Enterprise Application Integration Using Publish/Subscribe REST With IP Multicast

ฟาร์เดล เจ๊ะเต็ง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต (วิศวกรรมซอฟต์แวร์)
คณะสถิติประยุกต์
สถาบันบัณฑิตพัฒนบริหารศาสตร์
2558

Enterprise Application Integration Using Publish/Subscribe REST With IP Multicast

ฟาร์เดล เจ๊ะเต๊ง

คณะสถิติประยุกต์

ผู้ช่วยศาสตราจารย์.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(ดร.รัฐกร พูลทรัพย์)

คณะกรรมการสอบวิทยานิพนธ์ ได้พิจารณาแล้วเห็นสมควรอนุมัติให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรมหาบัณฑิต (วิศวกรรมซอฟต์แวร์)

ผู้ช่วยศาสตราจารย์.....ประธานกรรมการ

(ดร.สุเทพ ทองงาม)

ผู้ช่วยศาสตราจารย์.....กรรมการ

(ดร.รัฐกร พูลทรัพย์)

ผู้ช่วยศาสตราจารย์.....กรรมการ

(ดร.ปราโมทย์ กั่วเจริญ)

ผู้ช่วยศาสตราจารย์.....กรรมการ

(ดร.ธนิศา นุ่มนนท์)

อาจารย์.....คณบดี

(ดร.ศิวิกา ดุษฎีโหนด)

กันยายน 2558

บทคัดย่อ

ชื่อวิทยานิพนธ์ Enterprise Application Integration Using Publish/Subscribe REST With IP Multicast

ชื่อผู้เขียน นายฟาร์เดล เจ๊ะเต็ง

ชื่อปริญญา วิทยาศาสตร์มหาบัณฑิต (วิศวกรรมซอฟต์แวร์)

ปีการศึกษา 2558

เว็บเซอร์วิสเป็นเทคโนโลยีที่ถูกกำหนดมาตรฐานโดยองค์กร Word Wide Web Consortium หรือ W3C โดยมีจุดประสงค์เพื่อให้ซอฟต์แวร์สามารถแลกเปลี่ยนข้อมูลระหว่างกันโดยไม่ขึ้นอยู่กับภาษา เทคโนโลยีและแฟลตฟอร์ม ปัจจุบันถูกนำมาใช้งานอย่างแพร่หลายในการบูรณการโปรแกรมประยุกต์ต่างๆ ที่มีอยู่องค์กรให้สามารถสื่อสารแลกเปลี่ยนข้อมูลระหว่างกันได้ แม้ว่าโปรแกรมประยุกต์เหล่านั้นจะมีเทคโนโลยี แฟลตฟอร์มที่แตกต่างกันก็ตาม แต่เว็บเซอร์วิสมีข้อจำกัดประการสำคัญคือไม่ทราบสถานะล่าสุดของข้อมูลทำให้โปรแกรมที่เชื่อมต่ออยู่กันนั้นต้องคอยดึงข้อมูลเป็นระยะหรือตามช่วงเวลาเพื่อนำข้อมูลมาตรวจสอบเป็นผลให้มีการใช้งานเครือข่าย (Network) เกินความจำเป็นและยังอาจก่อให้เกิดปัญหาคอขวด (Bottle Neck) หรือส่งผลต่อเวลาตอบสนองโดยรวม (Response Time) ทำให้เครือข่ายทำงานช้าลงได้ จึงมีการศึกษาวิจัยเพื่อขจัดปัญหาดังกล่าวแต่ยังคงพบการใช้งานเครือข่ายที่เกินความจำเป็น ดังนั้นวิทยานิพนธ์ฉบับนี้จึงมีแนวคิดในการใช้งานเครือข่ายให้เกิดประสิทธิภาพสูงสุดโดยประยุกต์วิธีการ Publish/Subscribe ร่วมกับเว็บเซอร์วิสแบบ REST ส่งข้อมูลแบบ มัลติคาสต์ (Multicast) ซึ่งเป็นการส่งข้อมูลเพียงชุดเดียวและครั้งเดียวเข้าไปยังเครือข่ายและโปรแกรมประยุกต์ที่เกี่ยวข้องจะได้รับข้อมูลพร้อมๆ กัน ทั้งนี้ยังสามารถนำเอาไปประยุกต์ใช้ในการแลกเปลี่ยนข้อมูลบนอุปกรณ์เคลื่อนที่ (Mobile Device) ที่ได้รับความนิยมในการใช้งานในปัจจุบันได้

ABSTRACT

Title of Thesis	Enterprise Application Integration Using Publish/Subscribe REST With IP Multicast
Author	Mr.Fadel Cheteng
Degree	Master of Science (Software Engineering)
Year	2015

Web Services is a technology standardized by World Wide Web Consortium or W3C with the purpose of enabling software the ability to coordinate with one another regardless of the programming languages, technologies and platforms. Web Services is widely used in the integration of various applications and platforms in each organization so that the common data can be communicated and transferred to the silo applications on any platform. However, Web Services has significant limitations; which is its inability to indicate the latest state of the common data across applications. In order to do so, the connected application programs have to constantly retrieve the common data. This procedure not only overuses the network and causes a bottle neck issue, but it can also generally affect the response time, further causing the delay of the entire network. There were many other researches that try to resolve these problems, but were unable to resolve almost all of the aforementioned problems. Therefore, this research paper will be focusing on the network usage's most efficient method, which is to use public/subscribe Web Services on REST lightweight protocol along with the multicast data transmission that transfers a single packet of data only once to the networks so that all of the connected silo's applications will receive it simultaneously. The solution can be applied with mobile devices that maintain common data across the mobile devices.

กิตติกรรมประกาศ

วิทยานิพนธ์เรื่อง Enterprise Application Integration Using Publish/Subscribe REST With IP Multicast สำเร็จลุล่วงไปด้วยการสนับสนุนและความช่วยเหลือจากบุคคลหลายท่าน ที่กรุณาสละเวลาอันมีค่าให้คำเสนอแนะ ปรึกษาและให้กำลังใจเป็นอย่างดีมา โดยตลอดผู้วิจัยขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.รัฐกร พูลทรัพย์ อาจารย์ที่ปรึกษาและกรรมการวิทยานิพนธ์สำหรับความเมตตากรุณาในการประสิทธิประสาทความรู้ที่มีคุณค่า ความช่วยเหลือด้านต่างๆ รวมถึงคำแนะนำเป็นประโยชน์ ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.ปราโมทย์ กั่วเจริญ ผู้ช่วยศาสตราจารย์ ดร.ธนิศา นุ่มนนท์และผู้ช่วยศาสตราจารย์ ดร.สุเทพ ทองงาม ที่ได้สละเวลาอันมีค่าและภารกิจงานวิจัยมากมายมาเป็นกรรมการวิทยานิพนธ์

ขอขอบพระคุณคณาจารย์คณะสถิติประยุกต์ สถาบันบัณฑิตพัฒนบริหารศาสตร์ทุกท่านที่ให้ความเมตตาสั่งสอน และถ่ายทอดความรู้ให้แก่ผู้วิจัย รวมไปถึงคณาจารย์ที่ถ่ายทอดความรู้ความสามารถตั้งแต่ประถม มัธยม และปริญญาตรี ขอขอบพระคุณทางผู้บริหารบริษัทที่ผู้เขียนได้ทำงาน ที่มีเมตตา ส่งเสริม ให้อิสระและ เข้าใจถึงความยากลำบากในการทำงานและศึกษาไปพร้อมกันสุดท้ายนี้ขอขอบคุณกำลังใจจากครอบครัวที่เป็นกำลังใจสำคัญในยามท้อแท้

ด้วยการอนุมัติจากพระผู้เป็นเจ้าทำให้วิทยานิพนธ์นี้เสร็จสมบูรณ์

ฟาร์เดล เจ๊ะเต็ง

พฤษภาคม 2558

สารบัญ

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ **หน้า**

บทที่ 1

บทนำ

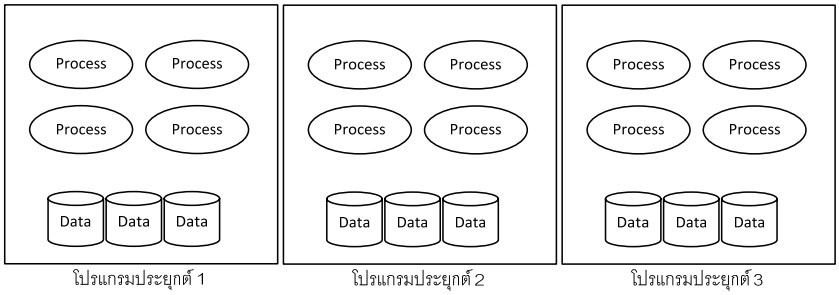
1.1 ความสำคัญของปัญหา

วิวัฒนาการเทคโนโลยีทางด้านฮาร์ดแวร์ (Hardware) และซอฟต์แวร์ (Software) ที่ถูกนำมาใช้งานในระดับองค์กรสามารถแบ่งออกเป็น 4 ช่วงยุคเวลา (Zuehlk, 2011: 1) คือ ยุคกำเนิดของเครื่องคอมพิวเตอร์สมัยใหม่ (Origin of Modern Computing) ยุคมินิคอมพิวเตอร์และไมโครคอมพิวเตอร์(Minicomputer and Microcomputer) ยุคการประมวลผลแบบกระจาย (Distributed System) และยุคปัจจุบันซึ่งเป็นยุคอินเตอร์เน็ตและเว็บ (Internet and World Wide Web : Web) โดยยุคกำเนิดของเครื่องคอมพิวเตอร์สมัยใหม่สามารถย้อนกลับไปในทศวรรษที่ 1700 เริ่มจากการคิดค้นบัตรเจาะรู (Punched Card) เพื่อเก็บข้อมูลที่ต้องการและการใช้งานในระดับองค์กรเริ่มขึ้นในทศวรรษที่ 1890 โดยสำนักสำรวจสำมะโนประชากรของสหรัฐอเมริกา (U.S. Census Bureau) นำบัตรเจาะรูมาประมวลผลข้อมูลเพื่อสำรวจประชากรของรัฐ หลังจากนั้นในทศวรรษที่ 1950 ได้กำเนิดเครื่องเมนเฟรมขึ้นมาโดยเครื่อง Manchester Mark 1 เป็นเครื่องแรกที่ถูกนำมาใช้งานในเชิงพาณิชย์ ซึ่งเครื่องเมนเฟรมในยุคแรกมีข้อจำกัดในการใช้งาน อันเนื่องจากไม่มีระบบปฏิบัติการที่สามารถทำงานได้ทีละโปรแกรมเท่านั้นและมีราคาแพง ต่อมาในตอนต้นทศวรรษ 1960 บริษัทไอบีเอ็ม (IBM) ได้พัฒนาระบบปฏิบัติการและภาษาโคบอล (Cobol) ขึ้นมาและเกิดภาษาสำหรับพัฒนาโปรแกรมขึ้นมาหลายภาษาในช่วงเวลาดังกล่าว เช่น Fortran, CPL, ALGOL 80 และ Basic เป็นต้น รวมทั้งมีการพัฒนาเครื่องซุปเปอร์คอมพิวเตอร์ (Supercomputer) ขึ้นมา ซึ่งถูกใช้ในงานวิจัยทางวิทยาศาสตร์และทางการทหารของหน่วยงานในภาครัฐเป็นหลัก

ต่อมาเป็นยุคของเครื่องมินิคอมพิวเตอร์และไมโครคอมพิวเตอร์โดยยุคนี้เกิดจากการพัฒนาแผงวงจรรวม (Integrated Circuit หรือ IC) ขึ้นมาใช้งานแทนหลอดสุญญากาศและทรานซิสเตอร์แบบเก่าทำให้เครื่องคอมพิวเตอร์มีขนาดเล็กลงมากจากที่ต้องใช้พื้นที่ห้องขนาดใหญ่เปลี่ยนเป็นสามารถนำมาวางบนโต๊ะสำนักงานได้ และยังมีความเร็วในการประมวลผลที่ดีกว่า แต่ยังคงมีราคา

แพงทำให้มีการใช้งานในองค์กรที่มีขนาดใหญ่เท่านั้นโดยเครื่องมินิคอมพิวเตอร์ถูกนำมาใช้เป็นเครื่องคอมพิวเตอร์ช่วยออกแบบ (Computer-aided Design หรือ CAD) ในภาคอุตสาหกรรมและ ในระยะเวลาต่อมามีการพัฒนาสารกึ่งตัวนำ (Semiconductor) ทำให้มีการผลิตหน่วยประมวลผลกลาง (Central Processing Unit หรือ CPU) ชนิด Microprocessor ขึ้นมาจนเกิดเครื่องไมโครคอมพิวเตอร์หรือคอมพิวเตอร์ส่วนบุคคล (Personal Computer หรือ PC) ซึ่งมีราคาถูกเป็นผลให้ได้รับความนิยมในการใช้งานทั้งในส่วนขององค์กรและผู้ใช้งานทั่วๆ ไป โดยในช่วงเวลานี้ได้มีการพัฒนาระบบปฏิบัติการภาษาและเทคโนโลยีในการพัฒนาโปรแกรม ระบบเครือข่ายที่ดีขึ้นจึงเกิดแนวแนวความคิดในการประมวลผลและใช้งานข้อมูลร่วมกัน จึงเกิดเป็นยุคประมวลผลแบบกระจาย กล่าวคือ เครื่องลูกข่าย (Client) จะทำการส่งข้อมูลเพื่อให้เครื่องแม่ขาย (Host/Server) ที่มีประสิทธิภาพมากกว่าประมวลผลข้อมูลให้ และในยุคนี้บริษัทผลิตซอฟต์แวร์ต่างๆ ได้จัดทำซอฟต์แวร์สำเร็จรูปขึ้นมา อาทิเช่น ซอฟต์แวร์บริหารการผลิต (Manufacturing Resource Planning หรือ MRP) ซอฟต์แวร์บริหารจัดการทรัพยากร (Enterprise Resource Planning หรือ ERP) หรือซอฟต์แวร์บริหารงานบุคคล (Human Resource หรือ HR) เป็นต้น เพื่อตอบสนองการดำเนินงานขององค์กร และในทศวรรษ 1960 มีการวิจัยเครือข่ายเพื่อใช้ในกระทรวงกลาโหมประเทศสหรัฐอเมริกาชื่อว่า ARPANET ซึ่งต่อมาได้กลายเป็นเครือข่ายอินเตอร์เน็ตที่ได้ความนิยมในการใช้งานและทำให้เกิดยุคต่อมา คือ ยุคของอินเตอร์เน็ตและเว็บ ส่งผลให้รูปแบบของโปรแกรมมีการเปลี่ยนแปลงโดยเกิดรูปแบบในการดำเนินธุรกิจขึ้นมา 2 แบบคือ ผู้ให้บริการโปรแกรมประยุกต์ (Application Service Provider หรือ ASPs) และซอฟต์แวร์เชิงบริการ (Software as a Services หรือ SaaS)

จากความเป็นมาที่ได้กล่าวในข้างต้นองค์กรต่างๆ นำเอาโปรแกรมประยุกต์มาใช้งานเพื่อสร้างสารสนเทศตามความต้องการของแต่ละส่วนงาน โดยโปรแกรมประยุกต์ (Application Program) เหล่านี้มีลักษณะการทำงานที่เป็นเอกเทศ ถูกพัฒนาด้วยภาษาเทคโนโลยีและฮาร์ดแวร์ที่แตกต่างกัน ไม่มีการสื่อสารหรือแลกเปลี่ยนข้อมูลระหว่างกัน ทำให้ข้อมูลและกระบวนการทางธุรกิจถูกจำกัดอยู่ในโปรแกรมประยุกต์ของตนเอง ดังภาพที่ 1.1 แต่ในปัจจุบันความต้องการสารสนเทศเพิ่มมากขึ้นอันเนื่องจากเป็นส่วนสำคัญในการดำเนินงานขององค์กร และรูปแบบการดำเนินงานเชิงอิเลคทรอนิกส์ (Electronic Business หรือ e-Business) ได้เข้ามามีบทบาททำให้เกิดรูปแบบการดำเนินธุรกิจใหม่ๆ เช่น การค้าระหว่างผู้ประกอบการกับผู้ประกอบการ (Business to Business หรือ B2B) การค้าระหว่างผู้ประกอบการกับลูกค้า (Business to Customer หรือ B2C) เป็นต้น ซึ่งต้องใช้สารสนเทศและกระบวนการทางธุรกิจระหว่างกันทำให้การบูรณการโปรแกรมประยุกต์ที่มีอยู่ในองค์กรกลายเป็นประเด็นสำคัญและมีความจำเป็นยิ่งขึ้นในปัจจุบัน



ภาพที่ 1.1 โปรแกรมประยุกต์ในองค์กร

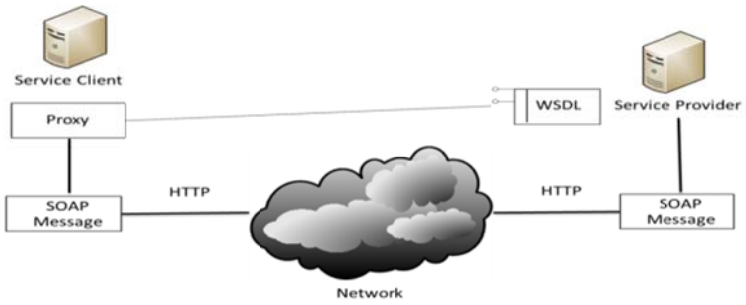
การบูรณการโปรแกรมประยุกต์ให้สอดคล้องกับความต้องการทั้งในและระหว่างองค์กรจะพบกับปัญหาอย่างมากอันเนื่องจากมีความแตกต่างกัน ยกตัวอย่างเช่น เลขจำนวนเต็มถ้าหากเป็นหน่วยประมวลผลกลางของบริษัทโมโตโรลา (Motorola) การจัดเรียงข้อมูลในหน่วยความจำจะเป็นแบบ Big Endean ในขณะเดียวกันหากเป็นของบริษัทอินเทล (Intel) จะเป็นแบบ Little Endean เป็นผลให้ต้องมีการแปลงข้อมูลจากลักษณะหนึ่งไปเป็นอีกลักษณะหนึ่งให้ถูกต้อง หรือโปรแกรมประยุกต์หนึ่งพัฒนาอยู่บนแฟลตฟอร์ม Java แต่ไม่สามารถเรียกใช้งานได้โดยเทคโนโลยี .NET เป็นต้น ดังนั้นจึงมีการศึกษาค้นคว้าและวิจัยแนวทางในการแก้ปัญหานี้ซึ่งเรียกว่า การบูรณการโปรแกรมประยุกต์ในองค์กร (Enterprise Application Integration หรือ EAI) ซึ่งในวิทยานิพนธ์ฉบับนี้จะใช้เทคโนโลยีเว็บเซอร์วิส (Web Services) มาประยุกต์ใช้ในการบูรณการเนื่องจากเป็นเทคโนโลยีที่ถูกออกแบบให้สามารถทำงานร่วมกันโดยไม่ขึ้นอยู่กับภาษาและแฟลตฟอร์ม โดยใช้วิธีการส่งข้อมูลแบบมัลติคาสต์ (Multicast) เพื่อให้การใช้งานเครือข่ายในการบูรณการให้เกิดประสิทธิภาพสูงสุด

1.2 วัตถุประสงค์ของวิทยานิพนธ์ฉบับนี้

เว็บเซอร์วิสเป็นเทคโนโลยีที่ถูกกำหนดมาตรฐานด้วยองค์กร World Wide Web Consortium หรือ W3C โดยมีจุดประสงค์ให้ซอฟต์แวร์สามารถทำงานร่วมกันโดยไม่ขึ้นอยู่กับเทคโนโลยีและแฟลตฟอร์มซึ่งใช้ภาษา XML ย่อมาจากคำว่า Extensible Markup Language เป็นรูปแบบข้อมูลในการแลกเปลี่ยนระหว่างกันภาษา XML ถูกออกแบบมาเพื่อใช้ในการเก็บข้อมูลเป็นหลักโดยเป็นเพียงข้อความธรรมดาซึ่งโครงสร้างของเอกสารที่จะถูกนิยามด้วยภาษา Document Type Definition หรือ DTD ทำให้เอกสารสามารถอธิบายตัวเอง (Self-description) ซึ่งเว็บเซอร์วิสมีองค์ประกอบสำคัญอยู่ 3 ส่วน (World Wide Web Consortium (W3C), n.d.b: 1) คือ SOAP, WSDL

และ UDDI ในการเรียกใช้งาน Web Services จะแบ่งผู้ใช้งานออกเป็น 2 ส่วนคือ ผู้ให้บริการ (Services Provider) และ ผู้รับบริการ (Service Client) การทำงานเริ่มจากผู้รับบริการจะนำเอาเอกสาร WSDL ที่ถูกบัญญัติโดยผู้ให้บริการมาสร้าง Proxy ซึ่งภายในจะประกอบด้วยรายละเอียดการใช้งาน รูปแบบข้อมูล เมธอด (Method) และที่ตั้งของผู้ให้บริการ (Endpoint) ตามที่ได้นิยามไว้แล้วเรียกใช้งานผ่าน Proxy นี้โดยข้อมูลที่ส่งผ่านแลกเปลี่ยนระหว่างกันจะอยู่ในรูปแบบโปรโตคอล SOAP ซึ่งภายในเป็นรูปแบบภาษา XML โดยจะทำงานบนโปรโตคอลอื่นอีกทีเช่น HTTP ดังภาพที่ 1.2

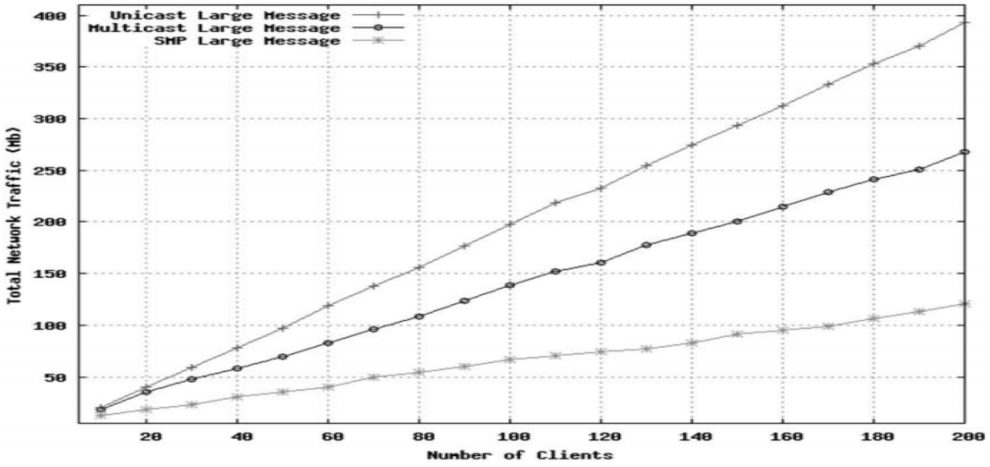
การรับส่งข้อมูลในรูปแบบ SOAP ทำให้ข้อมูลมีขนาดเพิ่มขึ้นและต้องเสียเวลาในการแปลงกลับไปมาจึงได้มีงานวิจัยในการใช้งานเมธอดของโปรโตคอล HTTP เช่น เมธอด GET แทนการเรียกข้อมูลร่วมกับ Universal Resource Identifiers (URIs) โดยตัวข้อมูลสามารถอยู่ในรูปแบบ XML หรือ JavaScript Object Notation (JSON) ซึ่งสามารถทำงานได้เร็วกว่าแบบ SOAP เรียกว่า Representational State Transfer หรือ REST แต่ถึงอย่างไรก็ตามทั้งสองรูปแบบจะพบปัญหาในการทำงานที่สำคัญ คือ ผู้รับบริการไม่ทราบสถานะล่าสุดของข้อมูลปัจจุบันว่ามีการปรับปรุงแก้ไขอย่างไรดังนั้นจึงต้องคอยเขียนโปรแกรมเพื่อดึงข้อมูลนำมาตรวจสอบในการทำงานและข้อมูล ถ้าหากจำนวนผู้รับบริการมีจำนวนมากหรือข้อมูลมีขนาดใหญ่จะทำให้มีการใช้งานเครือข่ายเกินความจำเป็น (Network Traffic) เกิดปัญหาคอขวด (Bottle Neck) และเวลาตอบสนอง (Response Time) ในการให้บริการของผู้ให้บริการได้



ภาพที่ 1.2 แสดงวิธีการเรียกใช้งาน Web Services

ดังนั้นวิทยานิพนธ์ฉบับนี้จึงมีแนวคิดในการประยุกต์ใช้เว็บเซอร์วิสเพื่อให้การบูรณการมีประสิทธิภาพสูงสุด โดยเปลี่ยนวิธีการเรียกใช้งานเว็บเซอร์วิสและการส่งข้อมูลเป็นแบบมัลติคาสต์ซึ่งเป็นงานใช้งานเครือข่ายได้อย่างมีประสิทธิภาพ จากงานวิจัยของ Phan, Tari and Bertok (2008)

ได้นำเอาเว็บเซอร์วิสแบบ SOAP ทำการส่งข้อมูลแบบมัลติคาสต์สามารถเปรียบเทียบการทำงานได้ดังภาพที่ 1.3



ภาพที่ 1.3 เปรียบเทียบการส่งข้อมูลแบบยูนิคาสต์ (Unicast) และมัลคิคาสต์ (Multicast)
แหล่งที่มา: Phan, Tari and Bertok, 2008: 102.

โดยในการใช้งานจะมีองค์ประกอบอยู่ 4 ส่วน คือ ผู้ให้บริการ (Services Provider) โบรกเกอร์ (Broker) ผู้รับบริการ (Service Client) และเอเยนต์ (Agent) โดยส่งข้อมูลแบบมัลติคาสต์ผ่านไอพีมัลติคาสต์ดังภาพที่ 3.9 ซึ่งจะอธิบายโดยละเอียดในบทถัดไป

บทที่ 2

ทฤษฎีและทบทวนวรรณกรรม

จากการสำรวจตลาดคอมพิวเตอร์ฮาร์ดแวร์ของประเทศไทยประจำปี พ.ศ. 2554 และประมาณการปี 2555 พบว่าตลาดคอมพิวเตอร์ฮาร์ดแวร์ในปี พ.ศ. 2555 จะเติบโตร้อยละ 4.3 โดยคิดมูลค่าตลาดรวม 91,174 ล้านบาท (สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ, 2557: 4) และผลการสำรวจตลาดซอฟต์แวร์และบริการซอฟต์แวร์ประจำปี พ.ศ. 2556 ประมาณการปี พ.ศ. 2557 จะเติบโตประมาณ 8.6% โดยคิดมูลค่าตลาดรวม 49,560 ล้านบาท (มูลนิธิสถาบันวิจัยเพื่อการพัฒนาประเทศไทย, 2557: 1) จะพบว่าองค์กรต่างๆ มีการลงทุนระบบสารสนเทศเพื่อสนับสนุนการดำเนินงานทางธุรกิจ

2.1 การบูรณการโปรแกรมประยุกต์ในองค์กร (Enterprise Application Integration หรือ EAI)

ปัจจุบันความต้องการสารสนเทศเพิ่มขึ้นมาก อันเนื่องจากเป็นส่วนสำคัญในการดำเนินงานขององค์กร กอปรกับรูปแบบการดำเนินธุรกิจเชิงอิเล็กทรอนิกส์ (Electronic Business หรือ e-Business) ได้เข้ามามีบทบาทสำคัญในการดำเนินงาน ทำให้มีรูปแบบการดำเนินธุรกิจใหม่ๆ เช่น การค้าระหว่างผู้ประกอบการกับผู้ประกอบการ (Business to Business หรือ B2B) หรือการค้าระหว่างผู้ประกอบการกับลูกค้า (Business to Customer หรือ B2C) เป็นต้น ซึ่งต้องใช้สารสนเทศและกระบวนการทางธุรกิจระหว่างกันมากขึ้น ยิ่งทำให้การบูรณการโปรแกรมทวีความสำคัญขึ้นไปด้วย แต่กระนั้นก็ดีก็ยังพบปัญหา เนื่องจากในองค์กรมีโปรแกรมประยุกต์ที่แตกต่างกัน ซึ่งต้องใช้เวลาและงบประมาณในการดำเนินงาน วิธีการหนึ่งที่ได้ถูกนำมาใช้ในการแก้ไขปัญหาเหล่านี้ คือ การบูรณการโปรแกรมประยุกต์ในองค์กร (Enterprise Application Integration หรือ EAI)

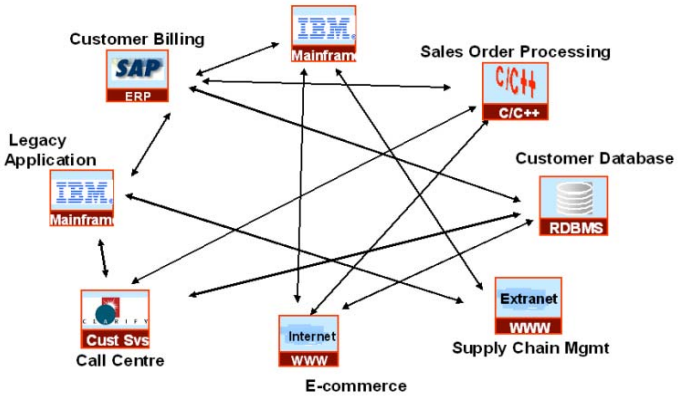
EAI เป็นวิธีการแก้ไขปัญหาโดยใช้เทคโนโลยีมาบูรณการโปรแกรมประยุกต์ที่มีความแตกต่างในเรื่องของภาษา เทคโนโลยีและแฟลตฟอร์มที่ใช้งานอยู่ในองค์กรที่มีการทำงานอย่างเป็น

เอกเทศให้สามารถประสานงานแลกเปลี่ยนข้อมูลและกระบวนการทางธุรกิจได้อย่างมีประสิทธิภาพ ผลของการบูรณการทำให้ผู้ใช้งานไม่ต้องป้อนข้อมูลชุดเดียวกันหลายครั้งเกิดเป็นผลดี คือ ทำให้ข้อมูลไม่มีความขัดแย้ง สามารถบริหารจัดการได้ง่ายและทำให้ระบบเดิม (Legacy System) ยังคงสามารถใช้งานร่วมกับโปรแกรมประยุกต์อื่นได้

สำหรับสถาปัตยกรรมในทำ EAI สามารถแบ่งออกเป็น 2 ประเภท (Fenner, 2002: 3) คือ การเชื่อมต่อแบบโดยตรง (Point-to-Point) และการเชื่อมต่อผ่านตัวกลาง (Middleware-based) และมีวิธีการในการทำ EAI แบ่งเป็น 4 วิธีการ (Linthicum, 1999: 21) คือ การบูรณการระดับข้อมูล (Data-level Integration) การบูรณการระดับส่วนติดต่อผู้ใช้งาน (User Interface-level Integration) การบูรณการระดับโปรแกรมประยุกต์ (Application-level Integration) และการบูรณการระดับวิธีการ (Method-level Integration) ซึ่งจะอธิบายในแต่ละหัวข้อดังต่อไปนี้

2.1.1 รูปแบบสถาปัตยกรรมของ EAI

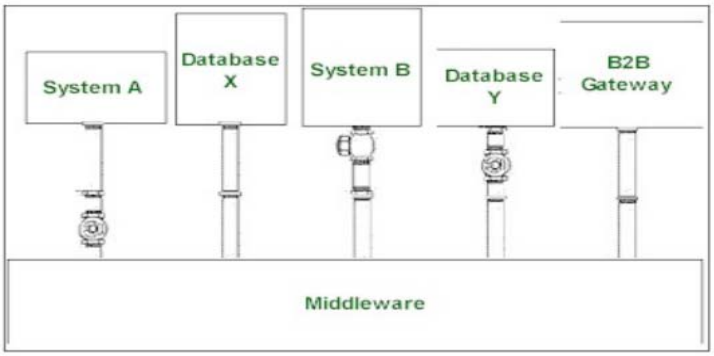
สถาปัตยกรรมในการทำ EAI จะพบรูปแบบอยู่ 2 ประเภทคือการเชื่อมต่อโดยตรง (Point-to-Point) และการเชื่อมต่อผ่านตัวกลาง (Middleware-based) ซึ่งการเชื่อมต่อโดยตรงเป็นวิธีการดั้งเดิมที่ใช้ในการบูรณการแลกเปลี่ยนข้อมูลจากโปรแกรมประยุกต์หนึ่งไปยังโปรแกรมประยุกต์หนึ่ง ซึ่งทำได้ง่ายและรวดเร็วเหมาะสำหรับการเชื่อมต่อโปรแกรมที่มีจำนวนน้อยเพราะหากเส้นของการเชื่อมเป็นดังรูปภาพที่ 2.1 จะพบปัญหาในการดูแลรักษาเพราะแต่ละโปรแกรมประยุกต์จะใช้เทคนิคและรูปแบบข้อมูลในการเชื่อมต่อที่แตกต่างกัน



ภาพที่ 2.1 การเชื่อมต่อแบบจุดต่อจุด

แหล่งที่มา: Fenner, 2002: 4.

ดังนั้นจึงมีแนวคิดในการบูรณการโดยเชื่อมต่อผ่านตัวกลางซึ่งแต่ละโปรแกรมประยุกต์จะเชื่อมต่อผ่านอินเตอร์เฟซ (Interface) มาตรฐานทั่วไปเพื่อให้สามารถรับส่งข้อมูลระหว่างกันทำให้เส้นของการเชื่อมต่อลดลง ดังภาพที่ 2.2



ภาพที่ 2.2 การเชื่อมต่อโดยใช้ตัวกลาง
แหล่งที่มา: Fenner, 2002: 5.

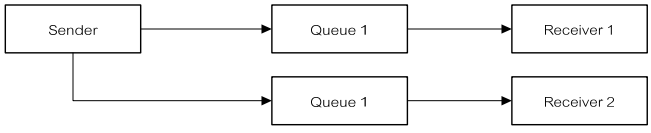
จากภาพข้างต้นจะพบว่ามี 5 โปรแกรมประยุกต์และจะมีเพียง 5 เส้นในการเชื่อมต่อเท่านั้น วิธีการนี้ส่งผลดี คือ ในการปรับปรุงแก้ไขโปรแกรมหนึ่งโปรแกรมใดจะไม่มีผลกระทบกับอีกโปรแกรมหนึ่งขณะที่ตัวกลางยังสามารถพัฒนาความสามารถต่างๆ เพิ่มขึ้น เช่น การแปลงข้อมูลให้อยู่ในรูปแบบที่เหมาะสม การส่งข้อมูลสามารถกำหนดเส้นทางของข้อมูลว่าต้องการให้มีลำดับในการส่งอย่างไร เป็นต้น ต่อมามีการพัฒนาโปรกเกอร์เพื่อเพิ่มความสามารถในการทำงานให้ตัวกลางทำให้ได้รับความนิยมในการใช้งานเนื่องจากมีส่วนติดต่อผู้ใช้งาน (Graphic User Interface หรือ GUI) ที่ดีใช้งานได้สะดวกและลดการเขียนโปรแกรมลงทำให้นักวิเคราะห์ธุรกิจ (Business Analysis) สามารถเข้ามาใช้งานและเพื่อกำหนดกระบวนการทางธุรกิจขององค์กรในปัจจุบัน

ตัวกลางสามารถแบ่งตามประเภทการบูรณการได้ 5 ประเภท (Bishop and Karne, n.d.: 2) คือ ตัวกลางเชิงกระบวนการ (Procedure Oriented Middleware) ตัวกลางเชิงวัตถุ (Object Oriented Middleware) ตัวกลางเชิงข้อความ (Message Oriented Middleware) ตัวกลางเชิงองค์ประกอบ (Component-based Oriented Middleware) และตัวแทน (Agent) ซึ่งตัวกลางแต่ละประเภทมีรายละเอียดดังนี้

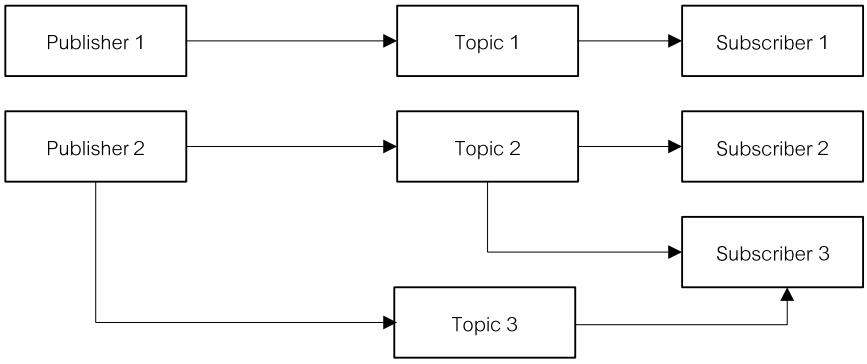
2.1.1.1 ตัวกลางเชิงกระบวนการ (Procedure Oriented Middleware) ตัวกลางเชิงกระบวนการจะใช้วิธีการสื่อสารแบบสมวาร (Synchronous) โดยมีลักษณะในการทำงานเป็น 2 ส่วนคือ Client Stub และ Server Skeleton ในทำงาน Client Stub จะทำการมาแชลลิ่ง (Marshaling) ข้อมูลจากโปรแกรมประยุกต์ของผู้เรียกส่งไปยัง Server Skeleton ซึ่งจะทำการแปลงข้อมูลที่ได้รับมาแล้วเรียกเมธอดที่อยู่บนเครื่อง Server และส่งผลลัพธ์กลับไปยัง Client Stub โดยจะมีการตรวจสอบและจัดการกับข้อผิดพลาดที่เกิดในการทำงาน แต่มีข้อจำกัดในการใช้งานเนื่องจากรูปแบบของการสื่อสารเพียงรูปแบบเดียวและไม่สามารถปรับขยายได้

2.1.1.2 ตัวกลางเชิงวัตถุ (Object Oriented Middleware) รูปแบบการสื่อสารแบบสมวาร (Synchronous) และอสมวาร (Asynchronous) ซึ่งสามารถใช้เธรด (Thread) ในการทำงานได้ โดยข้อเด่นประการสำคัญ คือ รองรับธุรกรรม (Transaction) ที่ทำให้หลายๆ การทำงานทำงานเสมือนเป็นงานเดียวกัน รองรับการทำงานพร้อมๆ กันและอ็อปเจ็คถูกสร้างและอยู่ในฝั่ง Server ซึ่งมีลักษณะการทำงาน คือ จะสร้างสตับ (Stub) โดยทำการมาแชลลิ่งข้อมูลที่ต้องการส่งไปยังโปรกเกอร์ หลังจากนั้นโปรกเกอร์จะพิจารณาถึงข้อมูลที่ได้รับมาโดยตรวจสอบจากเลข ID และส่งผ่านไปยังสเกลเลอทอน (Skeleton) ให้ทำงานและส่งผลลัพธ์ย้อนคืนตามลำดับ ข้อเสียของวิธีการนี้คือ ต้องมีการสร้างการเชื่อมต่อก่อนใช้งานได้และถ้าหากต้องใช้งานกับระบบเดิม (Legacy System) จะต้องเขียนโปรแกรมครอบการทำงานของระบบเดิม

2.1.1.3 ตัวกลางเชิงข้อความ (Message Oriented Middleware) ตัวกลางเชิงข้อความสามารถแบ่งออกเป็น 2 ประเภท คือ คิว (Queue) และ Publish/Subscribe โดยแบบคิวมีลักษณะการทำงาน คือ โปรแกรมที่ต้องการส่งข้อมูลจะส่งข้อมูลมายังตัวกลางโดยข้อมูลจะถูกจัดเก็บไว้ในคิว หลังจากนั้นโปรแกรมที่ต้องการใช้งานข้อมูลจะทำการเรียกใช้ข้อมูลจากคิวซึ่งมีลักษณะการดึงข้อมูลจากคิวแบบ First In First Out หรือ FIFO ดังภาพที่ 2.3 และแบบ Publish/Subscribe รูปแบบการทำงานจะมีการนิยามหัวข้อ (Topic) ผู้ที่ต้องการส่งข้อมูลเรียกว่าพับลิชเชอร์ (Publisher) และผู้ที่ต้องการรับข้อมูลเรียกว่า ซับสไครเบอร์ (Subscriber) ซึ่งผู้รับข้อมูลจะต้องทำการลงทะเบียน (Register) เพื่อให้รายละเอียดไว้กับที่ Server ตัวกลางและเมื่อ Publisher ทำการส่งข้อมูลตัวกลางจะทำการกระจายข้อมูลไปยัง Subscriber ที่ได้ลงทะเบียนไว้ดังภาพที่ 2.4



ภาพที่ 2.3 ตัวกลางเชิงข้อความแบบ Queue



ภาพที่ 2.4 ตัวกลางเชิงข้อความแบบ Publish/Subscribe

2.1.1.4 ตัวกลางเชิงองค์ประกอบ (Component-based Oriented Middleware) ตัวกลางเชิงองค์ประกอบ (Component) เป็นโปรแกรมที่ถูกออกแบบมาโดยมีฟังก์ชั่นเพื่อให้โปรแกรมอื่นๆ เรียกใช้งานได้โดยง่าย ตัวกลางเชิงองค์ประกอบนี้จะมีไลบรารี (Library) จำนวนมาก ซึ่งรองรับการทำงานได้หลายแฟลตฟอร์ม โดยสามารถกำหนดค่าต่างๆ ได้ และยังสนับสนุนการประมวลผลแบบกระจาย จุดเด่นของตัวกลางชนิดนี้ คือ ความยืดหยุ่นในการใช้งานและรองรับการทำงานของโปรแกรมประยุกต์จำนวนมาก

2.1.1.5 ตัวแทน (Agent) ตัวกลางชนิดนี้จะประกอบไปด้วย 3 องค์ประกอบ คือ เอนติตี้ (Entities) มีเดีย (Media) และกฎ (Law) โดยทั้งสามส่วนจะทำงานประสานกันโดยเอนติตี้เป็นส่วนต่างๆ ที่มีอยู่ในระบบ เช่น อ็อปเจ็ค (Object) เธรด (Thread) เป็นต้น มีเดียเป็นรูปแบบการติดต่อสื่อสารของตัวแทนกับส่วนอื่นๆ และกฎเป็นข้อตกลงในแต่ละตัวแทนว่าจะมีการติดต่อสื่อสารทำงานร่วมกันอย่างไร ตัวกลางชนิดนี้มีข้อเสียเนื่องจากทำความเข้าใจยากและมีความซับซ้อนในการทำงานซึ่งต้องใช้บุคคลที่มีความเชี่ยวชาญในการทำงาน

2.2 วิธีการในการทำ EAI

วิธีการในการทำ EAI จะประกอบด้วยวิธีการทำอยู่ 4 ระดับดังภาพที่ 2.5 ดังนี้

2.2.1 การบูรณการระดับข้อมูล (Data-level Integration)

วิธีการนี้เป็นการกระทำกับฐานข้อมูลโดยจะส่งข้อมูลจากฐานข้อมูลหนึ่งไปยังอีกฐานข้อมูลหนึ่งซึ่งมีรูปแบบการทำงานอยู่ 2 วิธีคือ วิธีการแบบ Push และวิธีการแบบ Pull โดย

วิธีการแบบ Push จะมีโปรแกรมหนึ่งทำการสร้างคิวรี่ (Query) แล้วส่งไปยัง Database Link หรือ Store Procedure เพื่อส่งข้อมูลไปยังฐานข้อมูลที่ต้องการและวิธีการแบบ Pull จะใช้ Trigger ในการทำงานโดยจะคอยตรวจสอบข้อมูลที่ต้องการหากมีการเปลี่ยนแปลงตัว Trigger ก็จะทำงานและส่งข้อมูลไปยังฐานข้อมูลที่ต้องการ

2.2.2 การบูรณการระดับส่วนติดต่อผู้ใช้งาน (User Interface (UI) – level Integration)

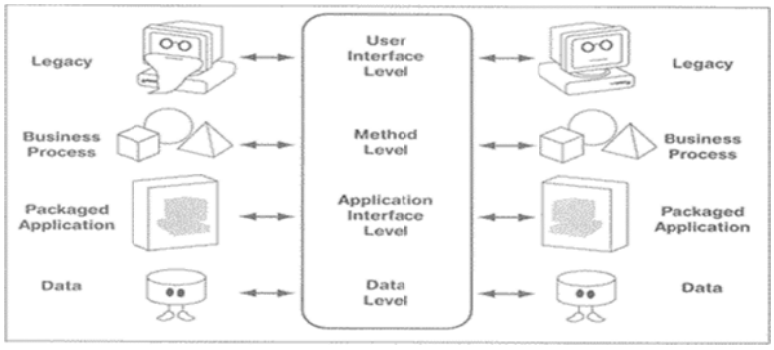
วิธีการนี้จะสร้าง Script หรือ Proxy แทรกเข้าไปในโค้ดของอีกโปรแกรมอื่นเพื่อให้เรียกใช้งาน Script หรือ Proxy นี้เมื่อเกิดการทำงาน วิธีการนี้ไม่เป็นนิยมใช้เนื่องจากทำให้โปรแกรมที่บูรณการกันนั้นขึ้นต่อกัน และเมื่อมีการแก้ไขอาจมีกระทบส่วนโปรแกรมอื่นที่เกี่ยวข้องได้ ทำให้ดูแลรักษาลำบาก วิธีการนี้จะใช้ก็ต่อเมื่อไม่สามารถเข้าไปยังฐานข้อมูลได้ยกตัวอย่าง เช่น ระบบเมนเฟรมซึ่งไม่เอื้อให้เข้าไปจัดการข้อมูลภายใน เป็นต้น

2.2.3 การบูรณการระดับโปรแกรมประยุกต์ (Application-level Integration)

วิธีการนี้เป็นทางเลือกที่เหมาะสมเนื่องจากจะเป็นการเรียกใช้งาน API ของโปรแกรมที่ต้องการบูรณการตัวอย่างเช่น เรียกใช้ API ของ SAP ในการเชื่อมต่อ เป็นต้น

2.2.4 การบูรณการระดับวิธีการ (Method-level Integration)

วิธีการนี้เป็นจะทำการแยกฟังก์ชั่นหรือเมธอดที่เหมือนกันในแต่ละโปรแกรมประยุกต์มาไว้ที่เดียวกัน แล้วสร้างอินเตอร์เฟซที่เหมือนกันและให้โปรแกรมประยุกต์มาเรียกใช้งาน อาจจะใช้แนวคิดของซอฟท์แวร์เชิงองค์ประกอบ เช่น CORBA, EJB หรือ COM เป็นต้น



ภาพที่ 2.5 วิธีการทำ EAI

แหล่งที่มา: Linthicum, 1999: 22.

2.3 เว็บเซอร์วิส (Web Services)

เว็บเซอร์วิสเป็นเทคโนโลยีที่ถูกกำหนดมาตรฐานขึ้นโดยองค์กร World Wide Web Consortium หรือ W3C โดยมีจุดประสงค์ให้ซอฟต์แวร์สามารถทำงานร่วมกันโดยไม่ขึ้นอยู่กับเทคโนโลยีและแฟลตฟอร์มซึ่งใช้ภาษา XML ย่อมาจากคำว่า Extensible Markup Language เป็นรูปแบบข้อมูลในการแลกเปลี่ยนระหว่างกัน ภาษา XML ถูกออกแบบมาเพื่อใช้ในการเก็บข้อมูลเป็นหลักโดยเป็นเพียงข้อความธรรมดาซึ่งโครงสร้างของเอกสารจะถูกนิยามด้วยภาษา Document Type Definition หรือ DTD ทำให้เอกสารสามารถอธิบายตัวเอง (Self-description) ในปัจจุบันได้รับความนิยมในการนำเอามาบูรณการโปรแกรมประยุกต์ที่แตกต่างกันทั้งทางด้านเทคโนโลยีและแฟลตฟอร์มให้สามารถสื่อสารแลกเปลี่ยนข้อมูลระหว่างกันได้โดยเว็บเซอร์วิสมีองค์ประกอบสำคัญอยู่ 3 ส่วน คือ SOAP, WSDL และ UDDI ซึ่งจะอธิบายแต่ละส่วนดังต่อไปนี้

2.3.1 Simple Object Access Protocol หรือ SOAP

SOAP เป็นโปรโตคอลที่ใช้การส่งข้อมูลระหว่างกันซึ่งจะทำงานอยู่บนโปรโตคอลอื่น เช่น Simple Mail Transfer Protocol (SMTP), Hypertext Transfer Protocol (HTTP) เป็นต้น โดยจะใช้รูปแบบภาษา XML เป็นพื้นฐานในการทำงานทำให้ SOAP ไม่ขึ้นอยู่กับภาษาและแฟลตฟอร์มมีองค์ประกอบสำคัญอยู่ 3 ส่วน (World Wide Web Consortium (W3C), n.d.a: 1)ได้แก่ SOAP Envelope, SOAP Header และ SOAP Body โดย SOAP Envelope เป็นส่วนที่ใช้นิยามซึ่งเป็นส่วนสำคัญที่ต้องถูกกำหนดอยู่ในเอกสารส่วน SOAP Headerจะเกี่ยวข้องกับข้อมูลผู้ใช้ซึ่งในกรณีที่ไม่จำเป็นสามารถละเว้นได้และสุดท้าย SOAP Body จะเป็นส่วนที่ข้อมูลที่ต้องการแลกเปลี่ยนระหว่างกันซึ่งอยู่ในรูปแบบภาษา XML

2.3.2 Web Services Description Language หรือ WSDL

WSDL เป็นเอกสารที่ถูกบัญญัติขึ้นซึ่งอยู่ในรูปแบบภาษา XML ใช้สำหรับอธิบายเว็บเซอร์วิสและวิธีการเข้าถึงเพื่อเรียกใช้งานประกอบด้วย 4 ส่วนสำคัญตามตารางที่ 2.1 ในส่วนของ PortType เป็นการกำหนดวิธีการปฏิบัติการของเว็บเซอร์วิสในการเรียกใช้งานซึ่งประกอบด้วย 4 รูปแบบ คือ One Way รูปแบบนี้ทำการส่งข้อความมาแล้วไม่รอผลตอบกลับ Request/Response รูปแบบนี้ทำการส่งข้อความมาแล้วรอผลลัพธ์จากการทำงาน Solicit/Response รูปแบบนี้ทำการส่งข้อความแล้วสามารถรอการประมวลผลเพื่อรอผลลัพธ์จากการทำงาน Notification รูปแบบนี้ทำการส่งข้อความแล้วไม่สามารถรอเพื่อประมวลผลลัพธ์ได้และในส่วนของ Binding เป็นการระบุรูปแบบ

โปรโตคอลและที่ตั้งของ Server ที่ให้บริการ (Endpoint) แสดงดังภาพที่ 2.6 จะพบว่าเว็บเซอร์วิสนี้มีการใช้โปรโตคอลแบบ SOAP ทำงานบนโปรโตคอล HTTP ในการสื่อสารแลกเปลี่ยนข้อมูลระหว่างกัน แสดงรายละเอียดของเมธอดและพารามิเตอร์ที่ต้องส่งและที่อยู่ของเว็บเซอร์วิส คือ http://example.com/getTerm ซึ่งจากข้อมูลดังกล่าวผู้รับบริการจะนำเอาไปสร้างสตับเพื่อเรียกใช้งานเว็บเซอร์วิสต่อไป

ตารางที่ 2.1 แสดงส่วนสำคัญของเอกสาร WSDL

รายการ	**คำอธิบาย**
Types	ประกอบด้วยข้อมูลที่ถูกนิยามซึ่งถูกใช้ในเว็บเซอร์วิส
Message	เป็นการนิยามชนิดข้อมูลก่อนที่จะมีการสื่อสาร
PortType	กลุ่มของเมธอดที่มีให้เรียกใช้งานของเว็บเซอร์วิส
Binding	เป็นรูปแบบของโปรโตคอลและข้อมูลรายละเอียดของ Port Type

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>

<binding type="glossaryTerms" name="b1">
   <soap:binding style="document"
   transport="http://schemas.xmlsoap.org/soap/http" />
   <operation>
     <soap:operation soapAction="http://example.com/getTerm"/>
     <input><soap:body use="literal"/></input>
     <output><soap:body use="literal"/></output>
  </operation>
</binding>
```

ภาพที่ 2.6 แสดงตัวอย่างเอกสาร WSDL

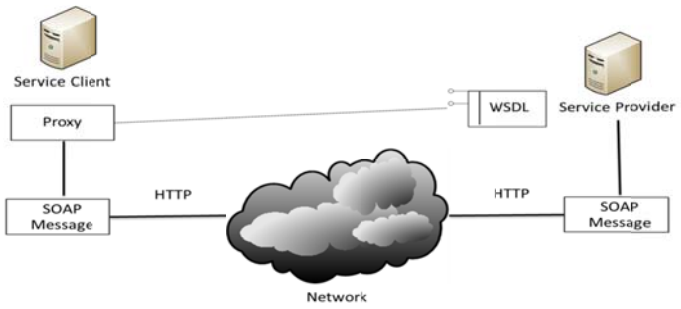
แหล่งที่มา: World Wide Web Consortium (W3C), n.d.c: 1.

2.3.3 Universal Description, Discovery and Integration หรือ UDDI

สำหรับผู้ให้บริการจะทำการลงทะเบียนเว็บเซอร์วิสของตนเอง เพื่อให้ผู้ต้องการเรียกใช้งานสามารถ มาค้นหาผ่านสาระบบ ซึ่งแต่ละเว็บเซอร์วิสถูกอธิบายด้วย WSDL และการติดต่อสื่อสารทั้งหมดจะกระทำผ่านโปรโตคอล SOAP

2.3.4 การเรียกใช้งานเว็บเซอร์วิส

ในการเรียกใช้งานเว็บเซอร์วิสจะแบ่งผู้ใช้งานออกเป็น 2 ส่วน คือ ผู้ให้บริการ (Services Provider) และผู้รับบริการ (Service Client) การทำงานเริ่มจากผู้รับบริการจะนำเอาเอกสาร WSDL ที่ถูกบัญญัติโดยผู้ให้บริการมาสร้าง Proxy ซึ่งภายในจะมีรายละเอียดการใช้งานรูปแบบข้อมูล เมธอดและที่ตั้งของผู้ให้บริการตามที่ได้นิยามไว้แล้วและเรียกใช้งานผ่าน Proxy นี้โดยข้อมูลที่ส่งผ่านแลกเปลี่ยนระหว่างกันจะอยู่ในรูปแบบของโปรโตคอล SOAP และทำงานอยู่บนโปรโตคอลอื่นอีกที เช่น HTTP ดังภาพที่ 2.7



ภาพที่ 2.7 แสดงวิธีการเรียกใช้งานเว็บเซอร์วิส

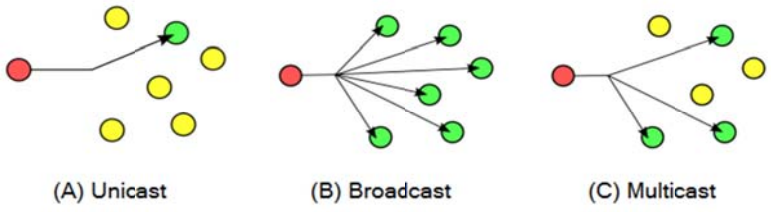
2.3.5 Representational State Transfer หรือ REST

การใช้งานเว็บเซอร์วิสแบบ SOAP จะทำงานได้ช้า อันเนื่องจากข้อมูลต้องอยู่ในรูปแบบ SOAP ทำให้มีขนาดเพิ่มขึ้นและต้องใช้เวลาในการแปลงข้อมูลไปมาในการรับและส่ง จากเหตุผลดังกล่าวจึงทำให้เกิดงานวิจัยโดย Roy Fielding (2000) งานวิจัยนี้ได้นำเสนอแนวความคิดในการใช้เมธอดของโปรโตคอล HTTP เป็นการทำงานของเว็บเซอร์วิส ยกตัวอย่างเช่น เมธอด GET ถูกใช้ในการดึงข้อมูลจากเว็บเซอร์วิส เมธอด PUT ถูกใช้ในการปรับปรุงข้อมูลในเว็บเซอร์วิส เป็นต้น โดยข้อมูลสามารถอยู่ในรูปแบบ XML หรือ JavaScript Object Notation (JSON) ร่วมกับ Universal Resource Identifiers (URIs) ในการทำงาน โดยจะมีเอกสาร WADL หรือ Web Application

Description Language ซึ่งมีหน้าที่ลักษณะคล้ายกับ WSDL ของ SOAP แต่จะเขียนและทำความเข้าใจได้ง่ายกว่า เว็บเซอร์วิสที่สร้างด้วยวิธีนี้ทำงานได้เร็วและกำลังได้รับความนิยมในปัจจุบัน ดังจะเห็นได้จากบริษัทใหญ่ เช่น Yahoo Google ต่างก็ใช้ REST แทนการใช้เว็บเซอร์วิสแบบ SOAP

2.4 มัลติคาสต์ (Multicast)

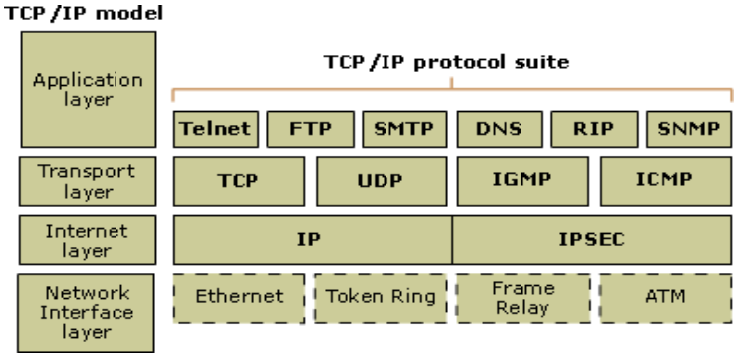
การสื่อสารในเครือข่ายรูปแบบการส่งข้อมูลสามารถแบ่งออกเป็น 3 ลักษณะใหญ่ (Wikipedia, 2002: 1) คือ การส่งข้อมูลแบบยูนิคาสต์ (Unicast) เป็นการส่งข้อมูลจากจุดหนึ่งไปยังอีกจุดหนึ่งเพียงจุดเดียว การส่งข้อมูลแบบบรอดคาสต์ (Broadcast) เป็นการส่งข้อมูลออกไปเพียงครั้งเดียวและทุกจุดในเครือข่ายจะได้รับข้อมูลพร้อมๆ กัน และการส่งข้อมูลแบบมัลติคาสต์ (Multicast) เป็นการส่งข้อมูลเพียงครั้งเดียวแต่เฉพาะจุดที่ระบุเท่านั้นที่สามารถรับข้อมูลได้ ซึ่งรูปแบบการส่งข้อมูลทั้ง 3 ลักษณะแสดงได้ดังภาพที่ 2.8



ภาพที่ 2.8 รูปแบบการส่งข้อมูลในเครือข่าย
แหล่งที่มา: Wikipedia, 2002: 1.

ปัจจุบันอินเตอร์เน็ตเป็นเครือข่ายใหญ่ที่สุดในโลกที่ใช้ในการแลกเปลี่ยนข้อมูลระหว่างกันโดยชุดโปรโตคอล TCP/IP (Transmission Control Protocol/Internet Protocol) จัดเป็นชุดโปรโตคอลพื้นฐานของเครือข่ายนี้ซึ่งถูกแบ่งออกเป็น 4 ชั้น คือ ชั้นแอพพลิเคชั่น (Application Layer) ชั้นทรานสปอร์ต (Transport Layer) ชั้นเน็ตเวิร์ก (Network Layer) และชั้นลิงค์ (Link Layer) ตามลำดับ ดังภาพที่ 2.9 การทำมัลติคาสต์ผ่านเครือข่าย TCP/IP จะใช้วิธีการที่เรียกว่าไอพีมัลติคาสต์ (IP Multicast) ซึ่งจะมีการกำหนด IP เป็น Multicast Address ต่างๆ เป็นช่วงๆ ซึ่งแต่ละช่วงจะมีการกำหนดวัตถุประสงค์เฉพาะเอาไว้ดังภาพที่ 2.10 เช่น Address ที่ 224.0.0.1 และ 224.0.0.2 ถูกสงวนไว้สำหรับ Local Network Control Block สำหรับอุปกรณ์เราเตอร์ (Routers)

ในการรับส่งข้อมูลเกี่ยวกับกลุ่มของมัลติคาสต์ในส่วนของซับเน็ต (Subnet) ตนเองและซับเน็ตรอบข้างให้สามารถทำงานแลกเปลี่ยนข้อมูลข้ามซับเน็ตกันได้นั้นในการทำงานไอพีมัลติคาสต์เครื่องที่ต้องการรับส่งข้อมูลนั้นจะต้องเข้ามาเป็นสมาชิกของกลุ่มมัลติคาสต์ก่อน ซึ่งจะใช้โปรโตคอล IGMP (Internet Group Management Protocol) เพื่อทำการลงทะเบียนที่เราเตอร์ว่าจะรับข้อมูลมัลติคาสต์มาจากแหล่งใด และเมื่อต้องการหยุดรับข้อมูลก็จะใช้โปรโตคอล IGMP ในการส่งไปยกเลิกการเป็นสมาชิกไปยังเราเตอร์ ด้วยเหตุนี้ทำให้เฉพาะเครื่องที่สนใจเท่านั้นสามารถรับข้อมูลที่ต้องการ



ภาพที่ 2.9 โปรโตคอล TCP/IP

แหล่งที่มา: Microsoft TechNet, n.d.: 1.

Range	Assignment
224.0.0.0-224.0.0.255	Local Network control block
224.0.1.0-224.0.1.255	Inter-network control block
224.0.2.0-224.0.255.0	AD-HOC block
224.1.0.0-224.1.255.255	ST Multicast groups
224.2.0.0-224.2.255.255	SDP/SAP block
224.252.0.0-224.255.255.255	DIS Transient block
225.0.0.0-231.255.255.255	Reserved
232.0.0.0-232.255.255.255	Source-specific multicast block
233.0.0.0-233.255.255.255	GLOP block
234.0.0.0-238.255.255.255	Reserved
239.0.0.0-239.255.255.255	Administratively scoped block

ภาพที่ 2.10 แสดงช่วงต่างๆ ของ IP Multicast Address

บทที่ 3

กรอบแนวคิดและระเบียบวิธีการวิจัย

เว็บเซอร์วิสถูกออกแบบมาเพื่อช่วยให้โปรแกรมหรือคอมโพเนนต์ (Component) ที่มีอยู่แล้วนำกลับมาใช้งานใหม่ได้ เช่น นำฟังก์ชั่นในคำนวณพิกัดที่มีความแม่นยำสูงมาประยุกต์ใช้ร่วมกับเว็บเซอร์วิสทำให้โปรแกรมอื่นสามารถเรียกใช้งานได้ เป็นต้น และช่วยให้โปรแกรมประยุกต์ที่มีความแตกต่างกันในเรื่องของภาษา เทคโนโลยีและแฟลตฟอร์มสามารถทำงานร่วมกันโดยส่งข้อมูลระหว่างกันได้ จึงนับเป็นเทคโนโลยีหนึ่งที่ได้รับความนิยมในการใช้งานบูรณการโปรแกรมประยุกต์ที่มีแตกต่างกันในองค์กรให้สามารถทำงานแลกเปลี่ยนข้อมูลระหว่างกันได้

3.1 กรอบแนวคิด

การทำงานร่วมกันโดยไม่ขึ้นอยู่กับภาษา เทคโนโลยีและแฟลตฟอร์มถือเป็นหัวใจสำคัญที่ทำให้เทคโนโลยีนี้ได้รับความนิยมในการใช้งาน โดยเว็บเซอร์วิสในช่วงแรกจะใช้ SOAP ซึ่งเป็นโปรโตคอลโดยลักษณะข้อมูลภายในอยู่ในรูปเอกสาร XML ซึ่งเป็นเพียงตัวอักษรธรรมดาที่ให้ความหมายและอธิบายตัวเองได้ (Self-description) ทั้งนี้โปรโตคอล SOAP เองยังสามารถทำงานได้บนโปรโตคอลเปิดอื่นๆ เช่น HTTP, STMP เป็นต้น แต่ในการใช้งานจริงจะพบข้อจำกัดบางประการของเว็บเซอร์วิสที่ทำให้มีการใช้งานเครือข่ายเกินความจำเป็น จึงได้มีการศึกษาและวิจัยเพื่อขจัดปัญหาดังกล่าวตามรายละเอียดหัวข้อด้านล่างนี้

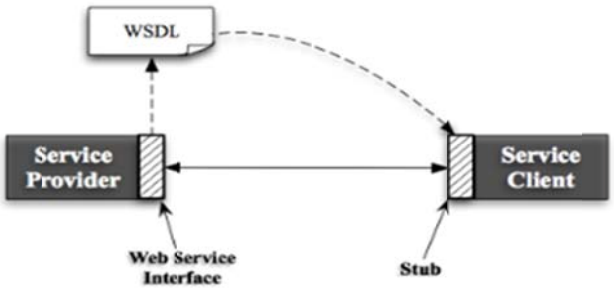
3.1.1 รูปแบบเรียกใช้งานเว็บเซอร์วิสโดยทั่วไป

การใช้งานเว็บเซอร์วิสจะประกอบด้วย ผู้ให้บริการ (Service Provider) และผู้รับบริการ (Service Client) โดยผู้ให้บริการจะบัญญัติเอกสารที่เรียกว่า WSDL (Web Services Description Language) ซึ่งเป็นเอกสารอธิบายว่ามีบริการรูปแบบข้อมูลและที่ตั้งของผู้ให้บริการ เป็นต้น

ผู้รับบริการจะนำเอกสารมาสร้างเป็นสตับ (Stub) หรือบางครั้งจะเรียกว่าพร็อกซี่ (Proxy) และใช้งานเว็บเซอร์วิสผ่านสตับนี้ดังภาพที่ 3.1 โดยรูปแบบการเรียกใช้งานจัดอยู่ในสถาปัตยกรรมแบบ Pull โดยผู้รับบริการจะร้องขอข้อมูลที่ต้องการไปยังผู้ให้บริการและรอข้อมูลที่จะถูกตอบกลับมาซึ่งจะพบปัญหาสำคัญอยู่ประการหนึ่งคือในส่วนของผู้รับบริการจะไม่ทราบสถานะล่าสุดของข้อมูลว่ามีการปรับปรุงเปลี่ยนแปลงอย่างไรบ้างทำให้ต้องคอยดึงข้อมูลมาเป็นระยะๆ หรือตามช่วงเวลาเพื่อตรวจสอบเป็นผลให้มีการใช้งานเครือข่ายเกินความจำเป็นและสามารถทำให้เกิดปัญหาคอขวดสำหรับผู้ให้บริการได้ในกรณีที่มีข้อมูลขนาดใหญ่หรือมีผู้รับบริการจำนวนมากร้องขอข้อมูลพร้อมๆ กัน และอาจส่งผลกระทบถึงประสิทธิภาพโดยรวมของเครือข่ายทำให้มีเวลาตอบสนองที่ช้าได้ ทำให้มีการศึกษาและวิจัยเพื่อขจัดปัญหาที่เกิดขึ้นซึ่งหนึ่งในแนวคิดนั้นคือการนำเอาสถาปัตยกรรมแบบ Publish/Subscribe มาประยุกต์ใช้งานกับเว็บเซอร์วิส

Brenna and Johansen (2005) ใช้ WAIF Proxy ซึ่งจะคอยดึงข้อมูลจากผู้ให้บริการแล้วนำมาพักไว้ที่เครื่องของตน หลังจากนั้นจะทำการส่งข้อมูลไปยังผู้รับบริการที่ได้มาลงทะเบียนไว้กับตนเอง ดังภาพที่ 3.2 ซึ่งข้อเสียของวิธีการนี้ คือ ต้องมีเครื่องคอยทำหน้าที่เป็น WAIF Proxy เพื่อคอยดึงและเก็บข้อมูลก่อนจะส่งไปยังผู้รับบริการ และยังคงทำให้ผู้ให้บริการเกิดปัญหาคอขวดได้หากมีการดึงข้อมูลบ่อยครั้งหรือข้อมูลมีขนาดใหญ่ ต่อมา Feng, Xue and Zhang (2009) ได้ประยุกต์ให้ผู้ให้บริการเว็บเซอร์วิสมาลงทะเบียนที่ระบบ UDDI ทำให้ผู้รับบริการไม่จำเป็นต้องหาเว็บเซอร์วิสเองเพียงแต่มาลงทะเบียนตามเรื่องหรือคีย์เวิร์ดที่ตนสนใจหลังจากนั้นเมื่อผู้ให้บริการมีรายการข้อมูลปรับปรุงก็จะส่งสัญญาณมายังระบบ UDDI จากนั้นระบบ UDDI จะส่งสัญญาณไปยังผู้รับบริการเพื่อไปร้องขอและรับข้อมูลจากผู้ให้บริการดังภาพที่ 3.3 อย่างไรก็ตามจะพบว่าข้อเสียของวิธีการนี้คือทำให้การบูรณการมีความยุ่งยากเนื่องจากแต่ละโปรแกรมประยุกต์ต้องหาวิธีการติดต่อกับระบบคิว

ต่อมา Skjervold, Hafsøe, Johnsen and Lund (2010) ได้ใช้วิธีการที่เรียกว่า Delay and Disruption Tolerant SOAP Proxy (DSProxy) ดังภาพที่ 3.4 โดยดีเอสพร็อคซี่ (DSProxy) จะอยู่ในฝั่งผู้ให้บริการและผู้รับบริการ โดยฝั่งผู้ให้บริการดีเอสพร็อคซี่จะคอยทำการตรวจสอบข้อมูลโดยดึงจากเว็บเซอร์วิสซึ่งหากข้อมูลมีการเปลี่ยนแปลงก็จะส่งสัญญาณไปยังดีเอสพร็อคซี่ที่อยู่ฝั่งผู้รับบริการให้ทราบว่าข้อมูลมีการเปลี่ยนแปลงแล้วเพื่อให้ผู้รับบริการมาดึงข้อมูลจากเว็บเซอร์วิสอีกทีวิธีการนี้ช่วยลดการใช้งานเครือข่ายที่ไม่จำเป็นลงไปได้ แต่ในการนำไปประยุกต์ใช้จะยังพบปัญหาอยู่ เนื่องจากไม่ได้ใช้โปรโตคอลปิดที่เป็นมาตรฐานทั่วไป ทำให้มีข้อจำกัดในการนำมาบูรณการโปรแกรมประยุกต์



ภาพที่ 3.1 แสดงรูปแบบการเรียกใช้งานเว็บเซอร์วิส

แหล่งที่มา: Thanisa, 2012: 27.

ภาพที่ 3.2 สถาปัตยกรรมแบบ Publish/Subscribe ใช้ WAIF Proxy

แหล่งที่มา: Thanisa, 2012: 22.

ภาพที่ 3.3 สถาปัตยกรรมแบบ Publish/Subscribe ใช้ UDDI Register System

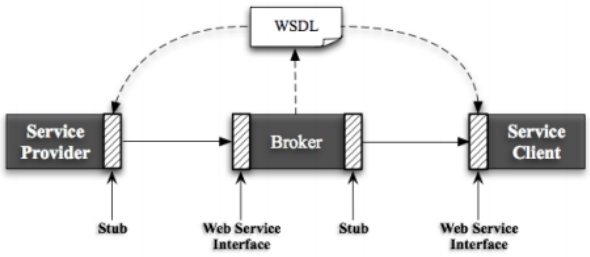
แหล่งที่มา: Thanisa, 2012: 23.

ภาพที่ 3.4 รูปแบบการทำงาน DSProxy

แหล่งที่มา: Thanisa, 2012: 24.

3.1.2 รูปแบบเรียกใช้งานเว็บเซอร์วิสแบบผกผัน

วิธีการเรียกใช้งานเว็บเซอร์วิสแบบยกผันของ Thanisa Noomnonda (2012) สามารถขจัดปัญหาคอขวดและมีการใช้งานเครือข่ายที่ดีขึ้น โดยใช้สถาปัตยกรรมแบบ Push-Based ในการทำงานดังภาพที่ 3.5 ลักษณะการทำงานจะประกอบด้วยโบรกเกอร์ (Broker) ผู้ให้บริการ (Service Provider) และผู้รับบริการ (Service Client) โดยโบรกเกอร์จะมีบทบาทในการกำหนดหัวข้อ (Topic) และบัญญัติเอกสาร WSDL ซึ่งเป็นมาตรฐาน (Canonical Message) สำหรับหัวข้อนั้นๆ เพื่อให้ผู้ให้บริการและผู้รับบริการนำไปใช้งานร่วมกัน

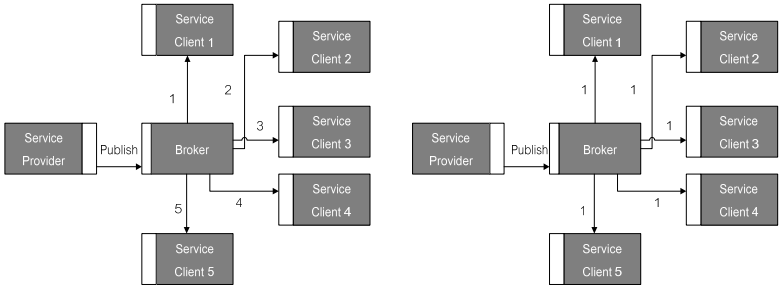


ภาพที่ 3.5 ภาพรวมในการเรียกใช้งานเว็บเซอร์วิสแบบ Push-Based
แหล่งที่มา: Thanisa, 2012: 31.

ในส่วนของผู้ให้บริการจะนำเอาเอกสาร WSDL ที่ต้องการส่งข้อมูลมาสร้างเป็น Proxy หรือ Stub เพื่อใช้ส่งข้อมูลผ่านมายังอินเตอร์เฟซของโบรกเกอร์และผู้รับบริการจะนำเอาเอกสาร WSDL เช่นเดียวกับผู้ให้บริการแต่จะมาสร้างเว็บเซอร์วิสขึ้นที่ฝั่งตนเองเพื่อใช้ในการรอรับข้อมูลที่ต้องการและทำการลงทะเบียนโดยแจ้งที่อยู่ (Endpoint) ของเว็บเซอร์วิสตนเองไปยังโบรกเกอร์ ดังนั้นเมื่อผู้ให้บริการมีการปรับปรุงข้อมูลก็จะเรียกใช้งาน Stub ส่งข้อมูลมายังอินเตอร์เฟซโบรกเกอร์และโบรกเกอร์ก็จะนำข้อมูลส่งต่อไปยังผู้รับบริการแต่ละรายที่ได้ทำการลงทะเบียนไว้ทำให้วิธีการนี้ขจัดปัญหาคอขวดเนื่องจากไม่มีการเขียนโปรแกรมเพื่อคอยตรวจสอบข้อมูลจากผู้รับบริการและผู้รับบริการได้รับข้อมูลที่ปรับปรุงล่าสุด

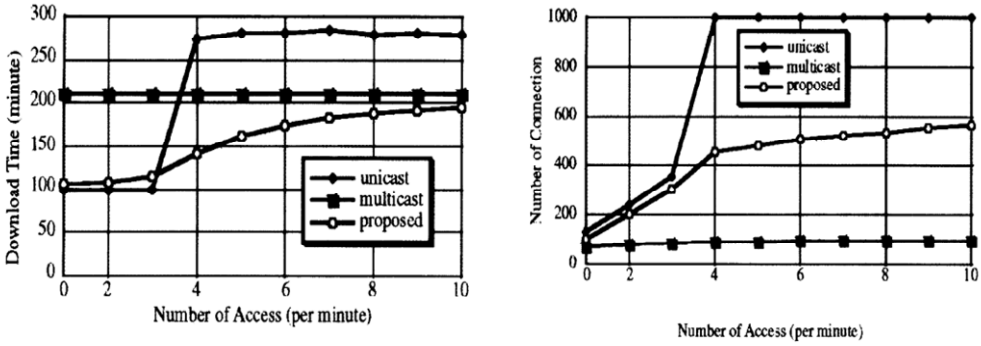
หากพิจารณาจะสังเกตว่า โบรกเกอร์ต้องทำการส่งข้อมูลตามจำนวนของผู้รับบริการที่มาลงทะเบียน เช่น ภาพที่ 3.6 โบรกเกอร์จะทำการส่งข้อมูลชุดเดียวกันไปยังผู้รับบริการจำนวน 5 ครั้ง โดยลักษณะการส่งข้อมูลเป็นแบบยูนิคาสต์ ซึ่งถ้าหากข้อมูลและหรือผู้รับบริการมีจำนวนมาก จะทำให้โบรกเกอร์ต้องส่งข้อมูล ชุดเดียวกันซ้ำๆ ในเครือข่าย เป็นผลให้มีการใช้งานเครือข่ายเกิน

ความจำเป็น ดังนั้นงานวิจัยฉบับนี้ ได้มีแนวคิด ต่อยอดในการประยุกต์ใช้เว็บเซอร์วิสให้เกิดประสิทธิภาพมากกว่า โดยเปลี่ยนรูปแบบการส่งข้อมูลจากยูนิคาสต์เป็นมัลติคาสต์ ซึ่งจะเป็นการส่งข้อมูลเพียงชุดเดียวและครั้งเดียวเข้าไปในเครือข่าย และผู้รับบริการได้รับข้อมูลพร้อมๆ กัน ดังภาพที่ 3.6



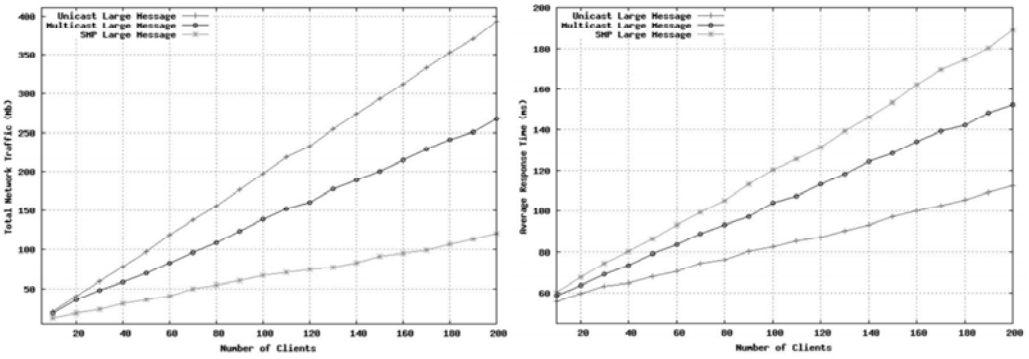
ภาพที่ 3.6 การส่งข้อมูลแบบยูนิคาสต์และมัลติคาสต์ของโบรกเกอร์

การส่งข้อมูลแบบมัลติคาสต์ถูกนำประยุกต์ใช้ในการประชุมระยะไกล ระบบกระจายข้อมูลโทรทัศน์ทางอินเตอร์เน็ต หรือการส่งข้อมูลจำนวนมาก เนื่องจากสามารถส่งข้อมูลชุดเดียวครั้งเดียวแต่มีผู้รับได้หลายคนพร้อมๆ กัน (Irie, Arakawa, Okamoto and Yamanaka, 2008: 5-9) เมื่อเปรียบเทียบประสิทธิภาพในการส่งเนื้อหา (Content) ระหว่างยูนิคาสต์และมัลติคาสต์พบว่า การส่งข้อมูลแบบมัลติคาสต์มีการทำงานที่ดีกว่า ดังภาพที่ 3.7 ได้นำเอาเว็บเซอร์วิสแบบ SOAP ทำการส่งข้อมูลโดยเปรียบเทียบการส่งแบบยูนิคาสต์และมัลติคาสต์ชี้ให้เห็นว่าแม้ขนาดข้อมูลหรือจำนวนผู้รับบริการจะมีขนาดเท่าใดก็ตามวิธีการส่งข้อมูลแบบมัลติคาสต์จะทำงานได้รวดเร็วและมีประสิทธิภาพมากกว่าดังภาพที่ 3.8



ภาพที่ 3.8 รูปภาพแสดงความเร็วในการโหลดข้อมูลและจำนวนของผู้ติดต่อ

แหล่งที่มา Irie, Arakawa, Okamoto and Yamanaka, 2008: 5, 8-9.

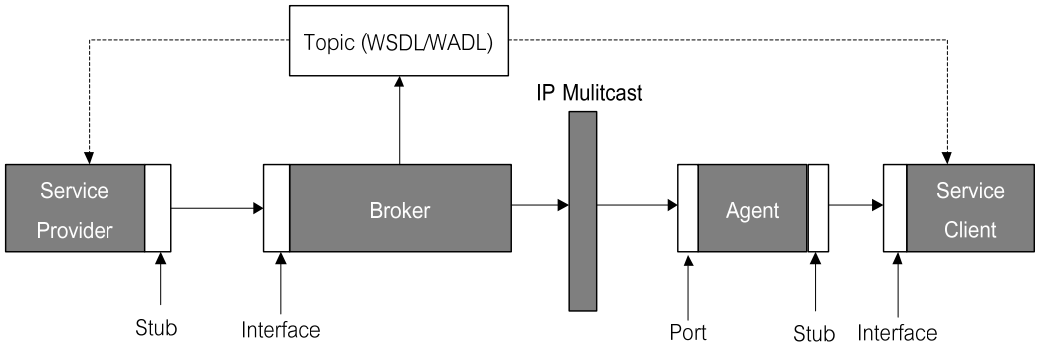


(ก.)เปรียบเทียบการส่งข้อมูล (ข.) ข้อมูลเฉลี่ยเปรียบเทียบการส่งข้อมูล

ภาพที่ 3.8 รูปภาพเปรียบเทียบความเร็วในการส่งข้อมูล

แหล่งที่มา: Phan, Tari and Bertok, 2008: 102.

ดังนั้นการทดลองในงานวิทยานิพนธ์ฉบับนี้จะนำเอาเว็บเซอร์วิสแบบ SOAP และ REST มาเปรียบเทียบด้วยเพื่อเป็นประโยชน์ในการเลือกนำมาใช้งานโดยจะมีองค์ประกอบดังภาพที่ 3.9 คือ ผู้ให้บริการ (Service Provider) โบรกเกอร์ (Broker) ผู้รับบริการ (Service Client) และเอเยนต์ (Agent) ซึ่งอธิบายรายละเอียดแต่ละองค์ประกอบตามตารางที่ 3.1



ภาพที่ 3.9 ภาพองค์ประกอบเว็บเซอร์วิสในการใช้งานแบบมัลติคาสต์

ตารางที่ 3.1 อธิบายแต่ละองค์ประกอบในการใช้งานเว็บเซอร์วิสแบบมัลติคาสต์

องค์ประกอบ	คำอธิบาย
โบรกเกอร์ (Broker)	ทำหน้าที่กำหนดหัวข้อ (Topic) และบัญญัติเอกสาร WSDL/WADL รับการละเบียนจากผู้รับบริการว่าได้เลือกหัวข้อไหนและที่อยู่ของเว็บเซอร์วิสเป็นอะไรจากนั้นเมื่อได้รับมาจากผู้ให้บริการจะทำการส่งข้อมูลผ่านไปยังไอพีมัลติคาสต์
ผู้ให้บริการ (Service Provider)	นำเอาเอกสาร WSDL/WADL สำหรับหัวข้อที่ต้องการส่งข้อมูลจากโบรกเกอร์มาสร้างเป็น Stub และเรียกใช้งานส่งข้อมูลผ่าน Stub
ผู้รับบริการ (Service Client)	นำเอาเอกสาร WSDL/WADL สำหรับหัวข้อที่ต้องการรับข้อมูลจากโบรกเกอร์แล้วมาสร้างเว็บเซอร์วิสเพื่อรอข้อมูลและลงทะเบียนรายละเอียดเว็บเซอร์วิสที่สร้างขึ้นไปยังโบรกเกอร์เพื่อรอข้อมูลที่จะถูกส่งมาจากเอเยนต์
เอเยนต์ (Agent)	อยู่ฝั่งเดียวกับผู้รับบริการทำหน้าที่รับข้อมูลจากไอพีมัลติคาสต์แล้วส่งไปยังเว็บเซอร์วิสที่ผู้รับบริการได้สร้างและลงทะเบียนไว้กับโบรกเกอร์

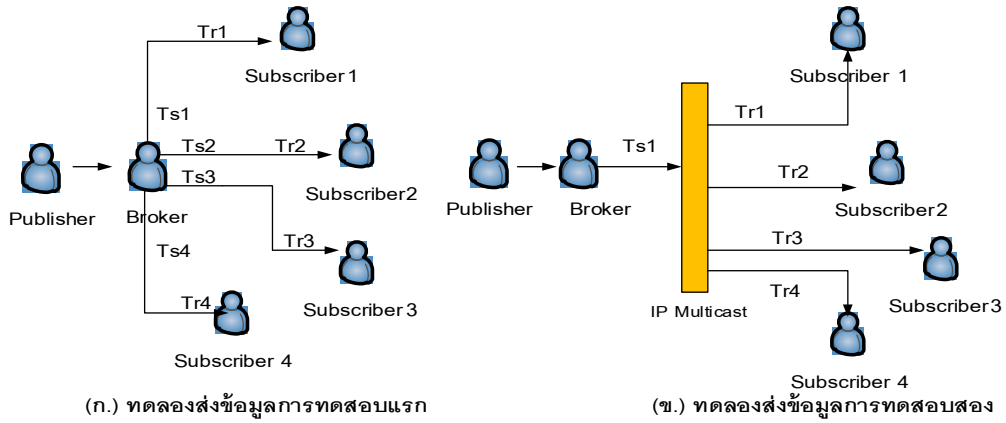
ในการทำงานโบรกเกอร์จะเป็นผู้กำหนดหัวข้อและบัญญัติเอกสาร WSDL/WADL ซึ่งผู้ให้บริการนำเอาไปสร้าง สตับไว้ที่ฝั่งของตน และผู้รับบริการนำเอามาสร้างเว็บเซอร์วิสที่ฝั่งของตน

โดยทำการลงทะเบียนที่อยู่ของเว็บเซอร์วิสที่สร้างไว้กับโบรกเกอร์ เมื่อข้อมูลมีการปรับปรุงเปลี่ยนแปลงผู้ให้บริการจะส่งข้อมูลมายังโบรกเกอร์ผ่านสตับที่ได้สร้างไว้ และเมื่อโบรกเกอร์ได้รับข้อมูลก็จะทำการส่งข้อมูลกระจายต่อไปยังผู้รับบริการที่ได้มาลงทะเบียนไว้ผ่านไอพีมัลติคาสต์ ซึ่งเป็นการส่งข้อมูลเพียงชุดเดียวและครั้งเดียว ในส่วนฝั่งผู้รับบริการจะมีโปรแกรมเอเยนต์คอยรับข้อมูลและส่งข้อมูลต่อไปยังเว็บเซอร์วิสตามที่ผู้รับบริการได้ลงทะเบียนไว้ ดังภาพที่ 3.9 ข้างต้น

3.2 ระเบียบวิธีการวิจัย

3.2.1 วิธีการวิจัย

จากแนวคิดในการเรียกใช้งานเว็บเซอร์วิสแบบผกผัน งานวิจัยชิ้นนี้ได้นำเสนอแนวคิดในการใช้งานเครือข่ายให้เกิดประสิทธิภาพสูงสุด โดยเปลี่ยนวิธีการส่งข้อมูลจากลักษณะเดิมที่เป็นแบบยูนิคาสต์เปลี่ยนเป็นแบบมัลติคาสต์โดยใช้ไอพีมัลติคาสต์ และผลการทดลองจะได้จากเวลาเปรียบเทียบการตอบสนองของเครื่องผู้รับบริการกับเวลาที่ถูกส่งไป โดยแบ่งการทดลองออกเป็น 2 ส่วน ซึ่งส่วนแรกใช้เว็บเซอร์วิสแบบ SOAP ส่งข้อมูลรูปแบบเดิมเปรียบเทียบกับการใช้งานผ่านไอพีมัลติคาสต์ และหัวข้อที่สองนำเว็บเซอร์วิสแบบ SOAP และ REST มาเปรียบเทียบโดยส่งข้อมูลผ่านไอพีมัลติคาสต์ ผลเวลาเปรียบเทียบของทั้งสองการทดลองจะเริ่มจับเวลาจากที่โบรกเกอร์เริ่มส่งข้อมูลและเวลาที่เครื่องผู้รับเริ่มได้รับข้อมูลจากนั้นมาเปรียบเทียบกันแล้วนำผลลัพธ์ที่ได้ของทุกๆ เครื่องที่ใช้ในการทดสอบครั้งนั้นมารวมกันเช่นทดสอบการส่งข้อมูลจำนวน 4 เครื่อง ดังภาพที่ 3.10 ผลที่ได้จากการทดลองมาจากเวลา



ภาพที่ 3.10 ตัวอย่างภาพที่ใช้ในการทดลอง

โดย Ts แทนเวลาที่ส่งไปยังผู้รับบริการแต่ละเครื่องและ Tr แทนเวลาที่เครื่องผู้รับข้อมูลแล้วนำผลมารวมกันเป็น Tt ดังนี้

$$Tt1 = (Ts1-Tr1) + (Ts2-Tr2) + (Ts3-Tr3) + (Ts4-Tr4)$$

ดังนั้นถ้ามีการทดสอบจำนวน 5 ครั้งโดยมีเครื่องทดสอบ 4 เครื่องผลการทดสอบจะมาจาก

$Tt1 = (Ts11 - Tr11) + (Ts12 - Tr12) + (Ts13 - Tr13) + (Ts14 - Tr14)$

$Tt2 = (Ts21 - Tr21) + (Ts22 - Tr22) + (Ts23 - Tr23) + (Ts24 - Tr24)$

$Tt3 = (Ts31 - Tr31) + (Ts32 - Tr32) + (Ts33 - Tr33) + (Ts34 - Tr34)$

$Tt4 = (Ts41 - Tr41) + (Ts42 - Tr42) + (Ts43 - Tr43) + (Ts44 - Tr44)$

$Tt5 = (Ts51 - Tr51) + (Ts52 - Tr52) + (Ts53 - Tr53) + (Ts54 - Tr54)$

สำหรับการส่งข้อมูลผ่านไอพีมัลติคาสต์เป็นลักษณะกระจายโดยส่งข้อมูลไปที่เดียวและครั้งเดียวที่ไอพีมัลติคาสต์ดังนั้นเวลาส่งทั้ง 4 เครื่องจะเป็นเวลาเดียวกันซึ่งผลการทดลองจะมาจากการคำนวณดังนี้

$$Tt1 = (Ts1-Tr1) + (Ts1-Tr2) + (Ts1-Tr3) + (Ts1-Tr4)$$

ดังนั้นถ้ามีจำนวนการทดสอบเหมือนกับข้างบนผลการทดสอบจะมาจาก

$Tt1 = (Ts1 - Tr11) + (Ts1- Tr12) + (Ts1 - Tr13) + (Ts1 - Tr14)$

$Tt2 = (Ts2 - Tr21) + (Ts2 - Tr22) + (Ts2 - Tr23) + (Ts2 - Tr24)$

$Tt3 = (Ts3 - Tr31) + (Ts3 - Tr32) + (Ts3 - Tr33) + (Ts3 - Tr34)$

$Tt4 = (Ts4 - Tr41) + (Ts4 - Tr42) + (Ts4 - Tr43) + (Ts4 - Tr44)$

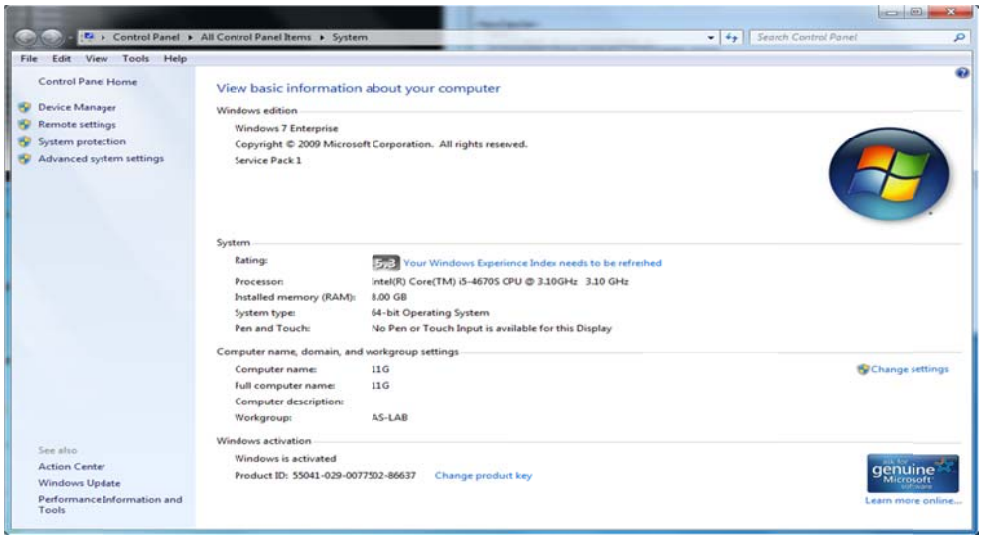
$Tt5 = (Ts5 - Tr51) + (Ts5 - Tr52) + (Ts5 - Tr53) + (Ts5 - Tr54)$

3.2.2 วิธีการทดลอง

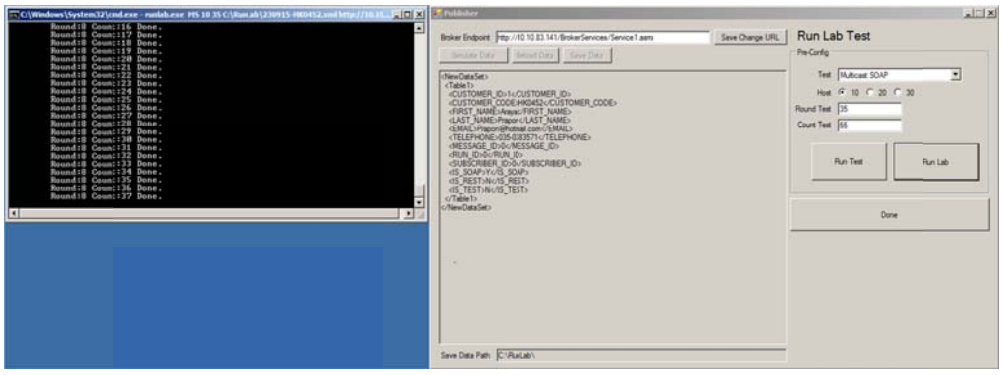
เครื่องที่ใช้ในการทดลองจะมีประสิทธิภาพทางด้านฮาร์ดแวร์และซอฟต์แวร์เหมือนกันคือใช้ระบบปฏิบัติการ Windows 7 Enterprise Sp1 หน่วยประมวลผลเป็น Intel Core i5-4670S หน่วยความจำขนาด 8 GB ดังภาพที่ 3.11 โดยโปรแกรมที่พัฒนาเพื่อใช้ในการทดลองครั้งได้ใช้ภาษา C# และ Visual Studio 2012 เป็นเครื่องมือในการพัฒนา ใช้ Microsoft SQL Server 2012 เป็นฐานข้อมูลในการเก็บข้อมูล

การทดสอบจะแบ่งเครื่องที่ใช้งานเป็น 3 ประเภทคือ เครื่องสำหรับ Publisher สำหรับโบรกเกอร์และสำหรับผู้รับบริการ โดยการทดสอบจะเริ่มจากเครื่อง Publisher ทำการส่งข้อมูลมายังเครื่องโบรกเกอร์ซึ่งใช้โปรแกรมดังภาพที่ 3.12 จะสังเกตเห็นว่ารูปแบบข้อมูลที่ส่งไปนั้นอยู่

ในรูปของ XML ซึ่งตลอดการทดสอบทั้งสองหัวข้อจะใช้ข้อมูลชุดเดียวกัน และในส่วนของโบรกเกอร์เมื่อรับข้อมูลจาก Publisher แล้วก็จะเริ่มทำการส่งข้อมูลไปยังเครื่องผู้รับบริการโดยในการส่งแต่ละครั้งจะมีการบันทึกเวลาลงในฐานข้อมูลดังตัวอย่างภาพที่ 3.13 ซึ่งภาพดังกล่าวเป็นการทดสอบแรกส่งข้อมูลแบบยูนิคาสต์จำนวน 10 เครื่องและสำหรับการทดสอบในส่วนที่เป็นไอพีมัลติคาสต์โปรแกรมที่ใช้ในการเขียนส่งข้อมูลไปยังไอพีมัลติคาสต์แสดงดังภาพที่ 3.14 สำหรับเครื่องผู้รับบริการได้แบ่งจำนวนเครื่องในการทดสอบเป็น 10 20 และ 30 เครื่องซึ่งเมื่อได้รับข้อมูลจากโบรกเกอร์ก็จะทำการบันทึกเวลาที่เริ่มได้รับข้อมูลลงฐานข้อมูลเพื่อให้ได้ผลลัพธ์ในการเปรียบเทียบผลการทดลอง



ภาพที่ 3.11 แสดงรายละเอียดเครื่องที่ใช้ในการทดลอง



ภาพที่ 3.12 แสดงโปรแกรม Publisher ที่ใช้ส่งข้อมูลไปยังโบรกเกอร์

	MESSAGE_ID	RUN_ID	CREATED_ON
1	1	1	2015-09-23 11:30:41.053

	MESSAGE_DTL_ID	MESSAGE_ID	SUBSCRIBER_ID	MESSAGE_SEND	MESSAGE_RECEIVE
1	1	1	1	2015-09-23 11:30:41.053	2015-09-23 11:30:41.110
2	2	1	2	2015-09-23 11:30:41.290	2015-09-23 11:30:41.333
3	3	1	3	2015-09-23 11:30:41.480	2015-09-23 11:30:41.537
4	4	1	4	2015-09-23 11:30:41.673	2015-09-23 11:30:41.713
5	5	1	5	2015-09-23 11:30:41.923	2015-09-23 11:30:41.980
6	6	1	6	2015-09-23 11:30:42.183	2015-09-23 11:30:42.260
7	7	1	7	2015-09-23 11:30:42.390	2015-09-23 11:30:42.447
8	8	1	8	2015-09-23 11:30:42.597	2015-09-23 11:30:42.657
9	9	1	9	2015-09-23 11:30:42.647	2015-09-23 11:30:42.687
10	10	1	10	2015-09-23 11:30:42.690	2015-09-23 11:30:42.740

ภาพที่ 3.13 แสดงข้อมูลที่ถูกจัดเก็บลงฐานข้อมูล

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using System.IO;
using System.IO.Compression;
using System.Net;
using System.Net.Sockets;

namespace TransmissionData
{
    public static class MulticastSend
    {
        public static void SendData(string ip, int port, DataSet data) {
            IPAddress mcIP = IPAddress.Parse(ip);
            IPEndPoint ipep = new IPEndPoint(mcIP, port);
            int ttl = 1;
            Socket sock = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
            sock.SetSocketOption(SocketOptionLevel.IP, SocketOptionName.MulticastTimeToLive, ttl);
            byte[] dataSend = ConversionData.Compress(data);
            sock.SendTo(dataSend, 0, dataSend.Length, SocketFlags.None, ipep);
            sock.Close();
        }

    }
}
```

ภาพที่ 3.14 แสดงตัวอย่าง Source Code ที่ส่งข้อมูลไปยังไอพีมัลติคาสต์

บทที่ 4

ผลการทดลอง

ในบทนี้เป็นการเปรียบเทียบผลการทดลองเพื่อพิสูจน์ว่าวิธีการใดสามารถประยุกต์ใช้งานเว็บเซอร์วิสให้มีประสิทธิภาพที่ดีกว่าซึ่งในการทดลองได้แบ่งการทดสอบออกเป็น 2 ส่วน คือ ส่วนแรกเป็นการนำเอาเว็บเซอร์วิสแบบ SOAP มาเปรียบเทียบการส่งข้อมูลแบบยูนิคาสต์กับมัลติคาสต์ และส่วนที่สองเปรียบเทียบเว็บเซอร์วิสแบบ SOAP และ REST โดยส่งข้อมูลแบบมัลติคาสต์ ซึ่งการทดลองแบบมัลติคาสต์ทุกส่วนจะกระทำผ่านไอพีมัลติคาสต์ การทดสอบในแต่ละส่วนจะทดลอง 10 ครั้งและในแต่ละครั้งจะเก็บข้อมูลการส่ง 10 ครั้ง มีหน่วยเป็น Millisecond ซึ่งผลการทดลองมีรายละเอียดดังต่อไปนี้

4.1 ผลการทดลองส่วนแรก

การทดลองในส่วนแรกเป็นการนำเอาเว็บเซอร์วิสแบบ SOAP มาเปรียบเทียบในการส่งข้อมูลยูนิคาสต์และมัลติคาสต์ซึ่งผลการทดสอบในการส่งข้อมูลแบบยูนิคาสต์แสดงดังตารางที่ 4.1 สำหรับจำนวนผู้รับบริการ 10 เครื่อง ตารางที่ 4.2 สำหรับผู้รับบริการ 20 เครื่องและตารางที่ 4.3 สำหรับผู้รับบริการจำนวน 30 เครื่องและผลการทดสอบในการส่งข้อมูลแบบมัลติคาสต์จะแสดงดังตารางที่ 4.4 สำหรับจำนวนผู้รับบริการ 10 ตารางที่ 4.5 สำหรับจำนวนผู้รับบริการ 20 เครื่องและตารางที่ 4.6 สำหรับผู้รับบริการ 30 เครื่องตามลำดับ

ตารางที่ 4.1 เว็บเซอร์วิสแบบ SOAP ส่งข้อมูลแบบยูนิคาสต์สำหรับผู้รับบริการ 10 เครื่อง

ครั้ง	รอบการทดสอบ									
ทดสอบ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	533	634	586	556	549	606	583	581	586	586

ตารางที่ 4.1 (ต่อ)

ครั้ง	รอบการทดสอบ									
ทดสอบ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	548	603	669	617	607	617	595	650	569	626
3	591	573	550	579	562	580	634	643	597	597
4	596	532	561	560	569	609	570	516	593	601
5	606	630	599	617	677	584	569	605	627	614
6	590	580	497	553	549	612	657	570	617	526
7	559	621	629	610	610	562	576	573	643	646
8	581	633	610	561	587	626	605	513	589	483
9	549	590	670	650	654	601	564	553	522	570
10	542	677	623	600	537	607	599	596	557	566

ตารางที่ 4.2 เว็บเซอร์วิสแบบ SOAP ส่งข้อมูลแบบยูนิคาสต์สำหรับผู้รับบริการ 20 เครื่อง

ครั้ง	รอบการทดสอบ									
ทดสอบ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1231	1202	1166	1236	1316	1180	1230	1135	1194	1087
2	1215	1022	1191	1145	1191	1094	1262	1196	1208	1125
3	1243	1163	1202	1266	1084	1149	1219	1159	1161	1138
4	1208	1157	1163	1224	1192	1289	1214	1202	1082	1226
5	1255	1086	1123	1134	1221	1106	1276	1081	1196	1132
6	1112	1209	1246	1093	1152	1265	1197	1137	1242	1187
7	1198	1188	1057	1170	1218	1116	1265	1233	1198	1212
8	1091	1218	1218	1184	1135	1172	1179	1122	1179	1168
9	1162	1106	1132	1243	1144	1282	1138	1241	1160	1136
10	1262	1278	1197	1150	1171	1167	1193	1129	1201	1215

ตารางที่ 4.3 เว็บเซอร์วิสแบบ SOAP ส่งข้อมูลแบบยูนิคาสต์สำหรับผู้รับบริการ 30 เครื่อง

ครั้ง	รอบการทดสอบ									
ทดสอบ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1853	1876	1848	1742	1829	1862	1763	1671	1872	1723
2	1876	1729	1722	1766	1741	1737	1783	1871	1900	1795
3	1869	1926	1824	1832	1825	1801	1820	1710	1783	1696
4	1726	1818	1745	1806	1712	1883	1746	1789	1745	1762
5	1732	1868	1781	1825	1912	1867	1717	1694	1897	1793
6	1754	1862	1803	1733	1695	1770	1687	1849	1916	1693
7	1787	1775	1888	1850	1790	1760	1848	1760	1787	1754
8	1664	1882	1817	1746	1859	1774	1776	1821	1839	1867
9	1793	1740	1758	1740	1736	1720	1717	1766	1656	1824
10	1766	1761	1807	1823	1703	1759	1750	1749	1915	1828

ตารางที่ 4.4 เว็บเซอร์วิสแบบ SOAP ส่งข้อมูลแบบมัลติคาสต์สำหรับผู้รับบริการ 10 เครื่อง

ครั้ง	รอบการทดสอบ									
ทดสอบ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	229	229	206	212	248	201	212	228	192	207
2	205	196	211	209	236	210	213	216	207	219
3	221	222	234	203	239	229	232	227	213	236
4	202	232	226	234	223	223	240	196	212	230
5	219	225	223	244	233	206	230	225	233	208
6	223	199	227	250	240	223	229	219	227	231
7	210	223	194	215	203	210	215	230	229	219
8	214	226	194	200	239	191	227	246	235	216
9	216	220	225	211	222	253	227	225	226	207
10	227	228	237	226	245	234	222	217	226	228

ตารางที่ 4.5 เว็บเซอร์วิสแบบ SOAP ส่งข้อมูลแบบมัลติคาสต์สำหรับผู้รับบริการ 20 เครื่อง

ครั้ง	รอบการทดสอบ									
ทดสอบ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	452	408	427	459	444	451	406	469	435	416
2	446	426	438	460	472	462	411	450	449	459
3	430	436	464	469	456	441	462	429	428	405
4	445	406	452	425	452	445	434	444	457	419
5	455	415	446	471	462	442	470	430	428	460
6	463	447	427	407	452	479	456	421	443	465
7	455	451	443	442	469	413	472	425	475	463
8	478	416	438	433	428	492	444	434	449	438
9	443	430	453	496	444	441	486	473	458	457
10	481	463	460	433	442	440	444	475	432	442

ตารางที่ 4.6 เว็บเซอร์วิสแบบ SOAP ส่งข้อมูลแบบมัลติคาสต์สำหรับผู้รับบริการ 30 เครื่อง

ครั้ง	รอบการทดสอบ									
ทดสอบ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	674	672	723	669	679	648	657	692	650	643
2	632	655	663	636	656	670	686	631	659	682
3	674	662	687	659	656	638	673	665	659	665
4	679	656	636	653	690	671	643	691	659	661
5	623	655	673	645	636	640	665	619	674	674
6	657	670	658	719	691	666	667	671	637	692
7	637	658	659	673	656	675	671	656	672	648
8	676	722	666	629	642	657	676	688	675	673
9	699	628	655	670	702	652	634	679	654	689
10	663	649	662	641	635	668	700	668	648	692

นำค่าเฉลี่ยของการทดลองของแต่ละตารางนำมาสร้างเป็นกราฟดังภาพที่ 4.1 จะพบว่าการใช้งานไอพีมัลติคาสต์ให้เวลาตอบสนองที่มีประสิทธิภาพมากกว่าการส่งข้อมูลรูปแบบเดิม

ภาพที่ 4.1 แสดงกราฟเปรียบเวลาตอบสนองในการทดลองแรก

4.2 ผลการทดลองส่วนที่สอง

การทดลองในส่วนที่สองเป็นการนำเว็บเซอร์วิสที่ใช้งานอยู่ในปัจจุบันคือ SOAP และ REST มาดำเนินการเปรียบเทียบส่งข้อมูลแบบมัลติคาสต์ ซึ่งผลการทดสอบของเว็บเซอร์วิสแบบ SOAP แสดงดังตารางที่ 4.7 สำหรับผู้รับบริการ 20 เครื่อง ตารางที่ 4.8 สำหรับผู้รับบริการ 20 เครื่อง และตารางที่ 4.9 สำหรับผู้รับบริการ 30 เครื่องและผลการทดสอบของเว็บเซอร์วิสแบบ REST แสดงดังตารางที่ 4.10 สำหรับผู้รับบริการ 10 เครื่อง ตารางที่ 4.11 สำหรับผู้รับบริการ 20 เครื่องและตารางที่ 4.12 สำหรับผู้รับบริการ 30 เครื่องตามลำดับ

ตารางที่ 4.7 เว็บเซอร์วิสแบบ SOAP ส่งข้อมูลแบบมัลติคาสต์สำหรับผู้รับบริการ 10 เครื่อง

ครั้ง	รอบการทดสอบ									
ทดสอบ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	202	216	223	216	218	263	230	234	228	220
2	208	216	220	223	259	236	212	210	228	242
3	233	233	254	227	231	229	222	223	229	236
4	209	212	239	232	226	242	230	233	219	226
5	236	215	209	209	238	222	210	230	233	218
6	210	238	226	218	208	198	217	210	254	223
7	246	229	235	232	219	218	206	222	223	227
8	239	224	223	212	210	237	210	199	220	233
9	238	207	230	226	220	209	226	219	239	233
10	239	209	215	218	210	240	222	222	230	219

ตารางที่ 4.8 เว็บเซอร์วิสแบบ SOAP ส่งข้อมูลแบบมัลติคาสต์สำหรับผู้รับบริการ 20 เครื่อง

ครั้ง	รอบการทดสอบ									
ทดสอบ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	438	449	414	479	454	452	464	414	459	475
2	442	438	454	414	419	420	419	423	417	425
3	411	448	436	428	455	427	454	443	475	465
4	455	439	463	456	461	420	447	415	436	459
5	451	446	476	424	439	443	461	449	442	462
6	422	427	496	439	449	444	460	458	446	432
7	472	438	442	441	455	442	420	437	425	458
8	442	482	427	443	499	462	427	407	454	472
9	442	438	444	439	453	433	452	432	451	449
10	454	461	455	463	413	430	484	447	471	442

ตารางที่ 4.9 เว็บเซอร์วิสแบบ SOAP ส่งข้อมูลแบบมัลติคาสต์สำหรับผู้รับบริการ 30 เครื่อง

ครั้ง	รอบการทดสอบ									
ทดสอบ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	702	677	654	704	642	676	615	662	650	654
2	685	673	672	706	615	658	627	643	631	651
3	691	689	671	679	696	684	631	662	671	640
4	662	664	676	682	690	637	645	684	636	680
5	705	662	672	704	664	642	667	640	642	623
6	675	675	682	647	647	632	679	659	679	716
7	696	669	693	649	667	640	685	683	653	708
8	656	674	685	674	638	618	675	650	658	699
9	636	679	666	678	668	696	668	692	694	669
10	664	678	681	638	656	679	659	690	659	697

ตารางที่ 4.10 เว็บเซอร์วิสแบบ REST ส่งข้อมูลแบบมัลติคาสต์สำหรับผู้รับบริการ 10 เครื่อง

ครั้ง	รอบการทดสอบ									
ทดสอบ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	52	69	61	51	45	54	52	51	70	69
2	52	69	59	58	55	52	66	56	65	51
3	55	46	59	48	52	65	55	62	48	51
4	65	66	49	56	52	63	54	54	51	62
5	59	58	62	66	52	51	62	55	54	61
6	58	55	53	49	55	48	57	61	69	69
7	63	48	58	49	52	52	54	49	52	51
8	61	64	54	52	65	45	54	61	59	52
9	64	61	66	52	62	51	61	65	58	63
10	51	45	45	53	57	58	53	46	58	66

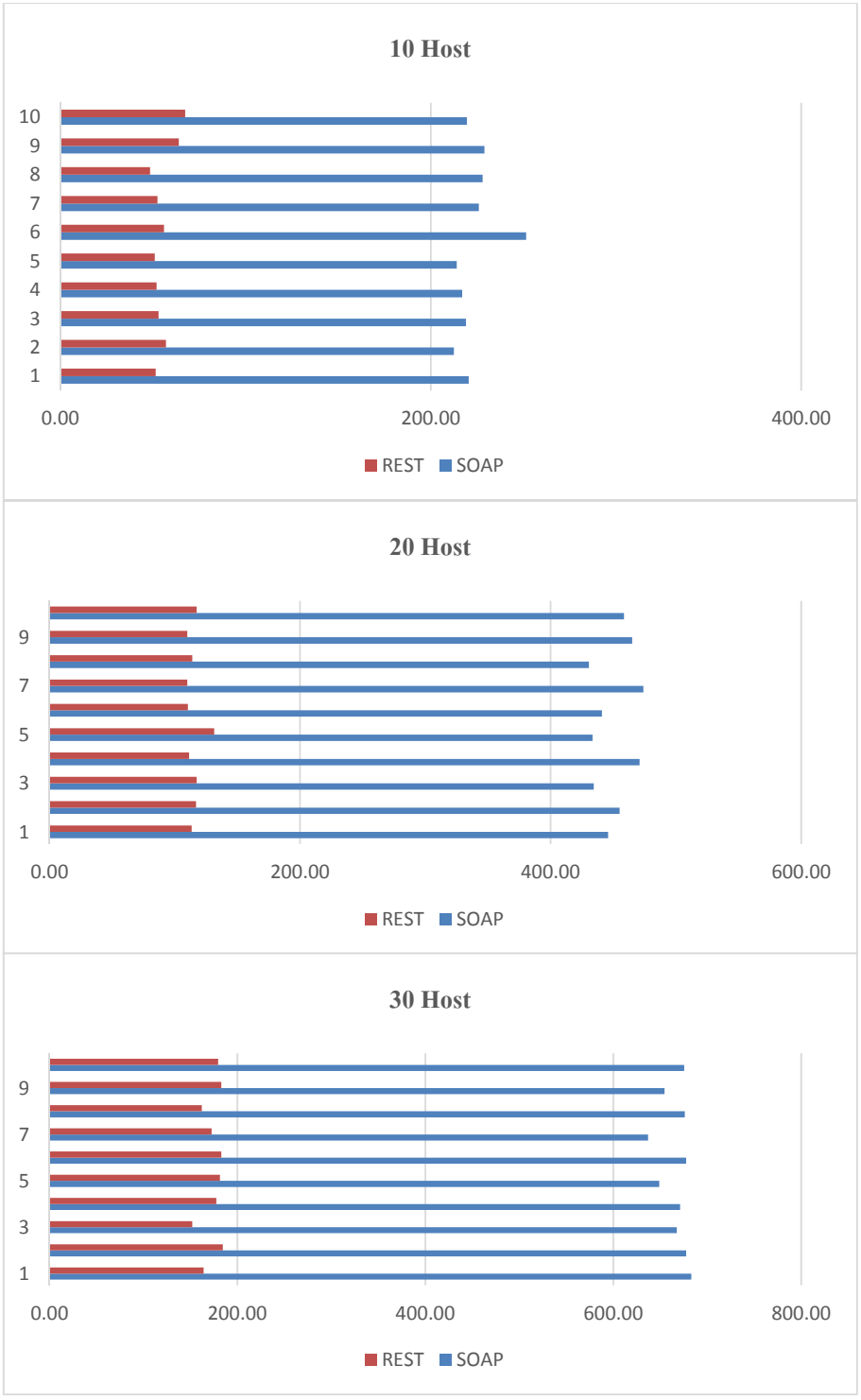
ตารางที่ 4.11 เว็บเซอร์วิสแบบ REST ส่งข้อมูลแบบมัลติคาสต์สำหรับผู้รับบริการ 20 เครื่อง

ครั้ง	รอบการทดสอบ									
ทดสอบ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	110	110	121	116	128	110	111	108	115	110
2	107	121	103	100	107	122	101	106	122	118
3	130	109	108	100	103	114	124	132	116	103
4	114	120	126	114	121	123	92	101	126	119
5	119	100	124	127	116	106	107	139	110	123
6	101	121	103	128	103	107	107	124	98	115
7	106	108	106	110	123	107	114	107	127	110
8	114	128	119	120	121	110	102	134	117	103
9	120	120	117	109	114	100	125	111	101	118
10	117	124	114	107	135	111	109	120	105	125

ตารางที่ 4.12 เว็บเซอร์วิสแบบ REST ส่งข้อมูลแบบมัลติคาสต์สำหรับผู้รับบริการ 30 เครื่อง

ครั้ง	รอบการทดสอบ									
ทดสอบ	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	170	189	168	180	190	186	185	151	183	193
2	179	185	151	169	177	176	160	173	188	159
3	177	157	169	173	178	177	167	170	167	173
4	164	153	185	171	165	171	159	160	141	169
5	187	180	175	155	168	155	157	155	192	178
6	160	160	194	179	182	172	162	172	164	167
7	172	150	154	179	153	180	176	154	187	147
8	173	165	189	168	162	172	175	170	156	182
9	153	182	161	168	165	159	173	147	173	162
10	158	180	136	175	173	180	160	173	183	166

เมื่อนำเอาแต่ละตารางของการทดลองมาหาค่าเฉลี่ยแล้วสร้างเป็นกราฟดังภาพที่ 4.2 จะพบว่าการใช้งานเว็บเซอร์วิสแบบ REST สามารถทำงานได้รวดเร็วกว่าเว็บเซอร์วิสแบบ SOAP



ภาพที่ 4.2 แสดงกราฟเปรียบเวลาตอบสนองในการทดลองที่สอง

บทที่ 5

สรุปผลวิจัยและข้อเสนอแนะ

มัลติคาสต์เป็นการส่งข้อมูลจากแหล่งหนึ่งและกระจายไปยังผู้รับพร้อมกันซึ่งเป็นการส่งข้อมูลออกไปเพียงครั้งเดียวและชุดเดียวโดยผู้รับนั้นจะได้รับข้อมูลพร้อมๆ กัน และจากการทดลองในบทที่ผ่านมาได้แบ่งการทดลองออกเป็น 2 ส่วนคือ ใช้เว็บเซอร์วิสแบบ SOAP เปรียบเทียบการส่งข้อมูลระหว่างยูนิคาสต์กับมัลติคาสต์และใช้เว็บเซอร์วิสแบบ SOAP และ REST มาเปรียบเทียบการส่งข้อมูลแบบมัลติคาสต์โดยจะใช้เวลาตอบสนองเปรียบเทียบผลการทดลองเพื่อให้มีการประยุกต์ใช้งานเว็บเซอร์วิสที่ใช้งานเครือข่ายอย่างมีประสิทธิภาพสูงสุด

5.1 สรุปผลวิจัย

จากผลการทดลองแรกที่นำเอาเว็บเซอร์วิสแบบ SOAP มาเปรียบเทียบการส่งข้อมูลแบบเดิมกับการใช้ไอพีมัลติคาสต์จะพบว่าการส่งข้อมูลโดยใช้ไอพีมัลติคาสต์ให้เวลาตอบสนองที่ดีกว่าเปรียบเทียบเป็นเปอร์เซ็นต์ดังตารางที่ 5.1, 5.2 และ 5.3 สำหรับการทดลองจำนวนเครื่อง 10 20 และ 30 ตามลำดับ จากการที่ไอพีมัลติคาสต์ให้เวลาตอบสนองที่ดีกว่าการส่งข้อมูลแบบยูนิคาสต์เนื่องจากตัวโปรโตคอลถูกออกมาให้รองรับการส่งข้อมูลขนาดใหญ่หรือข้อมูลจำนวนมากโดยเป็นการส่งข้อมูลออกไปเพียงชุดเดียวและครั้งเดียวจากจุดหนึ่งไปยังผู้รับหลายจุดซึ่งจะได้รับข้อมูลพร้อมๆ กันโดยไม่มีการตรวจสอบว่าผู้รับได้รับข้อมูลครบถ้วนหรือไม่ซึ่งจะแตกต่างจากการส่งข้อมูลแบบยูนิคาสต์ที่ต้องมีการสร้างทางเชื่อม ตรวจสอบข้อมูลของฝั่งผู้รับว่าได้รับครบถ้วนหรือเปล่าจึงทำให้การส่งข้อมูลในแต่ละครั้งนั้นใช้เวลามากกว่า

ตารางที่ 5.1 เปรียบเทียบความเร็วในการส่งข้อมูลการทดลองแรกจำนวน 10 เครื่องเป็นเปอร์เซ็นต์

ครั้งที่	เวลาเฉลี่ย 10 เครื่อง			
ทดสอบ	Unicast	Multicast	ผลต่าง	Multicast Better than (%)
1	537.50	228.00	309.50	57.58
2	655.50	228.50	427.00	65.14
3	604.50	221.50	383.00	63.36
4	578.00	219.00	359.00	62.11
5	543.00	246.50	296.50	54.60
6	606.50	217.50	389.00	64.14
7	591.00	217.00	374.00	63.28
8	588.50	222.50	366.00	62.19
9	571.50	209.00	362.50	63.43
10	576.00	217.50	358.50	62.24

ตารางที่ 5.2 เปรียบเทียบความเร็วในการส่งข้อมูลการทดลองแรกจำนวน 20 เครื่องเป็นเปอร์เซ็นต์

ครั้งที่	เวลาเฉลี่ย 20 เครื่อง			
ทดสอบ	Unicast	Multicast	ผลต่าง	Multicast Better than (%)
1	1246.50	466.50	780.00	62.58
2	1240.00	435.50	804.50	64.88
3	1181.50	443.50	738.00	62.46
4	1193.00	446.00	747.00	62.62
5	1243.50	443.00	800.50	64.37
6	1173.50	445.50	728.00	62.04
7	1211.50	425.00	786.50	64.92
8	1132.00	472.00	660.00	58.30
9	1197.50	433.50	764.00	63.80
10	1151.00	429.00	722.00	62.73

ตารางที่ 5.3 เปรียบเทียบความเร็วในการส่งข้อมูลการทดลองแรกจำนวน 30 เครื่องเป็นเปอร์เซ็นต์

ครั้งที่	เวลาเฉลี่ย 30 เครื่อง			
ทดสอบ	Unicast	Multicast	ผลต่าง	Multicast Better than (%)
1	1809.50	668.50	1141.00	63.06
2	1818.50	660.50	1158.00	63.68
3	1827.50	692.50	1135.00	62.11
4	1782.50	655.00	1127.50	63.25
5	1766.00	657.00	1109.00	62.80
6	1810.50	658.00	1152.50	63.66
7	1756.50	678.50	1078.00	61.37
8	1710.00	680.00	1030.00	60.23
9	1893.50	649.00	1244.50	65.72
10	1775.50	667.50	1108.00	62.40

ในส่วนการทดลองที่สองนำเว็บเซอร์วิสแบบ SOAP และ REST มาเปรียบเทียบการส่งข้อมูลแบบมัลติคาสต์จะพบว่าเว็บเซอร์วิสแบบ REST ให้เวลาตอบสนองที่ดีกว่าเปรียบเทียบเป็นเปอร์เซ็นต์ดังตารางที่ 5.4 5.5 และ 5.6 สำหรับการทดลองจำนวนเครื่อง 10 20 และ 30 ตามลำดับ การที่เว็บเซอร์วิสแบบ SOAP มีเวลาที่ตอบสนองที่ช้ากว่าเว็บเซอร์วิสแบบ REST เนื่องจากเว็บเซอร์วิสแบบ SOAP เวลารับส่งต้องแปลงให้อยู่รูปโปรโตคอล SOAP และทำให้ขนาดข้อมูลที่ส่งมีขนาดที่ใหญ่กว่าเว็บเซอร์วิสแบบ REST

ตารางที่ 5.4 เปรียบเทียบความเร็วในการส่งข้อมูลการทดลองสองจำนวน 10 เครื่องเป็นเปอร์เซ็นต์

ครั้งที่	เวลาเฉลี่ย 10 เครื่อง			
ทดสอบ	SOAP	REST	ผลต่าง	REST Better than (%)
1	220.50	51.50	169.00	76.64
2	212.50	57.00	155.50	73.18
3	219.00	53.00	166.00	75.80

ตารางที่ 5.4 (ต่อ)

ครั้งที่	เวลาเฉลี่ย 10 เครื่อง			
ทดสอบ	**SOAP**	**REST**	**ผลต่าง**	**REST Better than (%)**
4	217.00	52.00	165.00	76.04
5	214.00	51.00	163.00	76.17
6	251.50	56.00	195.50	77.73
7	226.00	52.50	173.50	76.77
8	228.00	48.50	179.50	78.73
9	229.00	64.00	165.00	72.05
10	219.50	67.50	152.00	69.25

ตารางที่ 5.5 เปรียบเทียบความเร็วในการส่งข้อมูลการทดลองสองจำนวน 20 เครื่องเป็นเปอร์เซ็นต์

ครั้งที่	เวลาเฉลี่ย 20 เครื่อง			
ทดสอบ	**SOAP**	**REST**	**ผลต่าง**	**REST Better than (%)**
1	220.50	51.50	169.00	76.64
2	212.50	57.00	155.50	73.18
3	219.00	53.00	166.00	75.80
4	217.00	52.00	165.00	76.04
5	214.00	51.00	163.00	76.17
6	251.50	56.00	195.50	77.73
7	226.00	52.50	173.50	76.77
8	228.00	48.50	179.50	78.73
9	229.00	64.00	165.00	72.05
10	219.50	67.50	152.00	69.25

ตารางที่ 5.6 เปรียบเทียบความเร็วในการส่งข้อมูลการทดลองสองจำนวน 30 เครื่องเป็นเปอร์เซ็นต์

ครั้งที่	เวลาเฉลี่ย 30 เครื่อง			
ทดสอบ	**SOAP**	**REST**	**ผลต่าง**	**REST Better than (%)**
1	683.00	164.00	519.00	75.99
2	677.50	184.50	493.00	72.77
3	667.50	152.00	515.50	77.23
4	671.00	177.50	493.50	73.55
5	649.00	181.50	467.50	72.03
6	677.50	183.00	494.50	72.99
7	637.00	172.50	464.50	72.92
8	676.00	162.00	514.00	76.04
9	654.50	183.00	471.50	72.04
10	675.50	179.50	496.00	73.43

5.2 ข้อเสนอแนะ

การทดสอบมัลติคาสต์โดยใช้ไอพีมัลติคาสต์มีข้อจำกัดคือเป็นรูปแบบการส่งข้อมูลทางเดียวซึ่งผู้ส่งทำการส่งข้อมูลออกไปแล้วไม่ทราบว่าผู้รับจะได้รับข้อมูลครบถ้วนหรือไม่ ดังนั้นหากมีโปรโตคอลหรือเทคนิคอื่นที่สามารถนำมาประยุกต์เพื่อยืนยันการรับส่งข้อมูลระหว่างกันจะทำให้การประยุกต์ใช้งานเว็บเซอร์วิสมีประสิทธิภาพสูงขึ้นไปอีก

บรรณานุกรม

สำนักงานส่งเสริมอุตสาหกรรมซอฟต์แวร์แห่งชาติ (องค์การมหาชน). 2555 (เมษายน 2555). **สรุปผลการสำรวจตลาดเทคโนโลยีสารสนเทศและการสื่อสารของประเทศไทย ประจำปี 2554 และประมาณการปี2555.** กรุงเทพฯ: กระทรวงเทคโนโลยีและสารสนเทศ.

มูลนิธิสถาบันวิจัยเพื่อการพัฒนาประเทศไทย. 2557. **โครงการสำรวจข้อมูลตลาดซอฟต์แวร์และบริการซอฟต์แวร์ประจำปี 2556 และคาดการณ์ปี 2557.** เอกสารวิจัย เสนอต่อสำนักงานส่งเสริมอุตสาหกรรมซอฟต์แวร์แห่งชาติ.

Bishop, Toni A. and Karne Ramesh K. n.d. **A Survey Of Middleware.** Retrieved January 31, 2012 from http://triton.towson.edu/~karne/research/middlew/surveym.pdf

Brenna, L. and Johansen D. 2005. **Engineering Push-Based Web Services.** Retrieved January 31, 2012 from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.61.649

Feng, Xiwei; Xue, Feng and Zhang, Tongwei. 2009. Research on Data Exchange Push Technology Based On Message-Driven. **International Join Conference On Artificial Intelligence.** Hainan Islan, Chaina: IEEE. Pp. 563-566.

Fenner, James. 2002. **Enterprise Application Integration Techniques.** Retrieved January 31, 2012 from http://www.cs.ucl.ac.uk/staff/ucacwxe/lectures/3C05-02-03/aswe21-essay.pdf

Fielding, Roy T. 2002. **Architectural Styles and Design of Network-based Software Architectures.** Doctoral dissertation, University of California, Irvine.

Irie, K.; Arakawa, Y.; Okamoto, S. and Yamanaka, N. 2008. Efficient Contents Delivery Method with Schedule Unicast and Multicast. **In APCC 2008. 14th Asia-Pacific Conference.** Tokyo: IEEE. Pp. 1-5.

Linthicum, David S. 1999. Enterprise Application Integration. Boston: Addison Wesley.

Makofske, David and Almeroth, Kevin. 2002. **Multicast Socket Practice Guide For Programmer.** San Francisco: Morgan Kaufmann.

Microsoft TechNet. n.d. **The TCP/IP Model.** Retrieved January 31, 2012 from https://technet.microsoft.com/en-us/library/cc786900(v=ws.10).aspx

Phan, Khoi A.; Tari, Zahir and Bertok Peter. 2008. Similarity-Based SOAP Multicast Protocol To Reduce Bandwidth and Latency In Web Services. **In Services Computing, IEEE Transaction.** Volume 1, No 2, April-June 2008: IEEE. Pp. 88-103.

Skjervold, E.; Hafsøe, T.; Johnsen, F.T. and Lund, K. 2010. Enabling Publish/Subscribe with COTS Web Services across Heterogeneous Networks. **2010 IEEE International Conference On Web Services.** Miami, FL: IEEE. Pp. 660-668.

Thanisa, Noomnonda. 2012. **Inversion of Web Service invocation Using Publish/Subscribe Push-based Architecture.** Doctoral dissertation, Nation Institute of Development Administration.

World Wide Web Consortium (W3C). n.d.a **SOAP.** Retrieved January 31, 2012 from http://www.w3schools.com/soap/default.asp

World Wide Web Consortium (W3C). n.d.b **Web Services.** Retrieved January 31, 2012 from http://www.w3schools.com/webservices/default.asp

World Wide Web Consortium (W3C). n.d.c **WSDL (Web Services Description Language).** Retrieved January 31, 2012 from http://www.w3schools.com/wsdl/default.asp

Wikipedia. 2001. **Internet Protocol Suite.** Retrieved January 31, 2012 from http://en.wikipedia.org/wiki/Internet_protocol_suite

Wikipedia. 2001. **Representational State Transfer.** Retrieved January 31, 2012 from http://en.wikipedia.org/wiki/Internet_protocol_suite

Wikipedia. 2002. **Multicast.** Retrieved January 31, 2012 from http://en.wikipedia.org/wiki/Multicast

Zuehlke, Lara. 2011. **The History of Enterprise Software.** Retrieved January 31, 2012 from http://ithistory.org/blog/?p=1660

ประวัติผู้เขียน

ชื่อ-ชื่อสกุล	นาย ฟาร์เดล เจ๊ะเต็ง
ประวัติการศึกษา	วิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์) มหาวิทยาลัยรามคำแหง ปีที่สำเร็จการศึกษา 2546
ประสบการณ์ทำงาน	พ.ศ. 2550-2558 ทีมพัฒนาระบบ RIS บริษัท มิราเคิล แอดวานซ์ เทคโนโลยี จำกัด ถนน ลาดพร้าว แขวง คลองเจ้าคุณสิงห์ กรุงเทพมหานคร 10310