# บทที่ 2

# ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

บทนี้กล่าวถึงความรู้พื้นฐานเกี่ยวกับกริดและเฟรมเวิร์คโกลบัสทูลคิทที่ใช้ในการพัฒนาระบบกริด ความรู้พื้นฐานเกี่ยวกับเครือข่ายเพียร์ทูเพียร์และเฟรมเวิร์คจักซ์ตาสำหรับพัฒนาโปรแกรมประยุกต์เพื่อทำงานบนเครือข่ายเพียร์ทูเพียร์ และความรู้พื้นฐานเกี่ยวกับจีไอเอสพีที่ใช้เป็นกลไกการทำดัชนีแบบกระจายในระบบร่วมใช้แฟ้มข้อมูลบนเครือข่ายเพียร์ทูเพียร์ที่ใช้ในงานวิจัยนี้

## 2.1 กริดและโกลบัสทูลคิท

### 2.1.1 กริด

กริดคือระบบเครือข่ายแบบกระจายขนาดใหญ่ที่มีความสามารถกำหนดองค์กรเสมือน (Virtual organization: VO) [18, 19] อันเกิดจากการตกลงกันระหว่างกลุ่มขององค์กรที่จะรวบรวมเอาทรัพยากรต่างๆเข้าด้วยกัน เพื่อบรรลุวัตถุประสงค์หนึ่งๆ โดยไม่ขึ้นกับโครงสร้างการเชื่อมต่อจริงของแต่ละองค์กร โดยยังคงสอดคล้องกับนโยบายบริหารทรัพยากรของแต่ละองค์กรที่แตกต่างกันไป กริดช่วยให้ผู้ใช้สามารถรวบรวมทรัพยากรเพื่อใช้สำหรับการประมวลผลบนกริด (Grid computing) ที่ให้คุณภาพของบริการได้ (Quality of Service: QoS) [20] ทรัพยากรอาจเป็นกลุ่มเครื่องคอมพิวเตอร์ เครื่องมืออุปกรณ์เฉพาะทาง ข้อมูล เครือข่ายและอื่นๆ ที่สามารถเชื่อมต่อเข้าสู่ระบบได้

การประมวลผลบนกริดนั้นมีแนวความคิดต้นแบบมาจากพลังงานไฟฟ้ากริด (Power grid) เราสามารถใช้พลังงานไฟฟ้ากริดโดยเสียบปลั๊กเครื่องใช้ไฟฟ้าเข้ากับเต้ารับที่มีอยู่ โดยไม่จำเป็นต้องรู้ถึงแหล่งต้นกำเนิดพลังงานที่แท้จริงภายในโครงสร้างกริด ความมุ่งหวังในกริดประมวลผลก็เป็นเช่นเดียวกัน

กริดแบ่งออกตามความต้องการใช้งานได้หลายประเภท ซึ่งขอบเขตนั้นไม่แน่ชัดและบ่อยครั้งที่เกิดจากการรวมหลายประเภทไว้ด้วยกัน ประเภทหลักของกริดมีดังนี้ [21]

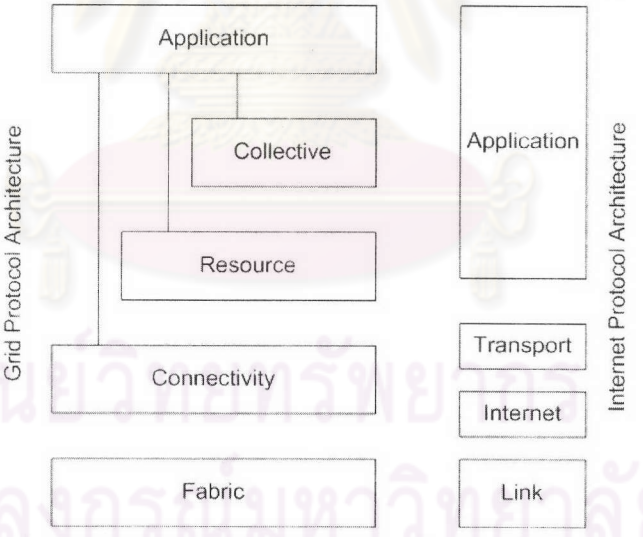
- กริดประมวลผล (Computational grid) มุ่งเน้นการจัดการทรัพยากรสำหรับการประมวลผล อุปกรณ์ส่วนใหญ่เป็นเครื่องคอมพิวเตอร์เซิร์ฟเวอร์ที่มีประสิทธิภาพสูง

- กริดรวบรวมทรัพยากร (Scavenging grid) มุ่งเน้นการรวบรวมทรัพยากรที่ใช้งานไม่เต็มประสิทธิภาพมาใช้งาน อุปกรณ์ส่วนใหญ่เป็นเครื่องคอมพิวเตอร์ส่วนบุคคลจากเจ้าของที่หลากหลาย

- กริดข้อมูล (Data grid) มุ่งเน้นการจัดเก็บและเข้าถึงข้อมูลระหว่างองค์กรเสมือน โดยผู้ใช้ไม่ต้องคำนึงถึงตำแหน่งจริงของข้อมูลที่ต้องการเข้าถึง

### 2.1.2 สถาปัตยกรรมกริด

กริดมีสถาปัตยกรรมแบบชั้น (Layered architecture) [21] ที่มีรูปร่างแบบนาฬิกาทราย แบ่งออกเป็น 5 ชั้น โครงสร้างรูปแบบนาฬิกาทรายช่วยให้เราสามารถกำหนดบริการแบบครอบคลุม (Global) ใหม่ที่อยู่เหนือคอคอด โดยไม่จำเป็นต้องเปลี่ยนแปลงบริการแบบเฉพาะที่ (Local) ที่อยู่เบื้องล่างคอคอดและมีความซับซ้อนน้อยกว่า ส่วนต่อประสาน (Interface) ที่นิยามไว้อย่างดีและง่าย ณ. บริเวณคอคอดของนาฬิกาทรายช่วยให้การเข้าใช้บริการเป็นไปอย่างมีรูปแบบและมีมาตรฐาน โดยมาตรฐานดังกล่าวจะถูกจำกัดให้แคบที่สุดเพื่อให้แน่ใจว่าส่วนต่างๆที่ถูกพัฒนาขึ้นภายในกริดจะสามารถติดต่อกันผ่านทางมาตรฐานกลางโดยไม่เกิดปัญหาอันเนื่องจากการใช้มาตรฐานที่แตกต่างกันในการติดต่อ ดังรูปที่ 2.1




รูปที่ 2.1 สถาปัตยกรรมแบบชั้นของกริด

1. ชั้นแฟบริค (Fabric layer) เป็นชั้นที่ทำหน้าที่เสมือนตัวเชื่อมต่อระหว่างระบบกริดกับทรัพยากรภายในระบบกริด เพื่อให้ระบบกริดสามารถติดต่อหรือใช้แต่ละทรัพยากรย่อยทำงานได้อย่างถูกต้องตามที่ต้องการ ตัวอย่างของชั้นนี้ได้แก่ส่วนต่อเชื่อมกับทรัพยากรฐานข้อมูล เป็นต้น

2. ชั้นภาวะเชื่อมต่อ (Connectivity layer) เป็นชั้นที่รับผิดชอบเกี่ยวกับมาตรฐานการติดต่อภายในระบบกริด โดยจะรวมถึงมาตรฐานในด้านการควบคุมความปลอดภัยด้วย ตัวอย่างมาตรฐานภายในชั้นนี้ได้แก่ ทีซีพี (Transport Control Protocol: TCP) หรือทีแอลเอส (Transport Layer Security: TLS) เป็นต้น
3. ชั้นทรัพยากร (Resource layer) เป็นชั้นที่รับผิดชอบบริการต่างๆที่เกี่ยวข้องกับแต่ละทรัพยากรย่อยหนึ่งๆ ตัวอย่างของบริการภายในชั้นนี้ได้แก่ การติดต่อขอเรียกดูข้อมูล หรือ การขอใช้ทรัพยากรภายในระบบกริด
4. ชั้นคอลเลคทีฟ (Collective layer) เป็นชั้นที่รับผิดชอบบริการต่างๆที่เกี่ยวข้องกับกลุ่มของทรัพยากรภายในระบบกริด ตัวอย่างของบริการภายในชั้นนี้ได้แก่ บริการหาทรัพยากร เป็นต้น
5. ชั้นโปรแกรมประยุกต์ (Application layer) เป็นส่วนที่จะอธิบายถึงโปรแกรมต่างๆที่จะเข้ามาใช้ทรัพยากรและบริการต่างๆภายในระบบกริด ตัวอย่างของโปรแกรมในชั้นนี้ได้แก่ โปรแกรมหาค่าพลังงานจากโครงสร้างโมเลกุลซึ่งต้องการหน่วยประมวลผลจำนวนมาก

### 2.1.3 โกลบัสทูลคิท

ผู้วิจัยเลือกใช้เครื่องมือในการติดตั้งและพัฒนากริดด้วยโกลบัสทูลคิทจากกลุ่มผู้พัฒนาโกลบัส (Globus Project) [22] เครื่องมือนี้เป็นที่แพร่หลายในกลุ่มผู้ทำงานวิจัยและผู้ใช้กริดเปิดเผยโค้ดต้นฉบับ (Open-source) และมีการพัฒนาอย่างต่อเนื่องโดยกลุ่มผู้ทำงานวิจัยและผู้ที่สนใจ ผู้วิจัยได้พัฒนากริดด้วยโกลบัสทูลคิทรุ่น 2 และรุ่น 3 ซึ่งในแต่ละรุ่นก็ประกอบไปด้วยรุ่นหลักและรุ่นย่อยซึ่งปรับเปลี่ยนไปตามการแก้ไขและพัฒนา

โกลบัสทูลคิทเวอร์ชัน 2 มีแกนหลักซึ่งประกอบด้วย

1. กลุ่มของเครื่องมือที่ให้บริการที่จำเป็นสำหรับการประมวลผลแบบกริดคือ
    - ความปลอดภัย (Security) ด้วยการพิสูจน์ตน (Authentication) การพิสูจน์สิทธิ์ (Authorization) การพิสูจน์ตนเพียงครั้งเดียว (Single sign-on) และการถ่ายโอนข้อมูลอย่างปลอดภัย (Secure data transfer)
    - การจัดการทรัพยากร (Resource management) ด้วยการจัดการงานและส่งมอบงานไปปฏิบัติงานแบบทางไกล

- การจัดการข้อมูล (Data management) ด้วยการเคลื่อนย้ายข้อมูลอย่างมีเสถียรภาพและปลอดภัย และระบบการทำซ้ำข้อมูล
- บริการสารสนเทศ (Information service) ด้วยไดเรกทอรีข้อมูลทรัพยากร บริการค้นหาทรัพยากร และบริการสอบถามสถานะของทรัพยากรเหล่านั้น

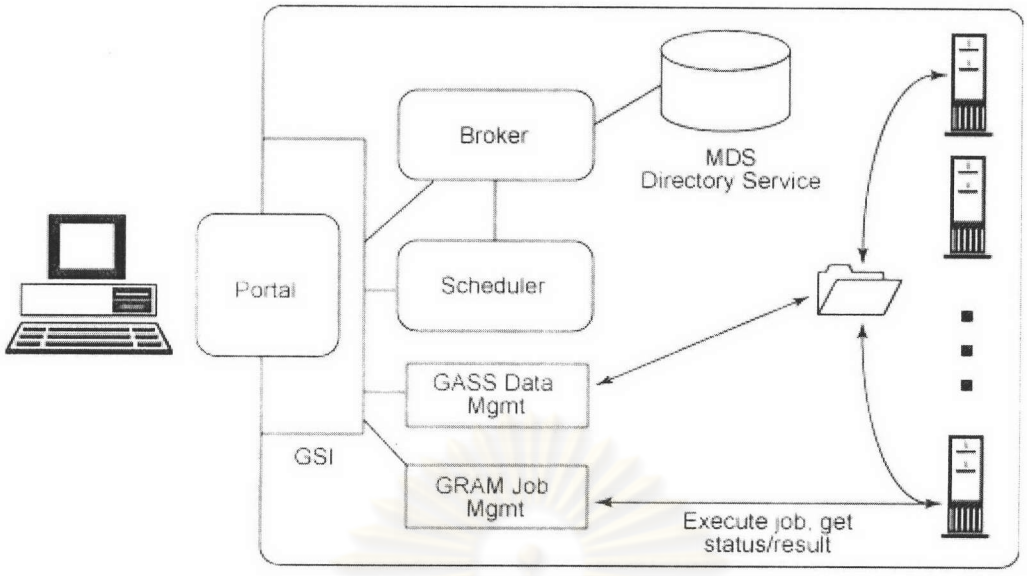
2. เอพีไอ (Application Programming Interface: API) สำหรับเรียกใช้บริการที่กล่าวมาข้างต้น
3. เฮดเดอร์ไฟล์ (Header file) เขียนด้วยภาษาซี สำหรับสร้างและแปลโปรแกรม (Compile)

โกลบัสทูลคิทรุ่น 2 นี้ประสบปัญหาด้านการใช้งานร่วมกันกับกริดที่ถูกพัฒนาจากองค์กรอื่น จำเป็นต้องมีการกำหนดมาตรฐานกริด กลุ่มผู้ทำงานวิจัยด้านกริดจึงร่วมกันสร้างมาตรฐานขึ้นมีชื่อว่าโอจีเอสเอ (Open Grid Services Architecture: OGSA) [18, 23, 24] ทำหน้าที่กำหนดมาตรฐานของทุกโครงสร้างและบริการในสภาพแวดล้อมกริด ทำให้เกิดความสามารถในการทำงานร่วมกัน (Interoperability) ระหว่างกริดที่ถูกออกแบบโดยใช้โอจีเอสเอได้ จากนั้นจึงสร้างโอจีเอสไอ (Open Grid Services Interface: OGSI) เป็นมาตรฐานกำหนดส่วนเชื่อมต่อและโพรโทคอลที่ใช้ระหว่างบริการต่างๆในสภาพแวดล้อมกริด

โกลบัสทูลคิทรุ่น 3 ถูกสร้างขึ้นตามมาตรฐานโอจีเอสเอ มาตรฐานนี้ได้ปรับเปลี่ยนรูปแบบการเขียนโปรแกรมไปสู่แนวคิดแบบเว็บเซอร์วิส (Web service) ที่มีความสะดวกและข้อดีอื่นๆอีกมากมาย รวมถึงการใช้วิธีการตามมาตรฐานเปิดและเป็นสามัญในการเข้าถึงบริการกริดต่างๆ เช่น ใช้มาตรฐานเอสโอเอพี (Simple Object Access Protocol: SOAP) และเอ็กซ์เอ็มแอล (Extention Message Language: XML) [25] มีมาตรฐานในการค้นหา บ่งชี้ และเข้าใช้งานบริการกริดใหม่เมื่อบริการพร้อม

### 2.1.4 องค์ประกอบกริด

องค์ประกอบของกริด (Grid component) ขึ้นอยู่กับการออกแบบและการใช้งานกริด บางองค์ประกอบอาจไม่จำเป็นต้องใช้ แต่ในบางครั้งอาจต้องมีการรวมความสามารถให้เกิดองค์ประกอบที่ซับซ้อนขึ้น ในที่นี้จะอธิบายองค์ประกอบของกริดอ้างอิงตามโกลบัสทูลคิท ดังรูปที่ 2.2




รูปที่ 2.2 มุมมองความสัมพันธ์ขององค์ประกอบภายในกริด

- พอร์ทอล (Portal) หรือส่วนเชื่อมต่อสำหรับผู้ใช้ (User interface)

เพื่อลดความยุ่งยากและซับซ้อนในการใช้งานกริด กริดพอร์ทอลใช้แนวคิดของเว็บพอร์ทอล (Web portal) ที่ผู้ใช้สามารถใช้งานผ่านเว็บบราวเซอร์ด้วยอินเทอร์เฟสเดียวเพื่อเข้าถึงแหล่งสารสนเทศที่มีอยู่มากมาย เช่นเดียวกัน กริดพอร์ทอลมีส่วนเชื่อมต่อสำหรับให้ผู้ใช้ประมวลผลแอพพลิเคชันที่ใช้ทรัพยากรและบริการต่างๆที่กริดมีให้ได้สะดวกและง่ายขึ้น

- จีเอสไอ (Grid Security Infrastructure: GSI)

ความต้องการหลักของการประมวลผลบนเครือข่ายกริดคือความปลอดภัย เพื่อให้มีการสื่อสารอย่างปลอดภัยระหว่างองค์กรภายในกริดประมวลผล โกลบัสทูลคิทมีจีเอสไอสำหรับการพิสูจน์ตน การพิสูจน์ตนทั้งสองฝ่าย (Mutual authentication) การพิสูจน์ตนเพียงครั้งเดียว การพิสูจน์สิทธิ์ การเข้ารหัสข้อมูล (Data encryption) เพื่อให้เกิดความปลอดภัยในการสื่อสารระหว่างขอบเขตองค์กรภายในเครือข่ายกริด โดยต้องไม่เป็นระบบความปลอดภัยซึ่งจัดการที่ศูนย์กลาง (Centrally-managed)

จีเอสไออยู่บนพื้นฐานของการเข้ารหัสแบบคีย์สาธารณะด้วยใบรับรองแบบ X.509 (X.509 certificate) และโพรโทคอลการสื่อสารเอสเอสแอล (Secure Sockets Layer: SSL) โดยการพัฒนาจีเอสไอในโกลบัสทูลคิทยึดตามเอพีไอจีเอสเอส (Generic Security Service API: GSS-API) ซึ่งเป็นเอพีไอมาตรฐานสำหรับระบบความปลอดภัย

- โบรกเกอร์ (Broker)

เมื่อผ่านการพิสูจน์ตนแล้ว ผู้ใช้สามารถทำการประมวลผลโปรแกรมประยุกต์บนทรัพยากรกริดที่พร้อมใช้งานและเหมาะสม โดยอาจมีการระบุจากผู้ใช้ผ่านพารามิเตอร์ ซึ่งการจัดหาทรัพยากรให้แก่การประมวลผลควรถูกจัดการโดยโบรกเกอร์ แม้ว่าโกลบัสทูลคิทจะยังไม่มีโบรกเกอร์ แต่ก็มีบริการข้อมูลสารสนเทศบนพื้นฐานของแอลแดพ (Lightweight Directory Access Protocol: LDAP) คือบริการจีไอเอส (Grid Information Service: GIS) เอ็มดีเอส (Monitoring and Discovery Service: MDS) ทำหน้าที่ให้ข้อมูลสารสนเทศและสถานะของทรัพยากรบนกริดที่พร้อมใช้งาน ทำให้สามารถพัฒนาบริการโบรกเกอร์โดยใช้ประโยชน์จากบริการติดตามและค้นหาได้

- สเกดยูลเลอร์ (Scheduler)

เมื่อได้ทำการระบุทรัพยากรที่ใช้ในการประมวลผลแล้ว งานจะถูกส่งไปประมวลผลบนทรัพยากรเหล่านั้น ถ้ากลุ่มงานที่ถูกประมวลผลไม่ขึ้นต่อกัน ก็ไม่จำเป็นต้องมีสเกดยูลเลอร์ที่เฉพาะเจาะจงเพื่อกลุ่มงานที่ขึ้นต่อกัน แต่หากถ้าผู้ใช้ต้องการจองทรัพยากรโดยเฉพาะเจาะจงหรือต้องประมวลผลหลายๆ งานในแอพพลิเคชันพร้อมกัน (Concurrent) ยกตัวอย่างเช่น ถ้ามีการสื่อสารระหว่างโพรเซส (Inter-process communication) แล้ว จำเป็นต้องมีสเกดยูลเลอร์เป็นผู้ประสานงานการประมวลผล โกลบัสทูลคิทไม่มีสเกดยูลเลอร์เช่นนี้มา แต่ก็มีสเกดยูลเลอร์มากมายที่ได้ผ่านการทดสอบในการใช้ร่วมกับสภาพแวดล้อมกริดของโกลบัส เราสามารถจัดสเกดยูลเลอร์ในกริดให้เป็นระดับได้ เช่นแทนคลัสเตอร์หนึ่งด้วยทรัพยากรเดียว โดยที่คลัสเตอร์อาจมีสเกดยูลเลอร์ของตนเองสำหรับช่วยจัดการโหนดต่างๆ ในคลัสเตอร์ สเกดยูลเลอร์ในระดับที่สูงกว่า (Meta scheduler) ทำหน้าที่จัดรายการงานให้แก่คลัสเตอร์ ในขณะเดียวกันสเกดยูลเลอร์ของคลัสเตอร์ก็จัดรายการงานบนกลุ่มของโหนดในคลัสเตอร์นั้นๆ

- จีเอเอสเอส (Grid Access to Secondary Storage: GASS)

เมื่อมีข้อมูลหรือโมดูลของแอพพลิเคชันต้องถูกเคลื่อนย้ายหรือมอบการยินยอมให้เข้าถึงตนไปยังโหนดหนึ่งที่งานของแอพพลิเคชันจะใช้ในการประมวลผล จำเป็นต้องมีวิธีการที่ปลอดภัยและเชื่อถือได้ในการเคลื่อนย้ายไฟล์หรือข้อมูลไปยังโหนดต่างๆ บนกริด โกลบัสทูลคิทมีองค์ประกอบการจัดการข้อมูลที่ให้บริการเหล่านี้

เรียกว่า จีเอเอสเอส รวมถึงเครื่องมือช่วยอำนวยความสะดวกเช่น กริดเอฟทีพี (GridFTP) กริดเอฟทีพีถูกสร้างบนโพรโทคอลเอฟทีพีมาตรฐาน (File Transfer Protocol: FTP) แต่เพิ่มความสามารถเพิ่มเติมและใช้จีเอสไอในการพิสูจน์ตนและพิสูจน์สิทธิ์ ดังนั้นหากผู้ใช้มีใบรับรองพร็อกซี่ (Proxy certificate) ที่ใช้ผ่านการพิสูจน์ตนได้แล้ว ผู้ใช้สามารถทำการถ่ายโอนข้อมูลบนกริดด้วยกริดเอฟทีพีโดยไม่ต้องผ่านกระบวนการพิสูจน์ตนยังโหนดต่างๆ ที่เกี่ยวข้อง กริดเอฟทีพีรองรับการถ่ายโอนไฟล์โดยบุคคลที่สาม (Third-party file transfer) คือโหนดหนึ่งสามารถเริ่มและควบคุมการถ่ายโอนไฟล์ระหว่างสองโหนดอื่นๆ ได้

- แกรม (Grid Resource Allocation Manager: GRAM)

  แกรมให้บริการการเริ่ม (Launch) ประมวลผลงานบนทรัพยากรที่เจาะจง ตรวจสอบสถานะของงาน และเอาผลลัพธ์จากการประมวลผลกลับคืนเมื่อการประมวลผลเสร็จสมบูรณ์

## 2.2 เพียร์ทูเพียร์ จักซ์ตา และจีไอเอสพี

### 2.2.1 เพียร์ทูเพียร์

เพียร์ทูเพียร์คือประเภทของระบบและโปรแกรมประยุกต์ที่ใช้ทรัพยากรที่กระจายตัวอยู่เพื่อทำงานในรูปแบบการจัดการแบบไม่รวมศูนย์กลาง (Decentralized) โดยทรัพยากรนี้ประกอบด้วยพลังประมวลผล ข้อมูล (เนื้อที่จัดเก็บและเนื้อข้อมูล) แบนด์วิดท์เครือข่าย การปรากฏตัว (คอมพิวเตอร์ มนุษย์ และทรัพยากรอื่นๆ) งานที่ทำอาจเป็นการประมวลผลแบบกระจาย การแบ่งร่วมข้อมูลหรือเฉพาะเนื้อข้อมูล การติดต่อสื่อสารและการทำงานร่วมกัน หรือบริการแพลตฟอร์ม ลักษณะไม่รวมศูนย์กลางด้วยอัลกอริทึมและข้อมูล หรืออย่างใดอย่างหนึ่ง เพียร์ทูเพียร์อาจมีบางส่วนของระบบเป็นแบบรวมศูนย์กลาง (Centralized) ได้เพื่อให้รองรับความต้องการของระบบ ปกติแล้วเพียร์ทูเพียร์วางตัวอยู่บนขอบนอกของอินเทอร์เน็ตหรือบนเครือข่ายเฉพาะกิจ (Ad-hoc)

**เป้าหมายของระบบเพียร์ทูเพียร์**

1. ร่วมใช้หรือลดค่าใช้จ่าย ระบบแบบขึ้นกับศูนย์กลางต้องแบกรับภาระการให้บริการไคลเอนต์จำนวนมาก อันเป็นค่าใช้จ่ายหลักของระบบ เมื่อค่าใช้จ่ายหลักนี้โตขึ้นเกินระบบจะรองรับได้ การกระจายค่าใช้จ่ายยังผู้ใช้บริการจึงเป็นทางออกที่เหมาะสม ระบบเพียร์ทูเพียร์กระจายค่าใช้จ่ายไปยังโหนดต่างๆ ในระบบ ค่าใช้จ่ายอาจเป็นการดูแลบำรุงระบบ ทรัพยากร องค์ประกอบของระบบ

2. เพิ่มความสามารถขยายตัว (Scalability) และความน่าเชื่อถือ (Reliability)
3. รวบรวม (Aggregation) และสร้างการทำงานร่วมกัน (Interoperability) ของทรัพยากร
4. เพิ่มภาวะอัตโนวัติ (Autonomy)
5. ภาวะปิดบัง (Anonymity) และภาวะส่วนตัว (Privacy)
6. ความพลวัต (Dynamism)
7. ช่วยให้เกิดการสื่อสารเฉพาะกิจและการร่วมมือ (Collaboration) ระบบมีความพลวัตสูงบ่งชี้ถึงการรองรับสภาพแวดล้อมแบบเฉพาะกิจที่สมาชิกของระบบสามารถเกิดขึ้นได้ตามตำแหน่งทางกายภาพในขณะนั้นๆ หรือความสนใจในขณะนั้นๆ ได้ ระบบเพียร์ทูเพียร์ไม่ผูกมัดกับโครงสร้างทางกายภาพ สามารถสร้างโครงสร้างทับซ้อน (Overlay infrastructure) ได้

**อนุกรมวิธานของเพียร์ทูเพียร์**

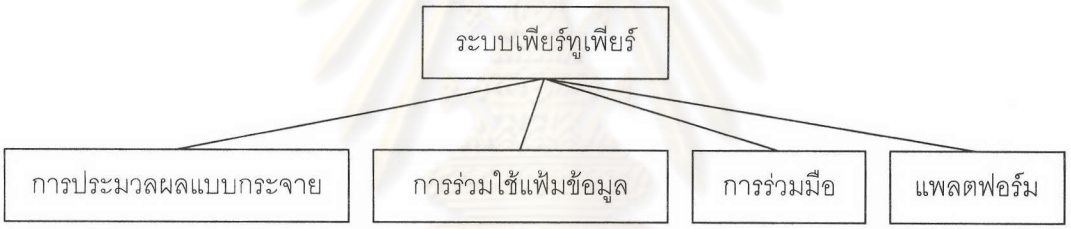
เพียร์ทูเพียร์จัดอยู่ในระบบแบบกระจายแบบหนึ่งของระบบคอมพิวเตอร์ดังรูปที่ 2.3


รูปที่ 2.3 อนุกรมวิธานของระบบคอมพิวเตอร์

แบบจำลองเพียร์ทูเพียร์แบ่งออกเป็นแบบบริสุทธิ์ (Pure) และแบบลูกผสม (Hybrid) ในแบบบริสุทธิ์ จะไม่มีเซิร์ฟเวอร์ศูนย์กลางเลย ตัวอย่างของระบบเช่น Gnutella และ FreeNet [26, 27, 28, 29] ในขณะที่แบบลูกผสมนั้น เซิร์ฟเวอร์ถูกใช้เพื่อเก็บสารสนเทศแบบเมทา (Meta-information) เช่นไอดีของเพียร์ ตัวอย่างของระบบเช่น Napster, Groove [30, 31, 32, 33], Aimster [34], Magi [35, 36], Softwax [37] และ iMesh [38] ทั้งนี้ยังมีวิธีแบบกึ่งกลางที่ใช้ซูเปอร์เพียร์ (Super-peer) [39] ซึ่งเป็นเพียร์ที่เก็บสารสนเทศที่เพียร์อื่นอาจไม่มีและเพียร์อื่นๆ สามารถมาใช้สำหรับการค้นหาได้ ตัวอย่างของระบบที่ใช้ซูเปอร์เพียร์คือ KaZaA

ซูเปอร์เพียร์เป็นแนวคิดใหม่ที่เป็นการนำเอาคุณลักษณะระบบแบบขึ้นกับศูนย์กลางมาใช้กับระบบเพียร์ทูเพียร์เพื่อลดการจราจรภายในเครือข่าย และเพื่อความน่าเชื่อถือของระบบ เพิ่มความสามารถขยายตัวของระบบ สามารถควบคุมและจัดการเป็นขอบเขต รวมถึงการเพิ่มประสิทธิภาพเครือข่ายด้วยการดุลภาระ (Load balancing) จัดการความปลอดภัยได้ง่าย ซูเปอร์เพียร์คือเพียร์ที่ทำหน้าที่คล้ายเซิร์ฟเวอร์ศูนย์กลางให้กับกลุ่มเพียร์ที่เป็นดั่งไคลเอนต์ของตน กลุ่มเพียร์เชื่อมโยงกันในรูปแบบบริสุทธิ์ กลุ่มเพียร์ไคลเอนต์นี้จะทำการสืบค้นและค้นคืนผ่านซูเปอร์เพียร์ที่เป็นตัวควบคุม ปรับแต่ง บริหารและดูแลเรื่องความปลอดภัยให้กับกลุ่มเพียร์นั้นๆ ซูเปอร์เพียร์จากแต่ละกลุ่มก็จะเชื่อมต่อกันในรูปแบบบริสุทธิ์ โดยมีค่าปรับแต่งการเชื่อมต่อที่แตกต่างจากกลุ่มเพียร์ไคลเอ็นต์ของแต่ละซูเปอร์เพียร์ แบบจำลองแบบซูเปอร์เพียร์นี้มีข้อเสียที่ได้รับจากระบบแบบขึ้นกับศูนย์กลางคือความล้มเหลวจากเพียงจุดเดียว (Single point of failure) เมื่อมีซูเปอร์เพียร์เพียงเพียร์เดียวสำหรับกลุ่มเพียร์ ซึ่งสามารถแก้ไขโดยด้วยการมีหลายซูเปอร์เพียร์สำหรับกลุ่มเพียร์

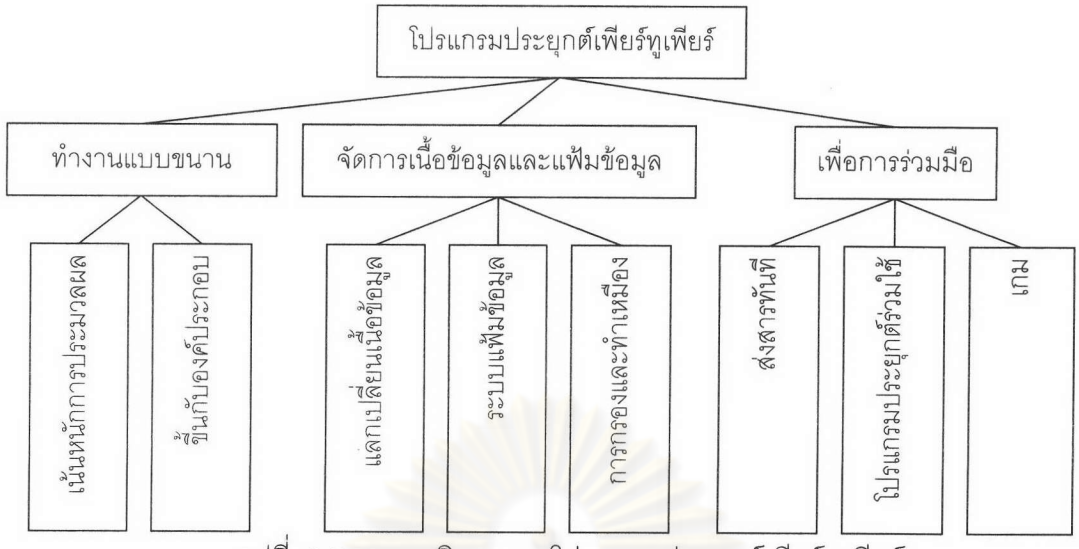
ระบบเพียร์ทูเพียร์แบ่งออกเป็นสี่ประเภทหลักๆ ดังในรูปที่ 2.4




รูปที่ 2.4 อนุกรมวิธานของระบบเพียร์ทูเพียร์

ตัวอย่างของการประมวลผลแบบกระจายได้แก่ SETI@home, Avaki [40], Entropia [41] ตัวอย่างของการร่วมใช้แฟ้มข้อมูลได้แก่ Napster, Gnutella, Freenet, Publius [42], Free Haven [43] ตัวอย่างของการร่วมมือได้แก่ Magi, Groove, Jabber [44] และตัวอย่างของแพลต์ฟอร์มได้แก่ จักซ์ตาและ .NET My Services [45]

โปรแกรมประยุกต์เพียร์ทูเพียร์แบ่งได้สามประเภทได้ดังรูปที่ 2.5




รูปที่ 2.5 อนุกรมวิธานของโปรแกรมประยุกต์เพียร์ทูเพียร์

โปรแกรมประยุกต์เพียร์ทูเพียร์ที่ทำงานแบบขนาน (Parallelizable) โปรแกรมประเภทนี้แบ่งงานขนาดใหญ่ออกเป็นงานชิ้นเล็กๆ ที่สามารถทำงานขนานกันได้บนหลายเพียร์ที่เป็นอิสระต่อกัน แบ่งลักษณะงานที่สามารถทำงานแบบขนานนี้ได้เป็นสองลักษณะคือ

- แบบขนานเพื่อเน้นหนักที่การประมวลผล (Compute-intensive) โดยมีแนวคิดหลักจากการใช้พลังประมวลผลจากเครื่องคอมพิวเตอร์ที่ว่างงานอยู่ซึ่งเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตมาช่วยแก้ปัญหาที่ต้องการการประมวลผลสูง มักเป็นงานเดียวกันบนแต่ละเพียร์ แต่ป้อนพารามิเตอร์ที่แตกต่างกัน ตัวอย่างเช่น การค้นหาสิ่งมีชีวิตนอกโลก (SETI@Home) ราคาหลักทรัพย์ การคำนวณประกันความเสี่ยง การประเมินตลาดและสินเชื่อ การวิเคราะห์ทางประชากรศาสตร์ เป็นต้น
- แบบขนานเพื่อเน้นองค์ประกอบ (Componentized) โปรแกรมประยุกต์ประเภทนี้จะส่งงานไปทำบนเพียร์ที่มีองค์ประกอบที่ตรงตามความต้องการของงาน โดยแต่ละเพียร์ต่างมีองค์ประกอบที่ต่างกันไป ตัวอย่างของรูปแบบการทำงานประเภทนี้ได้แก่กระแสงาน (Workflow) จาวาบีน (JavaBean) หรือเว็บเซอร์วิส (Web service)

โปรแกรมประยุกต์เพียร์ทูเพียร์จัดการเนื้อข้อมูลและแฟ้มข้อมูล (Content and file management) มุ่งเน้นที่การจัดเก็บและค้นคืนข้อมูลจากเพียร์ต่างๆ บนเครือข่าย แบ่งลักษณะการจัดการออกเป็นสามลักษณะคือ

- การแลกเปลี่ยนเนื้อข้อมูล (Content exchange) เช่น Napster และ Gnutella ช่วยให้เพียร์สามารถค้นหาและดาวน์โหลดแฟ้มข้อมูลที่เพียร์อื่นอนุญาตได้
- ระบบแฟ้มข้อมูล (File system) [46]

- การกรองและทำเหมืองข้อมูล (Filtering, Mining) มุ่งเน้นที่เทคนิคการกรองแบบร่วมมือกันด้วยการสร้างดัชนีที่ใช้ค้นหาได้ เช่น OpenCOLA [47], JXTA Search [48]

โปรแกรมประยุกต์เพื่อการร่วมมือช่วยให้ผู้ใช้สามารถทำงานร่วมกันโดยปราศจากเซิร์ฟเวอร์กลางที่ใช้รวบรวมและส่งต่อข้อมูลสารสนเทศ เช่น

- การส่งสารทันที (instant messaging) เช่น Yahoo! [49], AOL [50] และ Jabber
- การใช้โปรแกรมประยุกต์ร่วมกัน (Shared application) ช่วยให้ผู้ใช้สามารถโต้ตอบกันในขณะที่สามารถมองและแก้ไขข้อมูลสารสนเทศเดียวกันได้ในเวลาเดียวกัน เช่น Buzzpad [51] และ distributed PowerPoint [52]
- เกม การส่งข้อมูลกระจายไปยังเพียร์ต่างๆ ที่เกมทำงานอยู่โดยไม่ต้องใช้เซิร์ฟเวอร์กลาง เช่น NetZ 4.3 โดย Quazal [53], Scour Exchange โดย CenterSpan Descent [54] และ Cybiko [55]

### 2.2.2 จักซ์ตา

จักซ์ตาคือแพลตฟอร์มแบบเปิดสำหรับการประมวลผลบนเครือข่ายที่ถูกออกแบบสำหรับการประมวลผลแบบเพียร์ทูเพียร์ การสร้างแพลตฟอร์มนี้มีเป้าหมายเพื่อพัฒนาส่วนสร้าง (Building block) และบริการที่เป็นพื้นฐานสำหรับแอพพลิเคชันบนกลุ่มเพียร์ จักซ์ตามีชุดโพรโทคอลแบบเปิดและอิมพลีเมนเตชันอ้างอิงที่เปิดเผยต้นฉบับ สำหรับการพัฒนาโปรแกรมประยุกต์แบบเพียร์ทูเพียร์ โพรโทคอลจักซ์ตาสร้างมาตรฐานการจัดการของเพียร์ในเรื่อง

- การสืบค้น (Discover) เพียร์อื่นๆ
- การเข้าร่วมกลุ่มด้วยตนเอง
- การประกาศและสืบค้นบริการเครือข่าย
- การสื่อสารกับเพียร์อื่นๆ
- การเฝ้าสังเกต (Monitor) เพียร์อื่นๆ

โพรโทคอลจักซ์ตาถูกออกแบบให้เป็นอิสระจากภาษาเขียนโปรแกรมและเป็นอิสระจากโพรโทคอลการถ่ายโอน สามารถอิมพลีเมนต์โพรโทคอลด้วยภาษาเขียนโปรแกรมจาวา ซี ซีพลัสพลัส เพิร์ลและภาษาอื่นๆ บนโพรโทคอลการถ่ายโอนทีซีพี/ไอพี, เอชทีทีพี, บลูทูทหรือโพรโทคอลการถ่ายโอนอื่นๆ

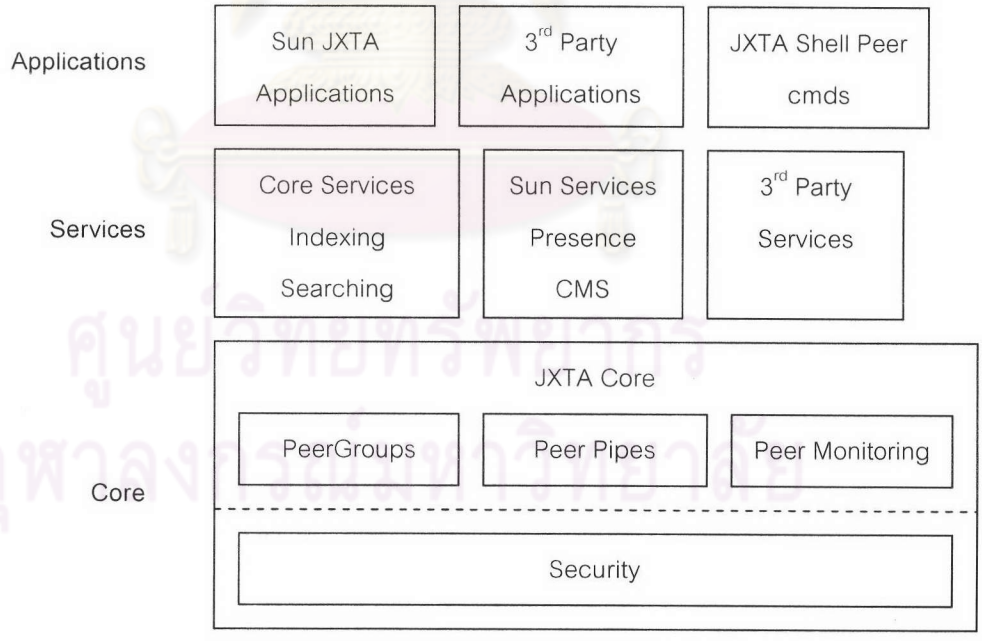
### 2.2.2.1 เป้าหมายของจักซ์ตา

เป้าหมายหลักของจักซ์ตาคือการมีแพลตฟอร์มที่มีการทำงานพื้นฐานที่จำเป็นต่อเครือข่ายเพียร์ทูเพียร์ และเอาชนะข้อบกพร่องบนระบบเพียร์ทูเพียร์ที่มีอยู่เดิม กล่าวคือ

- เพื่อให้เกิดความสามารถทำงานร่วมกันได้ (Interoperability) เทคโนโลยีจักซ์ตาถูกออกแบบให้เพียร์สามารถให้บริการเพียร์ทูเพียร์ที่หลากหลาย สามารถระบุตำแหน่งและติดต่อสื่อสารกับเพียร์อื่นๆ ได้
- เป็นอิสระจากแพลตฟอร์ม (Platform independence) เทคโนโลยีจักซ์ตาถูกออกแบบให้เป็นอิสระจากภาษาเขียนโปรแกรม โปรโทคอลการถ่ายโอน และแพลตฟอร์มที่นำไปใช้งาน
- ทำงานได้แพร่หลาย (Ubiquity) เทคโนโลยีจักซ์ตาถูกออกแบบมาให้ถูกใช้ได้ด้วยอุปกรณ์ใดๆ ที่มีจังหวะดิจิตอล (Digital heartbeat) ไม่จำกัดที่เครื่องคอมพิวเตอร์ส่วนบุคคลหรือแพลตฟอร์มที่กำหนดเท่านั้น

### 2.2.2.2 สถาปัตยกรรมจักซ์ตา

สถาปัตยกรรมจักซ์ตาแบ่งออกเป็นสามชั้นดังรูปที่ 2.6




รูปที่ 2.6 สถาปัตยกรรมซอร์ฟแวร์ของจักซ์ตา

- ชั้นแพลตฟอร์มหรือแกนจักซ์ตา (JXTA core) เป็นส่วนพื้นฐานร่วมที่จำเป็นสำหรับการทำงานของเครือข่ายเพียร์ทูเพียร์ ประกอบด้วยส่วนสร้างที่เป็นกลไกสำคัญสำหรับแอพพลิเคชันเพียร์ทูเพียร์ รวมถึงการสืบค้น (Discovery) การส่งถ่าย

(Transport) ที่มีไฟร์วอลล์ การสร้างของเพียร์และเพียร์กรุ๊ป และความปลอดภัยพื้นฐานที่เกี่ยวข้อง

- ชั้นบริการ (Services Layer) ประกอบด้วยบริการเครือข่ายที่ไม่จำเป็นอย่างแท้จริงสำหรับการทำงานของเครือข่ายเพียร์ทูเพียร์ แต่เป็นส่วนร่วมหรือที่ควรมีในสภาพแวดล้อมเพียร์ทูเพียร์ ตัวอย่างของบริการเครือข่ายนี้เช่น การค้นหา (Searching) และการทำดัชนี (Indexing) การสร้างไดเรกทอรี ระบบจัดเก็บข้อมูล (Storage system) การร่วมใช้แฟ้มข้อมูล ระบบแฟ้มข้อมูลแบบกระจาย การรวบรวมและให้เช่าทรัพยากร การแปลงโพรโทคอล (Protocol translation) การพิสูจน์ตนและบริการที่ใช้โครงสร้างพื้นฐานกุญแจสาธารณะ (Public key infrastructure)
- ชั้นโปรแกรมประยุกต์ (Application layer) ประกอบด้วยโปรแกรมประยุกต์ที่สร้างขึ้นที่สามารถ เรียกใช้บริการจากชั้นบริการ ตัวอย่างโปรแกรมประยุกต์เช่นการส่งสารทันทีแบบเพียร์ทูเพียร์ การร่วมใช้เอกสารและทรัพยากร การจัดการและนำส่งข้อมูลสาระบันเทิง ระบบอีเมลแบบเพียร์ทูเพียร์ ระบบประมูลแบบกระจาย และอื่นๆ

**องค์ประกอบจักซ์ตา**

เครือข่ายจักซ์ตาประกอบด้วยกลุ่มโหนดที่เชื่อมต่อกัน ซึ่งเรียกว่าเพียร์ เพียร์สามารถเข้าร่วมเพียร์กรุ๊ปที่มีกลุ่มบริการร่วมได้ด้วยตนเอง ตัวอย่างของบริการเช่นระบบร่วมใช้เอกสารหรือโปรแกรมประยุกต์สำหรับสนทนา

เพียร์จักซ์ตาประกาศบริการของตนด้วยเอกสารเอ็กซ์เอ็มแอลที่เรียกว่า แอดเวอทิสเมนต์ (Advertisement) แอดเวอทิสเมนต์ช่วยให้เพียร์อื่นบนเครือข่ายรู้วิธีการเชื่อมต่อและโต้ตอบกับบริการของเพียร์ได้

เพียร์จักซ์ตาใช้ไปป์ (Pipe) ในการส่งเมสเสจ (Message) ไปยังเพียร์อื่นๆ ไปป์เป็นกลไกการถ่ายโอนเมสเสจแบบอสมวาร (Asynchronous) และเป็นแบบทางเดียว (Unidirectional) ที่ใช้ในการสื่อสารกับบริการ เมสเสจเป็นเอกสารเอ็กซ์เอ็มแอลที่บรรจุเส้นทาง (routing) ไดเจท (Digest) และข้อมูลครีเดนเชียล (Credential) ไปป์ถูกเชื่อมโยงกับเอนพอยน์ท (Endpoint) ที่กำหนดได้ เช่นพอร์ตทีซีพีและแอดเดรสไอพี

**เกณฑ์เงื่อนไขของสถาปัตยกรรมจักซ์ตา**

เกณฑ์ที่สำคัญสามประการของสถาปัตยกรรมจักซ์ตาที่แตกต่างจากแบบจำลองเครือข่ายแบบกระจายอื่นๆ คือ

- การใช้เอกสารเอ็กซ์เอ็มแอล (แอดเวอทิสเมนต์) ในการอธิบายทรัพยากรเครือข่าย
- การกำหนดเชิงนามธรรมจากไปป์ไปสู่เพียร์ และจากเพียร์ไปสู่เอนพอยน์ทโดยไม่ต้องอาศัยตัวกลางการกำหนดชื่อและแอดเดรสแบบศูนย์กลางเช่น ดีเอ็นเอส (DNS)
- แบบแผนการกำหนดแอดเดรสเพียร์ที่รูปแบบเดียวกัน (เพียร์ไอดี)

2.2.2.3 **สถาปัตยกรรมเครือข่าย**

**การจัดระบบเครือข่าย**

เครือข่ายจักซ์ตาคือเครือข่ายที่ประกอบด้วยเพียร์ที่เชื่อมต่อกันแบบเฉพาะกิจ การเชื่อมต่อในเครือข่ายอาจเป็นแบบชั่วครู่ และการจัดเส้นทางเมสเสจระหว่างเพียร์เป็นแบบไม่กำหนด (Nondeterministic) เพียร์อาจเข้าร่วมหรือออกจากเครือข่ายในเวลาใดๆ ได้ และเส้นทาง (Route) อาจเปลี่ยนแปลงได้บ่อยครั้ง

เพียร์เหล่านี้อาจจัดรูปแบบใดๆ ก็ได้ตราบเท่าที่พวกมันยังคงติดต่อสื่อสารด้วยโพรโทคอลจักซ์ตา การจัดระบบเครือข่ายไม่ได้อยู่ในอาณัติของโครงร่างจักซ์ตา แต่ในทางปฏิบัติแล้วมีเพียร์ 4 ชนิดที่ใช้คือ

1. เพียร์ขอบขนาดย่อม (Minimal edge peer) เป็นเพียร์ที่สามารถรับส่งข้อความ แต่ไม่สามารถแคช (Cache) แอดเวอทิสเมนต์หรือจัดเส้นทางเมสเสจให้แก่เพียร์อื่นได้ ได้แก่ เพียร์ที่ทำงานบนอุปกรณ์ที่มีทรัพยากรจำกัด เช่น พีดีเอหรือโทรศัพท์แบบพกพา
2. เพียร์ขอบคุณลักษณะครบ (Full-featured edge peer) เป็นเพียร์ที่สามารถรับส่งข้อความและแคชแอดเวอทิสเมนต์ได้ เพียร์สามารถตอบกลับการร้องขอสืบค้นด้วยข้อมูลที่พบในแอดเวอทิสเมนต์ที่เก็บไว้ในแคชได้ แต่จะไม่ส่งต่อ (Forward) การร้องขอสืบค้น เพียร์ส่วนใหญ่เป็นเพียร์ประเภทนี้
3. เพียร์รันเดวู (Rendezvous peer) เป็นเพียร์ที่มีลักษณะเช่นเดียวกับเพียร์อื่น และดูแลแอดเวอทิสเมนต์ที่เก็บไว้ด้วย เพียร์รันเดวูจะส่งต่อการร้องขอสืบค้นเพื่อช่วยเพียร์อื่นสืบค้นทรัพยากร เมื่อเพียร์เข้าร่วมเพียร์กรุ๊ป เพียร์จะค้นหาหนึ่งเพียรรันเดวูโดยอัตโนมัติ หากไม่พบ เพียร์จะเป็นเพียร์รันเดวูสำหรับเพียร์กรุ๊ปนั้นๆ (เป็นไปอย่างพลวัต) แต่ละเพียร์รันเดวูจะดูแลรายการเพียร์รันเดวูอื่นที่รู้จักและเพียร์อื่นๆ ที่ใช้มันเป็นรันเดวู

เพียร์ขอบส่งคำร้องขอการค้นหาและการสืบค้นไปยังเพียร์รันเดวู เพียร์รันเดวูจะส่งต่อคำร้องขอที่ไม่สามารถตอบได้ไปยังเพียร์รันเดวูอื่นๆ ที่รู้จัก กระบวนการสืบค้นจะดำเนินต่อไปจนกระทั่งมีเพียร์ที่ให้คำตอบหรือคำร้องขอหมดอายุไป เมสเสจมีช่วงชีวิต

(Time-to-live: TTL) โดยปริยายเท่ากับ 7 ฮ็อปและป้องกันลูปแบ็ค (Loopback) โดยการรักษารายการเพียร์ไปตลอดเส้นทางเมสเสจ

4. เพียร์รีเลย์ (Relay peer) เป็นเพียร์ที่ดูแลข้อมูลเกี่ยวกับเส้นทางไปยังเพียร์อื่นๆ และจัดเส้นทางข้อความไปยังเพียร์อื่นๆ เพียร์จะดูข้อมูลเส้นทางในแคชของตนก่อน ถ้าไม่พบเพียร์จะส่งการค้นคืนไปยังเพียร์รีเลย์เพื่อถามข้อมูลเส้นทาง เพียร์รีเลย์จะส่งต่อข้อความในนามของเพียร์ที่ไม่สามารถระบุแอดเดรส (Address) เพียร์อื่นได้โดยตรง (เช่นในสภาวะที่ใช้เอ็นเอที)

เพียร์ใดๆ สามารถอิมพลีเมนต์บริการที่จำเป็นต่อการเป็นเพียร์รีเลย์หรือเพียร์รันเดวูได้ บริการรีเลย์และรันเดวูสามารถถูกอิมพลีเมนต์บนเพียร์เดียวกันได้

**ไฟร์วอลล์และเอนเอที**

เพียร์ที่อยู่หลังไฟร์วอลล์สามารถส่งเมสเสจไปยังเพียร์ที่อยู่นอกไฟร์วอลล์โดยตรงได้ แต่เพียร์ที่อยู่นอกไฟร์วอลล์นั้นไม่สามารถสร้างการเชื่อมต่อโดยตรงกับเพียร์ที่อยู่หลังไฟร์วอลล์ได้ เพื่อให้เพียร์จักซ์ตาสามารถติดต่อกับเพียร์อื่นข้ามไฟร์วอลล์ได้ เงื่อนไขเหล่านี้ต้องเกิดขึ้นคือ

- มีอย่างน้อยที่สุดหนึ่งเพียร์ในเพียร์กรุ๊ป อยู่ภายในไฟล์วอลล์ โดยเพียร์นี้ต้องทราบถึงอย่างน้อยที่สุดหนึ่งเพียร์ที่อยู่ภายนอกไฟร์วอลล์
- เพียร์ภายในและเพียร์ภายนอกไฟร์วอลล์ต้องทราบถึงกันและกันและรองรับโพรโทคอลเอชทีทีพี
- ไฟร์วอลล์ต้องอนุญาตการถ่ายโอนข้อมูลผ่านโพรโทคอลเอชทีทีพี

### 2.2.2.4 โพรโทคอลจักซ์ตา

จักซ์ตากำหนดชุดรูปแบบเมสเสจเอ็กซ์เอ็มแอลหรือก็คือโพรโทคอลสำหรับการสื่อสารระหว่างเพียร์ เพียร์ใช้โพรโทคอลนี้ในการสืบค้นเพียร์อื่น ใช้ประกาศและสืบค้นทรัพยากรเครือข่ายและใช้สื่อสารและจัดเส้นทางเมสเสจ ประกอบด้วย 6 โพรโทคอลคือ

- พีดีพี (Peer Discovery Protocol: PDP) เพียร์ใช้ในการประกาศทรัพยากรของตน (เช่น เพียร์ เพียร์กรุ๊ป ไปป์ หรือบริการ) และใช้สืบค้นทรัพยากรจากเพียร์อื่นๆ ซึ่งแต่ละทรัพยากรจะถูกอธิบายและเผยแพร่โดยใช้แอดเวอทิสเมนต์
- พีไอพี (Peer Information Protocol: PIP) เพียร์ใช้ในการขอข้อมูลสถานภาพ (Status) จากเพียร์อื่นๆ เช่นช่วงเวลาให้บริการ (Uptime), สถานะ, การจราจรล่าสุด เป็นต้น

- พีอาร์พี (Peer Resolver Protocol: PRP) ทำให้เพียร์สามารถส่งคำค้นคืนทั่วไปไปยังหนึ่งเพียร์หรือมากกว่า และรับการตอบสนอง (Response) ของการค้นคืน คำค้นคืนสามารถถูกส่งโดยตรงไปยังทุกเพียร์ในเพียร์กรุ๊ปหรือเฉพาะเพียร์ที่กำหนดในเพียร์กรุ๊ป โพรโทคอลนี้ช่วยให้บริการเพียร์กำหนดและแลกเปลี่ยนข้อมูลใดๆ ที่ไม่เจาะจงซึ่งเพียร์ต้องการได้ ต่างกับพีดีพีและพีอาร์พีที่ใช้ค้นคืนข้อมูลที่นิยามไว้ก่อนหน้าที่ต้องการ

- พีบีพี (Pipe Binding Protocol: PBP) เพียร์ใช้ในการตั้งช่องทางสื่อสารเสมือนหรือที่เรียกว่าไปป์ ระหว่างหนึ่งเพียร์หรือมากกว่า เพียร์ใช้พีบีพีในการ bind ปลายการเชื่อมต่อ (ไปป์เอนพอยน์ต) ทั้งสองหรือมากกว่าได้

- อีอาร์พี (Endpoint Routing Protocol: ERP) เพียร์ใช้ในการค้นหาเส้นทางไปยังพอร์ตปลายทางบนเพียร์อื่นๆ ข้อมูล routing ประกอบด้วยลำดับของไอดีของเพียร์รีเลย์ที่สามารถใช้ในการส่งเมสเสจไปยังปลายทางได้ (เช่น เมสเสจอาจถูกลำเลียงส่งไปยังเพียร์ ก ที่ส่งต่อไปยังเพียร์ ข ที่ส่งต่อไปยังปลายทางสุดท้ายได้)

- อาร์วีพี (Rendezvous Protocol: RVP) เป็นกลไกที่เพียร์สามารถสมัครหรือรับสมัครเข้าใช้บริการแพร่กระจาย (Propagation service) ในเพียร์กรุ๊ปนั้นเพียร์สามารถเป็นเพียร์รันเดวูหรือเพียร์ที่รับฟังเพียร์รันเดวู อาร์วีพีช่วยให้เพียร์สามารถส่งเมสเสจไปยังทุกบริการที่รับฟังอยู่ได้ อาร์วีพีถูกใช้โดยพีอาร์พีและพีบีพีในการแพร่กระจายเมสเสจ

ทุกโพรโทคอลจักซ์ตาเป็นแบบอสมวารและอยู่บนพื้นฐานแบบจำลองค้นคืนและตอบสนอง (Query/response model) เพียร์จักซ์ตาใช้โพรโทคอลในการส่งคำค้นคืนไปยังหนึ่งเพียร์หรือมากกว่าในเพียร์กรุ๊ปของตน เพียร์อาจไม่ได้รับการตอบสนอง หรือได้รับหนึ่งการตอบสนอง หรือมากกว่า จากการค้นคืน ตัวอย่างเช่น เพียร์อาจใช้พีดีพีในการส่งคำค้นคืนการสืบค้นเพียร์ที่รู้จักทั้งหมดที่อยู่ในเน็ตเพียร์กรุ๊ปปริยาย (Default Net Peer Group) ในกรณีนี้หลายเพียร์อาจตอบกลับด้วยการตอบสนองการสืบค้น หรือตัวอย่างเช่น เพียร์อาจส่งคำค้นคืนการสืบค้นไปป์ที่มีชื่อ aardvark ซึ่งถ้าไม่พบไปป์นี้ก็จะไม่มีการตอบสนองต่อการสืบค้น

เพียร์จักซ์ตาไม่จำเป็นต้องอิมพลีเมนต์ทั้งหกโพรโทคอล เพียร์สามารถอิมพลีเมนต์เฉพาะโพรโทคอลที่จำเป็นต่อการใช้งานเท่านั้น จักซ์ตาที่อิมพลีเมนต์ด้วยเจทูเอสอี (J2SE) ปัจจุบันรองรับทั้งหกโพรโทคอลพร้อมทั้งมีเอพีไอภาษาจาวาที่สามารถใช้ในการเข้าถึงกระบวนการที่รองรับในหกโพรโทคอลนี้ได้

### 2.2.3 จีไอเอสพี

จีไอเอสพี (Global Information Sharing Protocol: GISP) เป็นโพรโทคอลสำหรับการค้นคืนข้อมูลสำหรับการแลกเปลี่ยนที่ไม่จำเป็นต้องใช้การบรอดแคสท์ (Broadcast) เมสเสจ โดยใช้ตารางแฮชแบบกระจาย (Distributed hash table) เราสามารถค้นคืนข้อมูลต่างๆ ได้ด้วยคำหลัก (Keyword) โดยการคำนวณและเปรียบเทียบค่าแฮช (Hash value) ของคำหลักและค่าแฮชของเพียร์ไอดี (Peer ID) ด้วยวิธีการนี้ทำให้เราสามารถระบุตำแหน่งข้อมูลได้เนื่องจากข้อมูลที่มีคำหลักเดียวกันควรถูกเก็บลงบนเพียร์เดียวกัน

จีไอเอสพีช่วยให้เกิดความสามารถขยายตัว (Scalability) ความเรียบง่าย (Simplicity) และง่ายต่อการพัฒนาระบบ จีไอเอสพีถูกออกแบบมาเพื่อแก้ปัญหาที่พบใน Gnutella และมุ่งหวังให้เป็นโพรโทคอลเชิงปฏิบัติ สามารถใช้งานได้จริง โดยมีนโยบายของการออกแบบดังนี้

- เพื่อสร้างกลไกการร่วมใช้ข้อมูลสารสนเทศบนเครือข่ายเพียร์ทูเพียร์ที่มีประสิทธิภาพมากกว่าการบรอดแคสท์ในระดับโปรแกรมประยุกต์
- ทำงานบนเครือข่ายเพียร์ทูเพียร์แบบไม่ขึ้นกับศูนย์กลางเต็มรูปได้ (Fully decentralized) โดยยังคำนึงถึงความหลากหลายของความสามารถเพียร์
- เป็นโพรโทคอลที่ง่าย ไม่ซับซ้อน
- ใช้งานบนแพลตฟอร์มเพียร์ทูเพียร์ เช่นจักซ์ตา
- ทำงานบนเครือข่ายจริงได้โดยมีสมมุติฐานว่ามีการโจมตีเกิดขึ้นบ้าง

#### 2.2.3.1 แนวคิดของจีไอเอสพี

อธิบายแนวคิดของจีไอเอสพีด้วยลักษณะเด่นของจีไอเอสพีดังนี้

**ไม่ใช้การบรอดแคสท์เมสเสจ**

ปัญหาสำคัญที่มักพบในระบบเพียร์ทูเพียร์ปัจจุบันคือการทำงานที่ล่าช้าและความสามารถขยายตัวต่ำอันเนื่องจากการใช้การบรอดแคสท์เมสเสจในการสื่อสาร จีไอเอสพีใช้กลไกการทำดัชนี (Indexing) เพื่อหลีกเลี่ยงการใช้การบรอดแคสท์เมสเสจ และทุกเพียร์บนระบบใช้วิธีการเดียวกันในการเลือกเพียร์เป้าหมายสำหรับจัดเก็บและค้นคืนข้อมูลจากคำหลักของแต่ละข้อมูลนั้นๆ

**การทำดัชนีบนพื้นฐานการแฮช**

วิธีการเดียวกันที่เพียร์ร่วมใช้กันก็คือการใช้ค่าจากฟังก์ชันแฮชเป็นหลัก และมีนโยบายการเลือกเพียร์เป้าหมายด้วยค่าแฮช เช่นระบบเลือกใช้ฟังก์ชันแฮชที่ให้ค่าแฮชของคำหลักของข้อมูลและค่าแฮชของเพียร์ไอดีเป็นเลขจำนวนเต็มขนาด 128 บิตและเลือกเพียร์เป้าหมายที่มีค่าแฮชของเพียร์ไอดีใกล้ (เชิงเลขจำนวน) กับค่าแฮชของคำหลักของข้อมูลมากที่สุด

ทุกข้อมูลบนระบบจึงต้องมีคำหลักซึ่งอาจไม่ได้สัมพันธ์ในแง่ความหมายกับข้อมูลก็ได้ (ซึ่งไม่แนะนำ) วิธีการเลือกคำหลักจึงขึ้นกับผู้พัฒนาแอพพลิเคชัน

**การแทรก ค้นคืนและลบข้อมูล**

ด้วยการเลือกเพียร์เป้าหมายสำหรับการแทรกและค้นคืนด้วยการใช้ฟังก์ชันแฮชนั้น เพียร์ที่จัดเก็บข้อมูลและเพียร์ที่ค้นคืนข้อมูลจำเป็นต้องรู้จักกลุ่มเพียร์เดียวกันเพื่อให้ได้ข้อมูลที่ถูกต้อง กรณีที่รู้จักกลุ่มเพียร์ต่างกันจะต้องมีการจัดเส้นทาง

ส่วนการลบจะแตกต่างออกไป เพราะเมื่อเราเผยแพร่ข้อมูลออกไป เราจะสูญเสียการควบคุมข้อมูลและยากที่จะลบข้อมูลได้หมด จึงใช้การกำหนดวันหมดอายุ (Expiration date) กับทุกข้อมูล และทุกเพียร์ก็ควรตรวจสอบข้อมูลที่ตนเป็นระยะเพื่อลบข้อมูลที่หมดอายุ

**การทำซ้ำข้อมูล**

บนเครือข่ายเพียร์ทูเพียร์ เพียร์ต่างๆ เข้าร่วม (Join) และออกจาก (Leave) เครือข่ายอย่างต่อเนื่อง บ่อยครั้งที่เพียร์ออกจากเครือข่ายโดยไม่มีการแจ้งล่วงหน้าหรืออันเนื่องจากความล้มเหลวของเครือข่าย ซึ่งจะทำให้ไม่สามารถเข้าถึงข้อมูลที่เพียร์นั้นรับผิดชอบ การทำซ้ำข้อมูล (Data replication) จึงเป็นสิ่งจำเป็น เพียร์ควรแทรกหรือค้นคืนข้อมูลไม่เพียงจากเฉพาะเพียร์เป้าหมายเพียร์เดียวเท่านั้น แต่ควรทำกับกลุ่มของเพียร์เป้าหมายรอง (ค่าแฮชของเพียร์ไอดีใกล้เชิงเลขจำนวนน้อยกว่าเมื่อเทียบกับค่าแฮชของคำหลักของข้อมูล) โดยที่จำนวนเพียร์เป้าหมายรองอาจคงตัวหรือพลวัตได้

**การจัดเส้นทางเมสเสจ**

เมื่อเครือข่ายเพียร์ทูเพียร์มีขนาดใหญ่เกินที่เพียร์หนึ่งๆ จะเก็บข้อมูลเพียร์ของเพียร์อื่นๆได้ทั้งหมด อีกทั้งเมื่อเพียร์ที่เพิ่งเข้าเครือข่ายจะยังไม่เป็นที่รู้จักในช่วงเวลาหนึ่ง การจัดเส้นทางเมสเสจจึงเป็นสิ่งจำเป็น โดยเพียร์ที่ได้รับเมสเสจและรู้จักเพียร์ที่เหมาะสมกว่าตน (มีเพียร์อื่นที่มีค่าแฮชใกล้เคียงกับค่าแฮชของคำหลักของข้อมูลมากกว่า) เพียร์ที่ได้รับเมสเสจก็จะจัดเส้นทางเมสเสจไปยังเพียร์ที่เหมาะสมกว่า กลไกนี้ช่วยให้เพียร์ในระบบสามารถแบ่งความรับผิดชอบข้อมูลเพียร์ที่มีในระบบได้

เพื่อหลีกเลี่ยงการเกิดการวนกลับของเมสเสจ (Message looping) เพียร์ควรแนบรายการของเพียร์ที่เคยส่ง ไปกับเมสเสจด้วย

**ข้อมูลเพียร์ท้องถิ่น**

เพื่อลดการจราจรในเครือข่าย การจัดเส้นทางเมสเสจควรเกิดขึ้นน้อยที่สุดเท่าที่เป็นไปได้ เพื่อลดการจราจรในเครือข่าย เพียร์จึงควรเชื่อมต่อเป็นกลุ่มเดียวกัน และมีวิธีการเก็บรวบรวม (Collect) ข้อมูลเพียร์ท้องถิ่นที่ดี จีไอเอสพีมีกฎการจัดเก็บดังนี้

- เพียร์ควรจัดเก็บข้อมูลเพียร์ให้มากที่สุดเพื่อลดการจัดเส้นทางเมสเสจ
- เพียร์ควรละทิ้งข้อมูลเพียร์ของเพียร์ที่ไม่สามารถติดต่อได้
- หากเพียร์ไม่สามารถเก็บข้อมูลเพียร์ทั้งหมดได้ ก็สามารถละทิ้งบ้างได้
- เพียร์ควรเก็บข้อมูลเพียร์ของเพียร์ที่มีค่าแฮชใกล้เคียงไว้ให้มาก และเก็บข้อมูลเพียร์ของเพียร์ที่มีค่าแฮชห่างไว้ให้น้อย
- เพียร์ควรเก็บข้อมูลเพียร์ของเพียร์ต่างๆ ในเครือข่ายที่สามารถติดต่อเพียร์อื่นๆ ได้ภายในจำนวนฮ็อปที่สมเหตุสมผล

จีไอเอสพีนำเอาอัลกอริทึมการจัดเส้นทางของ Chord [56] มาใช้และเพิ่มเติมตารางจัดเส้นทาง (Routing table) เพื่อเป็นแคชของข้อมูลเพียร์ การจัดเส้นทางจึงใช้จำนวนเมสเสจอย่างมากที่สุด $O(\log^2 N)$ และบางกรณีต้องการจำนวนเมสเสจมากกว่าเล็กน้อยสำหรับการแคชข้อมูล

**การเข้าร่วมและการออกของเพียร์**

เมื่อเพียร์ใหม่ต้องการเข้าร่วมเครือข่าย เพียร์จำเป็นต้องรู้จักเพียร์อื่นที่อยู่ในเครือข่ายแล้วอย่างน้อยหนึ่งเพียร์เพื่อขอข้อมูลเพียร์ของเพียร์อื่นๆ ที่อยู่ในเครือข่าย การที่จะรู้จักเพียร์แรกนี้ขึ้นกับการอิมพลีเมนต์

เมื่อเพียร์ต้องการออกจากเครือข่าย เพียร์ควรแจ้งเตือนเพียร์อื่นๆ ก่อน อย่างไรก็ตามบางเพียร์ก็อาจออกจากเครือข่ายเนื่องจากเครือข่ายมีปัญหา สำหรับจีไอเอสพีการแจ้งเตือนจึงไม่ใช่ข้อบังคับ และแม้จะมีหลายๆ เพียร์ออกพร้อมกันก็จะไม่มีข้อมูลสูญหายเพราะมีการทำซ้ำข้อมูลไว้

**ความต่างแบบกันของเพียร์**

เนื่องจากแต่ละเพียร์ต่างมีความสามารถไม่เท่ากัน เช่นมีเนื้อที่จัดเก็บข้อมูลได้มากน้อยต่างกัน ในข้อมูลเพียร์ของเพียร์จึงมีค่าความทนของเพียร์ (Peer strength) ซึ่งเป็นเลขจำนวนเต็มที่ใช้ในการคำนวณระยะห่าง (Distance) ของเพียร์ด้วย

ในการออกแบบของจีไอเอสพี ระยะห่างระหว่างเพียร์สองเพียร์คำนวณได้ด้วย distance(X,Y)/($2^{L-1}2^{M-1}$) โดยที่ X และ Y เป็นค่าแฮชของเพียร์ทั้งสอง L และ M เป็นค่าความทนของเพียร์ทั้งสอง

**ทอพอโลยีเครือข่ายซ้อนทับ**

จีไอเอสพีช่วยให้เกิดเครือข่ายซ้อนทับ (Overlay network) ที่มีทอพอโลยีต่างจากทอพอโลยีของเครือข่ายจริง เช่นหนึ่งฮอปในเครือข่ายจีไอเอสพีอาจเป็นอีกฝั่งโลกก็เป็นได้

เพื่อลดค่าใช้จ่ายบนเครือข่าย (Network cost) ที่เกิดจากการจัดเส้นทางเมสเสจ ทอพอโลยีของเครือข่ายซ้อนทับควรได้รับอิทธิพลจากทอพอโลยีของเครือข่ายจริง เพราะเพียร์จะมีการติดต่อกับเพียร์ที่มีค่าแฮชใกล้เคียงกันมากกว่าเพียร์ที่มีค่าแฮชห่าง แต่เนื่องจากเพียร์ไอดีนั้นไม่ถูกผูกมัดกับสิ่งใด เพียร์จึงอาจพิจารณาว่าเพียร์ที่อยู่ใกล้กันในเครือข่ายจริงคือเพียร์ที่มีค่าแฮชของเพียร์ไอดีใกล้กัน

จีไอเอสพีแก้ปัญหากรณีนี้ด้วยกลไกเข้าร่วมบนพื้นฐานเวลาแฝง (Latency-based join mechanism) กล่าวคือเมื่อเพียร์กำลังเข้าสู่เครือข่าย เพียร์จะมองหากลุ่มของข้อมูลเพียร์ในเครือข่าย ซึ่งข้อมูลเพียร์เหล่านี้จะมีเพียร์ไอดีและเวลาแฝง (Latency) เพียร์ใหม่นี้จะคำนวณเพียร์ไอดีของตนโดยอ้างอิงค่าเวลาแฝงที่ได้ ซึ่งคาดหวังว่าเพียร์ที่อยู่ใกล้ในเครือข่ายจริงจะมีค่าแฮชใกล้เคียงกัน

**เพียร์ไม่พึงประสงค์**

จีไอเอสพีเป็นโพรโทคอลเปิดที่เปิดให้ผู้พัฒนานำไปใช้สร้างระบบได้ ทำให้เปิดโอกาสที่จะมีเพียร์ไม่พึงประสงค์ (Undesirable peer) หรือเพียร์ที่มุ่งโจมตีเพียร์อื่นเกิดขึ้นได้ ตัวอย่างเพียร์ที่ไม่พึงประสงค์คือ

- เพียร์ที่รับเมสเสจ แต่ไม่เคยส่งเมสเสจเลย
- เพียร์ที่ส่งเมสเสจที่มีรูปแบบไม่ถูกต้อง
- เพียร์ที่ส่งเมสเสจเป็นจำนวนมากเกินและไม่หยุด

แต่ละเพียร์ควรป้องกันตนเองจากเพียร์ที่ไม่พึงประสงค์ ควรสามารถจัดการกับเมสเสจที่ไม่คาดหวังและละทิ้งได้อย่างถูกต้อง ถ้าเพียร์พบการถ่าโถมของเมสเสจ เพียร์ควรละทิ้งข้อมูลเพียร์ของเพียร์ต้นทางทิ้ง การทำซ้ำข้อมูลก็จำเป็นสำหรับการรับมือการโจมตี เนื่องจากเพียร์ที่มุ่งโจมตีสามารถประเมินคุณค่าข้อมูลได้ด้วยเมสเสจจากเพียร์อื่นๆ

**การค้นหาแบบคลุมเครือ**

ถ้าเราค้นหาข้อมูลเกี่ยวกับ "car" นั่นหมายถึงเราก็ต้องการข้อมูลเกี่ยวกับ "Car" และ "CAR" ด้วยเช่นกัน แต่เนื่องจากจีไอเอสพีทำงานบนพื้นฐานของฟังก์ชันแฮชซึ่งค่าแฮชไวต่ออักษรใหญ่เล็ก (Case sensitive) และไม่สามารถรู้ถึงความหมายที่เหมือนกันของคำหลักที่ต่างกัน ผู้พัฒนาแอพพลิเคชันจึงต้องเป็นผู้จัดการเอง เช่นแปลงคำหลักที่มีอักษรใหญ่ (Upper case) ให้เป็นอักษรเล็ก (Lower case) ทั้งหมดก่อนจัดเก็บหรือค้นคืน รวมถึงคำหลักที่เป็นวลี (Phrase) เช่น "white car" ก็ให้จัดเก็บหรือค้นคืนด้วยสองคู่ข้อมูลที่มีคำหลักเป็น white กับ car

### 2.2.3.2 ข้อกำหนดของจีไอเอสพี

จีไอเอสพีคือโพรโทคอลที่ใช้สร้างตารางแฮชแบบกระจาย ตารางแฮชแบบกระจายประกอบด้วยคู่ข้อมูล (key, value) ที่เพียร์ต่างๆ ใช้ร่วมกัน ยิ่งกว่านั้นโพรโทคอลนี้ก็อนุญาตให้ใช้อิลิเมนต์เอ็กซ์เอ็มแอล (XML) ใดๆ มาใช้ร่วมกัน และใช้เอ็กซ์พาธ (XPath) ในการค้นคืนอิลิเมนต์เอ็กซ์เอ็มแอล

ข้อกำหนดมีสองส่วนได้แก่ โพรโทคอลและการผูกเชื่อม (Binding) โพรโทคอลนิยามรูปแบบเมสเสจและพฤติกรรมของเพียร์ ส่วนการผูกเชื่อมนิยามวิธีการการส่งเมสเสจในเครือข่ายและพฤติกรรมของเพียร์เพิ่มเติม การผูกเชื่อมมีสองแบบคือแบบเอนพอยน์ตจักซ์ตา (JXTA-Endpoint) และแบบยูดีพี (UDP) ข้อกำหนดนี้เป็นข้อกำหนดจีไอเอสพีรุ่น (version) 3.4 beta4

ในที่นี้จะอธิบายโพรโทคอลจีไอเอสพีด้วยคำศัพท์ นิยามของระยะระหว่างเพียร์ไอดี รูปแบบเมสเสจ พฤติกรรมพื้นฐาน พฤติกรรมขั้นก้าวหน้า พฤติกรรมที่ปลอดภัย

**คำศัพท์**

| | |
|---|---|
| เพียร์ (Peer) | คอมโพเนนท์ (component) ที่เป็นไปตามข้อกำหนดนี้ |
| เมสเสจ (Message) | ข้อมูลที่เคลื่อนย้ายระหว่างสองเพียร์ใดๆ |
| แอดเดรส (Address) | แอดเดรสของเพียร์ที่บอกถึงตำแหน่งเครือข่ายเดียวของเพียร์ |
| เพียร์ไอดี (PID) | หมายเลขที่กำหนดให้แต่ละเพียร์และแต่ละไอเท็ม (item) |
| ไอเท็ม | อิลิเมนต์เอ็กซ์เอ็มแอลใดๆ ที่มีเพียร์ไอดี เวลาหมดอายุ และรายการแอดเดรสที่ส่งล่าสุด กรณีปกติจะเป็นคู่ |

| | |
|---|---|
| | ข้อมูล (key, value) เพียร์ไอดีคำนวณได้จากค่าฟังก์ชันแฮช SHA-1 [57] ของ key |
| เพียร์รับผิดชอบ (Responsible peer) | เพียร์ใดๆ ในจำนวน n เพียร์ที่ใกล้กับไอเท็มมากที่สุด โดย n คือจำนวนที่สามารถกำหนดได้ที่บ่งชี้ถึงจำนวนครั้งทำซ้ำสำหรับแต่ละไอเท็ม |
| เพียร์หลัก (Primary peer) | เพียร์ที่ต้องเป็นส่วนหนึ่งของตารางจัดเส้นทางเพียร์ของเพียร์ |

**นิยามของระยะระหว่างเพียร์ไอดี**

เพียร์และไอเท็มอยู่ใกล้กันถ้าระยะห่างระหว่างเพียร์ไอดีของเพียร์และเพียร์ไอดีของไอเท็มอยู่ใกล้กัน โดยระยะระหว่างเพียร์ไอดีนิยามไว้ดังนี้

distance(a, b) = unsigned_integer(a XOR b)

ตัวอย่างเช่น

distance(0000, 0000) = 0000 = 0

distance(0000, 0001) = 0001 = 1

distance(0001, 0000) = 0001 = 1

distance(0010, 0000) = 0010 = 2

distance(0010, 0001) = 0011 = 3

distance(0011, 0001) = 0010 = 2

distance(0011, 0010) = 0001 = 1

distance(1001, 0011) = 1010 = 10

**รูปแบบเมสเสจ**

สารอยู่ในรูปเอกสารเอ็กซ์เอ็มแอล มีโครงสร้างพื้นฐานดังนี้

| | |
|---|---|
| `<message>` | เมสเสจสำหรับบรรจุข้อมูลทั้งหมด |
| `<src>` | แอดเดรสของเพียร์ต้นทาง |
| `<seen>` | เมสเสจที่เคยได้รับ |
| `<peer>` | ข้อมูลเพียร์ |

<search> ค้นหาเพียร์

<insert> แทรกไอเท็ม

<query> ค้นคืนไอเท็ม

<result> ผลของการค้นคืน

ตัวอย่างของเมสเสจสำหรับการผูกเชื่อมแบบยูดีพีเป็นดังนี้

```
<!-- an example of a message begins here -->
<message xmlns=http://gisp.jxta.org/protocol/3.4/beta4
        id="10938092892">
    <src>192.168.1.1:9420</src>
    <seen message="0192809283" />
    <peer>
        <pid>112233ddeeff...</pid>
        <addr>192.168.1.2:9394</addr>
        <ttl>3600000</ttl>
        <sent>192.168.2.1:9870</sent>
        <sent>192.168.2.2:9781</sent>
    </peer>
    <search>
        <pid>223344bbddee...</pid>
        <limit>100</limit>
    </search>
    <insert>
        <pid>223344dd...</pid>
        <xml>
            <item @key="car">...</item>
        </xml>
        <ttl>1800000</ttl>
        <sent>192.168.2.3:8970</sent>
        <sent>192.168.2.4:7981</sent>
    </insert>
    <query id="29429482094">
        <pid>334455ff...</pid>
        <qstr type="xpath">/item[key='car']</qstr>
    </query>
    <result query="20394802958">
        <xml>
            <item @key="cow">...</item>
```

```
            </xml>
            <peer>
                  <pid>334455aabbff...</pid>
                  <addr>192.168.1.7:8394</addr>
                  <ttl>1200000</ttl>
                  <sent>192.168.2.8:9307</sent>
                  <sent>192.168.2.9:9821</sent>
            </peer>
      </result>
</message>
<!-- an example of a message ends here -->
```

โดยเค้าร่างเอ็กซ์เอ็มแอลแบบสมบูรณ์ (XML schema) อยู่ที่ภาคผนวก ก

สังเกตว่าเราสามารถเพิ่มเติมรูปแบบเมสเสจต่อได้ ตราบที่ป้ายระบุ (Tag) ระบุด้วยเนมสเปส (Namespace) ที่แตกต่าง

อธิบายแต่ละอิลิเมนต์ในเมสเสจอย่างสังเขปดังนี้

`<message>` คือหนึ่งอิลิเมนต์ซึ่งใช้ส่งระหว่างเพียร์ **ต้องการ** (Required) เนมสเปส ส่วนแอทริบิวต์ `id` เป็น**ตัวเลือก** (Optional)

`<src>` บรรจุหนึ่งแอดเดรสเพียร์ต้นทาง รูปแบบแอดเดรสขึ้นอยู่กับการเชื่อมต่อ

`<seen>` บรรจุหนึ่งเมสเสจไอดี (Message id) ที่เพียร์เคยได้รับ

`<peer>` เป็นอิลิเมนต์ที่บรรจุข้อมูลหนึ่งเพียร์

`<pid>` บรรจุหนึ่งเพียร์ไอดีของเพียร์ แทนด้วยข้อความเลขฐานสิบหกแบบไม่มีเครื่องหมาย (Unsigned hex string) อาจต่อท้ายด้วย "/" และบิตไม่สนใจ (Don't care)

ตัวอย่างเช่น

| | | |
|---|---|---|
| "1234" | -> | 0001001000111000 |
| "fabc" | -> | 1111101010111100 |
| "4507/00f0" | -> | 01000101****0111 |
| "ac34/03c0" | -> | 101011****110100 |

`<addr>` บรรจุหนึ่งแอดเดรสของเพียร์

`<ttl>` บรรจุหนึ่งค่าจำนวนเต็มที่บ่งบอกเวลาที่เพียร์จะดำรงอยู่ หน่วยเป็นมิลลิวินาที

<sent> บรรจุหนึ่งแอดเดรสของเพียร์ที่เคยส่งข้อมูลไปแล้ว

<limit> บรรจุหนึ่งค่าจำนวนเต็มที่จำกัดจำนวนผลลัพธ์ของการค้นหา

<insert> หนึ่งอิลิเมนต์สำหรับการแทรกไอเท็ม

<xml> บรรจุอิลิเมนต์เอ็กซ์เอ็มแอลของหนึ่งไอเท็ม

<query> หนึ่งอิลิเมนต์สำหรับการค้นคืนไอเท็ม **ต้องการ**แอตทริบิวต์ id

<qstr> บรรจุสตริงของการค้นคืน ปกติเป็น XPath

<result> บรรจุหนึ่งผลลัพธ์การค้นคืน **ต้องการ**แอตทริบิวต์ query ที่บอกการค้นคืน <peer> ใน <result> คือหนึ่งเพียร์อื่นที่ได้ส่งการค้นคืน

**พฤติกรรมพื้นฐาน**

- การตื่น (Bootstrapping) ของเพียร์

เพียร์หนึ่งๆ จำเป็นต้องได้หนึ่งซีดแอดเดรส (Seed address) หรือมากกว่าจากภายนอก วิธีได้ซีดแอดเดรสขึ้นอยู่กับการผูกเชื่อม เมื่อเพียร์ได้ซีดแอดเดรสแล้ว เพียร์**ควร**ส่งเมสเสจที่**ต้องการ**อิลิเมนต์ <src> และ**ต้องการ**อิลิเมนต์ <search> เพียร์คาดหวังที่จะได้เมสเสจที่มีอิลิเมนต์ <peer> กลับมา ถ้าเพียร์ได้รับอิลิเมนต์ <peer> เพียงพอแล้ว มัน**ไม่ควร**ใช้ซีดแอดเดรสในการขออิลิเมนต์ <peer> เพิ่ม

- การบำรุง (Maintaining) ตารางจัดเส้นทางเพียร์ (Peer routing table)

เพียร์หนึ่งๆ **ต้อง** (Must) มีหนึ่งตารางจัดเส้นทางเพียร์ ซึ่งเก็บข้อมูลของกลุ่มของแอ็กทิฟเพียร์ (Active peer) ในเครือข่าย

เพียร์**ควร**เก็บข้อมูลเพียร์อื่นๆ ลงแคชให้มากที่สุดเท่าที่จะทำได้

เพียร์ (PID=pid1) **ต้อง**เก็บข้อมูลของกลุ่มเพียร์หลัก

เพียร์ (PID=pid2) เป็นเพียร์หลัก ถ้า $2^{(i-1)}$ <= distance(pid1,pid2) และไม่มีเพียร์อื่นที่มีเพียร์ไอดีอยู่ระหว่าง $2^{(i-1)}$ และ distance(pid1,pid2) โดย i =1, 2, ...

เพียร์**อาจ**ลบข้อมูลของเพียร์ที่ไม่ใช่เพียร์หลัก (Non-primary peer) ได้

ถ้าเพียร์ได้รับเมสเสจที่มีอิลิเมนต์ <peer> เพียร์**ควร**แทรกข้อมูลเพียร์ลงในตารางจัดเส้นทางเพียร์ของมัน

เพียร์**ควร**ส่งเมสเสจที่**ต้องการ**อิลิเมนต์ <search> ที่มีอิลิเมนต์ <pid> เป็น**ตัวเลือก** และมีอิลิเมนต์ <limit> เป็น**ตัวเลือก** เพื่อขอข้อมูลเพียร์เพิ่ม

ถ้าเพียร์ได้รับเมสเสจที่มีอิลิเมนต์ <search> เพียร์**ต้อง**ส่งเมสเสจที่มีอิลิเมนต์ <peer> ที่อยู่ในตารางจัดเส้นทางเพียร์ของมันกลับไป จำนวนข้อมูลเพียร์ที่ส่งกลับ**ต้องไม่**เกินค่าจำกัด ถ้ามีการระบุมาในอิลิเมนต์ <limit> ถ้าอิลิเมนต์ <limit> และอิลิเมนต์ <pid> ถูกระบุ ข้อมูล <peer> ที่ส่งกลับ**ต้อง**ใกล้เคียงกับ <pid> มากกว่าทุกเพียร์อื่นที่ไม่ส่งข้อมูลกลับ

เพียร์**ควร**ลบเพียร์ที่หมดอายุจากตารางจัดเส้นทางเพียร์ของมัน เวลาดำรงอยู่ในหน่วยมิลลิวินาทีซึ่งอยู่ในอิลิเมนต์ <ttl> ในอิลิเมนต์ <peer> เพียร์**อาจ**ลบเพียร์ที่ไม่หมดอายุได้ก็ต่อเมื่อมันเป็นเพียร์ที่ไม่ใช่เพียร์หลัก

เพียร์**ต้อง**ส่งเมสเสจที่มีอิลิเมนต์ <peer> ที่มีข้อมูลของตนไปยังเพียร์หลักทั้งหมดที่มีในตารางจัดเส้นทางเพียร์เป็นระยะๆ เพื่อให้เพียร์หลักเหล่านั้นรู้ว่าเพียร์ยังดำรงอยู่ เพียร**ควร**ส่งเมสเสจก่อนข้อมูลที่ส่งก่อนหน้าจะหมดอายุลง

- การแทรกและบำรุงไอเท็ม

เพียร์**ต้อง**มีฐานข้อมูลท้องถิ่น (Local database) สำหรับเก็บไอเท็ม

ถ้าเพียร์รับเมสเสจที่มีอิลิเมนต์ <insert> เพียร์**ต้อง**เพิ่มข้อมูลไอเท็มในอิลิเมนต์ <insert> ลงฐานข้อมูลท้องถิ่นของตน

ถ้าโปรแกรมประยุกต์แทรกไอเท็มผ่านเอพีไอ (API) เพียร์**ต้อง**เพิ่มข้อมูลไอเท็มลงในฐานข้อมูลท้องถิ่น

เพียร์จะหมั่นตรวจสอบฐานข้อมูลเฉพาะที่และปฏิบัติดังต่อไปนี้

- เพียร์**ต้อง**ลบไอเท็มที่หมดอายุ
- ถ้าเพียร์รับผิดชอบที่อยู่ระยะไกล (Remote responsible peer) ของไอเท็มบางเพียร์ไม่อยู่ในข้อมูลแอดเดรส sent เพียร์**ต้อง**ส่งเมสเสจที่มีอิลิเมนต์ <insert> เพื่อแทรกไอเท็มและปรับ (Update) ข้อมูลแอดเดรส sent
- ถ้าเพียร์ระยะไกลออกจากเครือข่ายหรือเกิดความผิดผลาด เพียร์**ต้อง**ลบแอดเดรสของเพียร์ออกจากข้อมูลแอดเดรส sent
- ถ้าเพียร์ไม่รับผิดชอบไอเท็มได้อีกต่อไป เพียร์**อาจ**ลบข้อมูลไอเท็มจากฐานข้อมูลท้องถิ่น

- การค้นคืนไอเท็ม

ถ้าเพียร์ได้รับเมสเสจที่มีอิลิเมนต์ <query> เพียร์**ต้อง**ส่งเมสเสจที่มีอิลิเมนต์ <result> ที่**ต้องการ**แอทริบิวต์ query ซึ่งมี id ของการค้นคืนกลับไปยังเพียร์ที่มีแอดเดรสตรงกับอิลิเมนต์ <src> ในเมสเสจที่ได้รับ อิลิเมนต์ <result> **ต้อง**บรรจุอิลิเมนต์ <xml> ซึ่งแต่ละอิลิเมนต์บรรจุผลลัพธ์ของการค้นคืน XPath อิลิเมนต์ <result> **ต้อง**บรรจุเพียร์ที่ใกล้กับเพียร์ไอดีของการค้นคืนมากที่สุดจำนวน n เพียร์ลงในอิลิเมนต์ <peer> จำนวน n อิลิเมนต์ด้วย โดยที่ n คือจำนวนที่ปรับเปลี่ยน (Configurable) ได้ บ่งชี้จำนวนเพียร์ที่ใช้ในการค้นคืนแบบรีเคอร์ซีฟ

ถ้าแอพพลิเคชันค้นคืนไอเท็มผ่านเอพีไอ เพียร์**ต้อง**ส่งคืนผลลัพธ์การค้นคืนที่ได้จากฐานข้อมูลท้องถิ่น และ**ต้อง**ส่งเมสเสจที่**ต้องการ**อิลิเมนต์ <src> และ**ต้องการ**อิลิเมนต์ <query> ที่**ต้องการ**แอตทริบิวต์ id ซึ่งเป็นสตริงอันไม่ซ้ำใคร ไปยังเพียร์จำนวน n เพียร์ที่อยู่ใกล้ที่สุดจากคีย์ของการค้นคืน โดย n คือจำนวนที่ปรับเปลี่ยนได้ บ่งชี้จำนวนเพียร์ที่ใช้ทำซ้ำการค้นคืน

ถ้าเพียร์ได้รับเมสเสจที่มีอิลิเมนต์ <result> ซึ่งมีแอตทริบิวต์ query ตรงกับ id ของการค้นคืนแล้ว เพียร์**ต้อง**ส่งข้อมูล <xml> กลับ และ**อาจ**ส่งเมสเสจที่มีอิลิเมนต์ <query> ไปยังทุกเพียร์ในอิลิเมนต์ <peer> ในอิลิเมนต์ <result> ถ้าเพียร์นั้นอยู่ใกล้กว่าเพียร์อื่นๆ ที่อยู่ในรายการเพียร์เคยส่งในเมสเสจ

- ข้อตกลงของการร่วมใช้คู่ (key, value)

รูปแบบเมสเสจไม่ได้มีไวยากรณ์พิเศษใดๆ ในการร่วมใช้คู่ (key, value)

คู่ (key, value) แสดงด้วย <item key="...">...<item> และเนมสเปซที่ตรงกับโพรโทคอล

เพียร์ไอดีของไอเท็ม คือค่าแฮชแบบ SHA-1 ของค่าคีย์

การค้นคืน XPath โดยคีย์จะแทนด้วย /*[local-name()='item' and @key='...']

- การส่งเมสเสจ

เพียร์**ควร**ส่งเมสเสจที่มีหลายๆ อิลิเมนต์ เพื่อช่วยให้จำนวนเมสเสจลดลง แม้ว่าจะทำให้ตัวเมสเสจเองมีขนาดใหญ่ก็ตาม วิธีคือเพียร์**ควร**มีคิวส่งที่แต่ละส่วนของเมสเสจจะถูกนำมาเก็บไว้ก่อนที่จะถูกส่งไปเป็นหนึ่งเมสเสจใหญ่เป็นระยะๆ ช่วย

ลดค่าใช้จ่าย (Overhead) ในการเข้ารหัสเอ็กซ์เอ็มแอล (XML Encoding) และลดค่าใช้จ่ายการจัดเส้นทางการถ่ายโอน (Transportation Routing Overhead)

เมื่อมีความจำเป็น เพียร์**ควร**กำหนดแอตทริบิวต์ `id` ในแต่ละเมสเสจและ**ต้องการ**อิลิเมนต์ `<src>` อันจะช่วยให้เราสามารถรู้ได้ว่าเมสเสจส่งไปถึงหรือไม่ เพียร์ที่ได้รับเมสเสจที่มีแอทริบิวต์ `id` **ต้อง**ส่งเมสเสจที่มีอิลิเมนต์ `<seen>` ที่ระบุ `id` ไปยังเพียร์ที่มีแอดเดรสอยู่ในอิลิเมนต์ `<src>` ถ้าเพียร์ต้นทางไม่ได้รับเมสเสจที่มีอิลิเมนต์ `<seen>` ที่มี `id` ตรงกับที่เพียร์ต้นทางกำหนดไว้ มัน**ควร**บันทึกความล้มเหลวไว้ และหากความล้มเหลวเกิดขึ้นต่อเนื่อง เพียร์**ควร**ลบข้อมูลเพียร์ปลายทางนั้นออกจากตารางจัดเส้นทางเพียร์

**พฤติกรรมขั้นก้าวหน้า**

เพียร์**อาจ**ทำงานในโหมดไคลเอนต์ (Client mode) ซึ่งเพียร์จะไม่ต้องรับผิดชอบกับส่วนใดๆในตารางแฮช โหมดไคลเอ็นต์**ควร**อนุญาตให้ใช้ได้ก็ต่อเมื่อมีข้อจำกัดเช่น มีแบนด์วิดท์ต่ำและทำงานเพียงระยะเวลาสั้นๆ ในโหมดไคลเอ็นต์เพียร์ไม่ต้องส่งข้อมูลของตนเองให้แก่เพียร์อื่นๆ แต่มัน**อาจ**ตอบรับการค้นหาเพียร์และการค้นคืนไอเท็มจากเพียร์อื่นๆ ก็ได้

เพียร์**อาจ**เปลี่ยนโหมดของตนจากโหมดไคลเอ็นต์ไปสู่โหมดปกติ จริงแล้ว **แนะนำ**ให้เริ่มต้นรันในโหมดไคลเอ็นต์ก่อน จากนั้นระยะเวลาหนึ่งจึงเปลี่ยนไปสู่โหมดปกติ วิธีนี้ช่วยกรองเพียร์ที่รันระยะเวลาสั้นๆ ออกจากการเป็นส่วนหนึ่งของตารางแฮช คุณสมบัติเด่นอื่นอันเป็น**ทางเลือก**คือการใช้เพียร์ไอดีซึ่งไวต่อเครือข่าย (Network-sensitive peer PID) ซึ่งคำนวณได้จากเพียร์ไอดีของเพียร์ต่างๆ ในเครือข่าย กล่าวคือเพียร์ไอดีของเพียร์จะใกล้เคียงกับเพียร์ไอดีของเพียร์ที่อยู่ใกล้กันในเครือข่าย ทอพอโลยีซ้อนทับจะยิ่งคล้ายคลึงกับทอพอโลยีเครือข่าย

**พฤติกรรมความปลอดภัย**

เพื่อความปลอดภัยของทั้งระบบแล้ว แต่ละเพียร์**ควร**รับมือกับการโจมตีที่อาจเกิดขึ้นได้ เพียร์**ควร**ตรวจจับเมสเสจที่ไม่พึงประสงค์ ซึ่งมีลักษณะตรงกับบางกรณีหรือทุกกรณีต่อไปนี้

- เมสเสจที่มีรูปแบบไม่ถูกต้อง
- เมสเสจที่มีอิลิเมนต์ `<peer>` หรือ `<insert>` ที่มีเวลาดำรงอยู่มากๆ
- เมสเสจหรืออิลิเมนต์ของมันที่เคยได้รับแล้วเมื่อเวลาไม่นาน

ทั้งหมดหรือบางส่วนของเมสเสจที่ไม่พึงประสงค์นี้**ควร**ถูกละทิ้ง เพียร์**อาจ**บันทึกเมสเสจที่ไม่พึงประสงค์หรือข้อมูลของมันไว้เพื่อจุดประสงค์แง่ความปลอดภัยในอนาคตได้

เพียร์**ควร**ตรวจจับเพียร์ที่ไม่พึงประสงค์ด้วย เช่น

- เพียร์ที่ไม่เคยตอบสนอง
- เพียร์ที่ตอบสนองยาก
- เพียร์ที่ส่งเมสเสจที่ไม่พึงประสงค์
- เพียร์ที่ส่งเมสเสจมากเกิน

เพียร์ที่ไม่พึงประสงค์**ควร**ถูกลบออกจากตารางจัดเส้นทางเพียร์ และ**ไม่ควร**ถูกรวมเข้าไปกับเมสเสจขาออก เพียร์**อาจ**บันทึกข้อมูลเพียร์ที่ไม่พึงประสงค์ไว้เพื่อจุดประสงค์แง่ความปลอดภัยในอนาคตได้

**การผูกเชื่อมแบบเอนพอยต์นจักซ์ตา**

- รูปแบบแอดเดรส

  `jxta://<JXTA-ID>/<service-name>/<jxta group>`

  ตัวอย่างเช่น `jxta://uuid-59616261646162614A78746150325033ADBC20EA9C84461B4EDB50A87C3693A03/GISP-3.4/jxta-NetGroup`

- การตื่นของเพียร์

  เพียร์**ควร**ประกาศแอดเวอทิสเมนต์แบบ JXTA ที่บรรจุข้อมูลแอดเดรสของตนไว้ ซึ่งเราเรียกว่าซีดแอดเดรส เพียร์ที่กำลังตื่น (boot) โดยไม่มีข้อมูลเพียร์อื่นๆ **ควร**ค้นหาแอดเวอทิสเมนต์แบบจักซ์ตาที่มีซีดแอดเดรสโดยใช้บริการค้นคืนจักซ์ตา (JXTA Discovery Service) **ไม่แนะนำ**ให้ทุกเพียร์ในเครือข่ายประกาศแอดเวอทิสเมนต์ เนื่องจากจะทำให้เกิดปัญหาด้านความสามารถขยายตัว (Scalability)

- การส่งถ่ายเมสเสจ (Message Transport)

  บริการเอนพอยน์ตจักซ์ตา (JXTA Endpoint Service) จะถูกใช้ในการส่งถ่ายเมสเสจ เมสเสจจะถูกจัดเก็บรวมเป็นหนึ่งอิลิเมนต์ที่มีชื่อว่า GISP-3.4 ลงในเมสเสจเอนพอยน์ตจักซ์ตา (JXTA Endpoint Message)

**การผูกเชื่อมแบบยูดีพี**

- รูปแบบแอดเดรส

  `<ipaddress or hostname>:<port number>`

  ตัวอย่างเช่น 192.168.12.34:5678 หรือ jxtafoo.dyndns.org:7890

- การตื่นของเพียร์

เพียร์**ควร**รับซีดแอดเดรสจากภายนอก เช่นจากพารามิเตอร์ของคอมมานด์ไลน์หรือโฮสต์ไฟล์หรือเซิร์ฟเวอร์ศูนย์กลาง

- การส่งถ่ายเมสเสจ

เมสเสจคือแพ็คเกจแบบยูดีพีรุ่นของเมสเสจควรถูกตรวจสอบที่เนมสเปสของเอกสาร `<message>`