200 กิโลบิตต่อวินาที จึงสามารถตรวจจับในชุดข้อมูลควบคุมได้เร็วกว่า



รูปที่ 4.4 ความเร็วในการตรวจจับเพียร์ด้วยเกณฑ์วัดร่วม $C + r_R$ ในชุดข้อมูลควบคุม

จำนวนเปอร์เซ็นต์สะสมของเพียร์ที่ตรวจจับได้มีความสอดคล้องกับจำนวนเปอร์เซ็นต์สะสมของปริมาณข้อมูลที่เพียร์มีการรับส่งกันก่อนถูกตรวจจับ (Transferred bytes until identified) โดยจากกราฟรูปที่ 4.4 ซึ่งเป็นชุดข้อมูลควบคุม ในช่วง 6 นาทีแรกจำนวนเปอร์เซ็นต์สะสมของปริมาณข้อมูลที่เกิดขึ้นมีจำนวนใกล้เคียงกับจำนวนเปอร์เซ็นต์สะสมของปริมาณข้อมูลที่เพียร์ทั้งหมดมีการรับส่งกัน (Total bytes of bittorrent peers) หลังจากนั้น เมื่อเพียร์ส่วนใหญ่ โดยเฉพาะเพียร์ที่มีการรับส่งข้อมูลจำนวนมากถูกตรวจจับได้ จำนวนเปอร์เซ็นต์สะสมของปริมาณข้อมูลที่เพียร์มีการรับส่งกันก่อนถูกตรวจจับมีอัตราการเพิ่มที่น้อยมาก และเมื่อเพียร์ทั้งหมดถูกตรวจจับได้ จำนวนเปอร์เซ็นต์สะสมในส่วนนี้จึงไม่มีการเพิ่มอีกต่อไป ซึ่งในช่วงระยะเวลาการทดลองของข้อมูลชุดนี้ พบว่าสามารถลดปริมาณข้อมูลที่เกิดจากเพียร์ทั้งหมดลงได้ 92% (สังเกตได้จากค่าผลต่างของกราฟทั้งสองเส้นที่แต่ละช่วงเวลา)

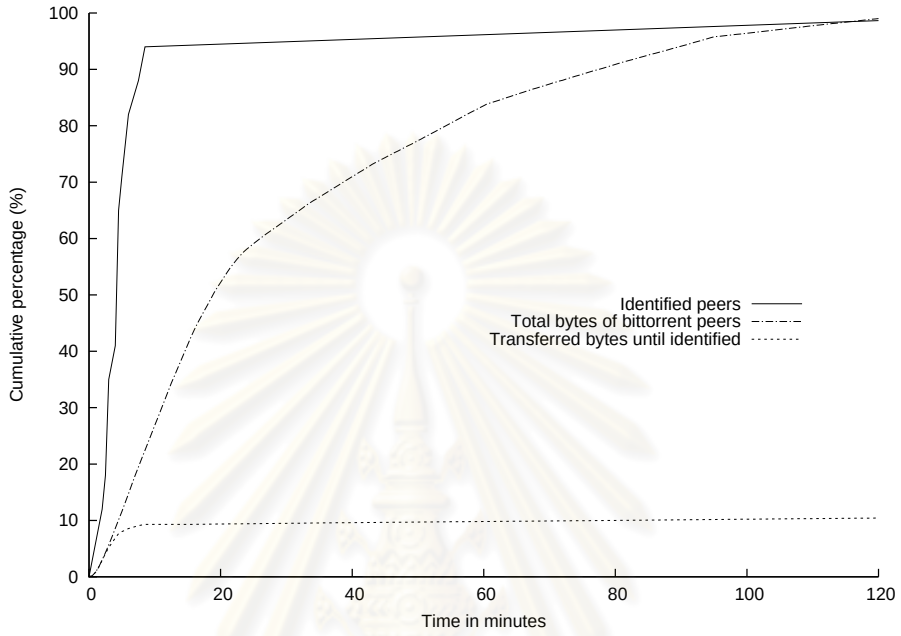
เช่นเดียวกันกับกราฟในรูปที่ 4.5 ซึ่งเป็นความเร็วของการตรวจจับในชุดข้อมูลปกติจากระบบเครือข่าย แต่เนื่องจากหมายเลขไอพีในกลุ่มนี้ไม่มีการควบคุมใดๆ ดังนั้นเพียร์จึงปรากฏในช่วงเวลาที่แตกต่างกัน เส้นกราฟจำนวนเปอร์เซ็นต์สะสมของปริมาณข้อมูลที่เกิดจากเพียร์ทั้งหมด และเส้นกราฟจำนวนเปอร์เซ็นต์สะสมของปริมาณข้อมูลที่เพียร์มีการรับส่งกันก่อนถูกตรวจจับจึงมีลักษณะเพิ่มขึ้นอย่างต่อเนื่อง แต่ยังคงมีความแตกต่างกับจำนวนเปอร์เซ็นต์สะสมของปริมาณข้อมูลที่เกิดขึ้นทั้งหมดอย่างเห็นได้ชัด แสดงให้เห็นว่าเกณฑ์วัดร่วม $C + r_R$ สามารถตรวจจับเพียร์ได้ก่อนที่เพียร์จะมีการรับส่งข้อมูลออกมาจำนวนมาก และถึงแม้มีเพียร์จำนวนหนึ่งที่ไม่สามารถตรวจจับได้ด้วยเกณฑ์วัดร่วม $C + r_R$ แต่ปริมาณข้อมูลที่เกิดจากเพียร์เหล่านี้มีน้อยมาก เมื่อเทียบกับปริมาณข้อมูลของบิตทอร์เรนต์ที่เกิดขึ้นทั้งหมด โดยสังเกตได้จากเส้นกราฟที่ความชันใกล้เคียง 0 หลังจากผ่านนาทีที่ 100 ไปแล้ว ซึ่งในช่วงระยะเวลาการทดลองของข้อมูลชุดนี้ พบว่าสามารถลดปริมาณข้อมูลที่เกิดจากเพียร์ทั้งหมดลงได้ 94.75%



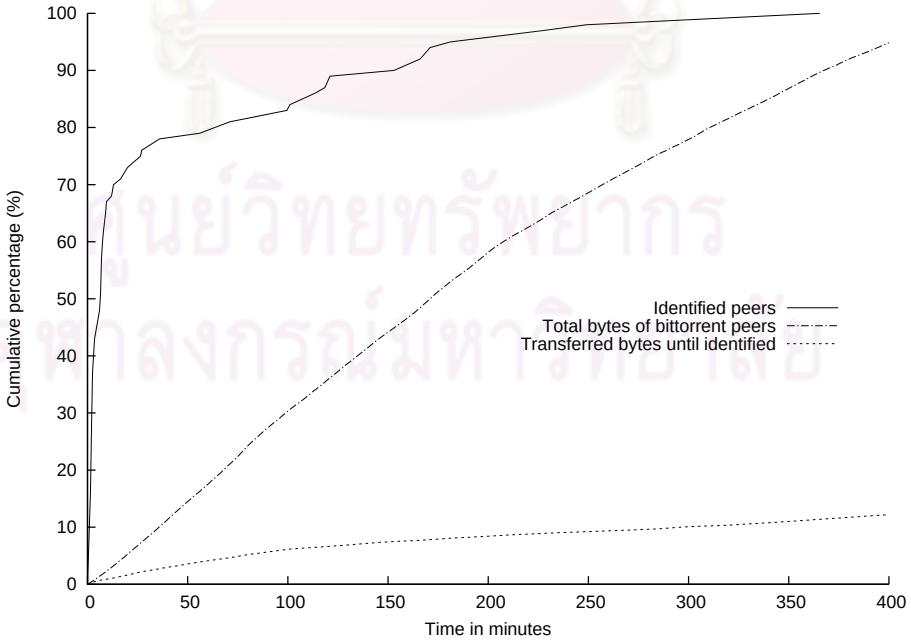
รูปที่ 4.5 ความเร็วในการตรวจจับเพียร์ด้วยเกณฑ์วัดร่วม $C + r_R$ ในชุดข้อมูลปกติ

จากกราฟรูปที่ 4.6 เกณฑ์วัดร่วม $C + r_A + r_R$ ใช้เวลาในการตรวจจับเพียร์มากกว่าเกณฑ์วัดร่วม $C + r_R$ ในชุดข้อมูลควบคุมเล็กน้อย โดย 95% ของเพียร์ที่ตรวจจับได้ใช้เวลาไม่เกิน 10 นาที แต่ในชุดข้อมูลปกติจากระบบเครือข่ายในกราฟรูปที่ 4.7 พบว่าต้องใช้เวลามากกว่าเกณฑ์วัดร่วม $C + r_R$ พอสมควร เนื่องจากมีเพียร์จำนวนหนึ่งที่ติดต่อกับหมายเลขไอพีจำนวนมาก แต่มีการเชื่อมต่อที่มีการรับส่งข้อมูลน้อย ทำให้ต้องใช้เวลานานกว่าจะผ่านการทดสอบด้วยเกณฑ์วัด r_A จึงทำให้มีเปอร์เซ็นต์สะสมของปริมาณข้อมูลที่เพียร์มีการรับส่งก่อนถูกตรวจจับมากกว่าของ

เกณฑ์วัดร่วม $C + r_R$ โดยในช่วงระยะเวลาการทดลองของชุดข้อมูลควบคุม พบว่าสามารถลดปริมาณข้อมูลที่เกิดจากเพียร์ลงได้ 87.27% และสามารถลดปริมาณข้อมูลลงได้ 89.19% ในชุดข้อมูลปกติ



รูปที่ 4.6 ความเร็วในการตรวจจับเพียร์ด้วยเกณฑ์วัดร่วม $C + r_A + r_R$ ในชุดข้อมูลควบคุม



รูปที่ 4.7 ความเร็วในการตรวจจับเพียร์ด้วยเกณฑ์วัดร่วม $C + r_A + r_R$ ในชุดข้อมูลปกติ

ปริมาณข้อมูลบิตทอร์เรนต์ที่สามารถลดลงได้ในแต่ละชุดข้อมูล ยังสามารถลดลงได้มากกว่านี้หากมีการทดลองที่ยาวนานขึ้น โดยเฉพาะในชุดข้อมูลปกติจากระบบเครือข่าย เนื่องจากเพียร์ที่ถูกตรวจจับได้เหล่านี้ยังคงมีการรับส่งข้อมูลกันอยู่หลังจากที่หยุดกระบวนการเก็บข้อมูลเพื่อนำมาทดลองแล้ว ซึ่งถือเป็นพฤติกรรมตามปกติของการใช้งานบิตทอร์เรนต์ที่มักมีช่วงเวลาของการดาวน์โหลดที่ยาวนาน

4.5 การวิเคราะห์อิทธิพลของค่าขีดเริ่มเปลี่ยน

จากผลการทดลองในหัวข้อที่ 4.3 การนำเกณฑ์วัดที่แตกต่างกันมาใช้ร่วมกันให้ความถูกต้องและความผิดพลาดที่แตกต่างกัน แสดงให้เห็นว่าแต่ละพฤติกรรมมีผลโดยตรงต่อประสิทธิภาพของตัวระบุเพียร์ ในหัวข้อนี้จึงเป็นการวิเคราะห์อิทธิพลของค่าขีดเริ่มเปลี่ยนของแต่ละเกณฑ์วัดที่มีผลต่อความแม่นยำในการตรวจจับ เพื่อหาค่าขีดเริ่มเปลี่ยนที่เหมาะสมของแต่ละเกณฑ์ โดยได้ทำการวิเคราะห์กับชุดข้อมูลที่สร้างขึ้นจากสภาพแวดล้อมควบคุม เนื่องจากเป็นชุดข้อมูลที่ทราบรายละเอียดอย่างชัดเจนว่าหมายเลขไอพีใดใช้งานบิตทอร์เรนต์ และหมายเลขไอพีใดไม่ได้ใช้งานบิตทอร์เรนต์ โดยได้ทำการประเมินประสิทธิภาพของการตรวจจับในด้านต่างๆ เมื่อมีการกำหนดค่าขีดเริ่มเปลี่ยนของเกณฑ์วัดที่แตกต่างกัน ดังนี้

1. จำนวนเพียร์ที่ตรวจจับได้ (True positive IPs) คือ จำนวนเพียร์ทั้งหมดที่ตรวจจับได้ในชุดข้อมูลควบคุม เมื่อมีการกำหนดค่าขีดเริ่มเปลี่ยนเป็นค่าต่างๆ

2. ปริมาณข้อมูลของเพียร์ที่ตรวจจับได้ (True positive bytes) คือ ปริมาณข้อมูลโดยรวมที่มีการรับส่งกันของเพียร์ที่ตรวจจับได้

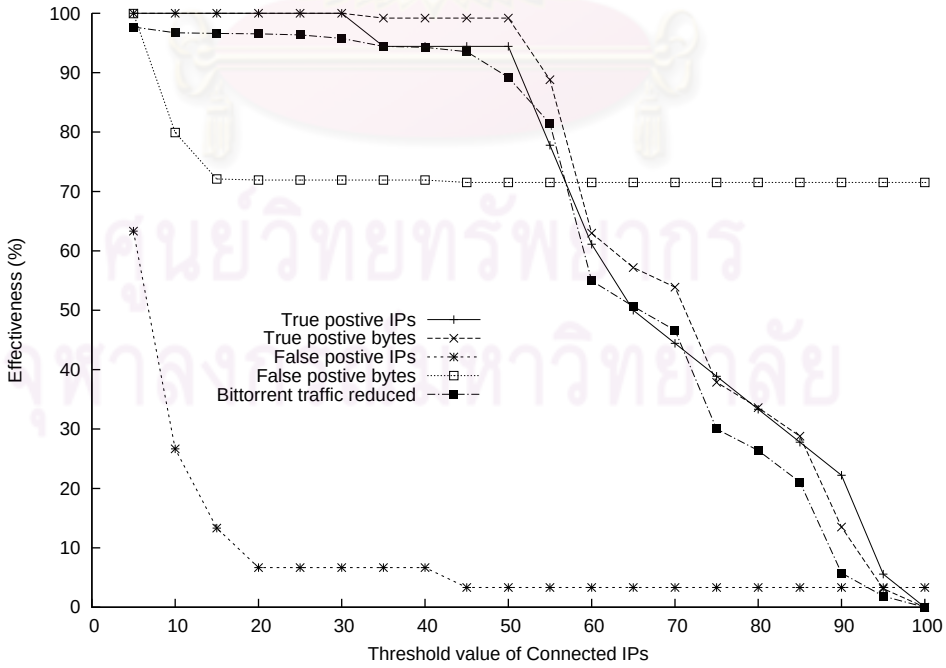
3. จำนวนหมายเลขไอพีที่ตรวจจับผิดพลาด (False positive IPs) คือ จำนวนของหมายเลขไอพีที่ไม่ได้ใช้งานบิตทอร์เรนต์ แต่ถูกระบุว่ามีการใช้งานบิตทอร์เรนต์

4. ปริมาณข้อมูลของหมายเลขไอพีที่ตรวจจับผิดพลาด (False positive bytes) คือ ปริมาณข้อมูลโดยรวมที่มีการรับส่งกันของหมายเลขไอพีที่ไม่ได้ใช้งานบิตทอร์เรนต์ แต่ถูกระบุว่ามีการใช้งานบิตทอร์เรนต์

5. ปริมาณข้อมูลบิตทอร์เรนต์ที่สามารถลดลงได้ (Bittorrent traffic reduced) โดยคำนวณจากปริมาณข้อมูลรวมของเพียร์ทั้งหมดในชุดข้อมูลควบคุม ลบด้วยปริมาณข้อมูลรวมของเพียร์ทั้งหมดในชุดข้อมูลควบคุม

เกณฑ์วัดที่นำมาวิเคราะห์อิทธิพลของค่าขีดเริ่มเปลี่ยนมี 4 เกณฑ์ คือ จำนวนหมายเลขไอพีที่ติดต่อด้วย (C) อัตราส่วนของการเชื่อมต่อที่มีการรับส่งข้อมูล (r_A) จำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูลสองทิศทาง (B) และอัตราส่วนความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลง (r_R) โดยทำการวิเคราะห์แต่ละเกณฑ์วัดแยกจากกัน เกณฑ์วัดอื่นที่ไม่เกี่ยวข้องจึงถูกกำหนดค่าขีดเริ่มเปลี่ยนเท่ากับ 0 เช่น กำหนดค่าขีดเริ่มเปลี่ยนของเกณฑ์วัด r_A, B และ r_R เท่ากับ 0 เพื่อวิเคราะห์อิทธิพลของค่าขีดเริ่มเปลี่ยนของเกณฑ์วัด C เป็นต้น ผลการวิเคราะห์ของแต่ละเกณฑ์วัดมีดังนี้

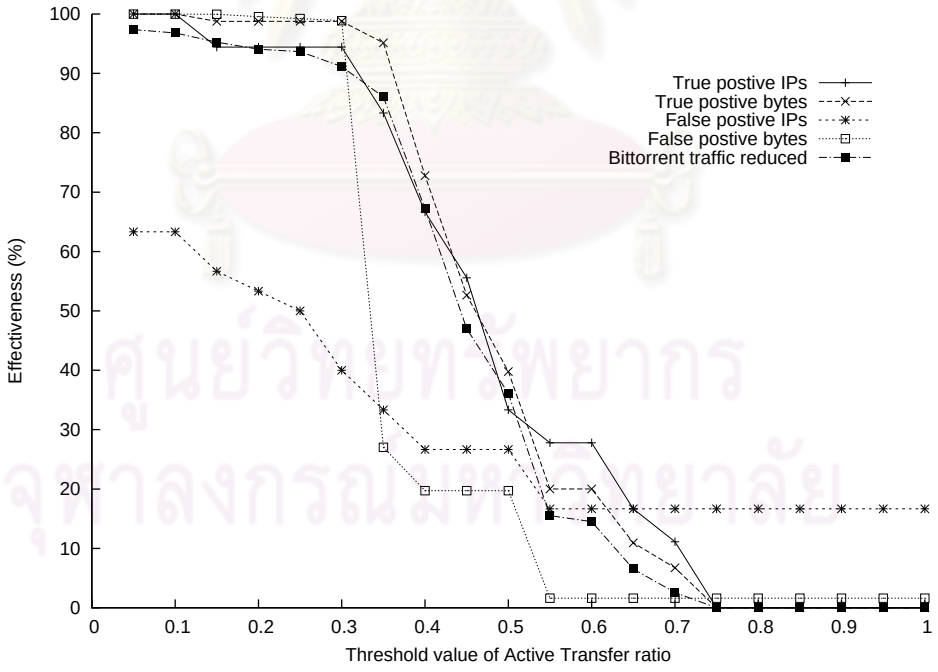
<u>เกณฑ์ที่ 1</u> จำนวนหมายเลขไอพีที่ติดต่อด้วย (Connected IPs: C) เนื่องจากเพียร์ของบิตทอร์เรนต์พยายามรักษาจำนวนเพียร์ในเซ็ตของเพียร์ไม่ให้ต่ำกว่า 20 อีกทั้งมีการแลกเปลี่ยนข้อความควบคุม เช่น ข้อความ Have กันอยู่ตลอดเวลาในช่วงการดาวน์โหลด จึงมีการติดต่อกับหมายเลขไอพีต่างๆ จำนวนมาก ดังนั้นเมื่อกำหนดค่าขีดเริ่มเปลี่ยนของเกณฑ์วัดไม่เกิน 30 จะสามารถตรวจจับเพียร์ (True positive IPs) ได้ทั้งหมด ดังแสดงในกราฟรูปที่ 4.8 โดยสามารถลดปริมาณข้อมูลบิตทอร์เรนต์ลงได้ (Bittorrent traffic reduced) มากกว่า 95% จากนั้นความถูกต้องจะเริ่มลดลงเมื่อกำหนดค่าขีดเริ่มเปลี่ยนมากขึ้น ทำให้ลดปริมาณข้อมูลบิตทอร์เรนต์ได้น้อยลงตามไปด้วย



รูปที่ 4.8 อิทธิพลของค่าขีดเริ่มเปลี่ยนของเกณฑ์วัด C ที่มีผลต่อความแม่นยำในการตรวจจับ

สำหรับแอพลิเคชันทั่วไป มักมีการติดต่อกับหมายเลขไอพีอื่นเท่าที่จำเป็น จำนวนของหมายเลขไอพีที่ระบุผิดพลาด (False positive IPs) จึงลดลงเหลือน้อยกว่า 7% เมื่อกำหนดค่าขีดเริ่มเปลี่ยนตั้งแต่ 20 ขึ้นไป ค่าขีดเริ่มเปลี่ยนที่เหมาะสมของเกณฑ์วัดนี้จึงอยู่ในช่วง 20-30 ซึ่งให้ค่าผลบวกแท้ 100% ในขณะที่ให้ค่าผลบวกลวงน้อยกว่า 7% อย่างไรก็ตาม แม้จำนวนหมายเลขไอพีที่ระบุผิดพลาดมีไม่ถึง 7% แต่ปริมาณข้อมูลที่เกิดจากหมายเลขไอพีเหล่านี้ (False positive bytes) กลับมีอัตราส่วนค่อนข้างมาก เนื่องจากในจำนวนนี้มีหมายเลขไอพีที่เป็นเครื่องให้บริการรวมอยู่ด้วย ซึ่งในบางช่วงเวลามีการติดต่อกับหมายเลขไอพีอื่นที่เข้ามาใช้บริการจำนวนมากเช่นกัน จึงถูกตรวจจับได้ด้วยเกณฑ์วัดนี้

<u>เกณฑ์ที่ 2</u> อัตราส่วนของการเชื่อมต่อที่มีการรับส่งข้อมูล (Active Transfer ratio: r_A) เพียร์ของบิตทอร์เรนต์สามารถดาวน์โหลดแต่ละชิ้นส่วนของไฟล์จากเพียร์ที่ต่างกันได้ในเวลาเดียวกัน จึงมีการเชื่อมต่อที่มีการรับส่งข้อมูลเป็นอัตราส่วนโดยตรงกับจำนวนของหมายเลขไอพีที่ติดต่อด้วย ดังนั้นเมื่อกำหนดค่าขีดเริ่มเปลี่ยนของเกณฑ์วัด r_A ไม่เกิน 0.35 จะทำให้สามารถตรวจจับเพียร์ได้อย่างมีประสิทธิภาพ ดังแสดงในกราฟรูปที่ 4.9

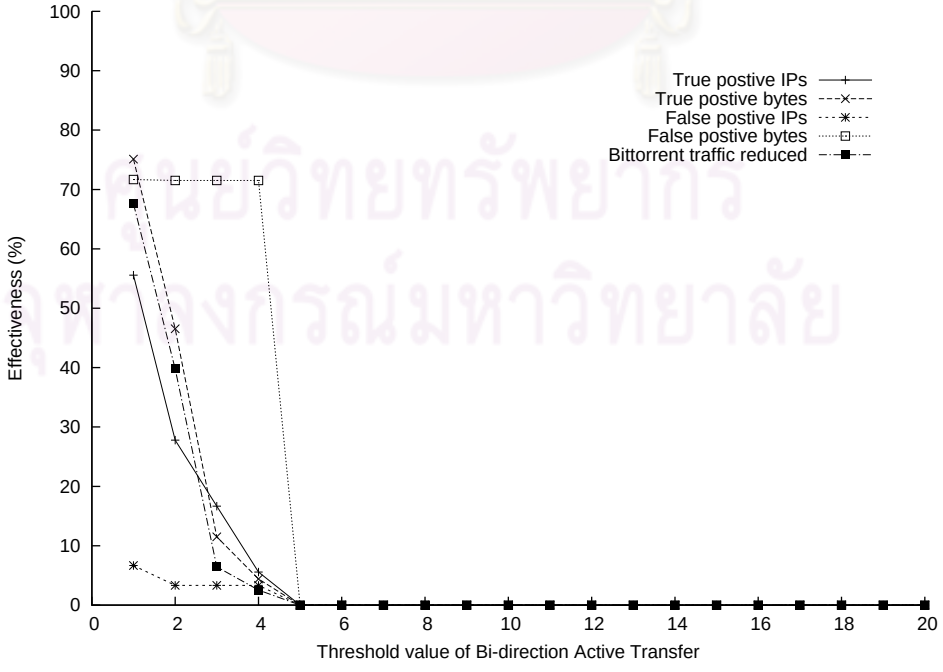


รูปที่ 4.9 อิทธิพลของค่าขีดเริ่มเปลี่ยนของเกณฑ์วัด r_A ที่มีผลต่อความแม่นยำในการตรวจจับ

ถึงแม้ว่ามีจำนวนหมายเลขไอพีที่ตรวจจับผิดค่อนข้างมาก แต่ปริมาณข้อมูลที่เกิดจากหมายเลขไอพีที่ตรวจจับผิดเหล่านี้กลับมีน้อยกว่าเมื่อเทียบกับเกณฑ์วัด c เนื่องจากหมายเลขไอ

พีของเครื่องให้บริการที่ติดต่อกับหมายเลขไอพีอื่นจำนวนมาก ซึ่งตรวจจับได้ด้วยเกณฑ์วัด C มีการเชื่อมต่อที่มีการรับส่งข้อมูลน้อยมาก เพราะการเชื่อมต่อกับหมายเลขไอพีส่วนใหญ่มีการรับส่งข้อมูลด้วยแพ็กเก็ตขนาดเล็ก จึงไม่ถูกตรวจจับด้วยเกณฑ์วัด r_A ส่วนสาเหตุที่มีจำนวนของหมายเลขไอพีที่ตรวจจับผิดพลาดค่อนข้างมาก เนื่องจากบางหมายเลขไอพีมีการติดต่อกับหมายเลขไอพีอื่นน้อย และในจำนวนนี้มีการรับส่งข้อมูลด้วย เช่น การใช้งานเอฟทีพี เป็นต้น ส่งผลให้หมายเลขไอพีเหล่านี้มีค่าของเกณฑ์วัด r_A ค่อนข้างสูง

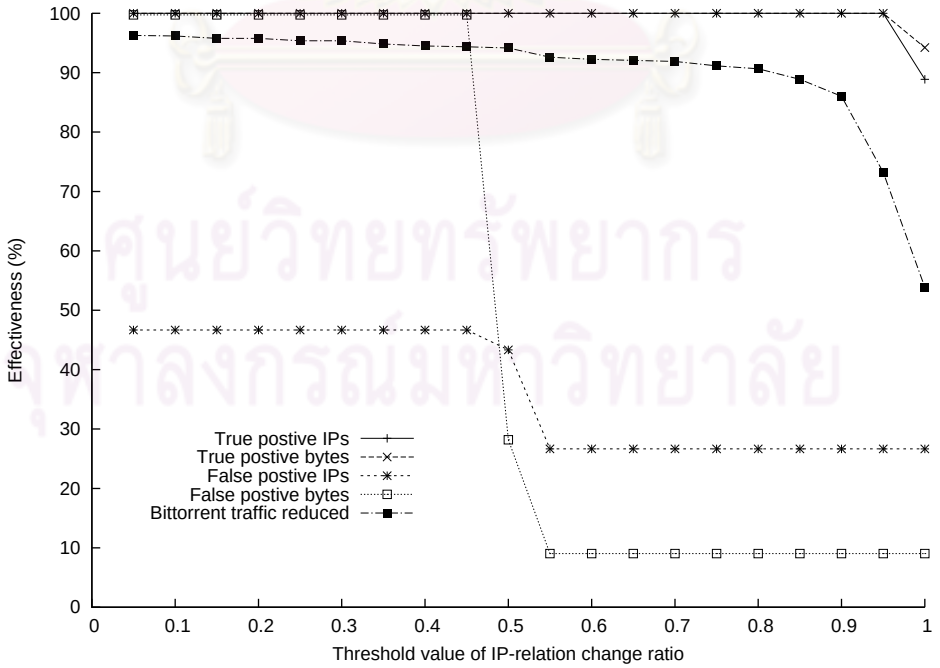
<u>เกณฑ์ที่ 3</u> จำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูลสองทิศทาง (Bi-directional Active Transfer: B) เพียร์ของบิตทอร์เรนต์พยายามเลือกอัพโหลดให้แก่เพียร์ที่ตนเองดาวน์โหลดได้เร็วที่สุด ทำให้การเชื่อมต่อระหว่างบางคู่เพียร์มีข้อมูลรับส่งกันทั้งสองทิศทาง แต่เนื่องจากการทดลองนี้ทำการดาวน์โหลดเพียร์ละ 1 ไฟล์ทอร์เรนต์ จึงมีโอกาสเกิดการถ่ายโอนข้อมูลสองทิศทางไม่เกิน 4 การเชื่อมต่อ และข้อมูลส่วนใหญ่ของแต่ละเพียร์ได้ดาวน์โหลดมาจากซีด (Seed) ดังนั้นโอกาสที่แต่ละเพียร์จะมีการรับส่งข้อมูลสองทิศทางจึงมีค่อนข้างน้อย อีกทั้งระบบเครือข่ายที่ใช้เก็บข้อมูลทั้งสองชุดมีอุปกรณ์ไฟร์วอลล์ติดตั้งอยู่ดังที่ได้อธิบายจากรูปที่ 4.1 และ 4.2 ทำให้ข้อมูลที่ใช้ทดลองทั้งสองชุดไม่มีกระแสข้อมูลที่เกิดจากเพียร์ภายนอกเป็นผู้ทำการเปิดการเชื่อมต่อ ปัจจัยเหล่านี้ส่งผลกระทบโดยตรงกับความสามารถในการตรวจจับเพียร์ของเกณฑ์วัด B ดังแสดงในรูปที่ 4.10



รูปที่ 4.10 อิทธิพลของค่าขีดเริ่มเปลี่ยนของเกณฑ์วัด B ที่มีผลต่อความแม่นยำในการตรวจจับ

จากรูปที่ 4.10 จะเห็นว่าทั้งความถูกต้องในการตรวจจับเพียร์และความผิดพลาดในการตรวจจับหมายเลขไอพีที่ไม่ได้ใช้งานบิตทอร์เรนต์มีค่าน้อยมาก โดยเฉพาะเมื่อมีการกำหนดค่าขีดเริ่มเปลี่ยนตั้งแต่ 5 ขึ้นไป ซึ่งทำให้ทั้งค่าผลบวกแท้และผลบวกลวงมีค่าเป็น 0 ดังนั้นเกณฑ์วัด B จึงไม่เหมาะสมกับการนำมาสร้างเป็นตัวระบุเพียร์เพื่อใช้ในระบบเครือข่ายที่มีอุปกรณ์คอยบอกปัดการเชื่อมต่อจากภายนอก หรือการตรวจจับเพียร์ที่มีการดาวน์โหลดจากไฟล์ทอร์เรนต์เพียง 1 หรือ 2 ไฟล์พร้อมกัน ดังนั้นในงานวิจัยนี้จึงไม่นำเกณฑ์วัด B มาสร้างเป็นเกณฑ์วัดร่วมเพื่อใช้ตรวจจับเพียร์

<u>เกณฑ์ที่ 4</u> อัตราส่วนความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลง (IP-relation Changes ratio: r_R) เพียร์ของบิตทอร์เรนต์จะมีการค้นหาเพียร์ที่ให้ค่าดาวน์โหลดที่ดีกว่าอยู่เสมอ ส่งผลให้มีพฤติกรรมการเปลี่ยนคู่ส่งอย่างต่อเนื่อง จำนวนของความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลงจึงเป็นอัตราส่วนโดยตรงกับจำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูล พฤติกรรมนี้ค่อนข้างชัดเจนกว่าพฤติกรรมอื่นที่นำเสนอ ดังนั้นเมื่อกำหนดค่าขีดเริ่มเปลี่ยนของเกณฑ์วัด r_R มีค่าต่างๆ จนถึง 0.9 จะยังคงสามารถตรวจจับเพียร์ได้ทั้งหมด แต่ค่าขีดเริ่มเปลี่ยนดังกล่าวส่งผลให้ปริมาณข้อมูลบิตทอร์เรนต์ที่สามารถลดลงได้เท่ากับ 85% ค่าขีดเริ่มเปลี่ยนที่เหมาะสมจึงอยู่ในช่วง 0.55-0.8 ซึ่งสามารถลดปริมาณข้อมูลบิตทอร์เรนต์ลงได้ 90% ดังแสดงในกราฟรูปที่ 4.11



รูปที่ 4.11 อิทธิพลของค่าขีดเริ่มเปลี่ยนของเกณฑ์วัด r_R ที่มีผลต่อความแม่นยำในการตรวจจับ

อย่างไรก็ตาม ผลที่ได้จากเกณฑ์วัด r_R ยังมีจำนวนของหมายเลขไอพีที่ตรวจจับผิดพลาดอยู่ค่อนข้างมาก เนื่องจากหมายเลขไอพีปกติเหล่านี้มีการรับส่งข้อมูลกับหมายเลขไอพีอื่นค่อนข้างน้อย และเมื่อมีบางการเชื่อมต่อกับหมายเลขไอพีเหล่านั้นมีการเปลี่ยนสถานะความสัมพันธ์ ค่าของเกณฑ์วัด r_R ที่เกิดขึ้นจึงค่อนข้างสูงเช่นเดียวกัน

จากการวิเคราะห์อิทธิพลของค่าขีดเริ่มเปลี่ยนของเกณฑ์วัดทั้ง 4 ด้วยการกำหนดค่าขีดเริ่มเปลี่ยนเป็นค่าต่างๆ พบว่าการใช้เพียงเกณฑ์วัดเดียวมาสร้างเป็นตัวระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ให้ประสิทธิภาพที่ไม่ดีนัก ดังนั้นการนำเกณฑ์วัดมาใช้ร่วมกันจะช่วยเพิ่มความถูกต้องและลดความผิดพลาดดังกล่าวลงได้อย่างมาก เช่น การนำเกณฑ์วัด C , r_A หรือ r_R มาใช้ร่วมกัน ซึ่งผลที่ได้คือเกณฑ์วัดร่วม $C + r_R$ และเกณฑ์วัดร่วม $C + r_A + r_R$ ซึ่งมีความสามารถในการตรวจจับเพียร์ในการทดลองกับทั้งสองชุดข้อมูลเป็นอย่างดี โดยช่วงของขีดเริ่มเปลี่ยนที่เหมาะสมของแต่ละเกณฑ์วัดแสดงได้ดังตารางที่ 4.9

ตารางที่ 4.9 ช่วงของค่าขีดเริ่มเปลี่ยนที่เหมาะสมของแต่ละเกณฑ์วัด

เกณฑ์วัด	ช่วงของค่าขีดเริ่มเปลี่ยนที่เหมาะสม
Connected IPs (C)	20 - 30
Active Transfer ratio (r_A)	0.3 - 0.35
Bi-directional Active Transfer ratio (B)	ไม่พบช่วงที่เหมาะสม
IP-relation Changes ratio (r_R)	0.55 - 0.8

หากพิจารณาจากตารางที่ 4.9 และผลการทดลองในหัวข้อที่ 4.3.1 ตัวระบุเพียร์ที่สร้างขึ้นเพื่อนำไปใช้งานจริงควรสร้างจากเกณฑ์วัดร่วม $C + r_R$ หรือเกณฑ์วัดร่วม $C + r_A + r_R$ โดยที่เกณฑ์วัดร่วม $C + r_R$ ควรมีค่าขีดเริ่มเปลี่ยนของเกณฑ์วัด C และ r_R อยู่ในช่วง 20-30 และ 0.55-0.8 ตามลำดับ และมีค่าขีดเริ่มเปลี่ยนของเกณฑ์วัด B และ r_A เท่ากับ 0 ในขณะที่เกณฑ์วัดร่วม $C + r_A + r_R$ ควรมีค่าขีดเริ่มเปลี่ยนของเกณฑ์วัด C , r_A และ r_R อยู่ในช่วง 20-30, 0.3-0.35 และ 0.55-0.8 ตามลำดับ และมีค่าขีดเริ่มเปลี่ยนของเกณฑ์วัด B และ r_A เท่ากับ 0

4.6 การเปรียบเทียบประสิทธิภาพกับงานวิจัยที่เกี่ยวข้อง

หัวข้อนี้เป็นการเปรียบเทียบประสิทธิภาพของตัวระบุเพียร์ของบิตทอร์เรนต์ที่นำเสนอกับงานวิจัยที่เกี่ยวข้อง ซึ่งเป็นงานวิจัยที่ระบุเพียร์ด้วยการวิเคราะห์ในระดับโฟล และมีความเป็นไปได้

ในการนำไปประยุกต์ใช้ในแบบออนไลน์ โดยพฤติกรรมที่ใช้ในการตรวจจับเพียร์ของงานวิจัยเหล่านี้คือ

1. เพียร์จะมีจำนวนของการเชื่อมต่อที่ไม่สำเร็จ (Failed connections: FC) ในอัตราค่อนข้างสูง [14, 16] เนื่องจากเพียร์พยายามเปิดการเชื่อมต่อไปยังเพียร์อื่นจำนวนมาก แต่เพียร์เหล่านั้นอาจไม่ได้อยู่ในระบบแล้วในขณะนั้น ต่างจากแอพลิเคชันทั่วไป เช่น เวิลด์ไวด์เว็บ ที่แต่ละเครื่องให้บริการเว็บไซต์มีช่วงเวลาให้บริการ (Uptime) ที่ยาวนาน จึงทำให้มีจำนวนของโฟลที่เชื่อมต่อไม่สำเร็จค่อนข้างน้อย โดยในงานวิจัย [16] ได้ให้การนิยาม (Definition) ของพฤติกรรมการเชื่อมต่อที่ไม่สำเร็จว่า มีการส่งแพ็กเก็ต SYN ภายในโฟลเดียวกันตั้งแต่ 4 แพ็กเก็ตขึ้นไป โดยไม่มีแพ็กเก็ต SYN/ACK ตอบรับ อย่างไรก็ตาม เมื่อทำการทดลองตรวจจับหมายเลขไอพีที่ใช้งานบิตทอร์เรนต์กับข้อมูลทั้งสองชุดด้วยนิยามของพฤติกรรมดังกล่าว ปรากฏว่าสามารถตรวจจับเพียร์ได้ไม่ถึง 1% งานวิจัยนี้จึงได้ปรับเปลี่ยนการนิยามของพฤติกรรมนี้บางส่วน โดยได้เปลี่ยนการนิยามของการเชื่อมต่อที่ไม่สำเร็จว่ามีการส่งแพ็กเก็ต SYN ตั้งแต่ 3 แพ็กเก็ตขึ้นไปภายในโฟลเดียวกันโดยไม่มีแพ็กเก็ต SYN/ACK ตอบรับ (FC_3)

2. เพียร์จะมีจำนวนของโฟลที่เชื่อมต่อกันด้วยหมายเลขพอร์ตที่ไม่ได้รับอนุญาตให้ใช้งาน (Unprivileged-to-Unprivileged port numbers: UP-UP) [16] หรือหมายเลขพอร์ตตั้งแต่ 1024 ขึ้นไป เป็นจำนวนมาก เนื่องจากแอพลิเคชันทั่วไปมักมีหมายเลขพอร์ตโดยปริยาย เช่น บริการเวิลด์ไวด์เว็บมีหมายเลขพอร์ต 80 เป็นหมายเลขพอร์ตโดยปริยาย แต่แอพลิเคชันแบบเพียร์ทูเพียร์มักมีการสุ่มหมายเลขพอร์ตที่สูงกว่า 1024 ในการทำงาน

3. เพียร์จะมีจำนวนของการเชื่อมต่อขาออกและการเชื่อมต่อขาเข้าในระดับที่สมดุลกัน (Balances of incoming and outgoing connections: BC) [16] เนื่องจากแต่ละเพียร์สามารถเป็นได้ทั้งผู้ใช้บริการและผู้ให้บริการในเวลาเดียวกัน เพียร์จึงมีทั้งการเชื่อมต่อขาออกและการเชื่อมต่อขาเข้า ต่างจากแอพลิเคชันปกติที่แต่ละเครื่องมักเป็นผู้ให้บริการหรือผู้ใช้บริการอย่างใดอย่างหนึ่ง จึงมีเฉพาะการเชื่อมต่อขาเข้าหรือการเชื่อมต่อขาออกเท่านั้น

แต่เนื่องจากระบบเครือข่ายที่ใช้เก็บข้อมูลทั้งสองชุดมีอุปกรณ์ไฟร์วอลล์ติดตั้งอยู่ จึงทำให้การเชื่อมต่อจากภายนอกไม่สามารถติดต่อเข้ามายังเพียร์ที่อยู่ภายในได้ ทำให้การตรวจจับหมายเลขไอพีที่ใช้งานบิตทอร์เรนต์โดยใช้พฤติกรรมความสมดุลของการเชื่อมต่อขาเข้าและขาออกไม่สามารถทำงานได้ ในหัวข้อนี้จึงแสดงการเปรียบเทียบกับ 2 พฤติกรรมแรกเท่านั้น โดยมีการ

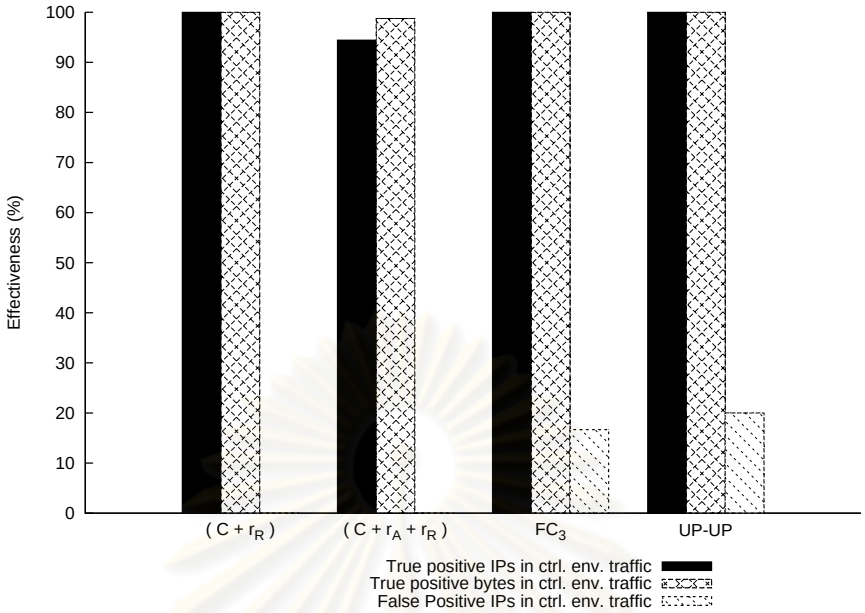
เปรียบเทียบประสิทธิภาพในด้านความถูกต้อง ความเร็วในการตรวจจับ และการใช้ทรัพยากรของระบบ

อย่างไรก็ดี วิธีทั้งสองใช้การตรวจสอบจากรูปแบบการเชื่อมต่อกันของโฟล โดยไม่ได้พิจารณาพฤติกรรมของเพียร์ในขณะที่มีการรับส่งข้อมูลจริง เพียร์ที่ตรวจจับได้บางตัวจึงอาจยังไม่มีการรับส่งข้อมูลหรือมีการรับส่งข้อมูลกันเพียงเล็กน้อย ซึ่งมีผลทางลบต่อปริมาณการใช้ข้อมูลในเครือข่ายไม่มากนัก จึงอาจยังไม่มีความจำเป็นต้องควบคุมเพียร์เหล่านั้น เนื่องจากเพียร์ที่จะก่อให้เกิดปัญหาเกี่ยวกับการใช้งานแบนด์วิดท์ของระบบเครือข่ายคือเพียร์มีการรับส่งข้อมูลจำนวนมาก การวิเคราะห์จากพฤติกรรมของเพียร์ในขณะที่มีการรับส่งข้อมูลจึงสามารถตรวจจับเพียร์ที่จะก่อให้เกิดปัญหาแก่ระบบเครือข่ายได้อย่างแท้จริง ดังจะเห็นได้จากผลการเปรียบเทียบความถูกต้องในหัวข้อถัดไป ซึ่งแม้ตัวระบุเพียร์ที่นำเสนอจะมีจำนวนเพียร์ที่ตรวจจับได้น้อยกว่าวิธีตรวจหาจากโฟล แต่ปริมาณข้อมูลที่เกิดจากเพียร์เหล่านั้นใกล้เคียงกับปริมาณข้อมูลที่เกิดจากเพียร์ซึ่งตรวจจับได้ด้วยวิธีตรวจสอบในระดับโฟลทั้งสอง แสดงให้เห็นว่าการตรวจจับเพียร์ที่มีการสร้างการเชื่อมต่อไว้โดยไม่มีการรับส่งข้อมูลได้ ไม่ช่วยให้ลดปริมาณข้อมูลที่เกิดขึ้นลงได้อย่างมีนัยสำคัญ อีกทั้งการพิจารณาจากการเชื่อมต่อของโฟลยังหลีกเลี่ยงได้ง่าย และมีโอกาสที่รูปแบบการเชื่อมต่อจะคล้ายกับแอพลิชันปกติ ทำให้เกิดการตรวจจับผิดพลาดได้

4.6.1 การเปรียบเทียบความถูกต้อง

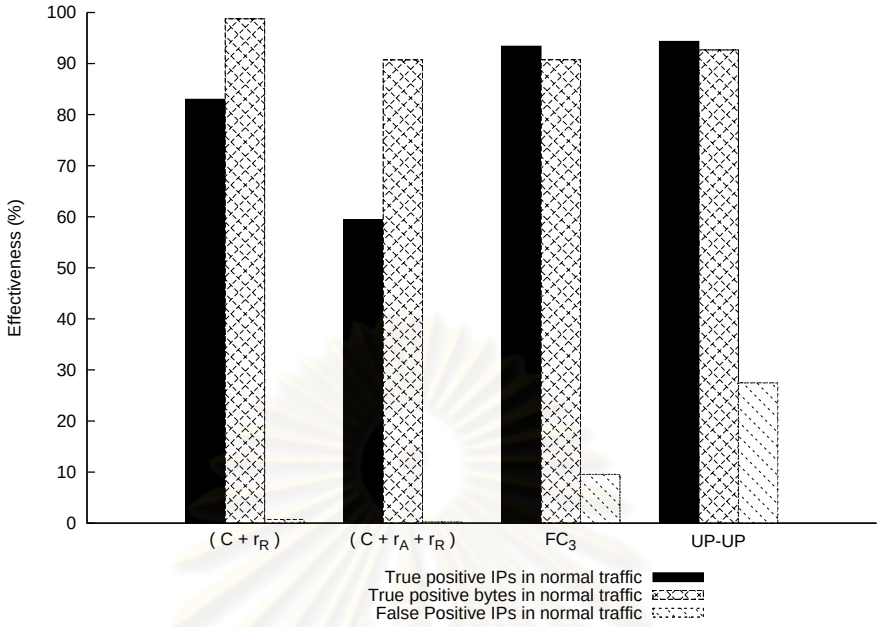
การเปรียบเทียบความถูกต้องในหัวนี้ ได้ทำการเปรียบเทียบในด้านผลบวกแท้ของหมายเลขไอพีที่ใช้งานบิตทอร์เรนต์หรือเพียร์ที่ตรวจจับได้ (True positive IPs) ผลบวกแท้ของปริมาณข้อมูลที่เพียร์มีการรับส่ง (True positive bytes) และผลบวกลวงของหมายเลขไอพีที่ตรวจจับ (False positive IPs) ในแต่ละชุดข้อมูล

ผลการเปรียบเทียบความถูกต้องในชุดข้อมูลควบคุมจากกราฟรูปที่ 4.12 แสดงให้เห็นว่าเกณฑ์วัดร่วมทั้งสองที่นำเสนอสามารถตรวจจับเพียร์ได้ดีใกล้เคียงกับวิธี FC_3 และ $UP-UP$ โดยเฉพาะเกณฑ์วัดร่วม $C+r_R$ ที่สามารถตรวจจับเพียร์ได้ทั้งหมดเช่นเดียวกัน และถึงแม้เกณฑ์วัดร่วม $C+r_A+r_R$ จะตรวจจับได้น้อยกว่า 1 ตัว แต่ปริมาณข้อมูลของเพียร์ที่ตรวจจับได้มีการรับส่ง (True positive bytes) ยังคงอยู่ในระดับเดียวกับวิธีอื่น



รูปที่ 4.12 การเปรียบเทียบความถูกต้องในชุดข้อมูลควบคุม

ส่วนผลการเปรียบเทียบในชุดข้อมูลปกติจากระบบเครือข่ายแสดงดังในรูปที่ 4.13 ทั้งวิธี FC_3 และ $UP-UP$ สามารถตรวจจับเพียร์ได้มากกว่าเกณฑ์วัดร่วมที่นำเสนอทั้งสอง โดยเฉพาะเกณฑ์วัดร่วม $C+r_A+r_R$ ซึ่งสามารถตรวจจับเพียร์ได้น้อยกว่าวิธีทั้งสองประมาณ 30% แต่เมื่อพิจารณาจากปริมาณข้อมูลของเพียร์ที่ตรวจจับได้มีการรับส่ง พบว่ามีจำนวนใกล้เคียงกับวิธี FC_3 และ $UP-UP$ โดยเฉพาะเกณฑ์วัดร่วม $C+r_R$ ซึ่งมีปริมาณข้อมูลที่เกิดจากเพียร์ที่ตรวจจับได้มากกว่าวิธี FC_3 และ $UP-UP$ แม้ว่าจะมีจำนวนของเพียร์ที่ตรวจจับได้น้อยกว่า เนื่องจากมีเพียร์ที่มีการรับส่งข้อมูลจำนวนมากบางตัว เปิดการเชื่อมต่อไปยังหมายเลขพอร์ตที่ต่ำกว่า 1024 ของหมายเลขไอพีอื่นจำนวนมาก วิธี $UP-UP$ จึงไม่สามารถตรวจจับได้ อีกทั้งการเชื่อมต่อเกือบทั้งหมดมีการส่งแพ็กเก็ต SYN ติดต่อกันน้อยกว่า 3 แพ็กเก็ต วิธี FC_3 จึงไม่สามารถตรวจจับได้เช่นกัน จากผลดังกล่าวแสดงให้เห็นว่า การตรวจจับเพียร์ด้วยพฤติกรรมของเพียร์ในขณะที่มีการรับส่งข้อมูลจริง สามารถตรวจจับเพียร์ที่อาจก่อให้เกิดปัญหาแก่ระบบเครือข่ายได้ทั้งหมด และถึงแม้การตรวจจับเพียร์ด้วยการพิจารณาในระดับโฟลจะตรวจจับเพียร์ได้มากกว่า แต่วิธีเหล่านี้ไม่ได้พิจารณาจากการรับส่งข้อมูลของเพียร์ จึงอาจไม่สามารถตรวจจับเพียร์ที่มีการรับส่งข้อมูลจำนวนมากบางตัวที่การปรับเปลี่ยนรูปแบบการเชื่อมต่อให้ใกล้เคียงกับแอพลิเคชันปกติได้



รูปที่ 4.13 การเปรียบเทียบความถูกต้องในชุดข้อมูลปกติ

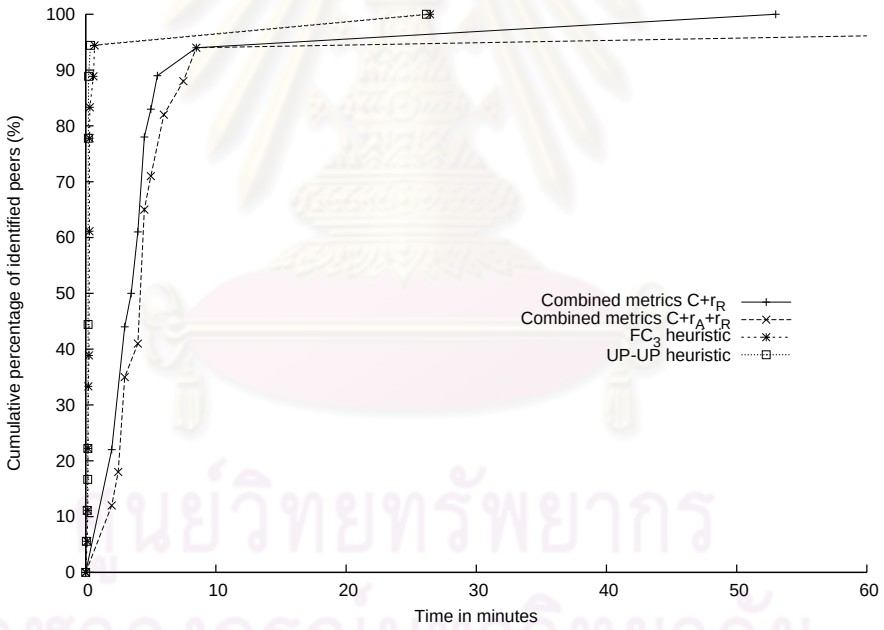
นอกจากนี้ แม้ว่าวิธี FC_3 และ $UP-UP$ จะมีความสามารถในการตรวจจับเพียร์ได้เป็นอย่างดี แต่ทั้งสองวิธีดังกล่าวก็สามารถหลีกเลี่ยงได้ง่ายเช่นเดียวกัน โดยเฉพาะวิธี $UP-UP$ ที่สามารถหลีกเลี่ยงได้ด้วยการกำหนดให้บิตทอร์เรนต์ไคลเอนต์ติดต่อกับหมายเลขไอพีอื่นเฉพาะที่หมายเลขพอร์ตต่ำกว่า 1024 กรณีเช่นนี้ถือเป็นตัวอย่างของการเปลี่ยนแปลงในระดับชั้นขนส่งที่ส่งผลกระทบโดยตรงต่อประสิทธิภาพการทำงานของตัวระบุเพียร์ของงานวิจัยที่เกี่ยวข้อง ส่วนตัวระบุเพียร์ที่นำเสนอในงานวิจัยนี้ทำการตรวจสอบในระดับชั้นเครือข่าย จึงไม่มีผลกระทบจากการเปลี่ยนแปลงในระดับชั้นขนส่งดังกล่าว

ทั้งเกณฑ์วัดร่วม $C+r_R$ และ $C+r_A+r_R$ มีความผิดพลาดในการตรวจจับ (False positive IPs) น้อยมาก โดยมีอัตราส่วนความผิดพลาดไม่ถึง 1% ในทั้งสองชุดข้อมูล ในขณะที่วิธี FC_3 และ $UP-UP$ ซึ่งอาศัยพฤติกรรมโดยทั่วไปของเพียร์ทูเพียร์ มีการตรวจจับหมายเลขไอพีที่มีการใช้งานแอพลิเคชันแบบเพียร์ทูเพียร์อื่นบางประเภท เช่น การใช้งานไออาร์ซี (IRC) และเกมส์ออนไลน์ (Online Game) ว่ามีการใช้งานบิตทอร์เรนต์ อีกทั้งในปัจจุบันมีการพัฒนาแอพลิเคชันใหม่ๆ อยู่ตลอดเวลา โดยที่แอพลิเคชันเหล่านี้มักใช้หมายเลขพอร์ตที่ไม่ได้อยู่ในช่วง 1-1024 เนื่องจากถูกพัฒนาขึ้นในภายหลัง ลักษณะเช่นนี้ส่งผลกระทบโดยตรงต่อประสิทธิภาพของวิธี $UP-UP$ ดังนั้นวิธี FC_3 และ $UP-UP$ จึงอาจไม่เหมาะสมในการนำมาใช้ตรวจจับเพียร์ของบิตทอร์เรนต์โดยเฉพาะ

4.6.2 การเปรียบเทียบความเร็วในการตรวจจับ

ในหัวนี้ได้ทำการเปรียบเทียบเวลาที่ใช้ในการตรวจจับเพียร์ และปริมาณข้อมูลที่เพียร์มีการรับส่งก่อนถูกตรวจจับด้วยแต่ละวิธี โดยใช้การเปรียบเทียบจำนวนเปอร์เซ็นต์สะสมของเพียร์ที่ตรวจจับได้ในช่วงเวลาต่างๆ ของการทดลอง ซึ่งเทียบอัตราส่วนจากจำนวนเพียร์ที่ตรวจจับได้ทั้งหมดในแต่ละชุดข้อมูล

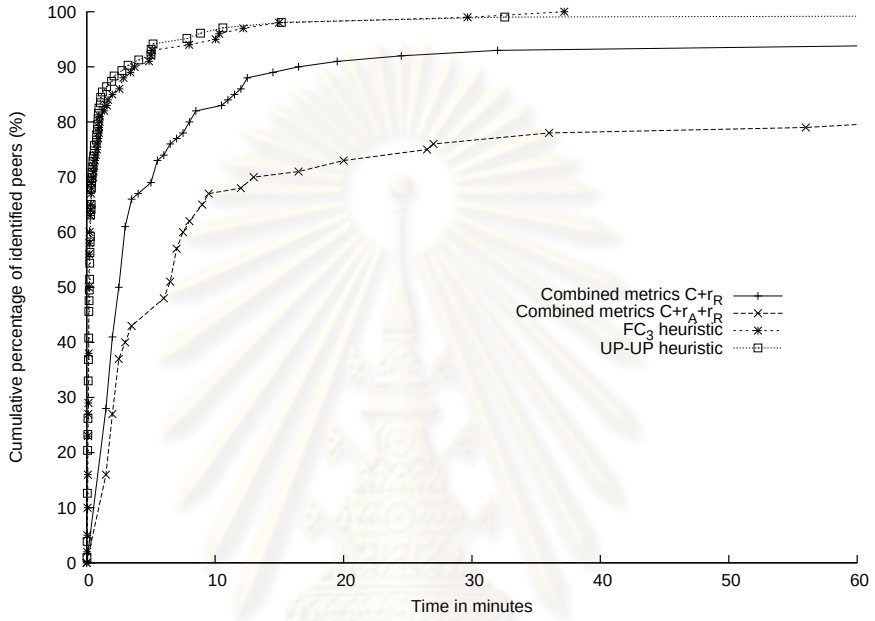
ผลการเปรียบเทียบเวลาที่ใช้ในการตรวจจับจากกราฟรูปที่ 4.14 แสดงให้เห็นว่าทั้งเกณฑ์วัดร่วม $C+r_R$ และ $C+r_A+r_R$ ใช้เวลาในการตรวจจับมากกว่าวิธี FC_3 และ $UP-UP$ โดยสำหรับวิธีทั้งสองนั้น 95% ของเพียร์ที่ตรวจจับได้ในชุดข้อมูลควบคุม ถูกตรวจจับได้ภายในเวลาไม่ถึง 1 นาที ในขณะที่เกณฑ์วัดร่วม $C+r_R$ และ $C+r_A+r_R$ ต้องใช้เวลา 8 นาที ซึ่งสอดคล้องกับเวลาที่ใช้ในการตรวจจับในชุดข้อมูลปกติดังแสดงในรูปที่ 4.15



รูปที่ 4.14 การเปรียบเทียบความเร็วในการตรวจจับในชุดข้อมูลควบคุม

สาเหตุที่เกณฑ์วัดร่วมทั้งสองใช้เวลาในการตรวจจับมากกว่าวิธี FC_3 และ $UP-UP$ เป็นเพราะตัวระบุเพียร์ที่นำเสนออาศัยพฤติกรรมของกระแสข้อมูลที่เกิดจากขั้นตอนวิธีการเค้น ซึ่งเกิดขึ้นเมื่อมีการรับส่งข้อมูลเท่านั้น จึงไม่สามารถตรวจจับได้ตั้งแต่ในช่วงเริ่มต้นซึ่งมีการเปิดการเชื่อมต่อไปยังเพียร์ต่างๆ อีกทั้งการทำงานของตัวระบุเพียร์ที่นำเสนอยังมีการคำนวณเป็นรอบทุก 30 วินาที รวมถึงการลดค่าผลบวกลวงด้วยการกำหนดให้หมายเลขไอพีที่ตรวจจับได้ต้องผ่านทุก

เกณฑ์วัดของเกณฑ์วัดร่วมติดต่อกัน 3 ครั้ง จึงทำให้ต้องใช้เวลาอย่างน้อย 90 วินาทีในการตรวจจับ ตรงกันข้ามกับวิธี FC_3 และ $UP-UP$ ที่อาศัยพฤติกรรมของการเปิดการเชื่อมต่อและรูปแบบการเชื่อมต่อกันของโฟลจากหมายเลขพอร์ต จึงสามารถตรวจจับได้ตั้งแต่ในช่วงเริ่มต้น เพราะเป็นช่วงที่เพียร์เริ่มทำการติดต่อไปยังเพียร์อื่นตามที่ได้รับมาจากแทรกเกอร์

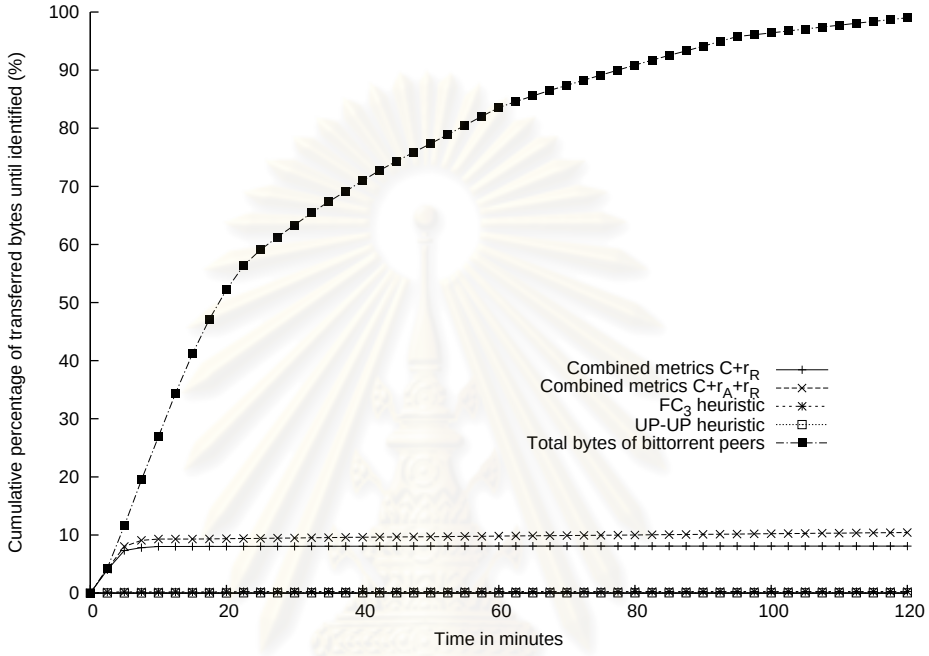


รูปที่ 4.15 การเปรียบเทียบความเร็วในการตรวจจับในชุดข้อมูลปกติ

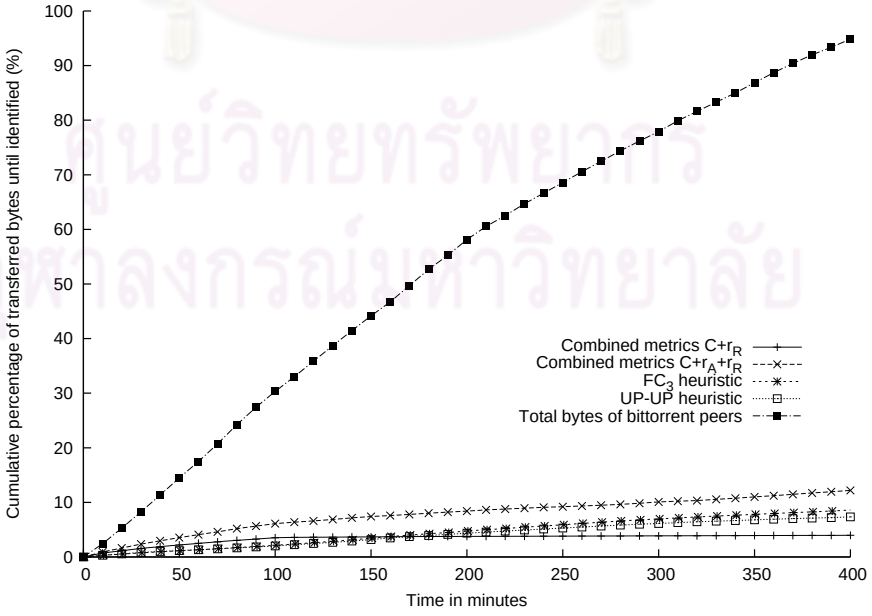
จากความเร็วในการตรวจจับดังกล่าว ส่งผลให้ปริมาณข้อมูลที่เพียร์มีการรับส่งก่อนถูกตรวจจับด้วยเกณฑ์วัดร่วม $C+r_R$ และ $C+r_A+r_R$ มีมากกว่าวิธี FC_3 และ $UP-UP$ ในชุดข้อมูลควบคุม ดังแสดงในกราฟรูปที่ 4.16 เนื่องจากวิธีทั้งสองสามารถตรวจจับเพียร์ได้ทั้งหมดตั้งแต่ในช่วงเริ่มต้นของการดาวน์โหลด

อย่างไรก็ตาม จากกราฟเปรียบเทียบความถูกต้องในการตรวจจับเพียร์ในชุดข้อมูลปกติดังรูปที่ 4.13 พบว่าทั้งวิธี FC_3 และ $UP-UP$ มีปริมาณข้อมูลที่เกิดจากเพียร์ที่ตรวจจับได้น้อยกว่าเกณฑ์วัดร่วม $C+r_R$ เนื่องจากไม่สามารถตรวจจับเพียร์บางตัวที่มีการรับส่งข้อมูลจำนวนมากและมีการเชื่อมต่อที่ใช้หมายเลขพอร์ตของแอพลิชันปกติจำนวนมาก ส่งผลให้ปริมาณข้อมูลที่เพียร์ทั้งหมดมีการรับส่งก่อนถูกตรวจจับด้วยวิธี FC_3 และ $UP-UP$ มีการเพิ่มขึ้นอย่างต่อเนื่องดังแสดงในกราฟรูปที่ 4.17 โดยในช่วงแรกที่เพียร์ดังกล่าวยังไม่ถูกตรวจจับ ปริมาณข้อมูลที่เพียร์มีการรับส่งก่อนถูกตรวจจับด้วยเกณฑ์วัดร่วมทั้งสองจะมีมากกว่าวิธี FC_3 และ $UP-UP$ แต่เมื่อถึงนาทีที่ 100 ซึ่งเพียร์ดังกล่าวถูกตรวจจับด้วยเกณฑ์วัดร่วม $C+r_R$ ทำให้ปริมาณข้อมูลที่เพียร์มีการ

รับส่งก่อนถูกตรวจจับมีการเพิ่มขึ้นในอัตราที่ต่ำมาก ในขณะที่วิธี FC_3 และ $UP-UP$ ยังคงไม่สามารถตรวจจับเพียร์ดังกล่าวได้ จึงมีปริมาณข้อมูลในส่วนนี้เพิ่มขึ้นอย่างต่อเนื่อง และเมื่อผ่านนาทีที่ 170 ไปแล้ว ปริมาณข้อมูลที่เพียร์มีการรับส่งก่อนถูกตรวจจับของเกณฑ์วัดร่วม $C+r_R$ จึงเริ่มน้อยกว่าวิธี FC_3 และ $UP-UP$



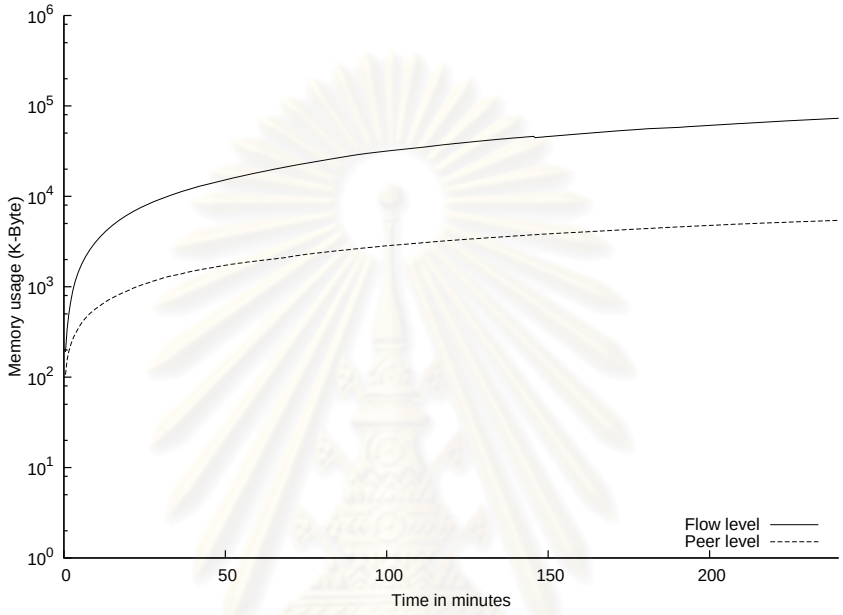
รูปที่ 4.16 การเปรียบเทียบปริมาณข้อมูลที่เพียร์มีการรับส่งก่อนถูกตรวจจับในชุดข้อมูลควบคุม



รูปที่ 4.17 การเปรียบเทียบปริมาณข้อมูลที่เพียร์มีการรับส่งก่อนถูกตรวจจับในชุดข้อมูลปกติ

4.6.3 การเปรียบเทียบการใช้ทรัพยากรของระบบ

ในหัวข้อนี้ เป็นการเปรียบเทียบขนาดของหน่วยความจำที่แต่ละวิธีใช้ในระหว่างการทำงาน โดยแกนตั้งคือขนาดของหน่วยความจำในมาตราส่วนลอการิทึม (Log scale) ที่ถูกใช้ในแต่ละช่วงเวลา ดังแสดงในรูปที่ 4.18



รูปที่ 4.18 การเปรียบเทียบขนาดของหน่วยความจำที่ใช้ในระหว่างการทำงาน

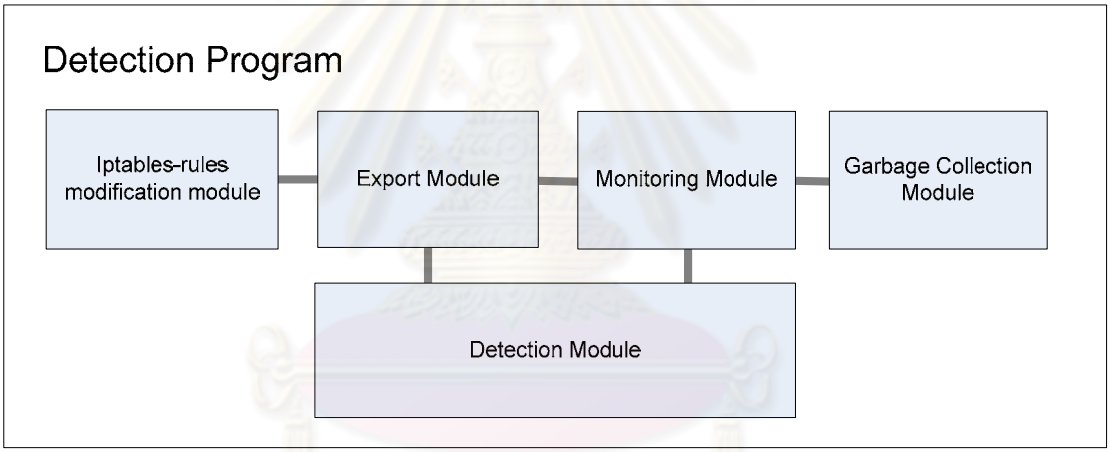
เนื่องจากตัวระบุเพียร์ที่นำเสนออาศัยข้อมูลจากระดับชั้นเครือข่าย ซึ่งเป็นการตรวจสอบในระดับเพียร์ (Peer level) ดังนั้นสถานะ (State) ที่ต้องจำในระหว่างการทำงานคือหมายเลขไอพีทั้งหมดที่แต่ละเพียร์มีการติดต่อด้วย ส่วนวิธี FC_3 และ $UP-UP$ อาศัยข้อมูลจากระดับชั้นขนส่ง ซึ่งเป็นการตรวจสอบในระดับโฟล (Flow level) สถานะที่ต้องจำในระหว่างการทำงานคือจำนวนโฟลทั้งหมดที่เกิดขึ้นของแต่ละเพียร์ ซึ่งนอกจากหมายเลขไอพีแล้ว ยังต้องจำหมายเลขพอร์ตที่ติดต่อกันด้วย อีกทั้งแต่ละคู่ของหมายเลขไอพีอาจมีการติดต่อกันมากกว่า 1 โฟลเพื่อรับส่งข้อมูลในแบบขนาน (Parallel) เช่น การติดต่อกันระหว่างเครื่องให้บริการเว็บไซต์กับเครื่องขอใช้บริการเป็นต้น จากเหตุผลดังกล่าว ทำให้การตรวจสอบในระดับเพียร์มีการใช้หน่วยความจำน้อยกว่าการตรวจสอบในระดับโฟล โดยในช่วงเวลาของการทดลองกับชุดข้อมูลที่สร้างขึ้นจากสภาพแวดล้อมควบคุม พบว่าตัวระบุเพียร์ที่นำเสนอมีการใช้หน่วยความจำน้อยกว่าวิธี FC_3 และ $UP-UP$ ประมาณ 10 เท่า

4.7 การทดลองด้วยการทำงานแบบออนไลน์

ในหัวข้อนี้เป็นการทดลองใช้ตัวระบุเพียร์ที่นำเสนอในแบบออนไลน์ (online) เพื่อทดสอบความเป็นไปได้ในการนำไปประยุกต์ใช้กับระบบเครือข่ายจริง

4.7.1 การออกแบบโปรแกรม

โปรแกรมที่ใช้ทดลองแบบออนไลน์ มีการออกแบบให้ทำงานร่วมกับ Iptables [19] เพื่อทำการสร้างกฎ (Rules) สำหรับควบคุมหมายเลขไอพีที่ตรวจจับได้ เช่น การยับยั้ง (Block) การรับส่งข้อมูล แต่เนื่องจากในการทดลองนี้มีการดักจับแพ็กเก็ตเพื่อนำมาวิเคราะห์เพิ่มเติมในภายหลังด้วย จึงทดลองสร้างกฎที่มีเป้าหมาย (Target) เป็นการยอมรับ (Accept) เท่านั้น ตัวโปรแกรมประกอบไปด้วยมอดูล (Module) ต่างๆ ดังแสดงในรูป 4.19



รูปที่ 4.19 ส่วนประกอบของโปรแกรมที่ใช้ในการทดลองแบบออนไลน์

1. Detection Module ทำหน้าที่ตรวจจับหมายเลขไอพีที่ใช้งานบิตทอร์เรนต์ในระบบเครือข่าย เมื่อตรวจจับได้จะทำการส่งต่อไปยัง Monitoring Module และ Export Module

2. Monitoring Module ทำหน้าที่ในการติดตามสถานะหมายเลขไอพีที่ถูกตรวจสอบได้จาก Detection Module และคอยตัดสินใจว่าจะยกเลิกการใช้กฎที่สร้างขึ้นสำหรับหมายเลขไอพีนี้เมื่อใด จากนั้นทำการส่งรายการของหมายเลขไอพีเหล่านี้ไปยัง Iptables-rules Modification Module ผ่านทาง Export Module นอกจากนี้ หากพบว่าหมายเลขไอพีใดที่ไม่มีการรับส่งข้อมูล

เป็นเวลานาน ซึ่งอาจเกิดจากการปิดเครื่องไปแล้ว จะทำการส่งหมายเลขไอพีเหล่านั้นต่อไปยัง Garbage Collection Module

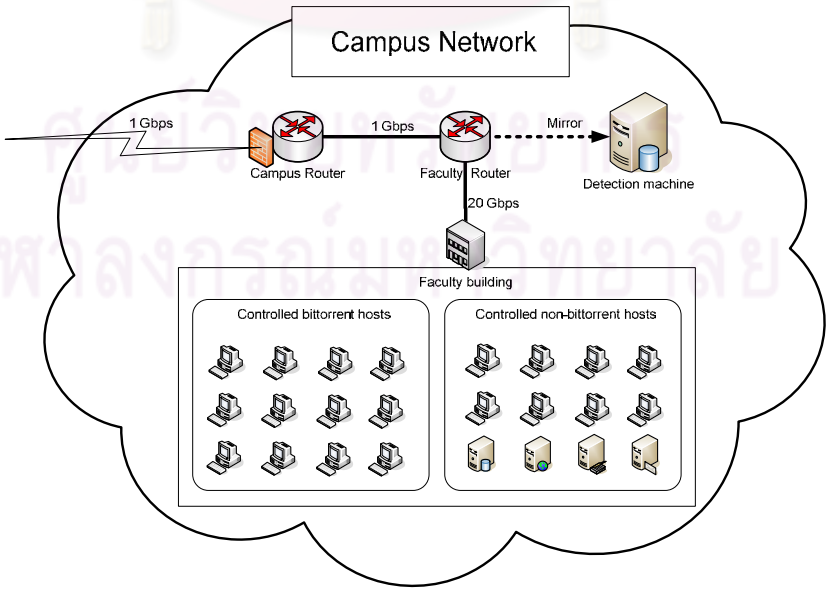
3. Garbage Collection Module ทำหน้าที่ในการลบหน่วยความจำที่ใช้ในการเก็บข้อมูลที่เกี่ยวข้องของหมายเลขไอพีที่ไม่มีการรับส่งข้อมูลเป็นเวลานาน ซึ่งได้รับมาจาก Monitoring Module อีกทีหนึ่ง เพื่อคืนหน่วยความจำที่ไม่ได้ใช้งานให้แก่ระบบ

4. Export Module ทำหน้าที่เป็นส่วนต่อประสาน (Interface) ระหว่างมอดูลภายในกับแหล่งข้อมูลภายนอก เช่น การบันทึกข้อมูลลงฐานข้อมูล เป็นต้น รวมถึงส่งข้อมูลไปยังมอดูลอื่นที่ต้องติดต่อกับแหล่งข้อมูลภายนอก ได้แก่ การส่งหมายเลขไอพีที่ต้องการสร้างกฎเพื่อควบคุมการใช้งานให้แก่ Iptables-rules Modification Module

5. Iptables-rules Modification Module ทำหน้าที่สร้างกฎสำหรับการควบคุมการใช้งานของหมายเลขไอพีที่ตรวจจับได้ ซึ่งได้รับมาจาก Export Module โดยทำหน้าที่เป็นส่วนต่อประสานระหว่างตัวโปรแกรมกับ Iptables

4.7.2 ระบบที่ใช้ในการทดลอง

การทดลองแบบออนไลน์ในหัวข้อนี้ดำเนินการบนระบบเครือข่ายของคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ดังแสดงในรูปที่ 4.20



รูปที่ 4.20 แผนผังของระบบเครือข่ายที่ใช้ในการทดลองแบบออนไลน์

เครื่องที่ใช้ทดสอบตัวระบุเพียร์ (Detection machine) เชื่อมต่อโดยตรงกับเราท์เตอร์หลักของคณะวิศวกรรมศาสตร์ โดยเครื่องคอมพิวเตอร์ดังกล่าวประกอบด้วยซีพียูเพนเทียม 4 ความเร็ว 2.66 กิกะเฮิร์ต หน่วยความจำหลัก 512 เมกะไบต์ และระบบปฏิบัติการลีนุกซ์เคอร์เนล 2.6

ในการทดลองได้แบ่งกลุ่มของหมายเลขไอพีเป็น 2 กลุ่ม กลุ่มแรกมีจำนวน 13 หมายเลข ที่เปิดใช้งานบิตทอร์เรนต์แบบมีการเข้ารหัส โดยมีการดาวน์โหลดข้อมูลจากไฟล์ทอร์เรนต์ที่แตกต่างกันทั้งหมด หมายเลขไอพีในกลุ่มนี้ใช้สำหรับประเมินผลบวกแท้และผลลบลวง ส่วนกลุ่มที่สองเป็นกลุ่มของหมายเลขไอพีที่ไม่มีการใช้งานบิตทอร์เรนต์ ประกอบไปด้วยหมายเลขไอพีของเครื่องใช้งานปกติและเครื่องให้บริการต่างๆ เช่น เครื่องให้บริการเว็บไซต์ เครื่องให้บริการอีเมล เครื่องให้บริการฐานข้อมูล และเครื่องคอมพิวเตอร์ของระบบคลัสเตอร์ เป็นต้น จำนวน 29 หมายเลข โดยหมายเลขไอพีในกลุ่มนี้ใช้สำหรับการประเมินผลลบแท้และผลบวกลวง

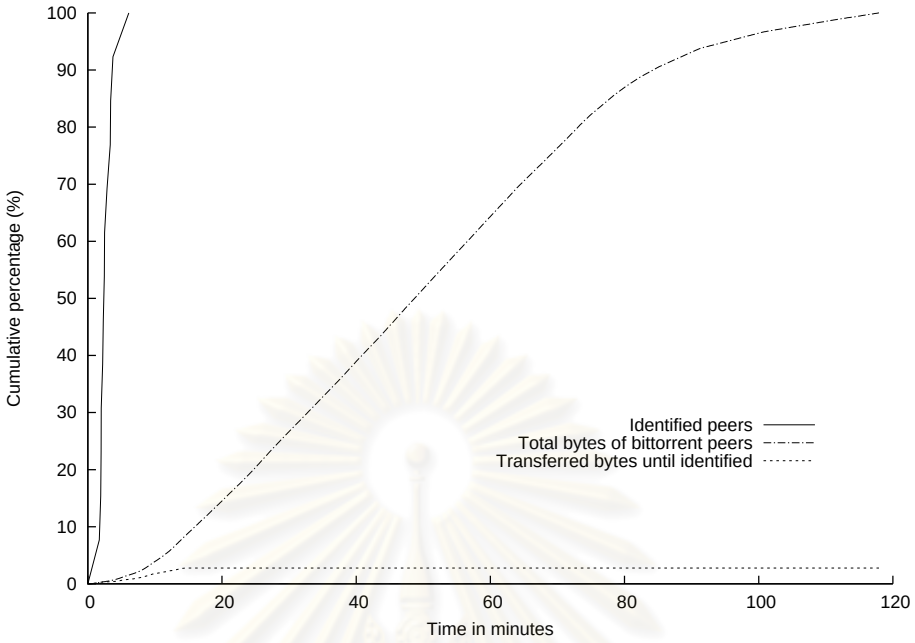
4.7.3 ผลการทดลองแบบออนไลน์

ตัวระบุเพียร์ของบิตทอร์เรนต์ในการทดลองนี้ สร้างขึ้นจากเกณฑ์วัดร่วม $C + r_R$ ซึ่งมีประสิทธิภาพมากที่สุดในการทดลองกับทั้งสองชุดข้อมูล ผลการตรวจจับเพียร์ในการทดลองแบบออนไลน์แสดงได้ดังตารางที่ 4.10 โดยโปรแกรมที่สร้างขึ้นสามารถตรวจจับเพียร์ได้ทั้งหมดเช่นเดียวกับการทดลองแบบออฟไลน์ (Offline) ในชุดข้อมูลควบคุม

ตารางที่ 4.10 ผลการทดลองแบบออนไลน์

	$C + r_R$	
	จำนวนหมายเลขไอพี	จำนวนไบต์ที่รับส่ง (GB)
ผลบวกแท้ (True positive)	**13 (100%)**	**57.89 (100%)**
ผลลบลวง (False negative)	0 (0%)	0 (0%)
ผลลบแท้ (True negative)	29 (100%)	21.55 (100%)
ผลบวกลวง (False positive)	**0 (0%)**	**0 (0%)**

จากการวิเคราะห์เพิ่มเติมจากแพ็กเก็ตที่ได้ดักจับไว้ขณะทำการทดลอง พบว่าโปรแกรมที่สร้างขึ้นสามารถตรวจจับเพียร์ทั้งหมดได้ภายในเวลา 6 นาที และมีปริมาณข้อมูลที่เพียร์ทั้งหมดมีการรับส่งก่อนถูกตรวจจับเท่ากับ 3% ของปริมาณข้อมูลที่เกิดจากเพียร์ทั้งหมดในช่วงการทดลอง ดังแสดงในรูปที่ 4.21



รูปที่ 4.21 ความเร็วในการตรวจจับเพียร์ในการทดลองแบบออนไลน์

นอกจากความถูกต้องและความเร็วในการตรวจจับแล้ว การทดลองแบบออนไลน์ในหัวข้อนี้ยังได้ประเมินการใช้งานทรัพยากรของตัวโปรแกรมที่สร้างขึ้นในด้านของการใช้งานซีพียูและหน่วยความจำโดยเฉลี่ยในแต่ละช่วงเวลาของระบบเครือข่ายคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ดังแสดงในตารางที่ 4.11

ตารางที่ 4.11 การใช้ทรัพยากรโดยเฉลี่ยขณะทำงานแบบออนไลน์

ช่วงเวลา	การใช้งานเครือข่าย (Mbps)	การใช้งานซีพียูโดยเฉลี่ย	การใช้งานหน่วยความจำโดยเฉลี่ย
08:00 – 12:00	100-150	20%	5%
12:01 - 17:00	250-300	45%	15%
17:01 – 21:00	150-200	30%	10%
21:01 – 8:00	35-45	5%	2%

<u>หมายเหตุ</u> การทดลองแบบออนไลน์ตามตารางที่ 4.11 จะไม่มีการประเมินการใช้งานฮาร์ดดิสต์ (Hard disk) เพราะในการใช้งานจริงโปรแกรมจะประมวลผลแต่ละแพ็กเก็ตที่เข้ามาแบบทันที (real-time) โดยไม่มีการสำเนาแพ็กเก็ตเก็บไว้วิเคราะห์ในภายหลัง

จากตารางที่ 4.11 พบว่าการใช้งานซีพียูและหน่วยความจำโดยเฉลี่ยนั้นมีความแตกต่างกันออกไป ขึ้นอยู่กับปริมาณการใช้งานของระบบเครือข่ายในช่วงเวลานั้น โดยในช่วงเวลา 12:00-17:00 น. เป็นช่วงเวลาที่มีการใช้งานระบบเครือข่ายมากที่สุด ส่งผลให้มีการใช้งานซีพียูและหน่วยความจำสูงตามไปด้วย โดยที่ความเร็ว 300 เมกะบิตต่อวินาที มีการใช้งานซีพียูและหน่วยความจำเฉลี่ยเท่ากับ 45% และ 15% ตามลำดับ ส่วนในช่วงเวลาที่มีการใช้งานน้อยที่สุด มีการใช้งานซีพียูและหน่วยความจำโดยเฉลี่ยเท่ากับ 5% และ 2% ตามลำดับ

จากปริมาณการใช้ทรัพยากรดังกล่าว แสดงให้เห็นว่าโปรแกรมตรวจจับเพียร์แบบออนไลน์ที่พัฒนาขึ้นมีการทำงานแบบเน้นการประมวลผลของซีพียู (CPU-Intensive) ซึ่งเมื่อพิจารณาจากผลการทดลองแล้ว พบว่าโปรแกรมที่นำเสนอสามารถทำงานได้ดีในระบบเครือข่ายขนาดกลางอย่างระบบเครือข่ายของคณะวิศวกรรมศาสตร์ เนื่องจากตัวโปรแกรมมีการใช้งานซีพียูเฉลี่ยในช่วงที่มีการใช้งานระบบเครือข่ายสูงสุดเพียงครึ่งหนึ่งเท่านั้น และหากคำนวณเพิ่มเติมจากผลการทดลองที่ได้พบว่า เครื่องที่ใช้ในการทดลองนี้สามารถนำไปใช้งานในระบบเครือข่ายที่มีการใช้งานสูงสุด 500 - 600 เมกะบิตต่อวินาทีได้

อย่างไรก็ตาม ปริมาณการใช้งานเครือข่ายที่เพิ่มขึ้นเท่าตัว (จาก 300 เมกะบิตต่อวินาที เป็น 600 เมกะบิตต่อวินาที) อาจทำให้โปรแกรมตรวจจับเพียร์มีการใช้งานซีพียูโดยเฉลี่ยเกินกว่า 2 เท่าตามที่ได้จากการคำนวณ (มากกว่า 90%) แต่ช่องว่างในส่วนนี้สามารถชดเชยได้ด้วยประสิทธิภาพของเครื่องคอมพิวเตอร์ที่สูงขึ้น เนื่องจากเครื่องที่ใช้ทดลองในหัวข้อนี้เป็นเพียงเครื่องคอมพิวเตอร์พีซีสำหรับการใช้งานทั่วไป มีราคาอยู่ในช่วง 20,000 – 25,000 บาท (ในปี พ.ศ. 2549) เท่านั้น ซึ่งไม่เหมาะที่จะนำมาใช้ทำงานเกี่ยวกับการตรวจสอบกระแสข้อมูลบนระบบเครือข่าย เพราะเครื่องดังกล่าวต้องมีการเปิดใช้งานอยู่ตลอดเวลา ดังนั้นโดยปกติแล้วเครื่องคอมพิวเตอร์ที่จะนำมาใช้งานในลักษณะนี้จึงมีประสิทธิภาพที่สูงกว่าเครื่องที่ใช้ทดลองในหัวข้อนี้ เช่น เครื่องคอมพิวเตอร์ในระดับเดียวกับเครื่องให้บริการเวิลด์ไวด์เว็บ อีกทั้งซีพียูในปัจจุบันมีแกนประมวลผลมากกว่า 1 แกน ซึ่งหากมีปรับแต่งตัวโปรแกรมนำเสนอให้สามารถใช้ประโยชน์จากจำนวนแกนประมวลผลที่เพิ่มขึ้นได้ จะช่วยให้การทำงานมีประสิทธิภาพมากขึ้นอีก ดังนั้นการนำโปรแกรมตรวจจับเพียร์แบบออนไลน์ไปใช้บนเครื่องดังกล่าว จึงมีความเป็นไปได้ในการนำไปใช้งานในระบบเครือข่ายที่มีอัตราการใช้งานเครือข่ายถึงระดับ 500-600 เมกะบิตต่อวินาทีหรือสูงกว่า

โดยสรุปแล้ว โปรแกรมตรวจจับเพียร์แบบออนไลน์ที่สร้างขึ้นจากงานวิจัยนี้เน้นการประมวลผลของซีพียูเป็นหลัก โดยขึ้นอยู่กับปริมาณการใช้งานระบบเครือข่ายในแต่ละช่วงเวลา

ดังนั้นเครื่องคอมพิวเตอร์ที่จะนำมาใช้กับโปรแกรมนี้จึงควรมีซีพียูที่มีประสิทธิภาพสูง เพื่อให้เหมาะสมกับปริมาณการใช้งานเครือข่ายสูงสุดในแต่ละองค์กร ในขณะที่หน่วยความจำไม่ควรต่ำกว่า 512 เมกะไบต์ตามที่ได้ทำการทดลองไว้ และต้องเป็นเครื่องที่มีระบบปฏิบัติการเป็นลีนุกส์เคอร์เนล 2.6 ขึ้นไป รวมถึงยังต้องเป็นเครื่องที่สามารถเปิดใช้งานได้ตลอดเวลาอีกด้วย

บทที่ 5

สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

งานวิจัยนี้ศึกษาเกี่ยวกับการระบุเพียร์ที่ใช้งานบิตทอร์เรนต์โดยอาศัยพฤติกรรมที่เกิดจากขั้นตอนวิธีการเค้น (Choke Algorithm) ซึ่งเป็นขั้นตอนวิธีหลักที่ควบคุมการแลกเปลี่ยนไฟล์ระหว่างเพียร์ของบิตทอร์เรนต์ และใช้การตรวจสอบในระดับชั้นเครือข่าย (Network Layer) ทำให้สามารถระบุหาเพียร์ได้แม้ว่าเพียร์จะมีการปรับเปลี่ยนรูปแบบการเชื่อมต่อในระดับโฟล (Flow) เป็นแบบใดก็ตาม และยังสามารถตรวจจับเพียร์ที่มีการเข้ารหัสข้อมูลได้ ไม่ละเมิดความเป็นส่วนตัว และไม่มีผลกระทบจากปัจจัยต่างๆ ที่เกี่ยวข้องในระดับขั้นขนส่ง โดยได้สร้างเกณฑ์วัด 4 ตัวจากแต่ละพฤติกรรมเพื่อใช้ตรวจจับเพียร์ ได้แก่ จำนวนหมายเลขไอพีที่ติดต่อด้วย (Connected IPs: C) อัตราส่วนของการเชื่อมต่อที่มีการรับส่งข้อมูล (Active Transfer ratio: r_A) จำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูลสองทิศทาง (Bi-directional Active Transfer: B) และอัตราส่วนความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลง (IP-relation Changes ratio: r_R)

ในส่วนของการทดลอง งานวิจัยนี้ได้ทำการเปรียบเทียบกระแสข้อมูลที่เกิดจากการเค้นของบิตทอร์เรนต์ไคลเอนต์และแอพลิเคชันทั่วไป และทำการทดลองตรวจจับเพียร์ในชุดข้อมูล 2 ชุด คือ ชุดข้อมูลที่สร้างขึ้นจากสภาพแวดล้อมควบคุม และชุดข้อมูลปกติจากระบบเครือข่าย นอกจากนี้ ยังได้ทำการวิเคราะห์อิทธิพลของค่าขีดเริ่มเปลี่ยนที่มีผลต่อความแม่นยำในการตรวจจับของเกณฑ์วัด รวมถึงเปรียบเทียบประสิทธิภาพกับงานวิจัยที่เกี่ยวข้อง และทำการทดลองในแบบออนไลน์ (online) ผลการวิจัยสามารถสรุปได้ดังนี้

1. กระแสข้อมูลที่เกิดจากการเค้นของบิตทอร์เรนต์ไคลเอนต์มีความแตกต่างกับกระแสข้อมูลที่เกิดจากการเค้นของแอพลิเคชันอื่นอย่างเห็นได้ชัด เนื่องจากขั้นตอนวิธีการเค้นออกแบบมาโดยเฉพาะสำหรับควบคุมการดาวน์โหลดและอัพโหลดระหว่างเพียร์ในบิตทอร์เรนต์ จึงมีการใช้งานในบิตทอร์เรนต์ไคลเอนต์เท่านั้น ทำให้กระแสข้อมูลที่เกิดจากแอพลิเคชันทั่วไปไม่มีลักษณะของกระแสข้อมูลที่เกิดจากการเค้น (หรือมี แต่ยังคงแตกต่างกับบิตทอร์เรนต์ไคลเอนต์อย่างมีนัยสำคัญ) โดยพฤติกรรมของกระแสข้อมูลบิตทอร์เรนต์ที่เด่นชัดที่สุดคือการเปลี่ยนคู่ถ่ายโอนข้อมูล ซึ่งทำให้ค่าของเกณฑ์วัด r_R โดยเฉลี่ยของแต่ละไคลเอนต์ใกล้เคียงกัน แม้ว่าจะมีการพัฒนาจากทีมงานต่างกัน

2. การนำเกณฑ์วัดมากกว่า 1 เกณฑ์มาสร้างเป็นตัวระบุเพียร์มีประสิทธิภาพในการตรวจจับดีกว่าการตรวจจับด้วยเกณฑ์วัดเดียว เนื่องจากแต่ละเกณฑ์วัดมีข้อดีและข้อด้อยในการตรวจจับที่แตกต่างกัน การนำเกณฑ์วัดมาใช้ร่วมกันเป็นเกณฑ์วัดร่วมสามารถนำข้อดีของเกณฑ์วัดหนึ่งมาชดเชยกับข้อด้อยของอีกเกณฑ์วัดหนึ่งได้ โดยในการทดลองกับข้อมูลทั้งสองชุดพบว่าเกณฑ์วัดร่วม $C + r_R$ และเกณฑ์วัดร่วม $C + r_R + r_A$ มีประสิทธิภาพในการตรวจจับมากที่สุด

3. ตัวระบุเพียร์ที่สร้างจากเกณฑ์วัดร่วม $C + r_R$ และเกณฑ์วัดร่วม $C + r_R + r_A$ สามารถตรวจจับเพียร์ที่มีการรับส่งข้อมูลจำนวนมากได้อย่างมีประสิทธิภาพ โดยเพียร์ที่ตรวจจับได้จากเกณฑ์วัดร่วม $C + r_R$ ในชุดข้อมูลควบคุมและชุดข้อมูลปกติมีการรับส่งข้อมูลรวมกันถึง 100% และ 98.75% ของปริมาณข้อมูลบิตทอร์เรนต์ทั้งหมดตามลำดับ ส่วนเพียร์ที่ตรวจจับได้จากเกณฑ์วัดร่วม $C + r_R + r_A$ มีการรับส่งข้อมูลรวมกันถึง 98.76% และ 90.73% ตามลำดับ ดังนั้นแม้ว่ามีเพียร์จำนวนหนึ่งที่ไม่สามารถตรวจจับได้ (ผลลบลวง) แต่ปริมาณข้อมูลที่เกิดจากเพียร์เหล่านี้มีค่อนข้างน้อย โดยเฉพาะอย่างยิ่งในชุดข้อมูลควบคุมซึ่งมีปริมาณข้อมูลของเพียร์ที่ไม่สามารถตรวจจับได้เพียง 1.25% เท่านั้น ดังนั้นการควบคุมเพียร์ที่ตรวจจับได้ จึงเท่ากับสามารถรักษาแบนด์วิดท์เกือบทั้งหมดเพื่อการใช้งานแอพลิเคชันอื่นบนระบบเครือข่าย

4. ตัวระบุเพียร์ที่นำเสนอมีจำนวนเพียร์ที่ตรวจจับผิดพลาด (ผลบวกลวง) น้อย โดยเกณฑ์วัดร่วมทั้ง 2 ไม่มีการตรวจจับผิดพลาดในชุดข้อมูลควบคุม และมีการตรวจจับผิดพลาดไม่ถึง 1% ในชุดข้อมูลปกติ แสดงให้เห็นว่าเกณฑ์วัดที่นำมาใช้สามารถแยกแยะความแตกต่างระหว่างกระแสข้อมูลบิตทอร์เรนต์ และกระแสข้อมูลของแอพลิเคชันอื่นได้เป็นอย่างดี

5. ตัวระบุเพียร์ที่นำเสนอสามารถตรวจจับเพียร์ได้อย่างรวดเร็ว โดยจากการทดลองในชุดข้อมูลควบคุมและการทดลองแบบออนไลน์ พบว่า 95% ของเพียร์ที่ตรวจจับได้ ใช้เวลาไม่เกิน 6 นาที ส่วนในชุดข้อมูลปกติพบว่า 90% ของเพียร์ที่ตรวจจับได้ ใช้เวลาไม่เกิน 10 นาที ซึ่งหากพิจารณาถึงปริมาณข้อมูลบิตทอร์เรนต์ที่เกิดขึ้นพบว่า หากมีการควบคุมเพียร์ที่ตรวจจับได้เหล่านี้จะช่วยลดปริมาณข้อมูลบิตทอร์เรนต์ลงได้กว่า 90%

6. จากการวิเคราะห์อิทธิพลของค่าขีดเริ่มเปลี่ยนที่มีผลต่อความแม่นยำในการตรวจจับพบว่า ค่าขีดเริ่มเปลี่ยนที่เหมาะสมของเกณฑ์วัด C อยู่ในช่วง 20-30 ในขณะที่ค่าขีดเริ่มเปลี่ยนที่เหมาะสมของเกณฑ์วัด r_A และ r_R อยู่ในช่วง 0.3-0.35 และ 0.55-0.8 ตามลำดับ ส่วนเกณฑ์วัด B ไม่พบช่วงของค่าขีดเริ่มเปลี่ยนที่เหมาะสม ซึ่งเป็นผลมาจากการทดลองดาวน์โหลดเพียง 1 ไฟล์ทอร์เรนต์ รวมถึงเพียร์ส่วนใหญ่มีการดาวน์โหลดข้อมูลจากซีด (Seed) เป็นส่วนใหญ่

7. ตัวระบุเพียร์ที่นำเสนอมีความเฉพาะเจาะจงกับบิตทอร์เรนต์มากกว่างานวิจัยที่นำมาเปรียบเทียบ เพราะตรวจจับจากพฤติกรรมที่เกิดจากขั้นตอนวิธีหลักของบิตทอร์เรนต์ ส่วนงานวิจัยที่นำมาเปรียบเทียบใช้การตรวจจับจากพฤติกรรมทั่วไปของเพียร์ทูเพียร์ในระดับขั้นขนส่ง จึงสามารถหลีกเลี่ยงได้ง่าย และมีสถานะที่ต้องจำขณะทำงานมากกว่าตัวระบุเพียร์ที่นำเสนอด้วย

8. จากการทดลองแบบออนไลน์พบว่า โปรแกรมตรวจจับเพียร์ที่พัฒนาขึ้นจากงานวิจัยนี้สามารถนำไปประยุกต์ใช้ในระบบเครือข่ายขนาดกลางที่มีระดับการใช้งานเครือข่ายสูงสุด 300 เมกกะบิตต่อวินาที (Mbps) ได้เป็นอย่างดี และมีความเป็นไปได้ในการนำไปใช้งานระดับที่สูงขึ้น เช่น 500-600 เมกะบิตต่อวินาทีหรือมากกว่า โดยมีองค์ประกอบสำคัญคือประสิทธิภาพของซีพียู (CPU) ของเครื่องที่ใช้งาน แต่เพื่อให้การทำงานเป็นไปอย่างราบรื่น เครื่องคอมพิวเตอร์ดังกล่าวจึงควรเป็นเครื่องคอมพิวเตอร์ในระดับเครื่องให้บริการ เพื่อให้สามารถเปิดใช้งานได้ตลอดเวลา

5.2 ข้อจำกัดและข้อเสนอแนะ

1. ตัวระบุเพียร์ที่นำเสนอไม่สามารถตรวจจับเพียร์ที่มีจำนวนเพียร์รับส่งข้อมูลด้วยน้อย เนื่องจากพฤติกรรมที่เกิดขึ้นมีลักษณะใกล้เคียงกับการใช้งานโพรโทคอลสำหรับถ่ายโอนข้อมูลทั่วไป เช่น เอฟทีพี (FTP) เป็นต้น

2. ตัวระบุเพียร์ที่นำเสนออาศัยพฤติกรรมของขั้นตอนวิธีการเค้น ซึ่งปรากฏในช่วงที่มีการรับส่งข้อมูลเท่านั้น จึงไม่สามารถตรวจจับได้ตั้งแต่ช่วงแรกที่มีการเปิดการเชื่อมต่อไปยังเพียร์ต่างๆ จึงต้องปล่อยให้เพียร์มีการรับส่งข้อมูลได้จำนวนหนึ่งก่อนจึงสามารถตรวจจับได้ ต่างจากงานวิจัยที่เกี่ยวข้องที่ใช้การพิจารณาจากรูปแบบการเชื่อมต่อของโฟล จึงตรวจจับได้เร็วกว่า

3. แนวคิดที่นำเสนอในงานวิจัยเป็นการระบุเพียร์ที่ใช้งานบิตทอร์เรนต์ ซึ่งไม่สามารถเจาะจงได้ว่าโฟล (Flow) ใดที่เป็นโฟลของบิตทอร์เรนต์บ้าง การควบคุมการใช้งานจึงทำได้เป็นรายเพียร์เท่านั้น ดังนั้นหากมีการพัฒนาตัวระบุเพียร์ที่นำเสนอเพิ่มเติม เช่น การวิเคราะห์ความสัมพันธ์ระหว่างโฟลในเพียร์ที่ตรวจจับได้ เพื่อหาว่าโฟลใดเป็นโฟลของบิตทอร์เรนต์ไคลเอนต์บ้าง จะช่วยใช้สามารถควบคุมการใช้งานได้ถึงระดับโฟล

4. โปรแกรมที่สร้างขึ้นจากงานวิจัยนี้เป็นโปรแกรมต้นแบบสำหรับทดสอบเกณฑ์วัดที่นำเสนอ จึงยังไม่มีการทำให้เหมาะที่สุด (Optimize) สำหรับการนำไปใช้งานจริง ซึ่งหากมีการออกแบบโปรแกรมให้มีประสิทธิภาพมากขึ้น จะช่วยให้สามารถนำไปใช้งานบนระบบเครือข่ายที่มีความเร็วสูงขึ้นได้

5.3 ปัญหาและอุปสรรค

การหาสถานที่เพื่อใช้ทดลองจริงโดยไม่มีการควบคุมทำได้ยาก เนื่องจากองค์กรทั่วไปไม่อนุญาตให้ใช้งานบิตทอร์เรนต์อย่างอิสระ รวมถึงจุฬาลงกรณ์มหาวิทยาลัยด้วย จึงมีการควบคุมการใช้งาน เช่น ติดตั้งเครื่องจำกัดปริมาณข้อมูลเพียร์ทูเพียร์ (Peer-to-Peer bandwidth shaper) ทำให้ไม่สามารถทดลองใช้งานบิตทอร์เรนต์ในสภาพแวดล้อมที่ไม่มีการควบคุมจากอุปกรณ์เหล่านี้ได้ เพราะอาจกระทบกระเทือนถึงเครือข่ายส่วนรวมทั้งหมด

รายการอ้างอิง

[1] Cohen, B. (2007). The BitTorrent Protocol Specification [Online]. Available from: http://www.bittorrent.org/protocol.html [20 May 2007]

[2] Sen, S., and Wang, J. (2002). Analyzing Peer-to-Peer Traffic Across Large Networks. Proceedings of ACM SIGCOMM Internet Measurement Workshop, p. 219-232.

[3] Saroiu, S., Gummadi, K.P., Dunn, R.J., Gribble, S.D., and Levy, H.M. (2002). An analysis of Internet content delivery systems. Proceedings of the 5th symposium on Operating systems design and implementation, p. 315-327.

[4] Karagiannis, T., Broido, A., Brownlee, N., Claffy, K.C., and Faloutsos, M. (2004). Is P2P dying or just hiding?. Global Telecommunications Conference.

[5] Sourceforge. (2008). Application Layer Packet Classifier for Linux [Online]. Available from: http://l7-filter.sourceforge.net [27 January 2008]

[6] Sen, S., Spatscheck , O., and Wang, D. (2004). Accurate, scalable in-network identification of p2p traffic using application signatures. Proceedings of the 13th international conference on World Wide Web, p. 512-521.

[7] Wikipedia. (2007). BitTorrent protocol encryption [Online]. Available from: http://en.wikipedia.org/wiki/BitTorrent_protocol_encryption [20 June 2007]

[8] Legout, A., Urvoy-Keller, G., and Michiardi, P. (2005). Understanding BitTorrent: An Experimental Perspective. Technical report, INRIA Sophia Antipolis / Institute Eurecom.

[9] Erman, D., Erman, D., and Popescu, A. (2005). Bittorrent session characteristics and models. Proceedings of the 3rd International Working Conference on Performance Modeling and Evaluation of Heterogeneous Networks.

[10] Milojicic, D.S., Kalogeraki, V., Lukose, R., Nagaraja1, K., Pruyne, J., Richard, B., Rollins, S., and Xu, Z. (2003). Peer-to-Peer Computing. Technical report, HP Laboratories Palo Alto.

[11] Horng, M.F., Chen, C.W., Chuang, C.S., and Lin, C.Y. (2006). Identification and Analysis of P2P Traffic- An Example of BitTorrent. Proceedings of the First

International Conference on Innovative Computing, Information and Control, p. 266-269.

[12] Karagiannis, T., Broido, A., Faloutsos, M., and Claffy, K. (2004). Transport layer identification of P2P traffic. Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, p. 121-134.

[13] Collins, M., and Reiter, M. (2006). Finding Peer-to-Peer File-Sharing Using Coarse Network Behaviors. Proceedings of the 11th European Symposium on research in computer security, p. 1-17.

[14] Karagiannis, T., Papagiannaki, K., and Faloutsos, M. (2005). BLINC: multilevel traffic classification in the dark. Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications, p. 229-240.

[15] Bartlett, G., Heidemann, J., and Papadopoulos, C. (2007). Inherent Behaviors for On-line Detection of Peer-to-Peer File Sharing. Proceedings of the IEEE Global Internet Symposium 2007, p. 55-60.

[16] IANA. (2007). PORT NUMBERS [Online]. Available from: http://www.iana.org/assignments/port-numbers/ [20 May 2007]

[17] Smith, F.D., Campos, F.H., Jeffay, K., and Ott, D. (2001). What TCP/IP protocol headers can tell us about the web. Proceedings of the ACM SIGMETRICS 2001, p. 245-256.

[18] Wikipedia. (2009). List of TCP and UDP port number [Online]. Available from: http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers [10 February 2009]

[19] Netfilter core team. (2008). iptables [Online]. Available from: http://www.netfilter.org/projects/iptables/ [28 April 2008]

ภาคผนวก

ภาคผนวก ก

ผลงานตีพิมพ์

ส่วนหนึ่งของงานวิทยานิพนธ์นี้ได้รับการตีพิมพ์เป็นบทความวิชาการในหัวเรื่อง “Bittorrent Peer Identification based on Behaviors of a Choke Algorithm” โดยวันชัย งิ้วลาย เฉลิมเอก อินทนากรวิวัฒน์ และ ยรรยง เต็งอำนวย ในงานประชุมวิชาการ “The 4th Asian Internet Engineering Conference (AINTEC 2008)” ซึ่งจัดขึ้น ณ โรงแรมพูลแมน บางกอก คิง พาวเวอร์ กรุงเทพมหานคร ประเทศไทย ระหว่างวันที่ 18-20 พฤศจิกายน 2551

และได้รับการตีพิมพ์ในวารสารเนคเทค (NECTEC Technical Journal) ในหัวเรื่อง “โปรแกรมตรวจหาและยับยั้งการใช้งานบิตทอร์เรนต์ในระบบเครือข่ายแบบเรียลไทม์โดยอาศัยพฤติกรรมของอัลกอริทึมการเค้น” โดยวันชัย งิ้วลาย ยรรยง เต็งอำนวย และเฉลิมเอก อินทนากรวิวัฒน์ ในงานประชุมวิชาการ “NECTEC Annual Conference and Exhibition 2008” ซึ่งจัดขึ้น ณ โรงแรมโซฟิเทลเซ็นทาราแกรนด์ กรุงเทพมหานคร ประเทศไทย ระหว่างวันที่ 24-25 กันยายน 2551

Bittorrent Peer Identification based on Behaviors of a Choke Algorithm

Wanchai Ngiwlay Chalermek Intanagonwiwat Yunyong Teng-amnuay

Information System Engineering Laboratory (ISEL)
Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand
g49wng@cp.eng.chula.ac.th, {Chalermek.I, Yunyong.T}@Chula.ac.th

ABSTRACT

Bittorrent is currently one of the most popular peer-to-peer (P2P) file sharing protocols. However, it incurs such excessive amount of traffic that it may adversely affect users of legacy internet applications. To limit this adverse impact, an efficient methodology for bittorrent identification may be needed. In this paper, we propose a novel approach to identify local bittorrent peers. Our approach is based on behaviors of the choke algorithm, a main algorithm used in bittorrent. An advantage of our approach is that we can identify bittorrent peers without examining the packet payload. Therefore, our approach is free from privacy issues and still effective even though the packet payload is encrypted. Unlike previous works, we identify bittorrent hosts at the peer level instead of the flow level. Given that we use only information from network layer instead of transport layer, our work maintains fewer states and achieves robustness to changes in the transport layer. Furthermore, our work is effective even in restricted environments such as networks equipped with NAT devices or firewalls without modifications to existing network equipments. Our experimental result indicates that our approach can efficiently identify most of excessive bandwidth-consuming peers (i.e., peers transfer a large amount of data in our traces) with a low false-positive rate.

Categories and Subject Descriptors

C.2.3 [Computer-Communication Networks]: Network Operations

General Terms

Algorithms, Measurement, Performance

AINTEC'08, November 18–20, 2008, Bangkok, Thailand.

Keywords

Peer-to-Peer, P2P, Bittorrent, Traffic Identification

1. INTRODUCTION

Bittorrent is currently one of the most popular P2P file sharing protocols. However, bittorrent traffic constitutes a significant share of the network bandwidth and adversely affects the users of legacy internet applications. Given that the number of bittorrent users dramatically increases, an efficient methodology to identify bittorrent peers is significantly crucial for limiting this adverse impact.

Traditional approaches identify bittorrent hosts based on default port numbers. These approaches are no longer effective because most bittorrent clients can use arbitrary listening ports. Therefore, some approaches identify bittorrent hosts based on a bittorrent-signature string in the handshaking packets [14, 8]. Even though the approaches perform reasonably well, they consume a significant amount of memory and processing power because they need to look into the packet payload. As a result, they may violate the privacy of users and may be ineffective when the packet payload is encrypted. Nowadays, the packet encryption is integrated into many bittorrent clients. Thus, it is almost impossible for the signature approaches to successfully identify the bittorrent hosts.

Recently, many research efforts focus more on traffic characteristics and behaviors of bittorrent applications. Using information from the transport layer, those recent approaches do not violate the privacy issues. In addition, the approaches can identify bittorrent hosts that encrypt their packet payload while consuming fewer resources. Flow statistics and patterns are heavily factored into the identification process. However, these approaches may not be effective in some situations because flow statistics (e.g., the average packet size, inter-arrival time between packets in a flow) may be significantly affected by behaviors of bittorrent's choke algorithm. Some approaches rely on the balancing number of incoming flows and outgoing flows. The approaches may not perform well in a restricted networks that equipped

with NAT devices or firewalls because these networks may refuse the TCP connection requesting from the outside (i.e., drop incoming TCP SYN packets). In such networks, connections can be successfully initiated only from the local peers but not from the outside. As a result, some flow characteristics may not be balanced.

In this paper, we propose a novel approach to identify local bittorrent peers. Our approach is based on behaviors of the choke algorithm, a main algorithm used in bittorrent. Unlike several previous approaches, our approach does not examine the packet payload. In addition, we simply examine the peer-level information rather than the flow-level information (i.e., network-layer information instead of transport-layer information). Therefore, our work maintains fewer states, achieves robustness to changes in the transport layer, and properly operates in restricted environments without modification to existing network equipments.

We have evaluated our approach on two traffic traces. The first trace has been collected from our controlled environment (i.e., we know exactly which hosts are bittorrent peers). The last trace has been collected from our faculty network in 2006. The experimental result indicates that our approach can efficiently identify most of the excessive bandwidth-consuming peers (i.e., peers that transfer a large amount of data in our traces). In terms of transferred bytes, we can identify bittorrent hosts with 100% true positive and no false positive for the controlled-traffic trace while we can achieve 98.75% true positive and 0.78% false positive for another trace.

The rest of this paper is organized as follows. Section 2 discusses some related works in identifying bittorrent and other P2P peers. Section 3 describes bittorrent and its behaviors. We then explain how these behaviors influence our methodology in Section 4. We present our experimental design in Section 5. The experimental result is later discussed in Section 6. Finally, we conclude this paper in Section 7.

2. RELATED WORK

Our work has been informed and influenced by a variety of research efforts as follow.

The traditional approaches that rely on default port numbers [15, 13] are no longer practically effective. Karagiannis et al. [8] have developed a payload-based technique to identify P2P traffics and have concluded that users tend to use arbitrary ports rather than default ports to avoid being detected. The idea is further developed into an online payload-based scheme [5] and a signature-based scheme [14].

However, those methodologies have some limitations, especially when the traffic is encrypted. To overcome those limitations, many research efforts rely more on the transport-layer information rather than the application-layer information [7]. Connection patterns and concurrent udp/tcp traffics are heavily factored into the identification process. Recently, Collins et al. [4] focus on identifying P2P traffic at the flow level. They classify all flows into three levels and use a level-specific technique for determining whether the flow is P2P or not.

A multilevel approach has been further developed to classify flows based on connection patterns on a particular port number [9]. Recently, Bartlett et al. [1] have proposed an on-line mechanism to detect Bittorrent and Gnutella hosts based on three inherent flow behaviors of P2P hosts: the number of failed connections, the use of unprivileged ports, and the balancing number of incoming and outgoing connections.

All of payload-based approaches examine the application signature within the packet content. Flow-based approaches identify P2P peers using port numbers, statistics, and connection patterns. Our approach differs from these works as follows.

- We do not examine the packet payload.
- Our approach is based on behaviors of bittorrent's main algorithm instead of statistical information or connection patterns.
- We simply examine the peer-level information rather than the flow-level information (i.e., network-layer information instead of transport-layer information).

3. BITTORRENT

3.1 Overview

Bittorrent [2, 10, 6] is a P2P protocol developed by Bram Cohen in 2001. It is designed to simultaneously transfer files among peers in the network. The original files are divided into small pieces at the starting peer. To prevent the last piece problem, bittorrent relies on the rarest first algorithm [3, 6, 11]. The mentioned algorithm make each peer holds a set of pieces that differ from others' as much as possible. Bittorrent has following important components.

- *Torrent file* is the .torrent extension file that contains information related with the original file (e.g., the original file size, piece size, total number of pieces, tracker url).
- *Tracker* is the centralized server that maintains the state of each peer and sends the list of available peers to the requesting peer.
- *Peers* are hosts that are interconnected with each other for downloading or uploading files. A group of peers that transfer data of the same .torrent file is called a *swarm*. Peers can be divided into two types: *seeds* and *leeches*. Seeds are peers with a complete set of pieces. They only upload pieces to

other peers in the swarm. At least one seed per swarm may be required in order to guarantee that the downloading can be completed. Leeches are peers with an incomplete set of pieces. They need to download the missing pieces from other peers. At the same time, they can upload their available pieces to other peers too.

To start using bittorrent, the torrent file must be downloaded from websites or other sources. The torrent file is then opened with a bittorrent client (e.g., BitComet, Azureus, uTorrent). The bittorrent client creates a tcp connection with a tracker whose url is specified in the torrent file. After the connection is established, the client sends its information (e.g., peer id, listening port, hash value of the torrent file) to the tracker. When the tracker receives this information and decides to accept this peer, it sends back a *peer set* to the bittorrent client. The peer set is a list of 50 available peers (default value) randomly selected from the swarm. The client then directly opens tcp connections to peers in the peer set.

After the connection is established, the local peer (i.e., user's host) exchanges a *bitfield* message with the remote peer (i.e., a peer in the peer set). The *bitfield* message contains information about pieces that the sending peer possesses. The local peer and remote peer initialize the state of each other as *choked* and *not interested*. Only peers with *unchoked* and *interested* states are allowed to download. If a peer B has some pieces that a peer A does not have, A will send an *interested* message to B (A's target peer). Once B receives this *interested* message, B assigns the state of the requesting peer A as *interested*. The target peer B decides whether to upload data to the requesting peer A or not based on the choke algorithm (Section 3.2). If B decides to upload data to A, B will assign A's state as *unchoked* and send A an *unchoked* message. Once receiving the *unchoked* message, the requesting peer A sends a *request* message to specify which pieces it wants. Consequently, the transferring process begins.

Each leech peer notifies all other peers in its peer set when it has a new piece by sending a *have* message. Each peer also sporadically updates its status and information (e.g., downloaded bytes, uploaded bytes) to the tracker. In addition, each peer attempts to maintain at least 20 peers in its peer set (default value) by asking the tracker for new peers when its peer-set size is lower than this threshold.

3.2 Choke Algorithm

The rarest-first algorithm and choke algorithm [3, 6, 10] are two main algorithms used in bittorrent. The objective of the choke algorithm is to eliminate free riders (i.e., peers that only download but do not upload to others) by balancing the uploading and downloading rate among the peers in the swarm. Free riders will be suspended from downloading or will be able to download but at a very slow speed. From the local peer's perspective, only four peers (default value) in its peer set are *unchoked* and *interested* at any given time. Therefore, only four peers are allowed to download data from a peer. The choke algorithm is described as follows.

- In every 10 seconds, all peers in the peer set are sorted in a descending order by their uploading rate to the local peer. The first four *interested* peers in the sorted list will be unchoked.
- In every three rounds (30 seconds), an *optimistic unchoked* mechanism will be activated. The operation is similar to the regular choke algorithm except that it selects only the first three interested peers to be unchoked and randomly selects one peer from the remaining peers regardless of their uploading rate. The objective of this mechanism is to give new peers a chance to download first in order to have pieces to upload later.

4. OUR METHODOLOGY

Our methodology is based on behaviors of a choke algorithm from the local peer's perspective. We observe four behaviors of bittorrent hosts and map each of them to a criterion for determining whether the hosts are using bittorrent or not. These criterions are described as follows.

4.1 Connected IPs

Our first criterion is based on the idea that bittorrent hosts are always connected to many ip addresses. In this paper, a connected ip address is an ip address of a host that transfers at least one tcp packet in both uploading and downloading direction. To join in a swarm, a peer exchanges several messages with every peer in its peer set. These messages include *choke* messages, *have* messages, and *keepalive* messages. Furthermore, each peer attempts to maintain at least 20 peers in its peer set. Consequently, each peer periodically sends several tcp packts to the same set of ip addresses.

Conversely, other internet applications establish connections to only a few of ip addresses as needed. For example, an ftp client is connected only to an ftp server. However, some applications (e.g., web applications) establish connections to several servers [16]. Specifically, the web page may contain some modules to fetch contents from several other sites (e.g., banner, rss feed). Nevertheless, web applications still differ from bittorrent because web clients do not maintain connections with servers after downloading.

4.2 Active Transfers

Given that bittorent splits the original files into small pieces, peers can simultaneously download different pieces from several different peers. This can be captured by the significant ratio of active transfers over connected IPs. Active transfers in this work are data transfers of at least five big tcp packets (i.e., packet size is around Maximum Transfer Unit or MTU). Given that bittorent is designed for transferring large files, it is likely to transfer data with the maximum packet size supported by the overlay network (e.g., 1500 bytes on Ethernet links) so that the number of packets required for transferring a file is minimized. As a result, the total overhead caused by all packet headers is also reduced.

Unlike bittorrent, other P2P file sharing protocols do not split files into small pieces. Although each peer in those protocols may be connected to many other peers, a local peer has to download the entire file from a single peer [17].

However, bittorrent peers do not always simultaneously transfer data with all peers in its peer set. This is due to the choke algorithm. At any given time, each peer uploads to at most four peers (i.e., the first four peers that give this peer the highest downloading rates). As a result, some connected peers may not transfer data packets with the local peer.

4.3 Bi-Directional Active Transfers

The selection process in the choke algorithm also leads to bi-directional active transfers because peers prefer to upload data to other peers from which they can download data.

Unlike bittorrent, other internet applications are usually clients or servers. Therefore, those applications normally transfer data in only one direction at a particular time. Furthermore, other P2P file sharing protocols do not have the reciprocation scheme that is influenced by the choke algorithm. Peers in those protocols do not need to upload their pieces to other peers from which they are downloading. As a result, there is no bi-directional active transfer in those protocols.

However, this bi-directional transfer of bittorrent occurs only in leeching peers that are connected to other leeching peers. This behavior will not happen in seeding peers or leeching peers that are not connected to any leeching peer at all.

4.4 IP-Relation Changes

In the choke algorithm, all peers in the peer set are sorted every 10 seconds in descending order of their uploading rate to the local peer. After sorting, the local peer will upload data to only the first four peers in the sorted list. Given that the uploading rate is quite dynamic, the choke algorithm may select some peers that were not selected in the previous round. Some previously selected peers may not be selected again either because some other peers may upload faster. Therefore, a pair of peers may have actively transferred data to one another but, suddenly, become inactive. The activity or relation between two peers is unavoidably dynamic. As a result, a bittorrent host can be identified by the significant ratio of IP-relation changes over active transfers.

Unlike bittorrent, other file sharing applications do not attempt to find a faster source during their downloading phases. Therefore, their activity or relation do not change because they keep downloading from certain sources until the files are complete.

4.5 Our Algorithm

We propose four metrics for capturing the above four behaviors: active transfer ratios, bi-directional active transfers, connected IPs, and IP-relation change ratios. The metrics are periodically calculated every 30 seconds and compared with their threshold in order to determine whether a host is a bittorrent peer or not. Note that each threshold can be finely tuned for the users' situation or environment so that the requirement of their network policy is satisfied.

Connected IPs (C): We count the number of peers that have communicated with the host. If the number of connected IPs is larger than or equal to the connected-IP threshold ($C_{threshold}$), the host will be identified as a bittorrent host (see Section 4.1).

$$C \geq C_{threshold}$$

Active Transfer Ratios (R_{AT}) : The active transfer ratio of a host is the ratio of active connected IPs (AT or the number of IPs that actively sending data to or receiving data from that host) over the total number of IPs connected to the host (C). If this ratio of a host is larger than or equal to the active-transfer threshold ($R_{ATthreshold}$), the host will be identified as a bittorrent host (see Section 4.2).

$$R_{AT} \geq R_{ATthreshold}$$

where

$$R_{AT} = \frac{AT}{C}$$

Bi-Direction Active Transfers ($BiAT$) : We measure the bi-directional active transfer of a host by counting the number of connected IPs that actively send data to and receive data from that host. If $BiAT$ of a host is larger than or equal to the bi-directional active-transfer threshold ($BiAT_{threshold}$), the host will be identified as a bittorent host (see Section 4.3).

$$BiAT \geq BiAT_{threshold}$$

IP-Relation Change Ratios (RC) : We measure the ip-relation changes of a host by counting the number of active connected IPs that become inactive and vice

versa. The ip-relation change ratio (R_{RC}) of a host is the ratio of the ip-relation changes (RC) over active transfers (AT). If the ratio of a host is larger than or equal to the ip-relation change threshold ($R_{RCthreshold}$), the host will be identified as a bittorent host (see Section 4.4).

$$R_{RC} \geq R_{RCthreshold}$$

where

$$R_{RC} = \frac{RC}{AT}$$

In our algorithm, only tcp packets are processed. We maintain all ip addresses in a hash table for quick retrieval later. Once a packet arrives, both source and destination ip addresses are determined whether they are already flagged as bittorrent hosts or not. Only unflagged ip addresses will be further processed. Our four metrics and their related information about source and destination ip are updated. If it is time to identify the bittorent hosts, all metrics of each ip will be compared with the corresponding thresholds. Flagged IPs will be unflagged after a timeout if there is no packet transferred within a certain period of time (20 minutes in our experiments).

5. EXPERIMENTAL DESIGN

We evaluate our approach on two different traffic traces: one from our controlled environment and another from our uncontrolled network. In the first trace, we know exactly which hosts run bittorrent applications. Conversely, we cannot be absolutely certain about which hosts are bittorent hosts in the second trace. This uncertainty complicates the accuracy evaluation of our approach. We believe that if we evaluate our approach only with the normal traffic trace (the second trace), the result may not be sufficiently conclusive.

Both traffic traces are collected from our engineering faculty network with 100 Mbps ethernet links. We collect our traces at the main router of the faculty network (see Figure 1). In these experiments, local peers are hosts in our faculty subnet.

5.1 Controlled environment traffic

In this traffic trace, there are two groups of ip addresses. The first group consists of 18 hosts that run bittorrent clients with the encryption feature enabled. Each host downloads a completely different torrent file whose size is between 700 MB and 4.3 GB. In this group, 16 hosts have successfully downloaded with a high rate (about 200 Kbps to 1 Mbps). Two remaining hosts can download at a low rate (about 1 Kbps to 30 Kbps). The last group consists of 30 non-bittorrent hosts. Some of them are mail servers, web servers, and clusters. The duration of this trace is about 140 minutes. There are

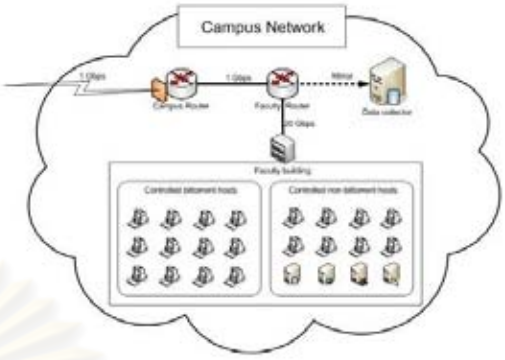


Figure 1: Network diagram of the traffic source and the collection point of our sources.

1,331 local IPs, 75,381 remote IPs, 2,360,269 flows, and 266,219,592 packets.

The objectives of the first group are for evaluating the ability to correctly identify bittorrent hosts (i.e., true positive) and the failure to identify these bittorent hosts (i.e., false negative). Conversely, the last group are for evaluating the ability to discard non-bittorrent hosts (i.e., true negative) and the failure to discard these non-bittorent hosts (i.e., missed identifcation or false positive). The result of this identifcation may be used to block or shape the traffics from these identified hosts so that the network policy is satisfied. Therefore, the false positive must be as low as possible so that non-bittorent hosts are not falsely identified and punished. Unsurpringly, our goal is to minimize the false positive rate as well as to maximize the true positive rate.

5.2 Normal traffic

This uncontrolled traffic is collected from our faculty network in 2006. The duration of this trace is about 7 hours. There are 1,519 local IPs, 1,226,424 remote IPs, 7,947,791 flows, and 610,310,682 packets. We examine the entire traffic using a signature-based scheme [14, 5] to classify hosts into two groups. The first group consists of only bittorrent hosts whose handshaking messages contain bittorrent signatures. The last group consists of likely non-bittorrent hosts whose messages do not contain bittorrent signatures. The objectives of grouping are similar to those in the controlled traffic. The first group is for evaluating the true positive and false negative rate whereas the last group is for evaluating the true negative and false positive rate.

Among the local ip addresses in this traffic trace, only 106 (6.98%) of these ip addresses are bittorrent hosts. However, in terms of bandwidth consumption, these bittorent hosts transfer up to 374.85 GB (92.43%) of the total bytes in this traffic trace.

6. EXPERIMENTAL EVALUATION

Our system are operated on a personal computer with 3 GHz Pentium 4 CPU, 1 GB of main memory, linux kernel 2.6 operating system, and libpcap [12]. The system sequentially processes each packet from the above traffic traces for performance evaluation that includes accuracy, detection speed, and sensitivity to the corresponding thresholds. The performance of our approach is compared with those of three existing flow-based heuristics: failed connections, use of unprivileged port numbers, and the balancing of incoming and outgoing connections. Even though these existing approaches rely on the transport-layer information, they also identify bittorrent peers, not bittorrent flows.

6.1 Accuracy

In our approach, there are four metrics for identifying a bittorrent host: connected IPs (C), active transfer ratios (R_{AT}), bi-directional active transfers ($BiAT$), and ip-relation change ratios (R_{RC}). Most of them perform reasonably well on our traces. However, the combinations of several metrics appear to perform significantly better than an individual one does. Due to the space limitation, we can only provide accuracy results of two promising combinations in this paper: $C + R_{RC}$ and $C + R_{AT} + R_{RC}$ (see Table 1). A host will be identified as a bittorrent host only if all metrics in the combination exceed their corresponding thresholds.

The combined metrics $C + R_{RC}$ can achieve an outstanding true positive rate of 100% for the controlled-environment trace and 83.02% for the normal-traffic trace. The results are even more impressive in terms of bytes transferred by bittorrent. These identified hosts are accounted for 100% of total bittorrent bytes transferred in the controlled-environment trace and 98.75% in the normal-traffic trace. Our results on both traces indicate that our approach can effectively identify peers that transfer a large amount of data.

Our approach cannot detect hosts that are connected to several peers but few connected peers are active and most of them stay inactive or active for a long period of time. As a result, their traffics are very similar to those of legacy internet applications (e.g., ftp, http). However, the traffics incurred by these undetected peers are so insignificant. Once the identified hosts are blocked, most of the network bandwidth should be available to other network applications.

Surprisingly, the second combined metrics $C+R_{AT}+R_{RC}$ perform worse than the first combined metrics $C+R_{RC}$ do in terms of the true positive rate. Even with one more metric, the second combination can achieve the true positive rate of 94.44% for the controlled environment trace and only 59.43% for the normal-traffic trace. However, in terms of transferred bytes, these identified hosts still constitute a significant share of the total bittorrent traffic, especially in the controlled trace (98.76% oftracetotal traffic). Nevertheless, the second combination outperforms the first combination in terms of the false positive rate. The second combination falsely identifies the hosts at only 0.21% whereas the first combination does at 0.78%. Our explanation is that the combination of three metrics is more conservative and more specific to bittorrent peers than that of two metrics is. Hence, the combination of three metrics acheives a conservative true-positive rate but rarely identifies a non-bittorrent host as a bittorrent host.

To minimize the false positive rate, our approach requires that a metric of a potential bittorent host must exceed its corresponding threshold for three consecutive rounds in order to identify the host as a bittorent host for that metric. Our result is very encouraging as we can totally eliminate the false positive rate for the controlled-environment trace as well as keep the false positive rate very low for the normal trace.

6.2 Detection Speed

We expect that, unlike previous approaches, our approach can operate on line as well as off line. Therefore, the detection speed is crucial to the viability of our online approach. To evaluate the detection speed for the controlled-traffic trace, our timer is started when the first packet is transferred to a tracker of each torrent file. However, we do not know which IPs are trackers in the normal-traffic trace. Therefore, our timer for the later trace is started when the first handshaking message is transferred to a particular host.

In addition to the detection time, we measure the total transferred bytes of all bittorrent peers before they are identified by our approach. We believe that it is necessary to detect and to block/shape their traffics before they transfer most of their data. Due to the space limitation, this paper only includes the detection speed of our most-efficient combined metrics ($C + R_{RC}$) for our two traffic traces.

Our result indicates that at least 95% of bittorent peers in the controlled-environment trace are identified within the first six minutes (Figure 2(a)). Once most of excessive bandwidth-consuming peers are detected, bytes transferred by bittorrent hosts are increased very slowly (if they do not stay the same). This indicates that traffics of undetected peers are insignificant, compared to the total bittorrent traffics. As a result, our combined metrics $C + R_{RC}$ can achieve up to 92% savings of bandwidth consumed by bittorrent hosts in this traffic trace.

Similarly, in the normal trace, at least 90% of detected peers are identified within the first 10 minutes and up to 94.75% of the bandwidth consumed by bittorrent peers can be saved (Figure 2(b)).

We believe that our bandwidth savings can be even

Table 1: Accuracy

	Summary		$C + R_{RC}$		$C + R_{AT} + R_{RC}$	
	Total IP	Total bytes (GB)	Total IP	Total bytes (GB)	Total IP	Total bytes (GB)
Controlled environment traffic	48	39.48				
Bittorrent hosts	18 (37.5%)	25.02 (63.37%)				
True positive			18 (100%)	25.02 (100%)	17 (94.44%)	24.71 (98.76%)
False negative			0 (0%)	0 (0%)	1 (5.56%)	0.31 (1.24%)
non-Bittorrent hosts	30 (62.5%)	14.46 (36.63%)				
True negative			30 (100%)	14.46 (100%)	30 (100%)	14.46 (100%)
False positive			0 (0%)	0 (0%)	0 (0%)	0 (0%)
Normal traffic	1,519	405.54				
Bittorrent hosts	106 (6.98%)	374.85 (92.43%)				
True positive			88 (83.02%)	370.16 (98.75%)	63 (59.43%)	340.11 (90.73%)
False negative			18 (16.98%)	4.69 (1.25%)	43 (40.57%)	34.74 (9.27%)
Likely non-bittorrent hosts	1,413 (92.02%)	30.69 (7.57%)				
True negative			1,402 (99.22%)	24.60 (80.16%)	1,410 (99.79%)	25.80 (84.07%)
False positive			11 (0.78%)	6.08 (19.84%)	3 (0.21%)	4.89 (15.93%)

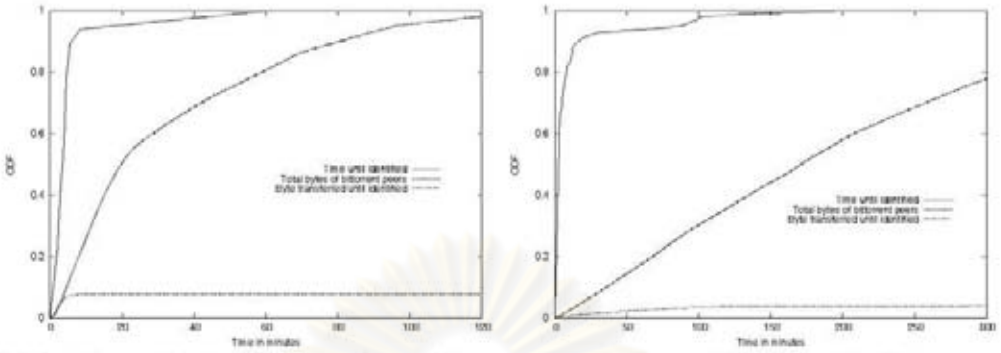
higher if the traces are longer because some detected bittorrent peers have not finished transferring data in our traces.

6.3 Sensitivity Analysis

In this section, we analyze the metric sensitivity to its corresponding threshold. For each metric, we evaluate the true positive IPs, true positive bytes, false positive IPs, false positive bytes, and bytes transferred until identified. The analysis is performed only on the controlled-environment trace. Therefore, these results may be slightly different from those on the normal trace.

Given that legacy internet applications connect to only a few of ip addresses as needed, the number of false positive ip addresses falls below 7% with the threshold of 20 or more connected IPs (Figure 3(a)). At the threshold of 20 connected IPs, we still achieve 100% true positive IPs because each bittorrent host attempts to maintain at least 20 peers in its peer set. However, the true positive rate is no longer perfect once the threshold is higher than 30. Consequently, the number of bytes transferred until identified is increased. Therefore, the optimal value of this metric is between 20 and 30. Nevertheless, even though this metric with the optimal threshold can reduce the false positive IPs, the number of bytes transferred by these IPs are still quite high. Some of these IPs are servers connected by several clients. As a result, this connected-IPs metric alone is not sufficient for identifying bittorrent hosts. It is necessary to combine this metric with others.

The active-transfer-ratio metric with a threshold of 0.35 or more outperforms the connected-IPs metric in terms of false positive bytes (Figure 3(b)). Hosts (including servers) that do not actively transfer a large amount of data with mutliple hosts will not be falsely identified using this ratio metric. However, the active-ratio metric with the threshold of 0.35 or more also significantly reduce the true positive rate. Therefore,



(a) Detection speed in the controlled-environment trace. (b) Detection speed in the normal-traffic trace.

Figure 2: Detection speed of the combined metrics $C + R_{RC}$.

this metric may also need to be combined with others.

In our faculty network, incoming connection requests toward unpriviledged ports are dropped by a firewall. Consequently, we lose some behaviors of leeching peers that initiate connections from outside. In addition, we found that most of our local peers downloaded data from seeds. This adversely effects the performance of our bi-directional-active-transfer metric. The impact is more evident when the threshold is five or more bi-directional active transfers as the true positive rate and false positive rate become zero (Figure 3(c)). Thus, this metric may not be suitable for detecting a bittorent host in this network environment. Additionally, it may not be useful to combine this metric with others either.

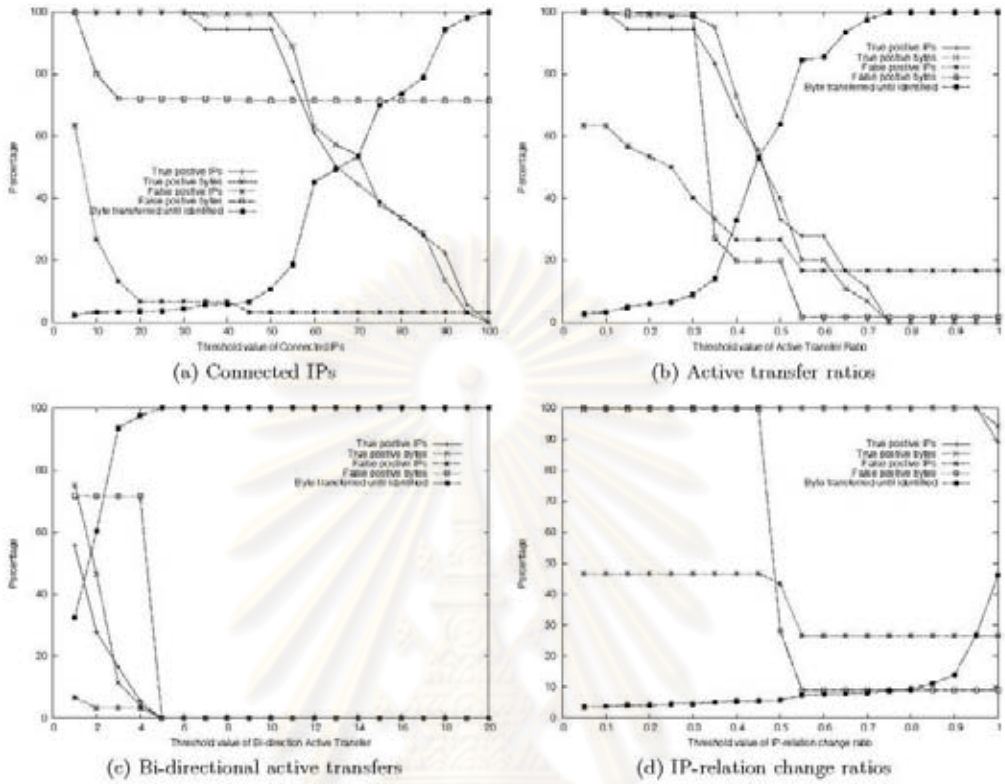
In our experiment, the IP-relation-change-ratio metric achieve the highest true positive rate among all metrics (Figure 3(d)). However, some hosts are still falsely identified as bittorent hosts. This is inherent to the property of a ratio. Given that some non-bittorrent hosts transfer data with a few of ip addresses, it is highly possible that they may stop transferring data with one of their connected IPs. This results in a change in IP relations. Even with one change, the ratio can be quite high with the small number of connected IPs. Reasonably, the IP-relation-change metric should be combined with the connected-IPs metric in order to eliminate this drawback.

6.4 Comparison with flow-based heuristics

In this section, we intend to compare our approach with three flow-based heuristics: failed connections [1, 4], unprivileged-to-unprivileged port numbers [1], and balances of incoming and outgoing connections [1]. However, the heuristic that relies on the balances of incoming and outgoing connections performs poorly in our faculty network that is equipped with firewalls. Therefore, the result of this balancing approach is not included in this paper.

Furthermore, we also need to slightly modify the definition of failed connections. In the original definition, the connection between two hosts is failed if the host cannot establish a connection with another host after sending four or more tcp SYN packets. However, this heuristic with the original definition cannot identify any bittorrent peer at all in our test. Therefore, in this paper, the failed connections are redefined to connections that cannot be established after three or more tcp SYN packets (instead of four or more).

The results on the controlled-environment trace indicate that our approach with combined metrics $C+R_{RC}$ perform as well as both flow-based heuristics do (Figure 4). They can achieve 100% of true positive IPs and bytes. On the normal-environment trace, these flow-based heuristics outperform both of our metric combinations ($C + R_{RC}$ and $C + R_{AT} + R_{RC}$) in terms of true positive rates. However, the false positive rates of our approaches are extremely low (0% in the controlled-environment trace and less than 1% in the normal-environment trace) whereas their false positive rates are quite high. Our explanation is that both flow-based approaches are designed for detecting P2P hosts in general whereas our approaches are specifically designed for detecting bittorrent hosts. Therefore, some P2P non-bittorrent hosts are detected by these flow-based approaches but not by our approaches. Examples of such hosts include hosts that run IRC applications or online multiplayer games. Furthermore, we believe that P2P applications can easily avoid the detection from the flow-based approach that relies only on unpriviledged-to-unpriviledged port numbers. The detection can be avoided by using only priviledged port numbers for outgoing connections.



(a) Connected IPs (b) Active transfer ratios

(c) Bi-directional active transfers (d) IP-relation change ratios

Figure 3: Sensitivity of each metric.

In this section, we also compare the resource usage of our approach with those of the flow-based approaches. Resource usage is directly proportional to the number of maintained states in each approach. Given that the flow-based approaches maintain and process the transport-layer information, their states are proportional to the number of flows. Unlike the flow-based approaches, our approach maintains and processes the network-layer information. Thus, the number of states in our approach is proportional to the number of connected IPs. Not surprisingly, the number of states maintained in the flow-based approaches is 10 times higher than ours (see a log-scale graph in Figure 5).

7. CONCLUSION AND FUTURE WORK

In this paper, we propose a novel approach for identifying bittorrent peers from the local-peer perspective. As far as we are concerned, this approach is the first attempt to identify bittorrent peers using only network-level information. Given that our approach relies on behaviors of the choke algorithm and processes only network-layer information, our approach still works even if the packet payload is encrypted, maintains fewer states, achieves robustness to changes at the transport layer, remains effective even in restricted networks. We also examine the sensitivity of each behavior to find the optimal threshold value of the corresponding metric. To attest the accuracy and performance, we evaluate our approach with two different traffic traces: the controlled-environment traffic and the normal traffic.

The experimental result indicates that our approach can efficiently identify excessive bandwidth-consuming peers in both traffic traces. We can achieve 100% and 98.75% true positive rate in terms of transferred bytes in the controlled-environment trace and the normal-traffic trace, respectively. In addition, our false positive rate is extremely low (i.e., less than 1%) in both traffic traces. Our detection speed is also quite fast as we can detect the bittorrent peers after they transfer only 7.8% (in the controlled-environment trace) and 4% (in the normal-traffic trace) of the total bittorrent traffic. Furthermore, our approach maintains 90% fewer states than the flow-

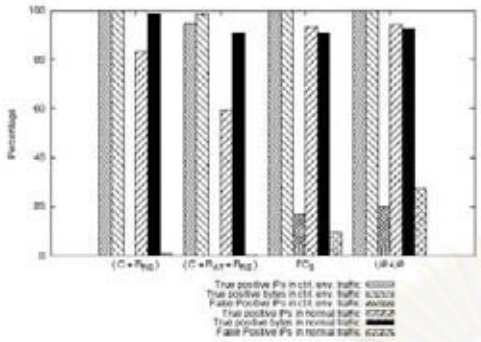


Figure 4: Accuracy comparison with flow-based heuristics.

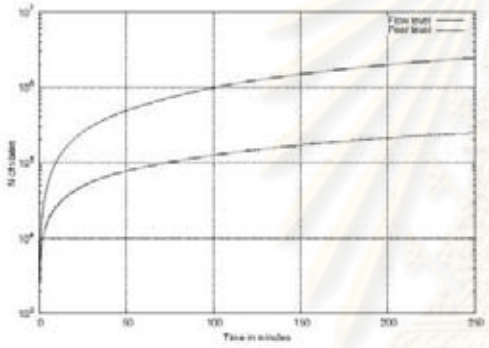


Figure 5: Resource usage comparision of peer-level and flow-level approaches.

based heuristics do. Hence, the approach is sufficiently fast and lightweight to operate on line. We plan to develope an on-line detection engine using the algorithm in this paper. We also plan to implement a real-time system to block/shape traffic from bittorrent hosts that are detected by our engine.

Like existing approaches, our approach is designed for detecting bittorrent hosts, not flows. Thus, we can only block/shape all traffic from the detected hosts. In some networks, this may be reasonable because bittorrent users can be forced to meet with network administrators before they can continue using the internet.

However, this may not be sufficiently flexible in other networks. Hence, we plan to use our algorithm to filter out the nonbittorent hosts first. Once we identify the bittorrent hosts, we can conduct some analysis only on flows from the detected hosts in order to identify the bittorrent flows later. This should consume less memory and time than conducting the analysis on all flows.

8. REFERENCES

[1] G. Bartlett, J. Heidemann, and C. Papadopoulos. Inherent Behaviors for On-line Detection of Peer-to-Peer File Sharing. In *the 10th IEEE Global Internet*, USA, May 2007.

[2] Bittorrent. http://www.bittorrent.org/protocol.html.

[3] B. Cohen. Incentives build robustness in bittorrent. In *workshop on P2PECON*, 2003.

[4] M. Collins and M.Reiter. Finding Peer-To-Peer File-sharing Using Coarse Network Behaviors. In *ESORICS 2006*, 2006.

[5] M.F. Horng, C.W. Chen, C.S. Chuang, and C.Y. Lin. Identification and Analysis of P2P Traffic-An Example of BitTorrent. In *the First ICICIC '06hp*, 2006.

[6] M. Izal, G. Urvoy-Keller, E.W. Biersack, P.A. Felber, A. Al Hamra, and L. Garc'es-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In *PAM'04*, France, April 2004.

[7] T. Karagiannis, M. Faloutsos A. Broido, and K. claffy. Transport Layer Identification of P2P Traffic. In *IMC 2004*, Italy, 2004.

[8] T. Karagiannis, A. Broido, and N. Brownlee. Is P2P Dying or Just Hiding? In *GLOBECOM '04*, Dallas, USA, 2004.

[9] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. In *ACM SIGCOMM 2005*, Pennsylvania, USA, 2005.

[10] A. Legout and G. Urvoy-Kelle. Rarest First and Choke Algorithms Are Enough. In *IMC 2006*, Rio de Janeriro, Brazil, 2006.

[11] A. Legout, G. Urvoy-Keller, and P. Michiardi. Understanding BitTorrent: An experimental perspective. Technical report, INRIA Sophia Antipolis / Institut Eurecom, November 2005.

[12] libpcap. http://sourceforge.net/projects/libpcap.

[13] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An Analysis of Internet Content Delivery Systems. In *OSDI 2002*, 2002.

[14] S. Sen, O. Spatscheck, and D. Wang. Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures. In *WWW*, 2004.

[15] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. In *IMC workshop 2002*, Marseille, France, 2002.

[16] F. Donelson Smith, F.H. Campos, K. Jeffay, and D. Ott. What TCP/IP Protocol Headers Can Tell Us About the Web. In *Proceedings of the ACM SIGMETRICS*, 2001.

[17] The Gnutella Protocol Specification. http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf.

โปรแกรมตรวจหาและยับยั้งการใช้งานบิตทอร์เรนต์ในระบบเครือข่ายแบบเรียลไทม์โดยอาศัยพฤติกรรมของอัลกอริทึมการเค้น

Real-time Bittorrent Detecting/Blocking Software Based on Behaviors of a Choke Algorithm

วันชัย จิ๋วลาย ยรรยง เต็งอำนวย และ เฉลิมเอก อินทนากรวิวัฒน์
ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ถนนพญาไท แขวงวังใหม่ เขตปทุมวัน กรุงเทพมหานคร 10330
g49wng@cp.eng.chula.ac.th , {Yunyong.T,Chalermek.I}@Chula.ac.th

ABSTRACT – **Bittorrent currently constitutes a significant share of network bandwidth in many networks. This may adversely affect the use of legacy internet applications. Most bittorrent-detection software attempts to examine the packet payload. Payload examination results in several limitations, such as consuming a great deal of processing power and memory, violating the user privacy, and likely failing to process encrypted packets. This paper presents a real-time bittorrent detecting and blocking program based on the behaviors of a choke algorithm, the main algorithm of bittorrent. Processing only network-layer information, our program does not suffer from the above limitations of the payload-based approach. Additionally, our approach maintains fewer states and achieves robustness to changes in the transport layer.**
KEY WORDS – Peer-to-Peer , P2P , Bittorrent, Traffic identification

บทคัดย่อ – ปัจจุบันกระแสข้อมูลที่เกิดจากแอพลิเคชันของบิตทอร์เรนต์ มีอัตราส่วนค่อนข้างมากเมื่อเทียบกับกระแสข้อมูลของแอพลิเคชันทั่วไป ซึ่งก่อให้เกิดผลกระทบกับการใช้งานบนระบบเครือข่ายในหลายองค์กร อีกทั้งโปรแกรมที่ใช้ตรวจหาการใช้งานบิตทอร์เรนต์ในปัจจุบันใช้การตรวจสอบจากเนื้อหาในแพ็กเก็ต ซึ่งใช้ทรัพยากรค่อนข้างมาก ละเมิดความเป็นส่วนตัว และไม่สามารถตรวจหาจากแพ็กเก็ตของบิตทอร์เรนต์ที่มีการเข้ารหัสได้ งานวิจัยนี้จึงนำเสนอโปรแกรมตรวจหาและยับยั้งการใช้งานบิตทอร์เรนต์ในระบบเครือข่าย โดยอาศัยพฤติกรรมของกระแสข้อมูลบิตทอร์เรนต์ที่เกิดจากอัลกอริทึมการเค้น ซึ่งเป็นอัลกอริทึมหลักของบิตทอร์เรนต์ การตรวจสอบจะใช้ข้อมูลจากระดับชั้นเครือข่าย จึงไม่มีข้อจำกัดของการตรวจสอบเนื้อหาในแพ็กเก็ต มีสถานะที่ต้องจำในขณะทำงานน้อยกว่างานวิจัยอื่นๆ ที่ตรวจสอบในระดับชั้นขนส่ง รวมถึงไม่มีผลกระทบจากปัจจัยต่างๆ ในระดับชั้นขนส่ง
คำสำคัญ – บิตทอร์เรนต์ เพียร์ทูเพียร์ การระบุกระแสข้อมูล

1. บทนำ

โพรโทคอลบิตทอร์เรนต์ (BitTorrent Protocol) [3, 6] เป็นโพรโทคอลสำหรับแลกเปลี่ยนข้อมูลระหว่างเครื่องบนระบบเครือข่ายที่กำลังได้รับความนิยมอย่างมากในปัจจุบัน ข้อมูลที่แลกเปลี่ยนกันด้วยโพรโทคอลนี้จะถูกส่งผ่านไปยังเครื่องต่างๆ ได้อย่างรวดเร็ว ซึ่งมีจำนวนเครื่องที่ร่วมทำการแลกเปลี่ยนข้อมูลเดียวกันอยู่มากเท่าไร ก็ยิ่งทำให้การแลกเปลี่ยนข้อมูลนั้นของระบบโดยรวมมีประสิทธิภาพมากขึ้นเท่านั้น

ด้วยประสิทธิภาพการทำงานของตัวโพรโทคอลบิตทอร์เรนต์ที่สามารถใช้ประโยชน์ทั้งช่องทางการสื่อสารขาเข้า (Downstream)

และช่องทางสื่อสารขาออก (Upstream) ได้อย่างเต็มประสิทธิภาพตลอดเวลา ประกอบกับจำนวนผู้ใช้งานที่เพิ่มมากขึ้นเรื่อยๆ [8] ส่งผลให้มีกระแสข้อมูล (Data Traffic) ที่เกิดจากแอพลิเคชันของบิตทอร์เรนต์อยู่ในระบบเครือข่ายค่อนข้างมาก ทำให้ส่งผลกระทบกับการใช้งานอินเทอร์เน็ตแอพลิเคชันตามปกติในองค์กร

การที่จะควบคุมการใช้งานบิตทอร์เรนต์เพื่อไม่ให้ส่งผลกระทบดังกล่าว จำเป็นต้องมีโปรแกรมที่มีความสามารถในการตรวจหาผู้ใช้งานบิตทอร์เรนต์ในระบบเครือข่ายได้ แต่โปรแกรมที่มีอยู่ในปัจจุบันนั้นใช้วิธีตรวจสอบจากเนื้อหาในแพ็กเก็ต ซึ่งมีข้อจำกัดอยู่หลายประการ เช่น ไม่สามารถตรวจสอบแพ็กเก็ตที่มีการเข้ารหัส (Encrypt) ได้

งานวิจัยนี้จึงนำเสนอโปรแกรมตรวจหาและยับยั้งการใช้งานบิตทอร์เรนต์ในระบบเครือข่าย โดยใช้พฤติกรรมของกระแสข้อมูลที่เกิดจากอัลกอริทึมการเค้น (Choke algorithm) ซึ่งเป็นอัลกอริทึมหลักของโพรโทคอลบิตทอร์เรนต์ ทำให้สามารถตรวจหาหมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์ได้แม้ว่าจะมีการเข้ารหัสส่วนของเนื้อหามาไว้

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

โพรโทคอลบิตทอร์เรนต์ (Bittorrent protocol) เป็นโพรโทคอลแบบเพียร์ทูเพียร์ที่ถูกออกแบบมาสำหรับการแลกเปลี่ยนไฟล์บนระบบเครือข่าย ไฟล์ต้นฉบับจะถูกสัดแบ่งออกเป็นชิ้นส่วน (Pieces) ที่เพียร์เริ่มต้น จากนั้นเพียร์อื่นๆ ที่ต้องการไฟล์นี้จะค่อยๆ ทำการดาวน์โหลดชิ้นส่วนของไฟล์ที่ถูกสัดแบ่งเหล่านี้ไปทีละชิ้นโดยการคำนวณจากอัลกอริทึมในการเลือกชิ้นส่วนของโพรโทคอลบิตทอร์เรนต์เรียกว่า Rarest First [2, 10] จุดประสงค์หลักเพื่อให้แต่ละเพียร์มีชิ้นส่วนของไฟล์แตกต่างกันมากที่สุด

การใช้งานโพรโทคอลบิตทอร์เรนต์ จะเริ่มจากการที่ผู้ใช้ดาวน์โหลดไฟล์ทอร์เรนต์จากเว็บไซต์ หรืออาจได้มาด้วยวิธีการอื่นๆ จากนั้นเปิดไฟล์ทอร์เรนต์นี้ด้วยโปรแกรมบิตทอร์เรนต์ไคลเอนต์ เช่น BitComet, Azureus เป็นต้น บิตทอร์เรนต์ไคลเอนต์จะทำการติดต่อไปยังแทรกเกอร์ด้วยโพรโทคอลทีซีพี (TCP) ตามยูอาร์แอล (URL) ที่ระบุไว้ในไฟล์ทอร์เรนต์ (.torrent) พร้อมทั้งส่งข้อมูลของตนเอง เช่น รหัสเพียร์ (Peer ID) หมายเลขไอพี หมายเลขพอร์ตที่เปิดรอ (Listening Port) รวมทั้งค่าแฮชของไฟล์ทอร์เรนต์ไปยังแทรกเกอร์ แทรกเกอร์จะทำการสุ่มรายชื่อเพียร์ในสวอร์ม (Swarm) จำนวน 50 เพียร์ (ค่าโดยปริยาย) ส่งมาให้บิตทอร์เรนต์ไคลเอนต์ เมื่อได้รับรายชื่อเพียร์จากแทรกเกอร์แล้ว บิตทอร์เรนต์ไคลเอนต์จะทำการเปิดการเชื่อมต่อด้วยโพรโทคอลทีซีพีไปยังเพียร์เหล่านั้น เรียกกลุ่มของเพียร์นี้ว่า เพียร์เซต (Peer Set)

การทำงานของโพรโทคอลบิตทอร์เรนต์ จะใช้ข้อความ (Message) ในการควบคุมและประสานงานการทำงานของแต่ละเพียร์ให้สอดคล้องกัน โดยในเริ่มต้นแต่ละเพียร์จะทำการพิสูจน์ตัวตนของอีกฝ่ายด้วย Handshake message จากนั้นจะกำหนดค่าสถานะเริ่มต้นให้แต่ละฝ่ายเป็น not interested และ choked เพียร์ที่มีสถานะเป็น interested และ unchoked เท่านั้นจึงจะมีสิทธิ์ดาวน์โหลดชิ้นส่วนได้

เมื่อเพียร์ต้องการดาวน์โหลดชิ้นส่วนใดๆ จากเพียร์อีกฝ่ายหนึ่ง จะทำการส่ง interested message ไปยังเพียร์นั้น ซึ่งเพียร์ปลายทางจะยอมอัพโหลดชิ้นส่วนให้หรือไม่ขึ้นอยู่กับการทำงานของอัลกอริทึมการเค้น ซึ่งมีกระบวนการคำนวณดังนี้

1. ในทุกๆ 10 วินาที เพียร์ในเพียร์เซตจะถูกเรียงลำดับตามอัตราค่าดาวน์โหลดที่เพียร์เฉพาะที่ (Local Peer) สามารถดาวน์โหลดได้ เพียร์ที่ให้อัตราค่าดาวน์โหลดสูงที่สุด 4 ลำดับแรกที่อยู่ในสถานะ interested จะถูก unchoked

2. ในทุกๆ รอบที่ 3 (หรือ 30 วินาที) ของการคำนวณอัลกอริทึม จะมีการทำ Optimistic Unchoked ซึ่งวิธีการเหมือนกับอัลกอริทึมการเค้นตามปกติ แต่ต่างกันตรงที่จะเลือกเพียง 3 ลำดับแรกที่ให้อัตราค่าดาวน์โหลดสูงที่สุดและอยู่ในสถานะ interested ส่วนอีก 1 เพียร์จะเลือกโดยการสุ่มเลือกจากเพียร์ทั้งหมดในเพียร์เซตโดยไม่ดูจากอัตราค่าดาวน์โหลด เพื่อเป็นโอกาสในการค้นหาเพียร์ที่อาจให้ค่าดาวน์โหลดที่ดีกว่า และเปิดโอกาสให้เพียร์ที่พึ่งเข้าร่วมในเพียร์เซต ซึ่งยังไม่มีชิ้นส่วนใดๆ ได้มีโอกาสเข้าร่วมในการดาวน์โหลด

ในระหว่างที่เพียร์เข้าร่วมอยู่ในสวอร์ม แต่ละเพียร์จะมีการส่งสถานะของตนเองไปยังแทรกเกอร์ เช่น ค่าดาวน์โหลด ค่าอัพโหลด เป็นต้น และจะรักษาจำนวนเพียร์ในเพียร์เซตของตนเองไม่ให้ต่ำกว่า 20 เพียร์ (ค่าโดยปริยาย) โดยถ้ามีจำนวนเพียร์ในเพียร์เซตต่ำกว่านี้ ก็จะทำการร้องขอรายชื่อเพียร์ชุดใหม่จากแทรกเกอร์ นอกจากนี้ เพียร์จะมีการส่งข้อความไปบอกเพียร์อื่นๆ ทั้งหมดในเพียร์เซตเมื่อตนเองมีชิ้นส่วนใหม่

2.2 งานวิจัยที่เกี่ยวข้อง

ที่ผ่านมามีหลายงานวิจัยที่นำเสนอวิธีการในการระบุกระแสข้อมูลเพียร์ทูเพียร์รวมถึงโพรโทคอลบิตทอร์เรนต์ โดยสามารถแบ่งได้เป็น 3 แนวทาง ได้แก่

1. การระบุกระแสข้อมูลโดยอาศัยหมายเลขพอร์ต (Port based) [13] และ [15] ได้ใช้หมายเลขพอร์ตโดยปริยาย (Default port) ของแต่ละแอพลิเคชันในการระบุกระแสข้อมูล แต่ในปัจจุบันแอพลิเคชันของเพียร์ทูเพียร์และบิตทอร์เรนต์มีการสุ่มหมายเลขพอร์ตในการทำงาน ดังนั้นงานวิจัยในแนวทางนี้จึงไม่สามารถตรวจหาผู้ใช้งานบิตทอร์เรนต์ในระบบเครือข่ายได้

2. การระบุกระแสข้อมูลด้วยการตรวจสอบจากเนื้อหาในแพ็กเก็ต (Payload based) [8, 14] ได้ทำการเปรียบเทียบหาลายเซ็นแอพลิเคชัน (Application Signature) ซึ่งเป็นลักษณะเฉพาะของแต่ละแอพลิเคชันจากแต่ละแพ็กเก็ต แต่วิธีการนี้มีข้อจำกัดอยู่หลายประการ เช่น ใช้ทรัพยากรในการประมวลผลค่อนข้างมาก และเกิดความเป็นส่วนตัว (Privacy) และไม่สามารถตรวจสอบแพ็กเก็ตที่มีการเข้ารหัสส่วนของเนื้อหาได้ [16]

3. การระบุกระแสข้อมูลด้วยการวิเคราะห์จากพฤติกรรมของกระแสข้อมูล (Traffic behaviors) โดยอาศัยข้อมูลจากส่วนหัวของระดับชั้นขนส่ง (Transport Layer) เพื่อหลีกเลี่ยงข้อจำกัดของการตรวจสอบจากเนื้อหาในแพ็กเก็ต

[4, 7, 9] ได้ใช้รูปแบบของการเชื่อมต่อกันในระดับโฟล (Flow) มาช่วยในการระบุกระแสข้อมูลเพียร์ทูเพียร์ แต่ยังไม่เหมาะสมที่จะนำมาใช้ในแบบเรียลไทม์เนื่องจากต้องใช้ข้อมูลในภาพรวม ซึ่งต้องใช้เวลานานในการรวบรวมข้อมูล ต่อมา [1] ได้พัฒนาเครื่องมือระบุกระแสข้อมูลเพียร์ทูเพียร์ในแบบเรียลไทม์ แต่ยังมีข้อจำกัดในการทำงานกับสภาพแวดล้อมที่มีอุปกรณ์ไฟร์วอลล์ (Firewall) หรือการแปลงเลขที่อยู่เครือข่าย (NAT)

อัลกอริทึมหลักของโปรแกรมนี้อาศัยพฤติกรรมของกระแสข้อมูลจึงไม่มีข้อจำกัดจากการตรวจสอบส่วนของเนื้อหาในแพ็กเก็ต นอกจากนี้ ยังทำการวิเคราะห์ในระดับชั้นเครือข่าย (Network layer) ซึ่งแตกต่างกับงานอื่นๆ ในแนวทางเดียวกันที่ทำการวิเคราะห์ในระดับชั้นขนส่ง จึงทำให้อัลกอริทึมหลักมีสถานะ (State) ที่ต้องจำน้อยกว่า รวมทั้งไม่มีผลกระทบจากปัจจัยต่างๆ ที่เกิดขึ้นในระดับชั้นขนส่งอีกด้วย

3. แนวคิดและวิธีการที่นำเสนอ

3.1 พฤติกรรมของกระแสข้อมูลและเกณฑ์วัด

กระแสข้อมูลของบิตทอร์เรนต์ซึ่งเกิดจากการทำงานของอัลกอริทึมการเค้นนั้น สามารถจำแนกได้เป็น 4 พฤติกรรม ซึ่งแต่ละพฤติกรรมสามารถนำมาสร้างเป็นเกณฑ์วัดได้ดังนี้

1. จำนวนหมายเลขไอพีที่เชื่อมต่อด้วย (Connected IP: *C*) บิตทอร์เรนต์จะรักษาจำนวนเพียร์ในเพียร์เซตไม่ให้ต่ำกว่า 20 ตลอดเวลา ดังนั้นจึงมีจำนวนของหมายเลขไอพีที่เพียร์มีการติดต่อด้วยจำนวนมาก

2. จำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูล (Active transfer: *AT*) บิตทอร์เรนต์สามารถดาวน์โหลดชิ้นส่วนจากหลายๆ เพียร์ได้ในเวลาเดียวกัน อัตราส่วนของการเชื่อมต่อที่มีการรับส่งข้อมูลต่อจำนวนหมายเลขไอพีที่เชื่อมต่อด้วยมักมีอัตราที่ค่อนข้างสูง (R_{AT}) โดยสามารถแสดงเป็นสมการได้ดังนี้

$$R_{AT} = \frac{AT}{C}$$

3. จำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูลสองทิศทาง (Bi-directional Active transfer: *BiAT*) บิตทอร์เรนต์สามารถดาวน์โหลดและอัพโหลดได้ในเวลาเดียวกัน และมีอัลกอริทึมการเค้นซึ่งทำให้แต่ละเพียร์พยายามอัพโหลดให้แก่เพียร์ที่ตนเองดาวน์โหลดได้เร็วที่สุด ทำให้บางคู่ของเพียร์มีการถ่ายโอนข้อมูลสองทิศทางในเวลาเดียวกันเกิดขึ้น

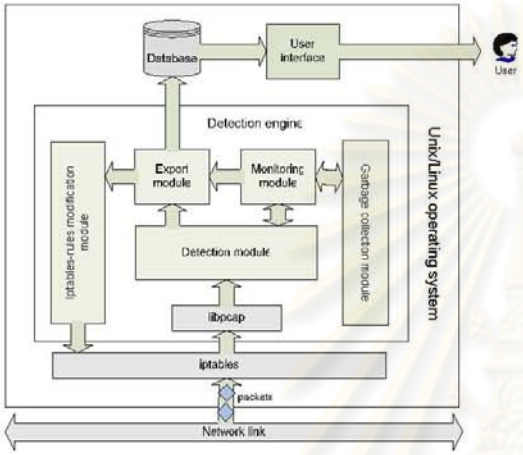
4. จำนวนของความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลง (IP-relation changes: *RC*) จากการทำงานของอัลกอริทึมการเค้นซึ่งทำให้เพียร์พยายามค้นหาเพียร์ใหม่ๆ ที่ให้ค่าดาวน์โหลดที่สูงกว่าอยู่เสมอ จึงมีการเปลี่ยนคู่ดาวน์โหลดกันอยู่ตลอดเวลา ทำให้อัตราส่วนความสัมพันธ์ของคู่หมายเลขไอพีที่เปลี่ยนแปลงต่อจำนวนของการเชื่อมต่อที่มีการรับส่งข้อมูลมักมีอัตราที่ค่อนข้างสูง (R_{RC}) โดยสามารถแสดงเป็นสมการได้ดังนี้

$$R_{RC} = \frac{RC}{AT}$$

ค่าของเกณฑ์วัดที่คำนวณได้จากแต่ละพฤติกรรม จะถูกนำไปเปรียบเทียบกับค่าขีดเริ่มเปลี่ยน (Threshold) ของเกณฑ์วัดนั้นๆ ในเวลาตรวจสอบ รายละเอียดเพิ่มเติมตลอดจนผลการตรวจจับแบบออฟไลน์ (Off-Line) ของเราสามารถอ่านได้จาก [12] ส่วนในบทความวิจัยนี้จะนำเสนอแต่โปรแกรมแบบเรียลไทม์ที่เราได้พัฒนาต่อมาและผลการตรวจจับแบบออนไลน์ (On-Line) ของเราเท่านั้น

3.2 การออกแบบโปรแกรม

โปรแกรมตรวจหาและยับยั้งการใช้งานบิตทอร์เรนต์ในระบบเครือข่ายที่นำเสนอ มีสถาปัตยกรรมของโปรแกรมดังรูปที่ 1



รูปที่ 1. แสดงสถาปัตยกรรมซอฟแวร์ของโปรแกรมตรวจหาและยับยั้งการใช้งานบิตทอร์เรนต์ในระบบเครือข่าย

โปรแกรมที่นำเสนอสามารถนำไปติดตั้งและใช้งานได้บนระบบปฏิบัติการลีนุกซ์หรือยูนิกซ์ เมื่อมีแพ็กเก็ตที่ได้รับอนุญาตให้ผ่านเข้ามาในระบบโดย iptables [11] แพ็กเก็ตจะถูกส่งต่อเข้ามาที่โปรแกรมผ่านทาง libpcap [5] เพื่อนำไปประมวลผล ตัวโปรแกรมประกอบไปด้วย 5 มอดูล ซึ่งทำหน้าที่ต่างๆ กันดังนี้

1. Detection module ทำหน้าที่ในการตรวจหาหมายเลขไอพีที่ใช้งานบิตทอร์เรนต์ในระบบเครือข่าย เมื่อตรวจพบจะทำการส่งต่อไปยัง Monitoring module และ Export module

2. Monitoring module ทำหน้าที่ในการติดตามสถานะหมายเลขไอพีที่ถูกตรวจสอบได้จาก Detection module และคอยตัดสินใจว่าจะยกเลิกการยับยั้งการใช้งานของหมายเลขไอพีนี้เมื่อใด แล้วทำการส่งรายการของหมายเลขไอพีเหล่านี้ไปยัง Export module นอกจากนี้ ยังทำการตรวจหาหมายเลขไอพีที่ไม่มีข้อมูลรับส่งเป็นเวลานาน ซึ่งอาจเกิดจากการปิดเครื่องไปแล้ว เพื่อส่งต่อให้ Garbage collection module ทำการลบหน่วยความจำอีกด้วย

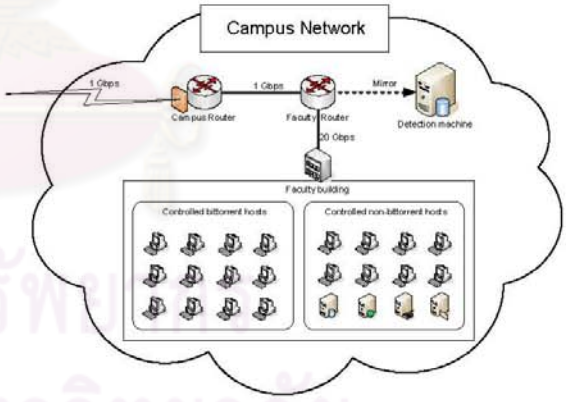
3. Garbage collection module ทำหน้าที่ในการลบหน่วยความจำของหมายเลขไอพีที่ไม่มีการรับส่งข้อมูลกันแล้ว ซึ่งได้รับมาจาก Monitoring module อีกทีหนึ่ง เพื่อเป็นการคืนหน่วยความจำที่ไม่ได้ใช้งานให้แก่ระบบ

4. Export module ทำหน้าที่เป็นส่วนต่อประสาน (Interface) ระหว่างมอดูลภายในกับแหล่งข้อมูลภายนอก ได้แก่ การบันทึกข้อมูลเข้าสู่ฐานข้อมูล MySQL รวมถึงมอดูลที่ติดต่อกับแหล่งข้อมูลภายนอก ได้แก่ การส่งหมายเลขไอพีที่ต้องการยับยั้งหรือยกเลิกการยับยั้งการใช้งานไปให้แก่ iptables-rules modification module

5. Iptables-rules modification module ทำหน้าที่สร้างกฎ (rule) สำหรับการยับยั้งหรือการยกเลิกการยับยั้งการรับส่งข้อมูลของหมายเลขไอพีที่ได้รับมาจาก Export module โดยเป็นส่วนต่อประสานระหว่างตัวโปรแกรมที่นำเสนอกับ iptables ซึ่งทำหน้าที่เป็นไฟร์วอลล์ (firewall) บนระบบปฏิบัติการลีนุกซ์และยูนิกซ์

4. ผลการทดลอง

การทดสอบประสิทธิภาพของโปรแกรมที่นำเสนอ เป็นการทดสอบโดยการใช้งานจริงแบบเรียลไทม์ (Real-time) โดยทำการทดลองในระบบเครือข่ายของคณะวิศวกรรมศาสตร์ ซึ่งมีแผนผังของระบบเครือข่ายดังรูปที่ 2



รูปที่ 2. แสดงแผนผังของระบบเครือข่ายที่ใช้ในการทดลอง

ในการทดลองประกอบไปด้วยกลุ่มของเครื่อง 13 เครื่อง ที่เปิดใช้งานบิตทอร์เรนต์แบบมีการเข้ารหัส ซึ่งหมายเลขไอพีในกลุ่มนี้ ใช้สำหรับประเมินความสามารถในการตรวจจับหมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์หรือผลบวกแท้ (True positive) และความผิดพลาดที่ไม่สามารถตรวจจับหมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์ได้หรือผลลบลวง (False negative)

หมายเลขไอพีลักกลุ่มหนึ่งเป็นกลุ่มของหมายเลขไอพีที่ไม่มีการใช้งานบิตทอร์เรนต์ ประกอบไปด้วยเครื่องพีซีของผู้ใช้ปกติ และเครื่องให้บริการต่างๆ เช่น เครื่องให้บริการเว็บไซต์ (Web server) เครื่องให้บริการอีเมล (Mail server) เครื่องให้บริการฐานข้อมูล (Database server) และเครื่องคลัสเตอร์ (Cluster) เป็นต้น จำนวน 29 เครื่อง โดยหมายเลขไอพีในกลุ่มนี้ใช้สำหรับการประเมินความสามารถในการแพ็กเฉยต่อหมายเลขไอพีที่ไม่มีการใช้งานบิตทอร์เรนต์หรือผลลบแท้ (True negative) และความผิดพลาดในการตรวจจับหมายเลขไอพี โดยระบุหมายเลขไอพีที่ไม่มีการใช้งานบิตทอร์เรนต์ว่ามีการใช้งานบิตทอร์เรนต์ หรือผลบวกเทียม (False positive)

ในงานวิจัยนี้ ได้ทำการประเมินความสามารถของโปรแกรมที่นำเสนอในด้านต่างๆ ได้แก่ ความถูกต้อง ความเร็วในการตรวจหา และทรัพยากรที่ใช้ในการทำงาน

4.1 ความถูกต้อง

ค่าขีดเริ่มเปลี่ยนของแต่ละเกณฑ์วัดในการทดลองนี้ เป็นค่าของเกณฑ์วัดที่ดีที่สุดที่เราได้จากการศึกษาความอ่อนไหว (Sensitivity) ของแต่ละพฤติกรรมแล้ว โดยรายละเอียดเราได้แสดงไว้ใน [12]

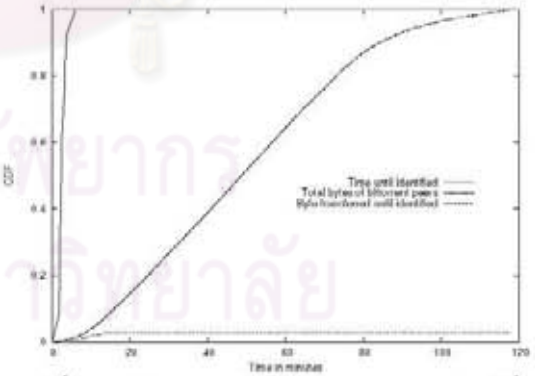
ตารางที่ 1. แสดงความถูกต้องของโปรแกรมที่นำเสนอ

	IP		Byte transferred	
	N	%	GB	%
All hosts	42		79.44	
Bittorrent hosts	13	30.95	57.89	72.87
True positive	13	100	57.89	100
False negative	0	0	0	0
Non-bittorrent hosts	29	69.05	21.55	27.13
True negative	29	100	21.55	100
False positive	0	0	0	0

ผลการทดลองจากตารางที่ 1 แสดงให้เห็นว่า โปรแกรมที่นำเสนอมีความสามารถในการตรวจหาหมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์ในระบบเครือข่ายได้อย่างมีประสิทธิภาพ โดยสามารถตรวจสอบพบหมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์ได้ทั้งหมด (True positive) ในขณะเดียวกัน โปรแกรมที่นำเสนอยังสามารถแยกแยะหมายเลขไอพีที่มีการใช้งานแอพลิเคชันอื่นๆ ที่ไม่ใช่บิตทอร์เรนต์ รวมถึงเครื่องให้บริการ ต่างๆ ได้อย่างมีประสิทธิภาพ ทำให้ไม่มีการระบุหมายเลขไอพีที่ไม่มีการใช้งานบิตทอร์เรนต์ผิด ว่ามีการใช้งานบิตทอร์เรนต์ (False positive) ซึ่งความถูกต้องในส่วนนี้มีความสำคัญมาก เพราะหากมีการระบุหมายเลขไอพีที่ไม่มีการใช้งานบิตทอร์เรนต์ว่ามีการใช้งานบิตทอร์เรนต์ จะทำให้หมายเลขไอพีเหล่านั้นถูกขับยั้งจาก iptables ไม่ให้มีการรับส่งข้อมูลได้ ซึ่งส่งผลให้หมายเลขไอพีนั้นไม่สามารถทำงานบนระบบเครือข่ายได้อีกต่อไป

4.2 ความเร็วในการตรวจจับ

โปรแกรมนี้ถูกออกแบบมาให้สามารถทำงานในแบบเรียลไทม์ได้ ดังนั้นนอกจากจะต้องตรวจสอบได้อย่างถูกต้องแล้ว จำเป็นต้องมีความสามารถในการตรวจหาหมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์ได้อย่างรวดเร็ว ในงานวิจัยนี้จึงได้มีการประเมินจำนวนของไบต์ซึ่งหมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์สามารถรับส่งกันได้ ก่อนที่จะถูกตรวจหาพบ เพราะการจะขับยั้งการใช้งานบิตทอร์เรนต์ได้อย่างมีประสิทธิภาพนั้น ต้องสามารถกระทำได้ก่อนที่หมายเลขไอพีเหล่านี้จะส่งข้อมูลออกมาจำนวนมาก ซึ่งอาจส่งผลกระทบกับการใช้งานแอพลิเคชันอื่นๆ ความเร็วในการตรวจหาหมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์ รวมถึงจำนวนไบต์ที่มีการรับส่งก่อนที่จะถูกตรวจหาพบแสดงได้ดังกราฟในรูปที่ 3



รูปที่ 3. กราฟแสดงความเร็วในการตรวจหาหมายเลขไอพีที่ใช้งานบิตทอร์เรนต์ จำนวนไบต์ทั้งหมดที่บิตทอร์เรนต์เพียร์มีการรับส่ง และจำนวนไบต์ที่บิตทอร์เรนต์เพียร์สามารถรับส่งได้ก่อนที่จะถูกตรวจหาพบและถูกขับยั้ง

ผลการทดลองแสดงให้เห็นว่า โปรแกรมที่นำเสนอมีความสามารถในการตรวจหาหมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์ได้อย่างรวดเร็ว โดยสามารถตรวจหาหมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์ทั้งหมดได้ภายในเวลา 6 นาที และเมื่อพิจารณาจากจำนวนไบต์ที่หมายเลขไอพีเหล่านี้จะสามารถรับส่งกันได้ (กราฟเส้นกลาง) และจำนวนไบต์ที่หมายเลขไอพีที่ตรวจหาได้มีการรับส่ง ก่อนที่จะถูกยับยั้งด้วยการเพิ่มกฎเข้าไปที่ iptables (กราฟเส้นล่าง) พบว่าในช่วงระยะเวลาทดลอง โปรแกรมที่นำเสนอสามารถลดแบนด์วิดธ์ของระบบเครือข่ายที่จะต้องเสียไปให้แก่หมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์เหล่านี้ลงไปได้ 97%

4.3 ทรัพยากรที่ใช้ในการทำงาน

การประเมินทรัพยากรที่ใช้ในการทำงาน จะทำการประเมินจากทรัพยากรสองส่วน คือ การใช้งานซีพียู (CPU usage) และการใช้งานหน่วยความจำ (Memory usage) โดยได้ทดสอบด้วยการเปิดใช้งานโปรแกรมที่นำเสนอบนระบบเครือข่ายจากรูปที่ 2 การทดสอบใช้เครื่องคอมพิวเตอร์ที่ประกอบด้วยซีพียู Pentium 4 2.66 GH_z หน่วยความจำหลัก 512 MB และระบบปฏิบัติการลีนุกซ์เคอร์เนล 2.6 โปรแกรมที่นำเสนอมีการใช้ทรัพยากรในการทำงานโดยเฉลี่ยแสดงได้ดังตารางที่ 2

ตารางที่ 2. แสดงทรัพยากรโดยเฉลี่ยที่ใช้ในการทำงาน

ช่วงเวลา	การใช้งานเครือข่าย (Mbps)	การใช้งานซีพียูโดยเฉลี่ย	การใช้งานหน่วยความจำโดยเฉลี่ย
08:00 – 12:00	100-150	20%	5%
12:01 - 17:00	250-300	45%	15%
17:01 – 21:00	150-200	30%	10%
21:01 – 8:00	35-45	5%	2%

ผลการทดลองจากตารางที่ 2 พบว่าการใช้งานของซีพียูและหน่วยความจำโดยเฉลี่ยนั้นมีความแตกต่างกันออกไป ขึ้นอยู่กับปริมาณการใช้งานของระบบเครือข่ายในช่วงเวลานั้น โดยในช่วงเวลา 12:00 น. ถึง 17:00 น. เป็นช่วงเวลาที่มีการใช้งานระบบเครือข่ายมากที่สุด ส่งผลให้มีการใช้งานซีพียูและหน่วยความจำสูงตามไปด้วย โดยที่ความเร็ว 300 Mbps จะมีการใช้งานซีพียูและหน่วยความจำโดยเฉลี่ยเท่ากับ 45% และ 15% ตามลำดับ ส่วนในช่วงเวลาที่มีการใช้งานน้อยที่สุด จะมีการใช้งานซีพียูและหน่วยความจำโดยเฉลี่ยเท่ากับ 5% และ 2% ตามลำดับ

5. บทสรุปและแนวทางในงานวิจัยเพิ่มเติม

งานวิจัยนี้นำเสนอโปรแกรมตรวจหาและยับยั้งการใช้งานบิตทอร์เรนต์ในระบบเครือข่ายแบบเรียลไทม์ โดยใช้พฤติกรรมของกระแสข้อมูลที่เกิดจากอัลกอริทึมการเค้น และใช้การตรวจสอบข้อมูลจากระดับชั้นเครือข่ายเท่านั้น ทำให้โปรแกรมที่นำเสนอใช้ทรัพยากรของระบบน้อยกว่างานวิจัยที่ผ่านมา ไม่มีข้อจำกัดด้านความเป็นส่วนตัว และยังสามารถตรวจสอบกระแสข้อมูลบิตทอร์เรนต์ที่มีการเข้ารหัสได้ จากผลการทดลองแสดงให้เห็นว่าโปรแกรมที่นำเสนอสามารถตรวจหาหมายเลขไอพีที่มีการใช้งานบิตทอร์เรนต์ได้อย่างมีประสิทธิภาพ ทั้งด้านความถูกต้อง ความรวดเร็วในการตรวจหา ซึ่งทำให้ลดแบนด์วิดธ์ของระบบที่จะต้องสูญเสียให้กับกระแสข้อมูลที่ไม่จำเป็นเหล่านี้ รวมถึงการใช้ทรัพยากรของระบบที่ไม่มากเกินไป ทำให้สามารถนำไปใช้งานได้จริงในแบบเรียลไทม์ได้

อย่างไรก็ตาม โปรแกรมที่นำเสนอนี้ได้ทำการยับยั้งไม่ให้หมายเลขไอพีที่ถูกตรวจสอบได้มีการรับส่งข้อมูลได้อีก แต่ในความเป็นจริงบางหน่วยงานอาจอนุญาตให้ใช้งานได้ในระดับหนึ่ง ดังนั้นแนวทางการวิจัยเพิ่มเติมจากโปรแกรมที่นำเสนอนี้ได้แก่ การเพิ่มความสามารถในการบีบกระแสข้อมูล (Traffic shaping) แทนที่จะเป็นการยับยั้งเพียงอย่างเดียว เพื่อเปิดโอกาสให้มีการใช้งานบิตทอร์เรนต์ให้เป็นไปตามนโยบายขององค์กร

เอกสารอ้างอิง

[1] G. Bartlett, J. Heidemann, and C. Papadopoulos. Inherent Behaviors for On-line Detection of Peer-to-Peer File Sharing. In the 10th IEEE Global Internet, USA, 2007.

[2] B. Cohen. Incentives build robustness in bittorrent. In workshop on P2PECON, 2003.

[3] B. Cohen, "The BitTorrent Protocol Specification", Available from: http://www.bittorrent.org/protocol.html. Access date: May 20, 2007.

[4] M. Collins and M. Reiter. Finding Peer-To-Peer File-sharing Using Coarse Network Behaviors. In ESORICS 2006, 2006.

[5] B. Fenner, G. Harris, and M. Richardson, "libpcap", Available from: http://sourceforge.net/projects/libpcap. Access date: March 15, 2008.

[6] M. Izal, G. Urvoy-Keller, E.W. Biersack, P.A. Felber, A. Al Hamra, and L. Garc'es-Erice. Dissecting BitTorrent: Five Months in a Torrent's Lifetime. In PAM'04, 2004.

[7] T. Karagiannis, M. Faloutsos A. Broido, and K. claffy. Transport Layer Identification of P2P Traffic. In IMC 2004, Italy, 2004.

[8] T. Karagiannis, A. Broido, and N. Brownlee. Is P2P Dying or Just Hiding? In GLOBECOM '04, Dallas, USA, 2004.

[9] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. In ACM SIGCOMM 2005, Pennsylvania, USA, 2005.

[10] A. Legout and G. Urvoy-Kelle. Rarest First and Choke Algorithms Are Enough. In IMC 2006, Brazil, 2006.

[11] Netfilter core team, "The netfilter.org 'iptables' project", Available from: http://www.netfilter.org/projects/iptables/. Access date: April 28, 2008.

[12] W. Ngiwlay, C. Intanagonwiwat, and Y. Teng-amnuay. Bittorrent Traffic Identification based on Behaviors of a Choke Algorithm. In AINTEC 2008. (under review)

[13] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. An Analysis of Internet Content Delivery Systems. In OSDI 2002, 2002.

[14] S. Sen, O. Spatscheck, and D. Wang. Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures. In WWW, 2004.

[15] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. In IMC workshop 2002, France, 2002.

[16] Wikipedia, "BitTorrent protocol encryption", Available from: http://en.wikipedia.org/wiki/bittorrent_protocol_encryption. Access date: June 20, 2007.

ประวัติผู้เขียนวิทยานิพนธ์

นายวันชัย งิ้วลาย เกิดเมื่อวันที่ 23 สิงหาคม พ.ศ.2525 ที่กรุงเทพมหานคร สำเร็จการศึกษาระดับมัธยมศึกษาตอนต้นจากโรงเรียนแจงร้อนวิทยา กรุงเทพมหานคร ได้รับคัดเลือกเป็นนักเรียนทุนโครงการส่งเสริมการผลิตครูที่มีความสามารถพิเศษทางด้านวิทยาศาสตร์และคณิตศาสตร์ (สควค.) เพื่อศึกษาต่อในระดับมัธยมศึกษาตอนปลายที่โรงเรียนสามเสนวิทยาลัย กรุงเทพมหานคร จนสำเร็จการศึกษาระดับปริญญาวิทยาศาสตรบัณฑิต (วท.บ.) เกียรตินิยมอันดับสองจากคณะวิทยาศาสตร์และเทคโนโลยี สาขาวิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยราชภัฏพระนครศรีอยุธยา ในปีการศึกษา 2547 และสำเร็จการศึกษาประกาศนียบัตรสาขาวิชาการสอนวิทยาศาสตร์ จากมหาวิทยาลัยมหิดล ในปีการศึกษา 2548 จากนั้นได้เข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต (วท.ม.) สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2549