เครื่องมือทดสอบความสามารถการทำงานของ WEB APPLICATION

WEB APPLICATION LOAD TESTING TOOL

พรภิณัฐ แก้วเมือง

พันธวัช สุทธิจำนงค์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2559

เครื่องมือทดสอบความสามารถการทำงานของ WEB APPLICATION

WEB APPLICATION LOAD TESTING TOOL

พรภิณัฐ แก้วเมือง

พันธวัช สุทธิจำนงค์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2559

ปริญญานิพนธ์ปีการศึกษา 2559

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องมือทดสอบความสามารถการทำงานของ WEB APPLICATION

WEB APPLICATION LOAD TESTING TOOL

ผู้จัดทำ

1. พรภิณัฐ แก้วเมือง รหัสนักศึกษา 56010808

2. พันธวัช สุทธิจำนงค์ รหัสนักศึกษา 56010841

_____ อาจารย์ที่ปรึกษา

(ดร. ธนัญชัย ตรีภาค)

เครื่องมือทดสอบความสามารถการทำงาน

ของ WEB APPLICATION

นายพรภิณัฐ	แก้วเมือง	56010808
นายพันธวัช	สุทธิจำนงค์	56010841
ดร. ธนัญชัย	ตรีภาค	อาจารย์ที่ปรึกษา

ปีการศึกษา 2559

บทคัดย่อ

เครื่องมือทดสอบความสามารถการทำงานของ Web Application เป็นชุดซอฟต์แวร์สำหรับทดสอบการทำงานของ Web Application เช่น Response Time ของระบบ ปริมาณการใช้งานสูงสุดของระบบที่สามารถให้บริการได้อย่างเหมาะสม ปริมาณการทำงานสูงสุดที่สามารถทำงานได้ในช่วงเวลาหนึ่ง ฯลฯ เพื่อให้ผู้ใช้งานนำข้อมูลที่ได้จากการทดสอบไปประเมินปัญหาในระบบ เช่น คอขวดในการทำงาน คุณภาพในการใช้ทรัพยากรของเครื่องรวมถึงสามารถปรับแต่งระบบให้สามารถทำงานได้อย่างเหมาะสม และวางแผนการใช้ทรัพยากรของระบบได้ โดยเครื่องมือที่พัฒนาขึ้นจะสามารถตั้งค่าการทำงานของเครื่องมือทดสอบสามารถได้ เช่น เป้าหมายที่ต้องการทดสอบ จำนวนสูงสุดของ Connection ระยะเวลาในการทดสอบ และยังสามารถจำลองพฤติกรรมการใช้งานระบบ ของผู้ใช้งานในลักษณะต่างๆ ได้

WEB APPLICATION LOAD TESTING TOOL

Mr. Ponpinut	Kaewmuang	56010808
Mr. Puntawath	Suttijumnong	56010841
Dr. Thanunchai	Threepak	Advisor

Academic Year 2559

ABSTRACT

The web application load testing tool is a software bundle for testing the process of web application such as response time, the maximum usages of the system while the server can work properly, the maximum works in the system which server can process in the short time, etc. Users can use the result of the test to indicate the problem in the system e.g. the bottleneck of process, quality of usage in systemic resource, reconfigure system to make it work properly and preform resource planning. We also can set the process of web application load testing tool e.g. the tested target, maximum connections, test time and also can simulate the usage behavior of user's different styles.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงได้ด้วยดี เนื่องจากได้รับคำปรึกษาและข้อชี้แนะจากอาจารย์ ดร.ธนัญชัย ตรีภาค ซึ่งเป็นอาจารย์ที่ปรึกษาในการจัดทำโครงงานชิ้นนี้ ทางกลุ่มผู้จัดทำรู้สึกซาบซึ้งในความอนุเคราะห์จากอาจารย์ จึงขอขอบพระคุณท่านเป็นอย่างสูง และขอขอบพระคุณสาขาวิชาวิศวกรรมคอมพิวเตอร์ที่ได้เอื้อเฟื้อสถานที่ และอำนวยความสะดวกในการทำโครงงานในภาคเรียนนี้ให้สำเร็จลุล่วงได้ด้วยดี

ขอขอบพระคุณอาจารย์ที่ช่วยอบรม สั่งสอน แนวทางที่ดีแก่ข้าพเจ้าเสมอมา ขอขอบคุณเพื่อนๆในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่คอยให้คำปรึกษาและแนะนำตลอดการทำงาน ขอกราบขอบพระคุณบิดา มารดา และครอบครัวของข้าพเจ้าอันเป็นที่เคารพรักยิ่ง ซึ่งได้ให้การเลี้ยงดู คอยอบรม เป็นกำลังใจและให้การสนับสนุน ซึ่งกลุ่มผู้จัดทำหวังเป็นอย่างยิ่งว่า การทำโครงงานในครั้งนี้ นอกจากจะช่วยพัฒนาความรู้ให้แก่ตัวข้าพเจ้าแล้ว ยังสามารถนำไปใช้เป็นประโยชน์เพื่อผู้อื่นและประเทศชาติได้

พรภิณัฐ แก้วเมือง

พันธวัช สุทธิจำนงค์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ในปัจจุบันเทคโนโลยีเครือข่ายได้เข้ามามีบทบาทอย่างมากในทุกๆด้าน ไม่ว่าจะเป็นด้านเศรษฐกิจ ด้านข่าวสาร ด้านการทหาร หรือแม้แต่ในชีวิตประจำวัน ส่งผลให้เกิดการแลกเปลี่ยนข้อมูลกันในเครือข่ายจำนวนมหาศาล แต่เมื่อมีการแลกเปลี่ยนข้อมูลปริมาณมากโดยที่เครื่องคอมพิวเตอร์แม่ข่ายมีประสิทธิภาพไม่สูงพอ มีทรัพยากรไม่เพียงพอ หรือตั้งค่าการทำงานไว้ไม่เหมาะสมจะส่งผลให้เครื่องคอมพิวเตอร์แม่ข่ายประมวลผลข้อมูลได้ช้าลงหรือไม่สามารถประมวลผลข้อมูลใดๆ ได้ จึงควรทำการวัดประสิทธิภาพการทำงานของระบบเพื่อประเมินระบบ, ปรับแต่งระบบและปรับปรุงระบบ เพื่อป้องกันความเสียหายที่อาจจะเกิดขึ้น เพื่อให้ระบบสามารถให้บริการอย่างเหมาะสมได้อย่างต่อเนื่อง

ทางคณะผู้วิจัยได้เล็งเห็นถึงปัญหาดังกล่าวจึงได้จัดทำเครื่องมือทดสอบความสามารถการทำงานของ Web Application ขึ้น เป็นชุดซอฟต์แวร์สำหรับทดสอบการทำงานของ Web Application เช่น ระยะเวลาในการตอบสนองของระบบ , ปริมาณการใช้งานสูงสุดของระบบที่สามารถให้บริการได้อย่างเหมาะสม ฯลฯ เพื่อให้ผู้ใช้งานนำข้อมูลที่ได้จากการทดสอบไปประเมินปัญหาในระบบ เช่น คอขวดในการทำงาน , คุณภาพของการใช้ทรัพยากร รวมถึงสามารถปรับแต่งระบบให้สามารถทำงานได้อย่างเหมาะสม และวางแผนการใช้ทรัพยากรของระบบได้ โดยระบบสามารถตั้งค่าการทำงานได้หลากหลาย สามารถจำลองพฤติกรรมการใช้งานระบบของผู้ใช้งานในลักษณะต่างๆ ที่ใกล้เคียงกับการใช้งานจริงให้มากที่สุด เพื่อให้ผลลัพธ์ที่ได้จากการทดสอบนั้น มีความเที่ยงตรงและน่าเชื่อถือมากที่สุด

1.2 วัตถุประสงค์

1) เพื่อสร้างโปรแกรมที่ใช้ในการทดสอบและวัดประสิทธิภาพการทำงานของระบบที่อยู่ภายใต้การใช้งานรูปแบบต่างๆ ทั้งแบบปกติหรือสูงกว่าที่คาดหวังไว้

2) เพื่อทราบความสามารถสูงสุดในการทำงานของระบบที่ต้องการ เช่น จำนวนการเชื่อมต่อที่ทำให้เกิดปัญหาคอขวดของระบบ (Bottleneck) ที่ส่งผลให้ระบบงานทำงานได้ช้าลง

3) เพื่อนำข้อมูลที่ได้รับจากการทดสอบ Load Testing ไปใช้วางมาตรการป้องกันหรือแก้ไขปัญหาที่อาจจะเกิดขึ้นกับระบบได้ในอนาคตเพื่อป้องกันความเสียหายที่อาจจะเกิดขึ้น

1.3 ขอบเขตงาน (Scope of Work)

1) พัฒนาโปรแกรม Load Testing ให้สามารถปรับแต่งค่าที่เกี่ยวข้องกับการทดสอบได้และสามารถรองรับ User ได้ไม่เกิน 200,000 users

2) พัฒนาโปรแกรมให้สามารถทำการทดสอบในรูปแบบ Master-Slave ได้

3) สร้างตัวติดตั้งโปรแกรม เมื่อทำการติดตั้งเรียบร้อยแล้ว โปรแกรมสามารถใช้งานได้เป็นปกติ

4) จัดทำคู่มือการใช้งานระบบเป็นภาษาไทย

5) พัฒนาโปรแกรมให้สามารถแสดงผล ในรูปแบบของกราฟ โดยแสดงค่าข้อมูลที่ใช้ในการวิเคราะห์ประสิทธิภาพหลักๆประกอบไปด้วย

a. เวลาที่ใช้ในการตอบสนอง (Response time)

b. ปริมาณงานที่ทำในช่วงเวลาหนึ่ง (Request/s)

c. ความสามารถในการรองรับ user ได้สูงสุด (Concurrent Users)

d. จำนวนที่ส่งสำเร็จในช่วงเวลาหนึ่ง (Success/s)

e. ความล่าช้าในการส่งข้อมูล (Latency)

f. จำนวนที่ส่งไม่สำเร็จในช่วงเวลาหนึ่ง (Error/s)

1.4 วิธีดำเนินการ

1) ระบุขอบเขตและวัตถุประสงค์ของโครงงาน

2) รวบรวมข้อมูลเกี่ยวกับเครื่องมือทดสอบภาระงานของ Web Application (Load Testing)

3) ศึกษาข้อมูลและวางแผนการทำโครงงาน

4) ทดลองสร้างเครื่องมือทดสอบภาระงาน และทดสอบตามแผนที่วางไว้

5) แก้ไขปัญหาที่เกิดขึ้นในการทดลองและปรับปรุงเครื่องมือทดสอบภาระงาน

1.5 ประโยชน์ที่ได้รับ

1) มีโปรแกรมที่ใช้ในการทดสอบและวัดประสิทธิภาพการทำงานของระบบซึ่งจะทำให้รู้ถึงความสามารถสูงสุดในการทำงานของเครื่องคอมพิวเตอร์แม่ข่ายที่ได้ทำการทดสอบ

2) เพื่อให้มีการรวบรวมข้อมูลจากกระบวนการ Load Testing นำไปวิเคราะห์ถึงปัญหาต่างๆ

3) นำข้อมูลที่ได้รับจากการทดสอบ Load Testing ไปใช้วางมาตรการป้องกันหรือแก้ไขปัญหาที่อาจจะเกิดขึ้นกับระบบได้ ในอนาคต

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ในการพัฒนาโปรแกรม Load Tester ตามเป้าหมาย จึงได้ออกแบบให้ระบบดังกล่าวใช้รูปแบบการติดต่อสื่อสารโดยมีระบบสำหรับควบคุม (Master) และระบบสำหรับทำการทดสอบตามคำสั่ง (Slave) โดยใช้ Socket เพื่อเป็นช่องทางในการติดต่อสื่อสารระหว่างเครื่อง Master และเครื่อง Slave ทำงานบนโปรโตคอลมาตรฐาน TCP/IP โดยได้ทำการเลือกใช้ภาษาในการพัฒนาคือ ภาษา C# ซึ่งทำงานบนสภาพแวดล้อมการทำงานของ .NET Framework ซึ่งมีเครื่องมือเอื้ออำนวยในการสร้างหน้าต่างติดต่อผู้ใช้ (UI) และ Library สำหรับการทำงานบน Socket

2.1 ภาษา C#

C# เป็นภาษาโปรแกรมเชิงวัตถุที่ถูกสร้างมาเพื่อใช้สำหรับการสร้างแอพพลิเคชั่นระดับองค์กรที่ทำงานบน .NET Framework ออกแบบโดยบริษัท Microsoft โดยภาษา C# นั้นมีวิวัฒนาการจากการนำภาษา C และ C++ มาพัฒนา เพื่อลดความซับซ้อนในโครงสร้างของภาษา C และ C++ นอกจากนี้ยังเป็นภาษาที่มีความปลอดภัยสูงและมี Garbage Collection รองรับการขยายขนาดและยังมีคุณสมบัติอื่นๆ ที่ทำให้การพัฒนาโปรแกรม สามารถทำได้เร็วและง่ายขึ้น [1] [2] [3]

2.1.1 C# เทียบกับภาษาอื่นๆ

1) การเขียนภาษา C# เหมาะกับการใช้โปรแกรม Visual Studio ของ Mircrosoft มากที่สุด เนื่องจากโปรแกรม Visual Studio มีฟังก์ชันที่ช่วยให้การเขียน C# นั้นง่ายขึ้น หากเทียบกับโปรแกรมอื่น แต่ภาษา C, C++ และ Java มีความหลากหลายของโปรแกรมที่มีฟังก์ชันช่วยในการเขียนได้ง่ายขึ้น ทำให้สามารถเลือกใช้โปรแกรมได้หลากหลายกว่า

2) C# ไม่มีปัญหาเรื่องการรั่วไหลของหน่วยความจำ ซึ่งเป็นปัญหาสำคัญใน C++ และ Java ไลบรารีระดับชั้นที่หลากหลายทำให้ง่ายต่อการใช้งาน โปรแกรมจะทำงานได้ดีเฉพาะในกรณีที่เครื่องติดตั้ง .NET framework

3) ภาษา C# มีความคล้ายคลึงกับภาษา Java มาก ทำให้นักเขียนโปรแกรมภาษา Java สามารถย้ายมาเขียนภาษา C# ได้ง่ายโดยศึกษาว่ามีสิ่งใดที่แตกต่างกันบ้าง [4]

4) C# มีความคล้ายกับภาษา Java แต่ภาษา C# มีเอกลักษณ์ที่ต่างไปหลายอย่างเช่น การจัดเก็บ Object ไว้ใน Stack การทำ Versioning พารามิเตอร์แบบ Reference และ Output นอกจากนี้ยังมีข้อดี เช่น Delegate Properties และ Operator Overloading ซึ่งจะไม่พบในภาษา Java [5]

2.2 การสื่อสารระหว่างโปรแกรมบนเครือข่าย

ซ็อกเก็ตเป็นหนึ่งในเทคโนโลยีขั้นพื้นฐานที่สุดของการเขียนโปรแกรมเครือข่ายคอมพิวเตอร์ ซึ่งซ็อกเก็ตอนุญาตให้แอพพลิเคชันซอฟต์แวร์เครือข่ายสื่อสารโดยใช้กลไกมาตรฐานที่มีอยู่ภายในฮาร์ดแวร์และระบบปฏิบัติการเครือข่าย [6] โดยปกติแล้วซ็อกเกต อนุญาตให้มีการสื่อสารระหว่าง 2 กระบวนการที่ต่างกันบนเครื่องเดียวกันหรือต่างเครื่อง วิธีที่สามารถพูดคุยกับคอมพิวเตอร์เครื่องอื่น ๆ โดยใช้ Standard Unix File ใน Unix ซึ่งทุกการดำเนินการ I/O ทำได้โดยการเขียนหรืออ่าน File Descriptor ซึ่ง File Descriptor นี้เป็นเพียงตัวเลขระบุตำแหน่งที่เชื่อมโยงกับไฟล์ที่เปิดอยู่และสามารถเชื่อมต่อกับเครือข่าย, Text File , Terminal

2.2.1 รูปแบบการทำงานแบบแม่ข่ายลูกข่าย

ในการสื่อสารข้อมูลจะพบว่า Socket เป็นช่องทางที่โปรแกรมใช้ติดต่อกับเครือข่ายสื่อสาร ซึ่งจะช่วยในเรื่องของการจัดการข้อมูลเช่น การเตรียม buffer สำหรับข้อมูลเข้าและออก การรับและส่งข้อมูล เป็นต้น ส่วนโพรโตคอลที่ใช้ยกตัวอย่าง เช่น TCP/IP ที่ใช้มีหน้าที่กำหนดวิธีการส่งผ่านข้อมูลในการสื่อสาร ส่วนบทบาทของผู้รับและผู้ส่งหรือจังหวะในการรับส่งข้อมูลนั้นไม่ได้ถูกกำหนดไว้ใน Specification ของทั้ง Socket และ Protocol เพื่อให้การสื่อสารข้อมูลทางคอมพิวเตอร์เป็นไปราบรื่นและมีประสิทธิภาพ ได้มีการจัดการจังหวะของการสื่อสารด้วยแนวคิดที่เรียกว่า "Client-Server Paradigm"

ประเด็นหลักที่ผลักดันให้เกิด "Client-Server Paradigm" คือปัญหาที่เรียกว่า Rendezvous ปัญหา Rendezvous เกิดขึ้นจากจังหวะการทำงานที่ไม่สอดคล้องกันของโปรแกรมที่สื่อสารกัน ยกตัวอย่างเช่น มีโปรแกรม 2 โปรแกรมต้องการสื่อสารข้อมูลผ่านโครงข่ายสื่อสาร หาก 2 โปรแกรมนั้นถูกผู้ใช้งานเรียกใช้งานพร้อม ๆกัน โปรแกรมทั้ง 2 นั้นก็จะสามารถสื่อสารกันได้ หากโปรแกรมทั้งสองนั้นถูกเรียกใช้งานไม่พร้อมกัน โปรแกรมที่ทำงานก่อนเมื่อเริ่มการสื่อสารก็จะพบว่าโปรแกรมที่จะสื่อสารด้วยนั้นไม่มีก็จะจบการทำงานไป ในขณะที่โปรแกรมที่เริ่มทำงานทีหลังพอเริ่มต้นการสื่อสารก็จะพบว่าโปรแกรมที่จะสื่อสารด้วยนั้นจบการทำงานไปแล้ว เป็นต้น Client-Server Model ได้ทำการแก้ไขปัญหานี้โดยการกำหนดบทบาทและจังหวะในการสื่อสาร โดย

กำหนดให้การสื่อสารระหว่างกันของคู่โปรแกรมนั้น จะต้องมีฝ่ายหนึ่งเมื่อเริ่มต้นทำงานแล้วจะต้องรอจนกว่าอีกฝ่ายหนึ่งจะติดต่อเข้ามาก่อนจึงจะเริ่มต้นกระบวนการสื่อสาร [7]

2.2.1.1 แม่ข่ายลูกข่าย

Client-Server Paradigm จะแบ่งประเภทของโปรแกรมที่สื่อสารกันออกเป็น 2 ประเภท โดยอาศัยการเริ่มต้นการสื่อสาร (Direction of initiation) เป็นเกณฑ์ในการแบ่งประเภท โดยทั่วไปโปรแกรมซึ่งเป็นผู้เริ่มต้นกระบวนการสื่อสารข้อมูลจะถูกเรียกว่า "Client" ในขณะที่โปรแกรมซึ่งรอให้โปรแกรมอื่นร้องขอการเชื่อมต่อจะเรียกว่า "Server" กระบวนการในการทำงานโดยทั่วไป Client ซึ่งส่วนใหญ่จะอยู่ในรูปของโปรแกรมประยุกต์เช่น web browser เป็นต้น เมื่อโปรแกรมประยุกต์ดังกล่าวถูกเรียกขึ้นมาใช้งาน ก็จะทำการติดต่อไปยัง Server, ส่งการร้องขอบริการ (Service request) ไปยัง Server จากนั้นก็จะรอการตอบสนองจาก Server ทางฝั่ง Server เมื่อได้รับการร้องขอบริการก็จะทำการประมวลผลตามที่ Client ร้องขอ เมื่อได้ผลลัพธ์แล้วก็จะส่งผลลัพธ์นั้นไปยัง Client

โปรแกรม Client โดยทั่วไปแบ่งได้เป็น 2 ลักษณะคือ Standard Client Software และ Nonstandard Client Software ความแตกต่างระหว่างโปรแกรม Client ทั้งสอง คือ Standard Client Software นั้นเป็นโปรแกรมที่ทำการติดต่อสื่อสารกับบริการที่เป็นมาตรฐาน และมีพอร์ตในการเชื่อมต่อ โดยทั่วไปตัวอย่างของบริการที่เป็นมาตรฐานได้แก่ WWW, FTP, SMTP เป็นต้น โดยปกติหมายเลขพอร์ตของบริการมาตรฐานจะมีค่าน้อยกว่า 1024 ซึ่งตัวอย่างหมายเลขพอร์ตมาตรฐานดังตารางที่ 2.1 ส่วน Nonstandard Client Software นั้นเป็นโปรแกรมที่มีการติดต่อกับบริการ ซึ่งให้บริการเป็นการเฉพาะ ใช้หมายเลขพอร์ตซึ่งกำหนดขึ้นมาเอง โดยส่วนมากหมายเลขพอร์ตจะมีค่ามากกว่า 1024

ตารางแสดงตัวอย่างของหมายเลข Port มาตรฐาน

Port	Service/Description	Port	Service/Description
20,21	FTP	514	Shell
22	Secure Shell	515	Print Service
23	Telnet	517	Talk
43	WHOIS	518	NTalk
52,54,56	XNS	520	Routing (RIP)
53	Domain Name System	530	RPC
70	Gopher	546,547	DHCPv6
79	Finger	554	Real Time Streaming Protocol
80	HTTP (WEB)	631	Internet Printing Protocol
88	Kerberos	636	Secured LDAP
110	POP3	666	DOOM (1st online, Game)
111	Sun RPC	691	MS Exchange
118, 156	SQL	711	Cisco TDP
123	Network Time Protocol	860	iSCSI
137,138,139	NetBIOS	873	rsync
143	IMAP	989,990	Secured FTP
389	LDAP	992	Secured Telnet
443	Secured HTTP	993	Secured IMAP
445	MS-SMB file sharing	995	Secured POP3

ตารางที่ 2.1 ตารางแสดงตัวอย่างหมายเลข port มาตรฐาน

กล่าวว่าโปรแกรม Server ยังอาจมีรูปแบบการสร้างแบ่งเป็น 2 ประเภท โดยแบ่งตามลักษณะการจัดการข้อมูลสถานะของการปฏิสัมพันธ์ระหว่าง Client และ Server ในระหว่างการสื่อสาร Server ซึ่งไม่มีการจัดเก็บข้อมูลสถานะระหว่างการสื่อสารนั้นจะเรียกว่า Stateless Servers ส่วน Server ที่มีการจัดเก็บข้อมูลสถานะระหว่างการสื่อสารจะเรียกว่า Stateful Servers

เพื่อให้เกิดความเข้าใจในเรื่องของ Stateless และ Stateful Servers จะยกตัวอย่างโปรแกรมซึ่งทำหน้าที่เป็น File Server หาก File Server นี้สร้างแบบ Stateless จะได้ว่า Server นี้จะไม่มีการเก็บข้อมูลสถานะระหว่างการเชื่อมต่อกับ Clients ดังนั้นในการร้องขอบริการเกี่ยวกับไฟล์ของ Clients ทุกครั้งจะต้องมีการระบุรายละเอียดสถานะด้วย เช่นในการร้องขอคัดลอกไฟล์จาก Server นั้น โปรแกรม Client จะต้องระบุชื่อไฟล์โดยสมบูรณ์ ระบุตำแหน่งในไฟล์ซึ่งจะให้ Server อ่านข้อมูล เพื่อส่งไปให้ Client ทำการคัดลอกและจะต้องทำเช่นนี้ทุกครั้งที่มีการร้องขอให้ Server ส่งข้อมูลในบล็อกถัดไปเพื่อทำการคัดลอกจนกว่าจะจบไฟล์ในขณะที่หาก File Server นั้นเป็น Stateful Server ตัว Server จะมีการจัดเก็บสถานะการทำงานระหว่างการสื่อสารข้อมูลกับ Client

ดังนั้นในการร้องขอคัดลอกไฟล์จาก Server ไม่จำเป็นที่จะต้องระบุชื่อไฟล์โดยสมบูรณ์และตำแหน่งในไฟล์ในทุก ๆ การร้องขอ ทั้งนี้เพราะ Server ได้ทำการจัดเก็บข้อมูลเหล่านั้นไว้แล้วเมื่อแรกเริ่มที่ Client ร้องขอการคัดลอกไฟล์

จากตัวอย่างจะพบว่า Stateful Servers นั้นจะให้ประสิทธิภาพที่ดีกว่า Stateless Servers แต่ในทางปฏิบัติ Stateful Server จะมีความยากและซับซ้อนในการสร้าง นอกจากนั้นปัญหาที่เกิดจากการสื่อสารเช่น การล่าช้าของข้อมูล การซ้ำกันของข้อมูล และลำดับของข้อมูล นั้นก็อาจเป็นสาเหตุในสถานะ ซึ่ง Server เก็บไว้นั้นผิดพลาดได้ [7]

2.2.2 ประเภทของการสื่อสารระหว่างโปรแกรมบนเครือข่าย

Sockets สามารถแบ่งออกเป็นประเภทตามลักษณะการรับส่งข้อมูลผ่านเครือข่ายได้เป็น 3 ประเภท [8] ดังนี้

• Stream Sockets หรือ Connection Oriented Sockets

เป็นซ็อคเก็ตที่จะกำหนดให้ทั้งสองฝ่ายสื่อสารต้องสร้างการเชื่อมต่อแบบซ็อกเก็ตหลังจากที่ข้อมูลใด ๆ ที่ส่งผ่านการเชื่อมต่อนั้นจะได้รับการรับประกันว่าจะได้รับตามลำดับเดียวกันกับที่มีการส่ง

• Datagram Sockets หรือ Connectionless Sockets

เป็นซ็อคเก็ตที่ทั้งสองฝ่ายสามารถส่งดาต้าแกรมได้ตามต้องการ และรอให้อีกฝ่ายหนึ่งตอบกลับ แต่ไม่รับประกันข้อมูลที่ส่ง ซึ่งข้อความอาจสูญหายในการรับส่ง โดยส่วนที่จัดการข้อมูลที่หายไม่ใช่หน้าที่ของซ็อกเก็ตดาต้าแกรม แต่เป็นส่วนของแอพพลิเคชัน การใช้ซ็อกเก็ตดาต้าแกรมจะทำให้แอพพลิเคชันบางตัวสามารถเพิ่มประสิทธิภาพการทำงานและเพิ่มความยืดหยุ่นได้ดีกว่าเมื่อเทียบกับการใช้ซ็อกเก็ตสตรีมในบางสถานการณ์

• Raw Sockets

เป็นซ็อกเก็ตที่ทำการรับส่งข้อมูลโดยตรงกับชั้นของ IP ไม่รองรับการสนับสนุนสำหรับโปรโตคอลมาตรฐาน เช่น TCP และ UDP ใช้สำหรับการพัฒนาโปรโตคอลระดับต่ำที่สามารถกำหนดเอง

2.2.3 การทำงานของการสื่อสารระหว่างโปรแกรมบนเครือข่ายใน .NET

ใน .NET การทำงานที่เกี่ยวข้องกับ Sockets จะต้องมีการเรียกใช้งาน Namespace ที่ชื่อว่า System.Net.Sockets ซึ่งเป็น Namespace ที่บรรจุ Classes ต่าง ๆ ที่สนับสนุนการทำงานกับ Sockets โดย Classes ที่เกี่ยวข้องกับการทำงานหลัก ๆ กับ Sockets [7] มีดังนี้

- MulticastOption: ประกอบด้วยค่า IPAddress ที่ใช้ในการเชื่อมต่อและยกเลิกการเชื่อมต่อกับ Multicast Group
- NetworkStream: ช่วยจัดการสตรีมสำหรับการเข้าถึงเครือข่าย
- TcpClient: ช่วยจัดการการเชื่อมต่อของ Client สำหรับ TCP network services
- TcpListener: รอการเชื่อมต่อจาก TCP network clients
- UdpClient: ช่วยจัดการ UDP network services
- SocketException: ข้อยกเว้นเมื่อมี Error ของ Socket เกิดขึ้น
- Socket: ใช้ Berkeley Socket Interface

โดย Class Socket จะมี function พื้นฐานต่าง ๆ ที่จำเป็นสำหรับการสร้าง Application ซึ่งใช้งาน Sockets สำหรับProperties ที่สำคัญสำหรับ System.Net.Sockets ได้แก่

- o AddressFamily: รับค่าที่อยู่ของ Socket
- o Available: รับข้อมูลที่ได้รับจากเครือข่ายและสามารถอ่านได้
- o Blocking: รับหรือกำหนดค่าที่ระบุว่า Socket อยู่ในโหมดปิดกั้นหรือไม่
- o Connected: รับค่าที่ระบุว่า Socket มีการเชื่อมต่อกับ Remote Host
- o LocalEndPoint: รับค่า Local endpoint
- o ProtocolType: รับค่า Protocol type ของ Socket
- o RemoteEndPoint: รับค่า Remote endpoint
- o SocketType: รับค่าประเภทของ Socket

และ Method ที่สำคัญของ System.Net.Sockets.Socket มีดังนี้

- o Accept(): สร้าง Socket ใหม่สำหรับการเชื่อมต่อที่ถูกสร้างขึ้นใหม่
- o Bind(): เชื่อมต่อ Socket กับ Local Endpoint
- o Close(): ปิดการเชื่อมต่อ Socket และปล่อยการใช้ Resources ทั้งหมด

o Connect(): สร้างการเชื่อมต่อกับ Remote host

o GetSocketOption(): คืนค่าของ Socket option ที่ถูกระบุไว้

o IOControl(): ตั้งค่าโหมดการทำงานระดับต่ำสำหรับ Socket โดยใช้ Numerical Control Codes

o Listen(): สถานะของ Socket รอการตอบกลับ

o Receive(): รับข้อมูลจาก Socket ไปเก็บไว้ที่ Buffer

o Poll(): กำหนดสถานะของ Socket

o Select(): กำหนดสถานะของ Socket มากกว่าหนึ่งตัว

o Send(): ส่งข้อมูลไปที่ Socket

o SetSocketOption(): ตั้งค่า Socket ที่ระบุให้เป็นตามค่าที่กำหนด

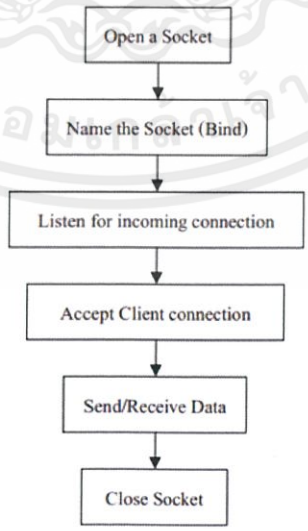
o Shutdown(): ปิดการส่งและรับบน Socket

2.2.4 รูปแบบการทำงานของการสื่อสารระหว่างโปรแกรมบนเครือข่าย

รูปแบบการทำงานของ Socket สามารถแบ่งออกเป็น 2 ประเภทหลัก ดังนี้

2.2.4.1 Passive Sockets

Passive Sockets หรือ Sockets ซึ่งใช้ใน Server Application นั้นมีขั้นตอนการสร้าง ดังรูปที่ 2.1



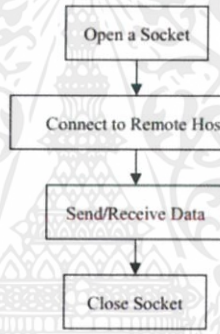
รูปที่ 2.1 Diagram ขั้นตอนการสร้าง Passive Sockets

จากรูป 2.1 เราสามารถอธิบายได้ดังนี้

1. เมื่อเริ่มต้นการทำงานตัว Server จะทำการเปิดการเชื่อมต่อแบบ socket
2. ทำการกำหนด Port ที่จะใช้ในการเชื่อมต่อ
3. รอการเชื่อมจากภายนอก ผ่าน Port นั้น
4. เมื่อได้รับการเชื่อมต่อ ก็จะส่งการตอบรับกลับไปยังต้นทาง
5. เมื่อการเชื่อมต่อสมบูรณ์ จึงจะเริ่มทำการส่งข้อมูล
6. เมื่อการส่งข้อมูลเสร็จสิ้น ก็จะทำการปิดการเชื่อมต่อแบบ socket นั้น

2.2.4.2 Active Sockets

Active Sockets หรือ Socket ซึ่งใช้งานใน Client Application มีขั้นตอนการสร้าง ดังรูปที่ 2.2



รูปที่ 2.2 Diagram ขั้นตอนการสร้าง Active Sockets

จากรูป 2.1 เราสามารถอธิบายได้ดังนี้

1. เมื่อเริ่มต้นการทำงานตัว Client จะทำการเปิดการเชื่อมต่อแบบ socket
2. ทำการกำหนด Port และ IP ที่จะใช้ในการเชื่อมต่อ
3. ส่งคำขอการเชื่อมและรอการตอบรับของปลายทางผ่าน Port นั้น
4. เมื่อการเชื่อมต่อสมบูรณ์ จึงจะเริ่มการส่งข้อมูล
5. เมื่อการส่งข้อมูลเสร็จสิ้น ก็จะทำการปิดการเชื่อมต่อแบบ socket นั้น

2.3 .NET Framework

.NET Framework คือ รูปแบบการพัฒนาโปรแกรมที่ไมโครซอฟท์ได้พัฒนาขึ้นโดยมีจุดประสงค์สำคัญคือ สามารถใช้งานโปรแกรมได้แม้ว่าสภาวะฮาร์ดแวร์หรือระบบปฏิบัติการจะแตกต่างกันก็สามารถใช้งานได้อย่างไม่มีปัญหา (เช่น เครื่องพีซีกับเครื่องแมคหรือระบบปฏิบัติการวินโดว์กับลีนุกซ์) และสามารถพัฒนาโปรแกรมใหม่ๆ ได้ด้วยภาษาอะไรก็ได้แต่ยังคงสามารถทำงานร่วมกันได้ (เช่น ภาษา C กับ Java เป็นต้น) รวมถึงเป็นเครื่องมือในการพัฒนาโปรแกรมให้สามารถเชื่อมต่อกับโปรแกรมต่างๆ ของไมโครซอฟท์ได้โดยง่าย ซึ่งก็รวมไปถึงการทำงานภายในของระบบปฏิบัติการวินโดว์เองด้วย ผู้พัฒนาจึงสามารถพัฒนาโปรแกรมใหม่ๆ ได้โดยง่ายและรวดเร็ว โดยไม่ติดข้อจำกัดต่างๆ อย่างเช่นการพัฒนาโปรแกรมข้ามแพลตฟอร์มเหมือนในสมัยก่อน [9] [10]

2.3.1 วัตถุประสงค์ของ .NET Framework

.NET Framework เป็นเทคโนโลยีที่สนับสนุนการสร้างและใช้งานแอพพลิเคชั่น หรือ XML Web Service รุ่นใหม่ ๆ โดย .NET Framework ได้รับการออกแบบเพื่อให้บรรลุวัตถุประสงค์ดังต่อไปนี้ [11]

- เพื่อให้มีสภาพแวดล้อมของการเขียนโปรแกรมเชิงวัตถุที่สอดคล้องกัน ไม่ว่าจะมีการจัดเก็บและประมวลผลโปรแกรมภายในเครื่อง หรือดำเนินการในเครื่องแต่มีการเผยแพร่ทางอินเทอร์เน็ตหรือดำเนินการจากระยะไกล

- เพื่อให้มีสภาพแวดล้อมในการประมวลผลโปรแกรมที่ช่วยลดความยุ่งยากในการใช้งานซอฟต์แวร์และปัญหาการขัดแย้งกันทางด้านเวอร์ชัน

- เพื่อให้มีสภาพแวดล้อมในการประมวลผลโปรแกรมที่สนับสนุนการทำงานของโปรแกรมอย่างปลอดภัย รวมถึงโปรแกรมที่สร้างขึ้นโดยบุคคลที่ไม่รู้จักหรือบุคคลที่สามที่กึ่งเชื่อถือได้

- เพื่อให้มีสภาพแวดล้อมในการประมวลผลโปรแกรม ที่ช่วยลดปัญหาด้านประสิทธิภาพของ scripted หรือ interpreted environments.

- เพื่อให้นักพัฒนาซอฟต์แวร์พบกับประสบการณ์ใหม่ในการพัฒนาโปรแกรมที่มีความสอดคล้องกันกับแอ็พพลิเคชันที่แตกต่างกันอย่างกว้างขวาง เช่น แอพพลิเคชันที่ใช้ Windows และแอพพลิเคชันบนเว็บ

- เพื่อสร้างการสื่อสารทั้งหมดบนมาตรฐานอุตสาหกรรมเพื่อให้แน่ใจว่า code ที่ใช้ .NET Framework สามารถทำงานร่วมกับ code อื่น ๆ ได้

2.3.2 ประโยชน์ที่ได้เมื่อเลือกใช้โปรแกรมที่พัฒนาบน .NET Technology

ในส่วนของข้อดีและประโยชน์ของ .NET Framework นั้นเรา สามารถสรุปได้เป็นข้อๆ ดังนี้

1. เป็นระบบที่มี library ที่เป็นมาตรฐานเดียวกัน เนื่องจากมี library ที่เป็นมาตรฐานเดียวกันทั้งหมดทำให้เราไม่ต้องกังวลว่าภาษาที่ใช้เขียนนั้นมี library ตัวนั้นตัวนี้หรือไม่ รวมทั้งไม่ต้องคอยกังวลว่าถ้าใช้ Library ของภาษาหนึ่งแล้วอีกภาษาหนึ่งจะไม่มี library ตัวนั้น

2. ไม่ขึ้นกับ ระบบปฏิบัติการ (OS) เนื่องจากระบบปฏิบัติการที่แต่ละบุคคลหรือองค์กรใช้นั้นย่อมไม่เหมือนกัน แต่ภายใน .NET Framework จะไม่มีปัญหานี้ของเพียงแค่มีระบบ .NET Framework ก็จะทำให้สามารถใช้งานโปรแกรมต่างๆได้ ซึ่งเป็นข้อดีตรงที่เราจะสามารถใช้โปรแกรมต่าง ๆได้ทุกระบบปฏิบัติการ

3. ใช้ในการพัฒนาได้ทุกภาษาทำให้เราไม่จำเป็นต้องศึกษาภาษาใหม่ๆ เมื่อต้องการสร้างโปรแกรมในแต่ละครั้ง นอกจากนั้นผู้พัฒนายังสามารถเลือกใช้ภาษาที่ถนัดที่สุดในการพัฒนาโปรแกรมต่างๆได้ด้วย

4. มีการควบคุมสิ่งแวดล้อมในการทำงานเป็นอย่างดี เนื่องจากเป็นระบบที่เป็นมาตรฐานทำให้การควบคุม จัดสรรระบบต่างๆทำได้ง่ายขึ้น ไม่ว่าจะเป็นการจัดสรรหน่วยความจำด้านการใช้งานเครื่องก็มีความรวดเร็วมากขึ้น ลดโอกาสที่เครื่องจะค้างได้เป็นอย่างดี

5. ความปลอดภัยที่มีมากขึ้น .NET Framework สามารถกำหนดสิทธิ์การใช้งานหรือ permission ของผู้ใช้งานได้มากขึ้นทำให้สามารถกำหนดว่าจะให้โปรแกรมในส่วนใดใช้งานได้หรือไม่ได้ แล้วแต่เฉพาะบุคคล

2.3.3 ผลกระทบเมื่อเลือกใช้งาน .NET Framework Technology

1. .NET Framework เป็นส่วนเสริมที่ไม่ได้ติดตั้งให้ทันทีเมื่อลงระบบปฏิบัติการวินโดว์ XP หรือวินโดว์ 2000 แต่เป็นส่วนเสริมหนึ่งที่สามารถติดตั้งเพิ่มจากแผ่นติดตั้งหรือ Download เพื่อติดตั้งเองได้ ทั้งนี้การติดตั้งนี้ไม่มีค่าใช้จ่ายเรื่องลิขสิทธิ์หรือมีผลกระทบต่อโปรแกรมที่ได้ติดตั้งอยู่แต่เดิม

2. โปรแกรมที่พัฒนาจาก .NET Application จะได้ภาษากลางที่เรียกว่า Intermediate Language (IL) ที่จะต้องส่งให้ .NET Platform เป็นตัวกลางในการแปลภาษาแล้วจึงกลายเป็นภาษาเครื่อง (Machine code) ซึ่งแตกต่างจากโปรแกรมที่ไม่ได้พัฒนาด้วย .NET Technology ที่เมื่อพัฒนาแล้วได้ภาษาเครื่องออกมาทันที ซึ่งประสิทธิภาพของโปรแกรมที่ทำงานบน .Net Framework นั้นจะได้ประมาณ 80% ของโปรแกรมที่ไม่ได้พัฒนาด้วย .NET Technology (เช่น delphi หรือ Visual Basic 6.0) ทั้งนี้ความแตกต่างด้านประสิทธิภาพของโปรแกรมนั้นขึ้นอยู่กับประสิทธิภาพของคอมพิวเตอร์ที่ใช้งานด้วย

3. โปรแกรมที่พัฒนา .NET Application จะถูกควบคุมให้อยู่ในสภาพแวดล้อมที่ที่กำหนดไว้บน .NET Framework ซึ่งเป็นข้อดีในแง่ของความน่าเชื่อถือของระบบ และผลกระทบที่อาจเกิดขึ้นว่าจะไม่กระทบต่อการทำงานส่วนอื่นๆ

4. โปรแกรมที่พัฒนาด้วย .NET Technology โดยส่วนใหญ่แล้วจะไม่สามารถเชื่อมต่อโดยตรงให้เข้ากับโปรแกรมที่ไม่ได้พัฒนาด้วย .NET Technology ได้ การใช้งานร่วมกันระหว่างโปรแกรมจึงเกิดขึ้นเฉพาะระหว่างโปรแกรมที่พัฒนาด้วย .NET Technology แต่อย่างไรก็ตามทาง Microsoft ได้ออกแบบให้มีทางออกในการเชื่อมต่อกับโปรแกรมอื่นๆ ได้โดยง่ายผ่านเทคโนโลยี Web Service ซึ่งทำให้รูปแบบการทำงานระหว่างโปรแกรมอยู่ในรูปแบบที่เป็นมาตรฐานและเปิดกว้างมากขึ้น

2.4 TCP/IP

TCP/IP (Transmission Control Protocol/Internet Protocol) เป็นชุดของโปรโตคอลที่ถูกใช้ในการสื่อสารผ่านเครือข่ายอินเทอร์เน็ต โดยมีวัตถุประสงค์เพื่อให้สามารถใช้สื่อสารจากต้นทางข้ามเครือข่ายไปยังปลายทางได้ และสามารถหาเส้นทางที่จะส่งข้อมูลไปได้เองโดยอัตโนมัติ ถึงแม้ว่าในระหว่างทางอาจจะผ่านเครือข่ายที่มีปัญหา โปรโตคอลก็ยังคงหาเส้นทางอื่นในการส่งผ่านข้อมูลไปให้ถึงปลายทางได้ [12] [13] [14]

2.4.1 วัตถุประสงค์ของ TCP/IP Protocol

TCP/IP มีจุดประสงค์ของการสื่อสารตามมาตรฐาน 3 ข้อดังนี้

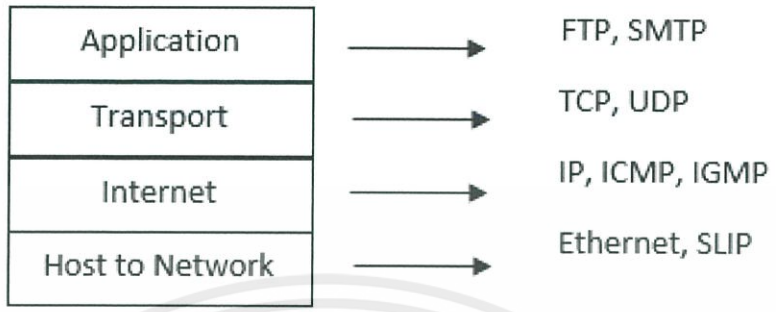
1. เพื่อใช้ติดต่อสื่อสารระหว่างระบบที่มีความแตกต่างกัน

2. ใช้แก้ไขปัญหาที่เกิดขึ้นในระบบเครือข่าย เช่น ในกรณีที่ผู้ส่งและผู้รับยังคงมีการติดต่อกันอยู่ แต่โหนดกลางที่ใช้เป็นผู้ช่วยรับ-ส่งเกิดเสียหายใช้การไม่ได้ หรือสายสื่อสารบางช่วงถูกตัดขาด กฎการสื่อสารนี้จะต้องสามารถจัดหาทางเลือกอื่นเพื่อทำให้การสื่อสารดำเนินต่อไปได้โดยอัตโนมัติ

3. มีความคล่องตัวต่อการสื่อสารข้อมูลได้หลายชนิดทั้งแบบที่ไม่มีความเร่งด่วน เช่น การจัดส่งแฟ้มข้อมูลและแบบที่ต้องการรับประกันความเร่งด่วนของข้อมูล เช่น การสื่อสารแบบ real-time และทั้งการสื่อสารแบบเสียง (voice) และข้อมูล (data)

2.4.2 โครงสร้าง TCP/IP

โครงสร้างของ TCP/IP สามารถแบ่งออกเป็น 4 ชั้นได้ดังรูป 2.3



รูปที่ 2.3 โครงสร้าง TCP/IP

จากรูปที่ 2.3 สามารถอธิบายรายละเอียด ดังนี้

1) ชั้นโฮสต์-เครือข่าย (Host-to-Network Layer)

โพรโตคอลสำหรับการควบคุมการสื่อสารในชั้นนี้เป็นสิ่งที่ไม่มีการกำหนดรายละเอียดอย่างเป็นทางการ หน้าที่หลักคือการรับข้อมูลจากชั้นสื่อสาร IP มาแล้วส่งไปยังโหนดที่ระบุไว้ในเส้นทางเดินข้อมูล ทางด้านผู้รับก็จะทำงานในทางกลับกันคือรับข้อมูลจากสายสื่อสารแล้วนำส่งให้กับโปรแกรมในชั้นสื่อสาร

2) ชั้นสื่อสารอินเทอร์เน็ต (The Internet Layer)

ชั้นสื่อสารอินเทอร์เน็ตจะใช้ประเภทของระบบการสื่อสารที่เรียกว่า ระบบเครือข่ายแบบสลับช่องสื่อสารระดับแพ็กเก็ต (packet-switching network) ซึ่งเป็นการติดต่อแบบไม่ต่อเนื่อง (Connectionless) หลักการทำงานคือการปล่อยให้ข้อมูลขนาดเล็กที่เรียกว่าแพ็กเก็ต (Packet) สามารถไหลจากโหนดผู้ส่งไปตามโหนดต่างๆ ในระบบจนถึงจุดหมายปลายทางได้โดยอิสระ แต่ถ้าหากมีการส่งแพ็กเก็ตออกมาเป็นชุดโดยมีจุดหมายปลายทางเดียวกันในระหว่างการเดินทางในเครือข่าย แพ็กเก็ตแต่ละตัวในข้อมูลชุดนี้จะเป็นอิสระแก่กัน ดังนั้นแพ็กเก็ตที่ส่งไปถึงปลายทางไม่จำเป็นต้องเรียงตามลำดับ

3) ชั้นสื่อสารนำส่งข้อมูล (Transport Layer)

แบ่งเป็นโพรโตคอล 2 ชนิดตามลักษณะของการส่งข้อมูล ชนิดแรกเรียกว่า Transmission Control Protocol (TCP) เป็นแบบที่มีการกำหนดช่วงการสื่อสารตลอดระยะเวลาการสื่อสาร (connection-oriented) ซึ่งจะยอมให้มีการส่งข้อมูลเป็นแบบ byte stream ที่ไว้ใจได้โดยไม่มีข้อผิดพลาด ข้อมูลที่มีปริมาณมากจะถูกแบ่งออกเป็นส่วนเล็กๆ เรียกว่า message ซึ่งจะถูกส่งไปยังผู้รับผ่านทางชั้นสื่อสารของอินเทอร์เน็ต โดยฝ่ายผู้รับจะนำ message เหล่านั้นมาเรียงต่อกันตามลำดับเป็นข้อมูลตัวเดิมและนอกจากนี้ TCP ยังมีความสามารถในการควบคุมการไหลของข้อมูลเพื่อป้องกันไม่ให้ผู้ส่งส่งข้อมูลเร็วเกินกว่าที่ผู้รับจะทำงานได้ทันอีกด้วย

โปรโตคอลการส่งข้อมูลแบบที่สองเรียกว่า UDP (User Datagram Protocol) เป็นการติดต่อแบบไม่ต่อเนื่อง (connectionless) มีการตรวจสอบความถูกต้องของข้อมูลแต่จะไม่มีการแจ้งกลับไปยังผู้ส่ง จึงถือได้ว่าไม่มีการตรวจสอบความถูกต้องของข้อมูล อย่างไรก็ตาม วิธีการนี้มีข้อดีในด้านความรวดเร็วในการส่งข้อมูลจึงนิยมใช้ในระบบผู้ให้และผู้ใช้บริการ (client/server system) ซึ่งมีการสื่อสารแบบถาม/ตอบ (request/reply) นอกจากนั้นยังใช้ในการส่งข้อมูลประเภทภาพเคลื่อนไหวหรือการส่งเสียง (voice) ทางอินเทอร์เน็ต

4. ชั้นสื่อสารการประยุกต์ (Application Layer)

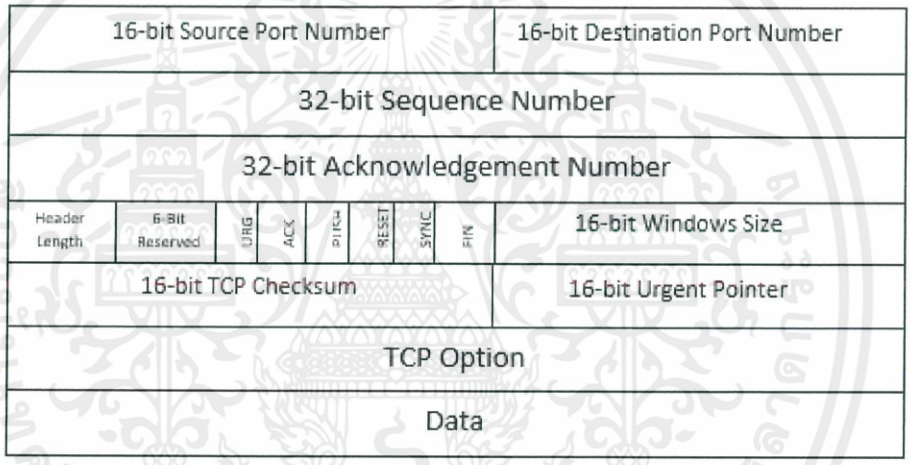
เป็นชั้นสื่อสารที่เกี่ยวข้องกับการทำงานของโปรแกรมประยุกต์ต่างๆ ที่ใช้สำหรับการเข้าถึงเครือข่ายโดยจะมีอินเตอร์เฟซเพื่อใช้ในการโต้ตอบกันระหว่างผู้ใช้งานกับคอมพิวเตอร์ จากนั้นคอมพิวเตอร์จะแปลงข้อมูลที่ได้รับจากผู้ใช้เข้าสู่ระบบ โดยมีโพรโตคอล เช่น โพรโตคอลสำหรับสร้างจอเทอร์มินัลเสมือน เรียกว่า TELNET ซึ่งช่วยให้ผู้ใช้สามารถติดต่อกับเครื่องโฮสต์ที่อยู่ไกลออกไปโดยผ่านอินเทอร์เน็ต และสามารถทำงานได้เสมือนกับว่ากำลังนั่งทำงานอยู่ที่เครื่องโฮสต์นั้น, โพรโตคอลสำหรับการจัดการแฟ้มข้อมูล เรียกว่า FTP โพรโตคอลสำหรับการจัดการแฟ้มข้อมูลช่วยในการคัดลอกแฟ้มข้อมูลมาจากเครื่องอื่นที่อยู่ในระบบเครือข่ายหรือส่งสำเนาแฟ้มข้อมูลไปยังเครื่องใดๆก็ได้และโพรโตคอลสำหรับการให้บริการจดหมายอิเล็กทรอนิกส์เรียกว่า SMTP โดย โพรโตคอลสำหรับให้บริการจดหมายอิเล็กทรอนิกส์ช่วยในการจัดส่งข้อความไปยังผู้ใช้ในระบบหรือรับข้อความที่มีผู้ส่งเข้ามา

2.4.3 การทำงานของ TCP/IP

TCP / IP คือการรวมกันของ 2 โปรโตคอลที่แยกกัน คือ Transmission Control Protocol (TCP) และ Internet Protocol (IP) [15] ซึ่งแต่ละโพรโตคอลมีการทำงานดังนี้

2.4.3.1 TCP (Transmission Control Protocol)

ในส่วนของ Transmission Control Protocol มีหน้าที่รับผิดชอบในการสร้างความน่าเชื่อถือในการเชื่อมต่อ/ส่งข้อมูลผ่านเครือข่ายที่เชื่อมต่อกับอินเทอร์เน็ต โดย TCP จะตรวจสอบแพ็คเก็ตสำหรับข้อผิดพลาดและส่งคำขอสำหรับการส่งใหม่ถ้าหากพบข้อผิดพลาดนั้น และมีกลไกควบคุมการรับส่งข้อมูลให้มีความถูกต้อง (reliable) และมีการสื่อสารอย่างเป็นกระบวนการ (connection-oriented) [13] [15]



รูปที่ 2.4 TCP Header

จากรูปที่ 2.4 สามารถอธิบายรายละเอียด ดังนี้

• Source Port Number : หมายเลขพอร์ตต้นทางที่ส่งเซกเมนต์นี้

• Destination Port Number : หมายเลขพอร์ตปลายทางที่จะเป็นผู้รับเซกเมนต์

• Sequence Number : ฟิลด์ที่ระบุหมายเลขลำดับอ้างอิงในการสื่อสารข้อมูลแต่ละครั้ง เพื่อใช้ในการแยกแยะว่าเป็นข้อมูลของชุดใด และนำมาจัดลำดับได้ถูกต้อง

• Acknowledgment Number : ทำหน้าที่เช่นเดียวกับ Sequence Number แต่จะใช้ในการตอบรับ

• Header Length : โดยปกติความยาวของเฮดเดอร์ TCP จะมีความยาว 20 ไบต์ แต่อาจจะมากกว่านั้น ถ้ามีข้อมูลในฟิลด์ option แต่ต้องไม่เกิน 60 ไบต์

• Flag : เป็นข้อมูลระดับบิตที่อยู่ในเฮดเดอร์ TCP โดยใช้เป็นตัวบอกคุณสมบัติของแพ็กเก็ต TCP ขณะนั้นๆ และใช้เป็นตัวควบคุมจังหวะการรับส่งข้อมูลด้วย ซึ่ง Flag มีอยู่ทั้งหมด 6 บิต แบ่งได้ดังนี้

Type	Description
URG	ใช้บอกความหมายว่าเป็นข้อมูลด่วน และประกอบไปด้วยข้อมูลพิเศษ (อยู่ใน Urgent Pointer)
ACK	แสดงว่าข้อมูลในฟิลด์ Acknowledge Number นำมาใช้งานได้
PSH	เป็นการแจ้งให้ผู้รับข้อมูลทราบว่าควรจะส่งข้อมูล Segment นี้ไปยัง Application ที่กำลังรออยู่โดยเร็ว
RST	ยกเลิกการติดต่อ (Reset) เนื่องจากในกรณีที่เกิดการสับสนขึ้นด้วยเหตุผลต่างๆ เช่น โฮสต์มีปัญหา ให้เริ่มสื่อสารใหม่
SYN	ใช้ในการเริ่มต้นขอติดต่อกับปลายทาง
FIN	ใช้ส่งเพื่อแจ้งให้ปลายทางทราบว่ายุติการติดต่อ

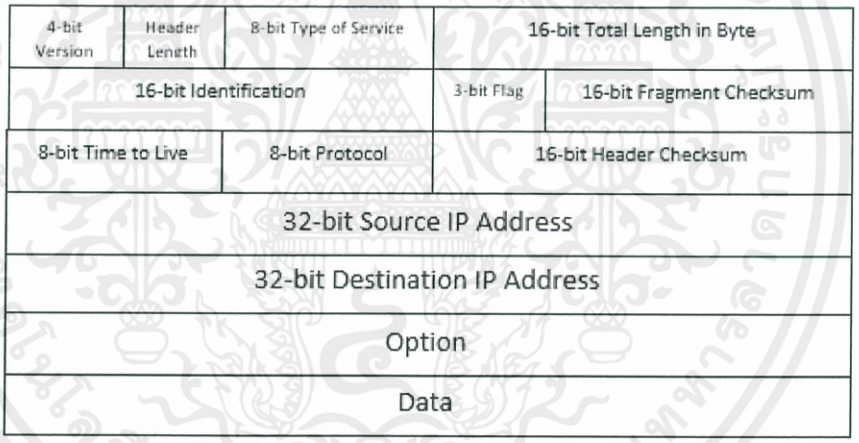
ตารางที่ 2.2 ตารางแสดงชนิดของ Flag

Flag ในเฮดเดอร์ของ TCP มีความสำคัญในการกำหนดการทำงานของ TCP segment เนื่องจากข้อมูลในเฮดเดอร์ของ TCP จะมีข้อมูลครบถ้วนทั้งการรับและการส่งข้อมูล ซึ่งในการทำงานแต่ละอย่างจะมีการใช้งานฟิลด์ไม่เหมือนกัน Flag จะเป็นตัวกำหนดว่าให้ใช้งานฟิลด์ไหน เช่น ฟิลด์ Acknowledgment number จะไม่ถูกใช้ในขั้นตอนการเริ่มต้นการเชื่อมต่อ แต่จะมีข้อมูลในฟิลด์ซึ่งเป็นข้อมูลที่ไม่มีความหมายใดๆ ซึ่งถ้าไม่มี flag เป็นตัวกำหนดก็อาจจะมีการนำข้อมูลมาใช้ และก่อให้เกิดความผิดพลาด

2.4.3.2 IP (Internet Protocol)

Internet Protocol เป็นโปรโตคอลในระดับเน็ตเวิร์คเลเยอร์ มีหน้าที่รับผิดชอบในการส่งข้อมูล จัดการเกี่ยวกับแอดเดรสและข้อมูลอีกทั้งยังควบคุมการส่งข้อมูลบางอย่างที่ใช้ในการหาเส้นทางของแพ็กเก็ต ซึ่งกลไกในการหาเส้นทางของ IP จะมีความสามารถในการหาเส้นทางที่ดีที่สุด และสามารถเปลี่ยนแปลงเส้นทางได้ในระหว่างการส่งข้อมูลโดยตัว IP นั้นมีวิธีการที่ช่วยให้คอมพิวเตอร์บนอินเทอร์เน็ตสามารถส่งต่อแพ็คเก็ตไปยังคอมพิวเตอร์เครื่องอื่นซึ่งอยู่ในระยะหนึ่งช่วงหรือมากกว่านั้นใกล้กับเครื่องผู้รับ

การเชื่อมต่อของ IP เพื่อทำการส่งข้อมูล จะเป็นแบบ connectionless หรือเกิดเส้นทางการเชื่อมต่อในทุกๆครั้งของการส่งข้อมูล 1 เซกเมนต์ โดยจะไม่ทราบถึงข้อมูลเซกเมนต์ที่ส่งก่อนหน้าหรือส่งตามมา แต่การส่งข้อมูลใน 1 เซกเมนต์ อาจจะเกิดการส่งได้หลายครั้งในกรณีที่มีการแบ่งข้อมูลออกเป็นส่วนย่อยๆ (fragmentation) และถูกนำไปรวมเป็นเซกเมนต์เดิมเมื่อถึงปลายทาง



รูปที่ 2.5 IP Header

จากรูปที่ 2.5 สามารถอธิบายรายละเอียด ดังนี้

เฮดเดอร์ของ IP โดยปกติจะมีขนาด 20 bytes ยกเว้นในกรณีที่มีการเพิ่ม option บางอย่าง ฟิลด์ของเฮดเดอร์ IP จะมีความหมายดังนี้

• Version : หมายเลขเวอร์ชันของโปรโตคอล ที่ใช้งานในปัจจุบันคือ เวอร์ชัน 4 (IPv4) และเวอร์ชัน 6 (IPv6)

• Header Length : ความยาวของเฮดเดอร์ โดยทั่วไปถ้าไม่มีส่วน option จะมีค่าเป็น 5 (5*32 bit)

• Type of Service (TOS) : ใช้เป็นข้อมูลสำหรับเราเตอร์ในการตัดสินใจเลือกการเราต์ข้อมูลในแต่ละเซกเมนต์ แต่ในปัจจุบันไม่มีการนำไปใช้งานแล้ว

• Length : ความยาวทั้งหมดเป็นจำนวนไบต์ของเซกเมนต์ ซึ่งด้วยขนาด 16 บิตของฟิลด์ จะหมายถึงความยาวสูงสุดของเซกเมนต์ คือ 65535 bytes (64k) แต่ในการส่งข้อมูลจริง ข้อมูลจะถูกแยกเป็นส่วนๆตามขนาดของ MTU ที่กำหนดในลิงค์เลเยอร์ และนำมารวมกันอีกครั้งเมื่อส่งถึงปลายทาง แอพพลิเคชั่นส่วนใหญ่จะมีขนาดของเซกเมนต์ไม่เกิน 512 bytes

• Identification : เป็นหมายเลขของเซกเมนต์ในกรณีที่มีการแยกเซกเมนต์เมื่อข้อมูลส่งถึงปลายทางจะนำข้อมูลที่มี identification เดียวกันมารวมกัน

• Flag : ใช้ในกรณีที่มีการแยกเซกเมนต์

• Fragment Offset : ใช้ในการกำหนดตำแหน่งข้อมูลในเซกเมนต์ที่มีการแยกส่วน เพื่อให้สามารถนำกลับมาเรียงต่อกันได้อย่างถูกต้อง

• Time to Live (TTL) : กำหนดจำนวนครั้งที่มากที่สุดที่เซกเมนต์จะถูกส่งระหว่าง hop (การส่งผ่านข้อมูลระหว่างเน็ตเวิร์ค) เพื่อป้องกันไม่ให้เกิดการส่งข้อมูลโดยไม่สิ้นสุด โดยเมื่อข้อมูลถูกส่งไป 1 hop จะทำการลดค่า TTL ลง 1 เมื่อค่าของ TTL เป็น 0 และข้อมูลยังไม่ถึงปลายทาง ข้อมูลนั้นจะถูกยกเลิก และเราเตอร์สุดท้ายจะส่งข้อมูล ICMP แจ้งกลับมายังต้นทางว่าเกิด timeout ในระหว่างการส่งข้อมูล

• Protocol : ระบุโปรโตคอลที่ส่งในเซกเมนต์ เช่น TCP, UDP หรือ ICMP

• Header Checksum : ใช้ในการตรวจสอบความถูกต้องของข้อมูลในเฮดเดอร์

• Source IP Address : หมายเลข IP ของผู้ส่งข้อมูล

• Destination IP Address : หมายเลข IP ของผู้รับข้อมูล

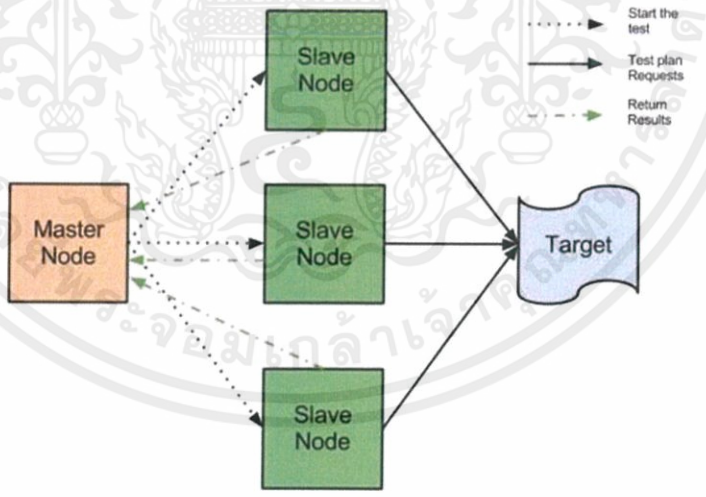
• Data : ข้อมูลจากโปรโตคอลระดับบน

2.5 Load Testing

Load Testing คือกระบวนการทดสอบโดยการกำหนดความต้องการของระบบแล้ววัดการตอบสนอง โดยการทดสอบโหลดจะดำเนินการเพื่อหาพฤติกรรมของระบบภายใต้สภาวะปกติและโหลดสูงสุด ช่วยให้สามารถระบุขีดความสามารถในการดำเนินงานของแอปพลิเคชัน รวมถึงปัญหาคอขวดและพิจารณาว่าองค์ประกอบใดที่ทำให้เกิดปัญหาเมื่อในระบบมีโหลดสูงกว่าปกติ จะเป็นการทดสอบการตอบสนองของระบบที่มีโหลดสูงสุดผิดปกติ จะเรียกว่า Stress Testing และสามารถนำผลของการทดสอบไปแก้ไขตัวระบบให้สามารถรองรับการทำงานได้ดียิ่งขึ้น [16] [17]

2.6 Master-Slave

เป็นรูปแบบการดำเนินงานอย่างหนึ่งซึ่งจะแบ่งออกเป็น 2 ส่วน คือ master กับ slave (บางครั้งเรียกว่า boss กับ worker) โดยเมื่อ user ทำการใช้งานที่ตัว master จะนำคำสั่งส่งต่อให้ slave เพื่อดำเนินงานตามคำร้องเหล่านั้น โดยปกติแล้ว master จะควบคุมเฉพาะการสั่งการ slaves แต่ในส่วนของการดำเนินงาน slave จะควบคุมด้วยตัวเอง และ slave แต่ละตัวจะดำเนินงานอย่างเป็นอิสระจาก slave อื่น [18] [19]



รูปที่ 2.6 การสื่อสารระหว่าง Master-Slave และ Target

จากรูป 2.6 สามารถอธิบายได้ว่า

- Master เป็นตัวควบคุม และสรุปผลการทำงานจาก slave
- Slave เป็นตัวทำงานจริง สามารถอยู่ภายในและภายนอก Network และมีได้มากกว่า 1 ตัว

- Target จะเป็นเครื่องเป้าหมาย ที่จะถูกทำการทดสอบ

การใช้รูปแบบการดำเนินงานแบบ Master-Slave มีข้อดีข้อเสียสรุปได้ดังนี้

ข้อดี เหมาะสำหรับการประมวลผลของงานที่ต้องการการประมวลผลมากๆ เนื่องจากมีตัวประมวลผลหลายตัวในระบบ จึงทำให้สามารถกระจายงานที่ต้องประมวลผลได้ ส่งผลให้การประมวลผลรวดเร็วขึ้น อีกทั้งยังสามารถเพิ่มตัวประมวลผลได้อย่างไม่มีจำกัด และเนื่องจากมีการแยกการทำงานระหว่างส่วนควบคุมและส่วนประมวลผล ทำให้ส่วนโปรแกรมประยุกต์ของผู้ใช้ที่เป็นส่วนควบคุมกับส่วนที่ทำการประมวลผลแยกจากกัน ถ้าหากมีส่วนประมวลผลบางตัวเกิดข้อผิดพลาดก็จะไม่ส่งผลต่อส่วนควบคุมหลักและงานของตัวประมวลที่ผิดพลาดก็จะถูกถ่ายเทไปให้ตัวประมวลผลตัวอื่นต่อไป

ข้อเสีย เนื่องจากแยกการทำงานระหว่างส่วนควบคุมและส่วนประมวลผลจึงจำเป็นต้องมีการใช้ Network ในการติดต่อสื่อสารระหว่าง Master กับ Slave ทำให้ประสิทธิภาพของระบบนั้นขึ้นอยู่กับ Network ด้วย และในระหว่างการทำงานหน่วยประมวลผลของเครื่อง Master อาจทำงานหนักเกินไป เนื่องจากต้องประมวลผลลัพธ์ที่ได้จาก Slave จำนวนหลายตัวในขณะที่ตอบสนองและแสดงผลโปรแกรมประยุกต์ที่ User เรียกใช้งาน ซึ่งถ้าหากเครื่อง Master มีประสิทธิภาพต่ำ จะทำให้เกิดปัญหาคอขวดที่เครื่อง Master ซึ่งส่งผลต่อการทดสอบทั้งด้านการสื่อสาร Slave และการประมวลผลลัพธ์ที่ได้รับมาจาก Slave ทำให้การประมวลผลลัพธ์อาจทำได้ไม่ถูกต้อง แม้ว่าเครื่อง Slave จะสามารถทำงานได้อย่างสมบูรณ์ก็ตามหรือในกรณีที่เลวร้ายที่สุด อาจส่งผลให้เครื่อง Master หยุดทำงานได้

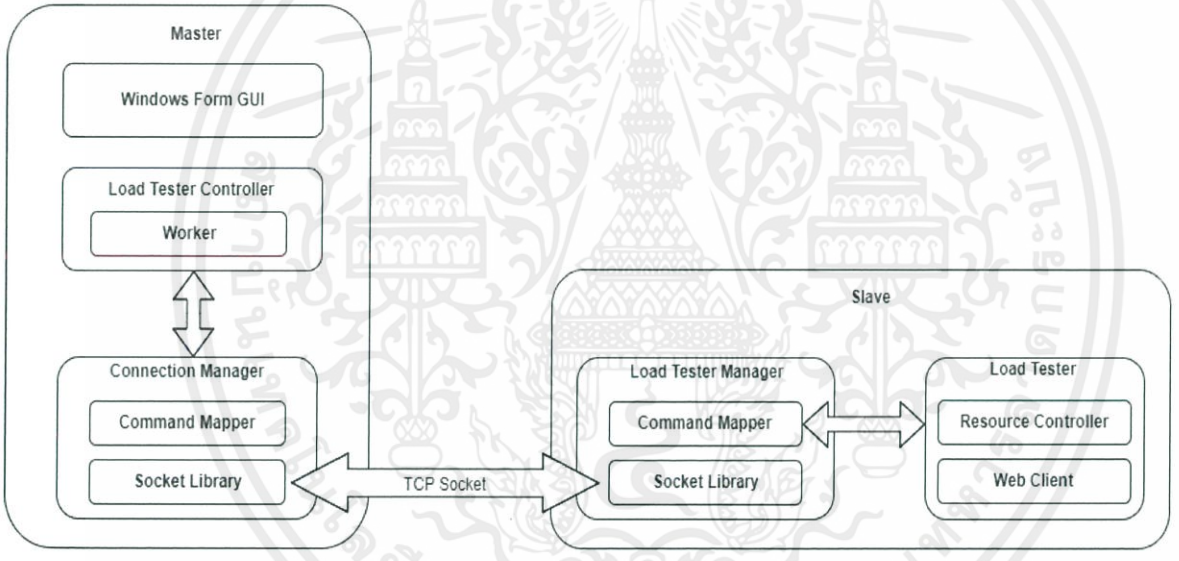
บทที่ 3

การออกแบบและการพัฒนา

ในการพัฒนาเครื่องมือทดสอบความสามารถการทำงานของ Web Application ที่รองรับการทำงานจำนวนมาก โดยการใช้คอมพิวเตอร์หลายเครื่องช่วยในการทดสอบในแต่ละครั้ง ได้มีการกำหนดรูปแบบการเชื่อมต่อเป็นแบบ Master-Slave เข้ามาช่วยในการติดต่อระหว่างเครื่องที่เป็นตัวสั่งการกับเครื่องที่ดำเนินการทดสอบ โดยมีภาพรวมและการออกแบบระบบดังต่อไปนี้

3.1 ภาพรวมของระบบ

ในส่วนนี้เป็นการอธิบายภาพรวมของระบบ Load Testing และการติดต่อกันของแต่ละส่วนของโปรแกรมดังรูปที่ 3.1



รูปที่ 3.1 ภาพรวมของระบบ

จากรูปที่ 3.1 ระบบจะประกอบด้วยสองส่วนหลักคือส่วน Master ที่อยู่ทางฝั่งซ้ายของรูปภาพและส่วนของ Slave ที่อยู่ทางขวาของรูปภาพ โดยมีหลักการทำงานดังนี้

3.1.1 Master โดยส่วนของ Master ทำงานโดยติดต่อกับ Slave ด้วย Socket โดยทำหน้าที่เป็น Client คอยส่งข้อมูล/คำสั่งแก่เครื่อง Slave โดยใช้ Library Socket จาก .Net Framework ซึ่งระบบในส่วนของ Master ประกอบด้วยโมดุลการทำงาน 3 โมดุล ได้แก่ Window Form GUI, Load Tester Controller, Connection Manager ซึ่งมีรายละเอียดการทำงานดังนี้

3.1.1.2 การทำงานของ Window Form GUI ใน Master ทำหน้าที่รับค่าการตั้งค่าส่วนต่างๆ จาก User เพื่อนำมาส่งต่อให้ส่วนอื่นที่จำเป็นต้องใช้และแสดงผล

3.1.1.3 การทำงานของ Load Tester Controller ใน Master ทำหน้าที่ควบคุม Slave ระหว่างการทำงาน

3.1.1.4 การทำงานของ Connection Manager ใน Master ทำหน้าที่ดูแลการเชื่อมต่อและการรับส่งข้อมูลระหว่าง Master กับ Slave

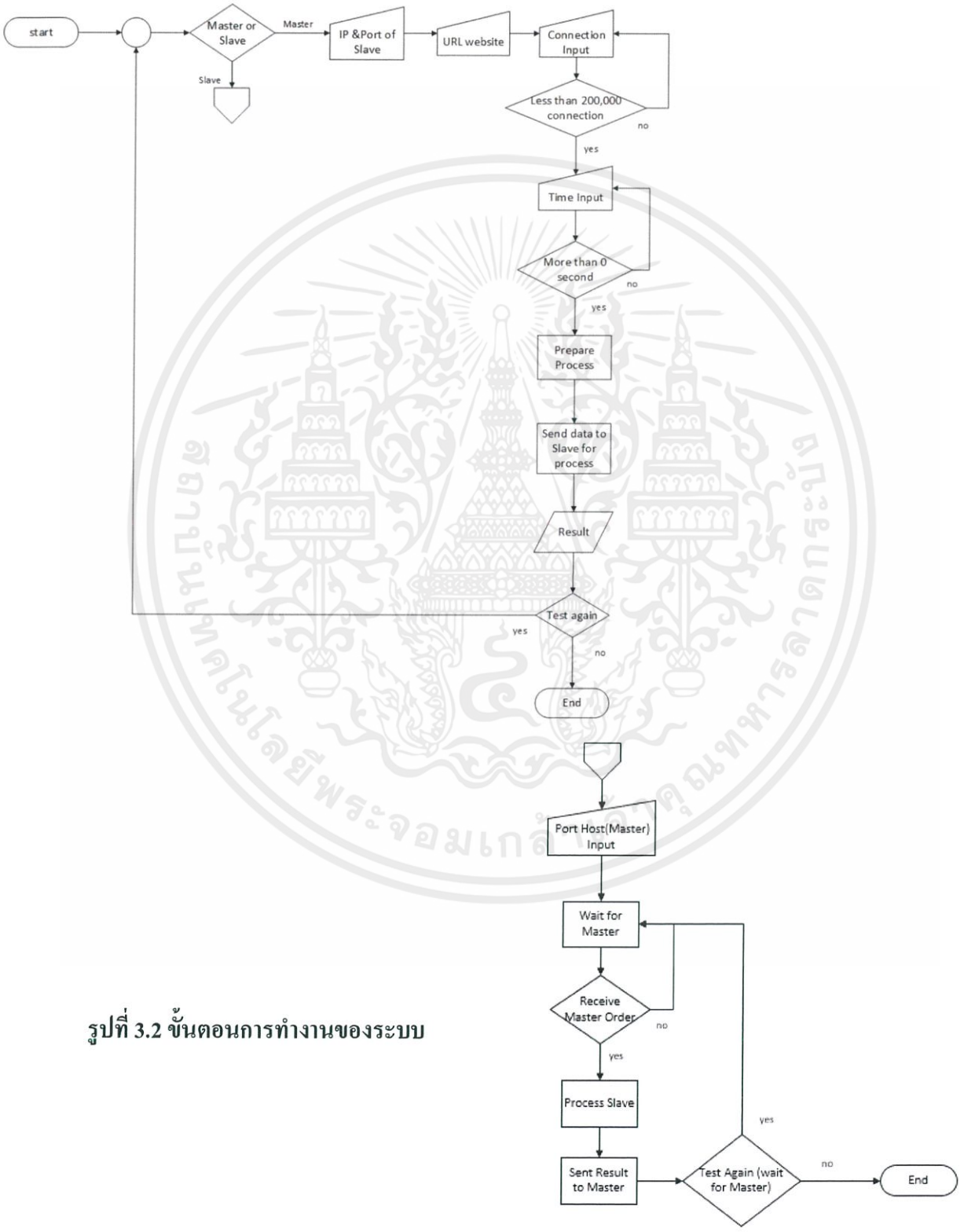
3.1.2 Slave ในส่วนของ Slave ทำงานโดยติดต่อกับ Master ด้วย Socket โดยทำหน้าที่เป็น Server คอยรับข้อมูล/คำสั่งจากเครื่อง Master โดยใช้ Library socket จาก .Net Framework แล้วส่งค่าต่าง ๆ ให้กับ Load Tester เป็นตัวทดสอบซึ่งอยู่ภายใน Slave ซึ่งระบบในส่วนของ Slave ประกอบด้วย 2 โมดุล ได้แก่ Load Tester, Load Tester Manager ซึ่งมีรายละเอียดการทำงานดังนี้

3.1.2.2 การทำงานของ Load Tester ใน Slave ทำหน้าที่สร้าง Connection ขึ้นมาตามจำนวนที่กำหนดเพื่อใช้วัดประสิทธิภาพตามที่กำหนด โดยในการสร้างการเชื่อมต่อจะใช้ HttpWebRequest ซึ่งเป็น Library ใน .Net Framework แล้วควบคุมจำนวนที่สร้างให้เป็นไปตามที่กำหนด

3.1.2.3 การทำงานของ Load Tester Manager ใน Slave ทำหน้าที่ดูแลการเชื่อมต่อและการรับส่งข้อมูลระหว่าง Master กับ Slave รวมถึงควบคุมตัวการทำงานของ Load Tester เพื่อให้มีจำนวน Concurrent Connection เป็นไปตามค่าที่กำหนด

3.2 ขั้นตอนการทำงานของระบบ

ขั้นตอนการทำงานของระบบทั้งหมดแบ่งเป็น 2 ส่วนคือ ส่วนของ Master และส่วนของ Slave แสดงดังรูปที่ 3.2



รูปที่ 3.2 ขั้นตอนการทำงานของระบบ

จากรูป 3.2 สามารถอธิบายขั้นตอนการทำงานซึ่งจะแบ่งเป็น 2 ฝั่งคือ Master และ Slave ได้ดังนี้

ฝั่ง Master

- ทำการ Set IP และ Port ของ Slave ที่จะเพิ่มเข้าไปในระบบ
- จากนั้นทำการจำลองการเข้าเว็บเพื่อเก็บค่า URL Website
- Set ประเภทและจำนวนของ Connection และตั้งค่าเวลาที่ใช้ในการทดสอบ
- เมื่อ Set ค่าทั้งหมดที่เครื่อง Master เสร็จแล้ว เครื่อง Master จะส่งข้อมูลทั้งหมดให้เครื่อง Slave เป็นตัวดำเนินการทำงานต่อไป
- ในระหว่างการทำงานนั้น Master จะรอรับข้อมูลจากเครื่อง Slave นำมาประมวลผล
- เมื่อการทำงานเสร็จสิ้น Master จะรวบรวมข้อมูลและแสดงผลลัพธ์ออกมา

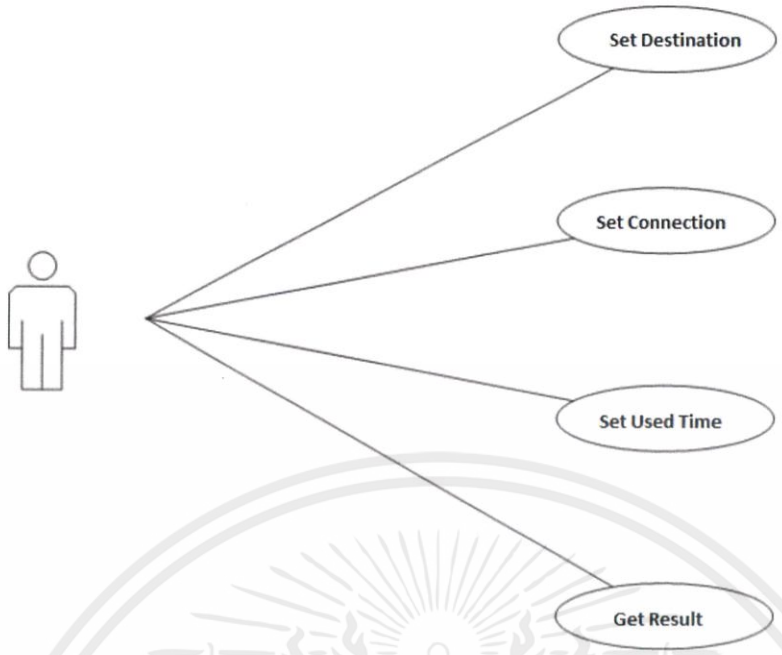
ฝั่ง Slave

- เซ็ตค่า Port ของ Slave ที่ใช้เชื่อมต่อกับเครื่อง Master
- รอการเชื่อมต่อจากเครื่อง Master
- เมื่อการเชื่อมต่อกับ Master สมบูรณ์ จะทำการรอ/ดำเนินการตามคำสั่งที่ Master ส่งมา
- ในระหว่างการทำงาน Slave จะส่งข้อมูลกลับมาที่เครื่อง Master
- เมื่อ Slave ทำงานเสร็จ ก็จะปิดการเชื่อมต่อกับเครื่อง Master

3.3 Use Case Diagram

ความสามารถของโปรแกรม Load Tester ที่ User สามารถกำหนดค่าหรือใช้งานมีดังนี้

1) การกำหนดเป้าหมายในการทดสอบ โดยการกรอก URL ของ Website ที่ต้องการจะทดสอบลงในโปรแกรม
2) การตั้งค่าจำนวน Concurrent Connection ที่จะใช้ในการทดสอบสามารถเลือกการทดสอบได้ 3 แบบ คือ
   - Constant load Test: การทดสอบแบบนี้จะใช้ค่า Concurrent Connection ที่เป็นค่าคงที่ในการทดสอบ
   - Peak load Test: การทดสอบแบบนี้จะทำการเปลี่ยนค่า Concurrent Connectionให้ต่างกันเป็นอย่างมากในระยะเวลาหนึ่ง
   - Increase load Test: การทดสอบแบบนี้จะทำการเพิ่มค่า Concurrent Connection แบบคงที่
3) การตั้งค่าเวลาที่ใช้ในการทดสอบทั้งหมด
4) เมื่อทำการทดสอบเสร็จแล้วจะสามารถเรียกดูผลลัพธ์ของการทดสอบในรูปแบบของกราฟได้ สามารถเขียนเป็น Use Case Diagram ได้ดังรูปที่ 3.3



รูปที่ 3.3 Use Case Diagram

จากรูป 3.3 เราสามารถอธิบาย Use Case Diagram ได้ดังนี้

1) ผู้ใช้กำหนดเป้าในการทดสอบ (Set Destination) โดยใช้ URL หรือ IP ของ Website ที่ต้องการจะทดสอบ

2) ผู้ใช้ทำการกำหนดจำนวน Concurrent Connection (Set Connection) สูงสุดที่ใช้ในการทดสอบ

3) ผู้ใช้ทำการกำหนดเวลา (Set Used Time) ทั้งหมดที่จะใช้ในการทดสอบ

4) ผู้ใช้สามารถเรียกดูผลลัพธ์ (Get Result) ของการทดสอบในแต่ละครั้งได้

3.4 Use Case Specification

ความสามารถของโปรแกรม Load Tester ที่ User สามารถกำหนดค่าหรือใช้งานมีดังนี้

1) Use Case: กำหนดเป้าหมาย (Set Destination)

เพื่อให้ผู้ใช้ทำการกำหนดเป้าหมายในการทดสอบ โดยใช้ IP Address ของ Server ที่ต้องการจะทดสอบ

2) Use Case: กำหนดจำนวน Connection (Set Connection)

เพื่อให้ผู้ใช้ทำการกำหนดจำนวน Connection สูงสุดที่ใช้ในการทดสอบ

3) Use Case: กำหนดเวลาที่ใช้ (Set Used Time)

เพื่อให้ผู้ใช้ทำการกำหนดเวลาทั้งหมดที่จะใช้ในการทดสอบ

4) Use Case: เรียกดูผลทดสอบ (Get Result)

เพื่อให้ผู้ใช้สามารถเรียกดูผลลัพธ์ของการทดสอบในแต่ละครั้งได้

3.5 การออกแบบ

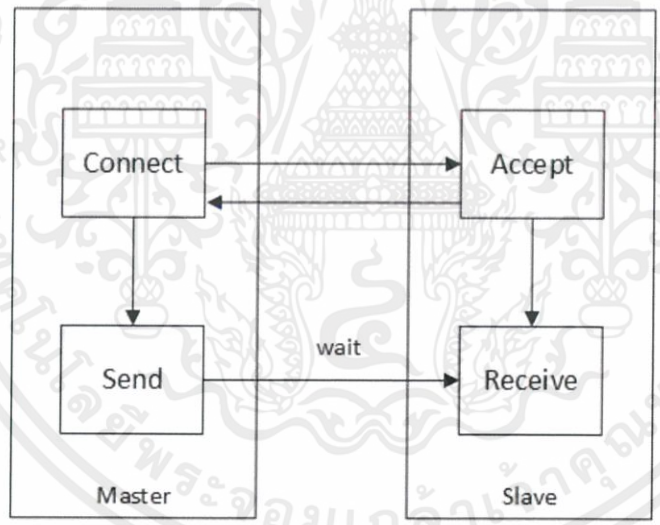
ในการพัฒนาโปรแกรม Load Tester ทางผู้ทดลองได้ทำการแบ่งการออกแบบเป็น 2 ส่วนหลักๆ คือ

1) ส่วนการเชื่อมต่อ (Connection) อยู่ในส่วนของ Connection Manager และ Load Tester Manager ดังรูปที่ 3.1 ซึ่งทำหน้าที่ดูแลการเชื่อมต่อและรับส่งข้อมูลระหว่าง Master กับ Slave

2) ส่วนการควบคุม (Control) อยู่ในส่วนของ Load Tester Controller และ Window Form GUI ดังรูปที่ 3.1 ซึ่งทำหน้าที่รับค่าการตั้งค่า User และควบคุม Slave ระหว่างการทำงาน

3.5.1 ส่วนการเชื่อมต่อ (Connection)

ในการสร้างระบบเพื่อให้เครื่องอุปกรณ์ต่างๆ ติดต่อกันได้ ทางผู้จัดทำได้ใช้การเชื่อมต่อด้วยวิธี Socket Programming โดยรูปแบบการเชื่อมต่อนั้นเป็นประเภท Master-Slave ผู้จัดทำจะขออธิบายการเชื่อมต่อในกรณีที่มี Slave เพียงหนึ่งตัวเท่านั้น เนื่องจากการเชื่อมต่อแบบมี Slave หลายตัวก็มี หลักการที่คล้ายคลึงกัน โดยมีขั้นตอนการทำงานดังรูปที่ 3.4

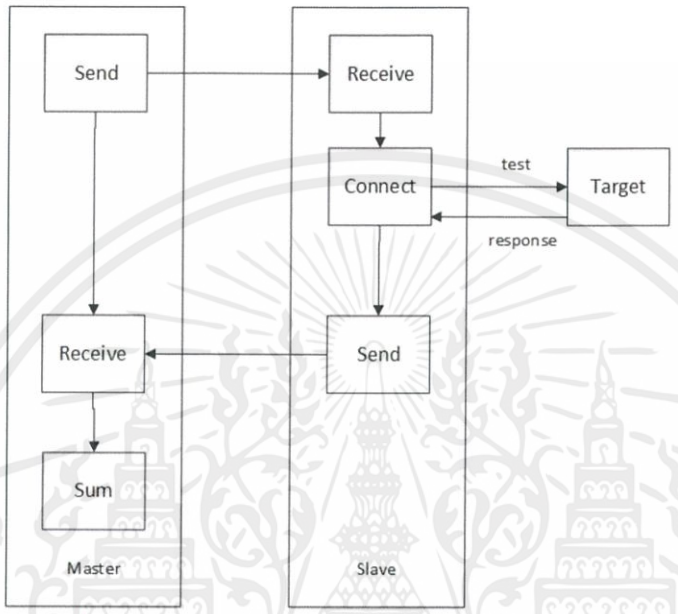


รูปที่ 3.4 การทำงานของ Network ในส่วนการเชื่อมต่อ

จากรูปที่ 3.4 เมื่อเริ่มต้นทำการกำหนดหน้าที่ของแต่ละเครื่องเรียบร้อยแล้ว ขั้นตอนการทำงานนั้นจะเริ่มจากการที่ Master จะส่งการร้องขอไปยัง Slave เพื่อให้เชื่อมต่อกับตน จากนั้น Slave จะตอบรับการร้องขอ เมื่อเชื่อมต่อกันเรียบร้อยแล้วหลังจากนั้น Slave ก็จะทำการรอรับคำสั่งจาก Master ต่อไป

3.5.2 ส่วนการควบคุม (Control)

จากที่ได้กล่าวไว้ข้างต้นว่า รูปแบบการเชื่อมต่อนั้น เป็นประเภท Master-Slave ดังนั้นการเริ่มต้นการทดสอบความสามารถการทำงานของ Web Application จะต้องสั่งการผ่านเครื่องที่เป็น Master เท่านั้น โดยมีขั้นตอนการทำงานดังรูปที่ 3.5



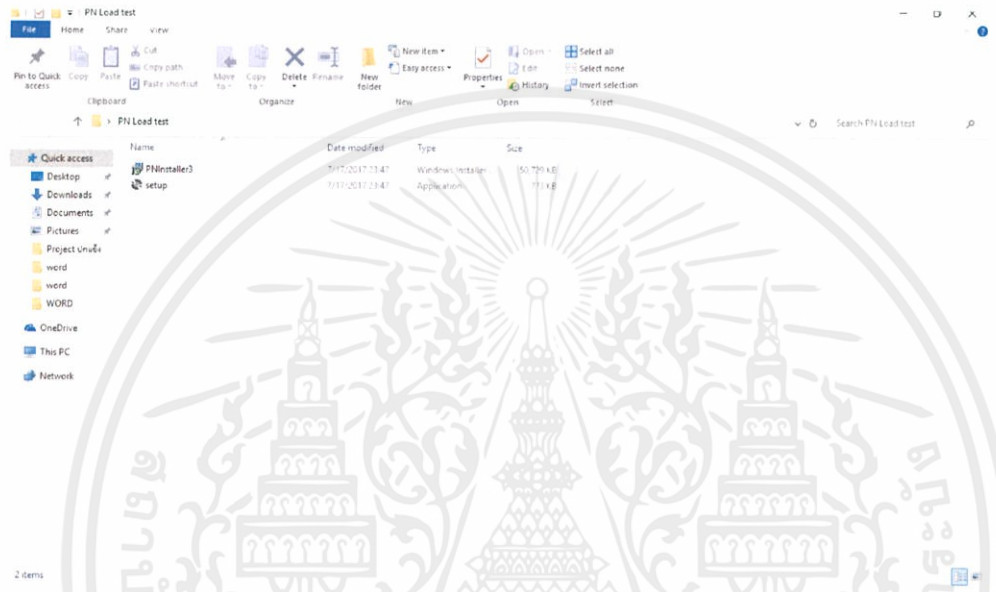
รูปที่ 3.5 การทำงานของ Network ในส่วนการควบคุม

จากรูปที่ 3.5 เมื่อเริ่มต้นทำการกำหนดค่าที่ใช้ในการทดสอบที่ Master จากนั้นส่งค่าของการตั้งค่าทั้งหมดให้ Slave ซึ่ง Slave จะเป็นเครื่องจริงที่ใช้ในการทดสอบ Server ปลายทาง เมื่อทำการทดสอบเสร็จแล้ว Slave จะส่งข้อมูลที่ได้จากการทดสอบทั้งหมดให้ Master นำไปรวบรวมแล้วแสดงผลต่อไป

3.6 การออกแบบ: ส่วนการติดตั้งโปรแกรม

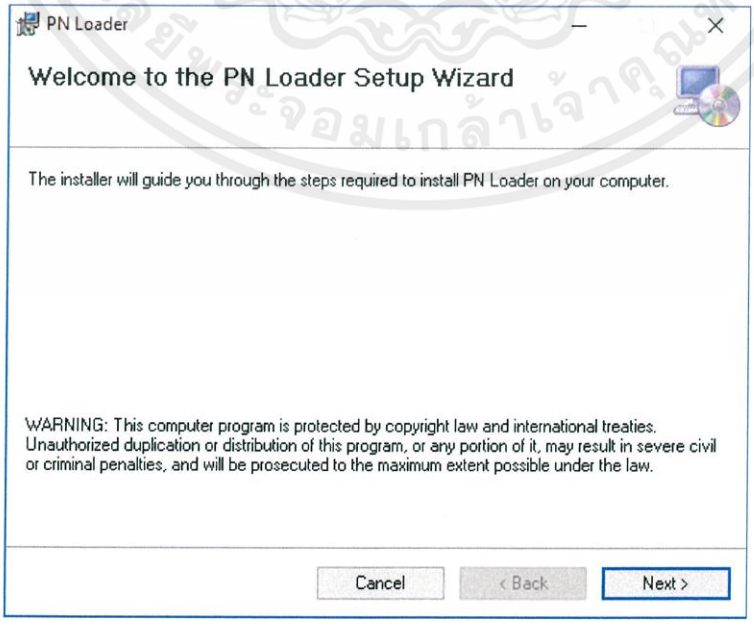
ในการใช้งานโปรแกรม Load Tester ได้มีการออกแบบให้ทำการติดตั้งก่อน ถึงจะสามารถใช้งานได้ ซึ่งใช้ไฟล์ตั้งตัวเดียวกันในการติดตั้งทั้งเครื่อง Slave และเครื่อง Master โดยมีรายละเอียดขั้นตอนการติดตั้งดังนี้

1) ดับเบิลคลิกที่ File ชื่อ PNInstaller3 ดังรูปที่ 3.6



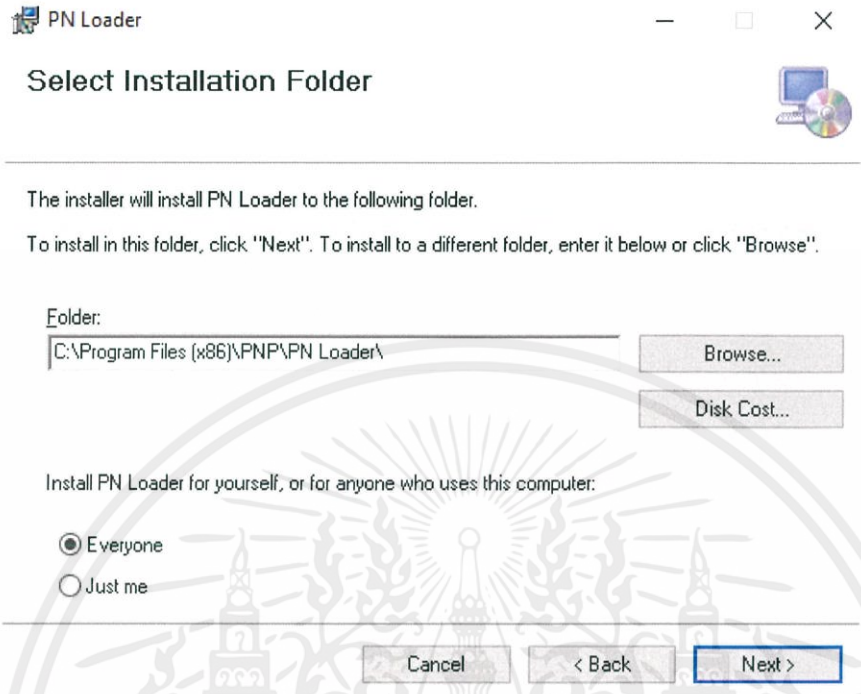
รูปที่ 3.6 ชื่อไฟล์ install โปรแกรม PN Loader

2) ในหน้าต่าง Welcome to the PN Loader Setup Wizard ให้คลิกที่ Next ดังรูปที่ 3.7



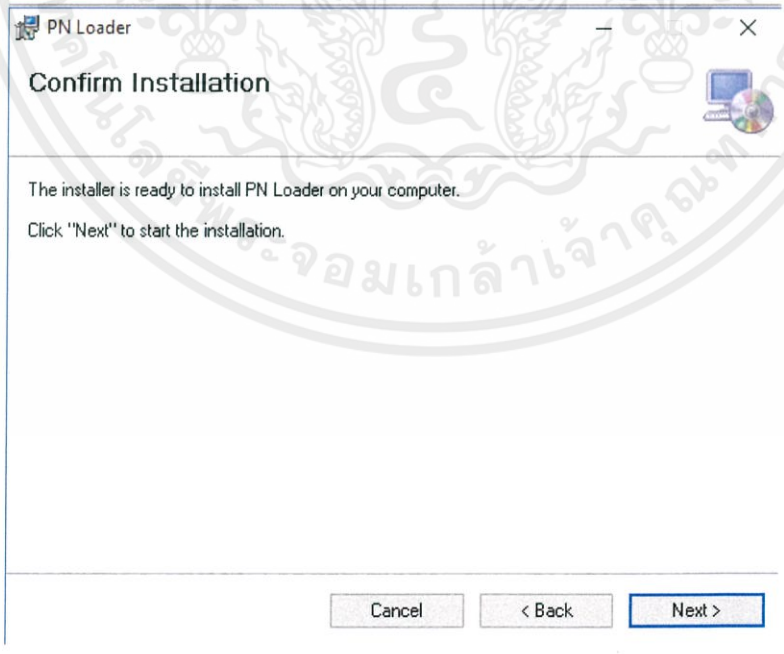
รูปที่ 3.7 หน้าต่าง Welcome to the PN Loader Setup Wizard

3) ในหน้าต่าง Select Installation Folder สามารถเลือก Path ในการลงตัวโปรแกรมได้โดยคลิกที่ Browse เมื่อเลือกเสร็จแล้ว ให้คลิกที่ Next ดังรูปที่ 3.8



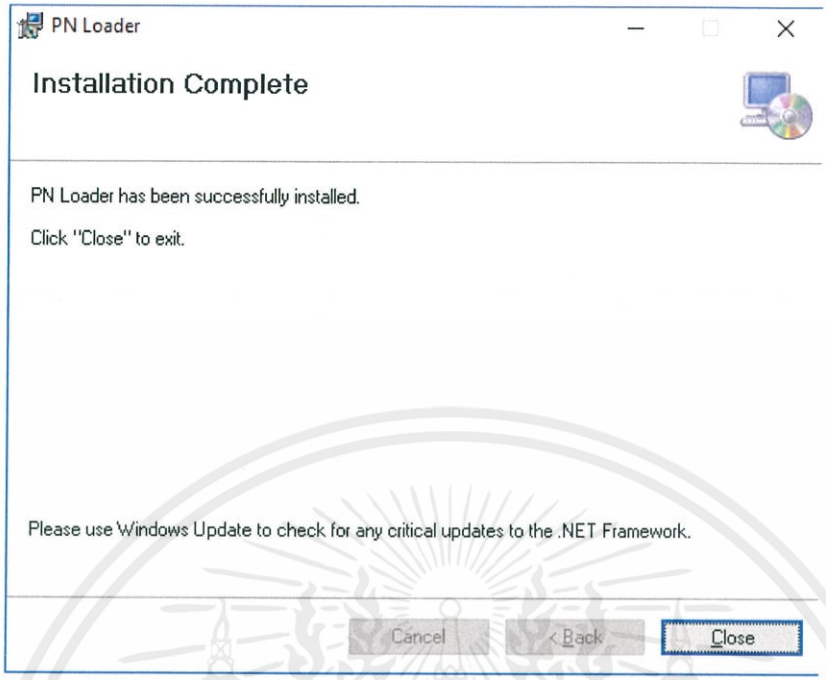
รูปที่ 3.8 หน้าต่าง Select Installation Folder

4) หน้าต่าง Confirm Installation ให้คลิกที่ Next ดังรูปที่ 3.9



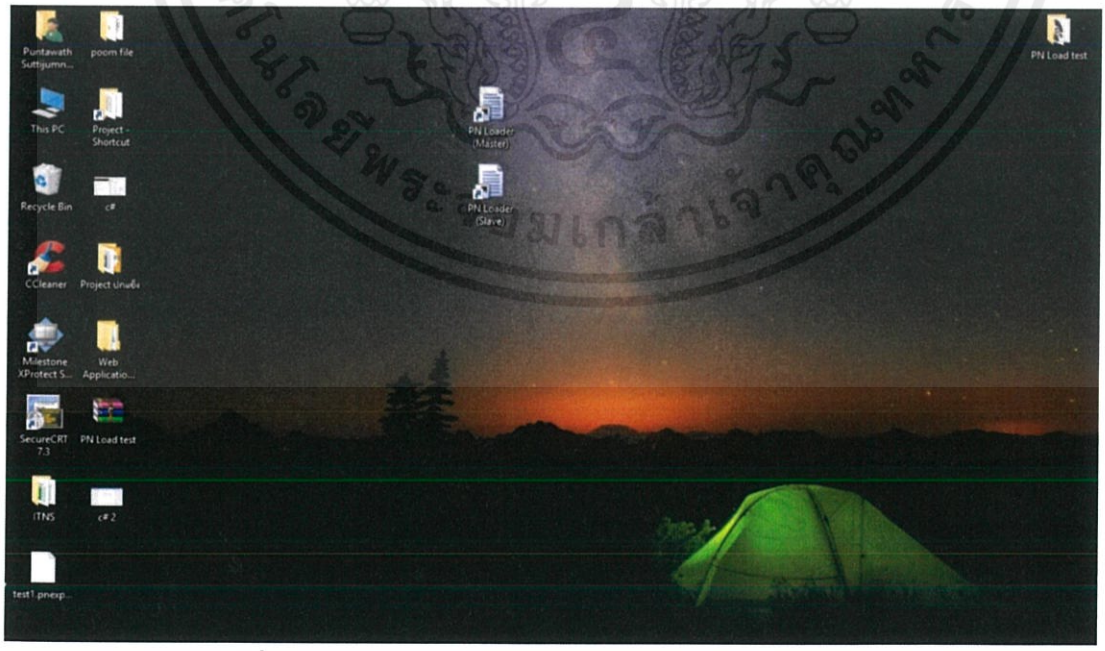
รูปที่ 3.9 หน้าต่าง Confirm Installation

5) เมื่อทำการติดตั้งเสร็จจะขึ้นหน้าต่าง Installation Complete ให้คลิกปุ่ม Close ดังรูปที่ 3.10



รูปที่ 3.10 หน้าต่าง Installation Complete

6) Shortcut ของโปรแกรม PN Loader จะแสดงบนหน้าจอดัง หากต้องการใช้งานบนเครื่อง Slave ให้เลือกที่ไฟล์ PN Loader (Master) แต่หากเป็นเครื่อง Slave ให้เลือกที่ไฟล์ PN Loader (Slave) ดังรูปที่ 3.11



รูปที่ 3.11 Shortcut ของโปรแกรม PN Loader บน Desktop

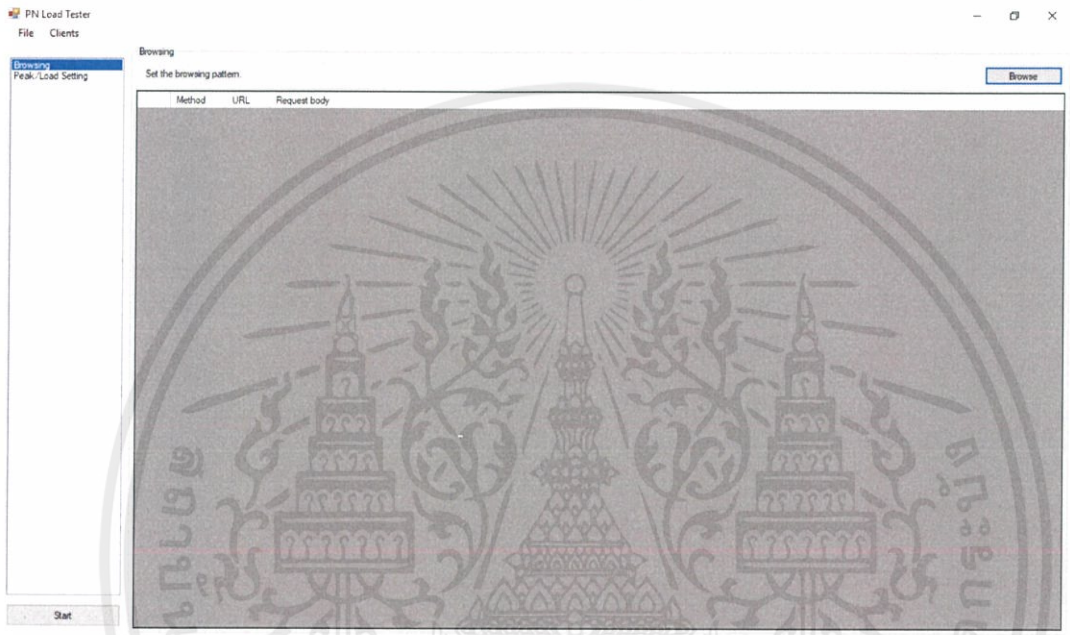
3.7 User Interface ของโปรแกรม

การแสดงผลทาง User Interface ของโปรแกรมแบ่งออกเป็น 2 ส่วนคือ ส่วน Master และส่วน Slave

3.7.1 เมื่อเครื่องถูกกำหนดให้ทำหน้าที่เป็น Master

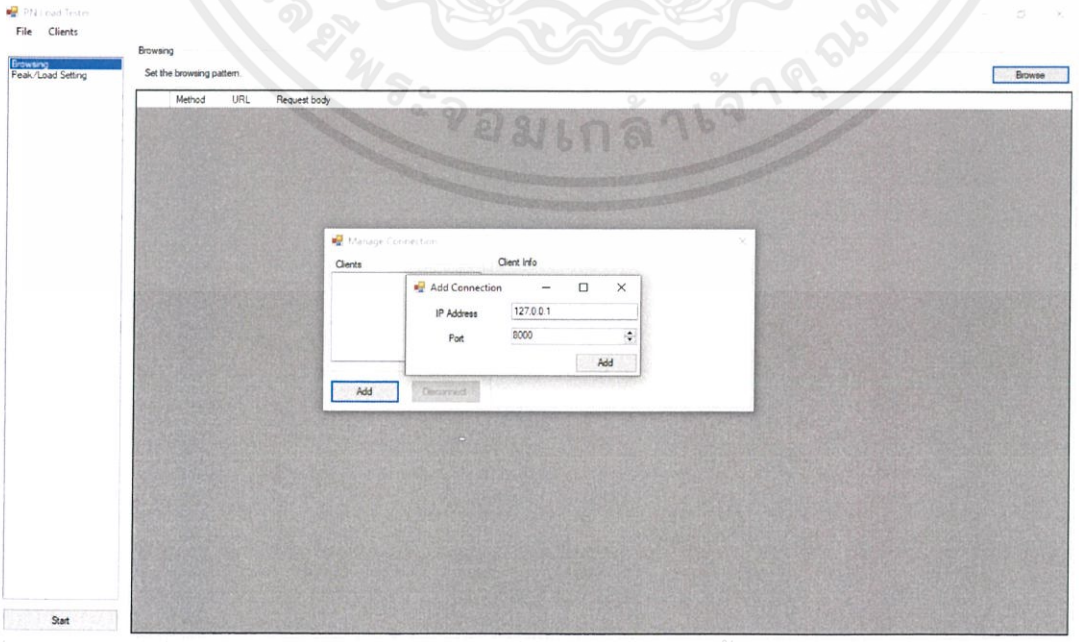
เครื่องที่เป็น Master มี User Interface ของแต่ละขั้นตอนการใช้งานดังนี้

1) เมื่อเข้าโปรแกรม หน้าต่างของโปรแกรมจะเป็นดังรูปที่ 3.12



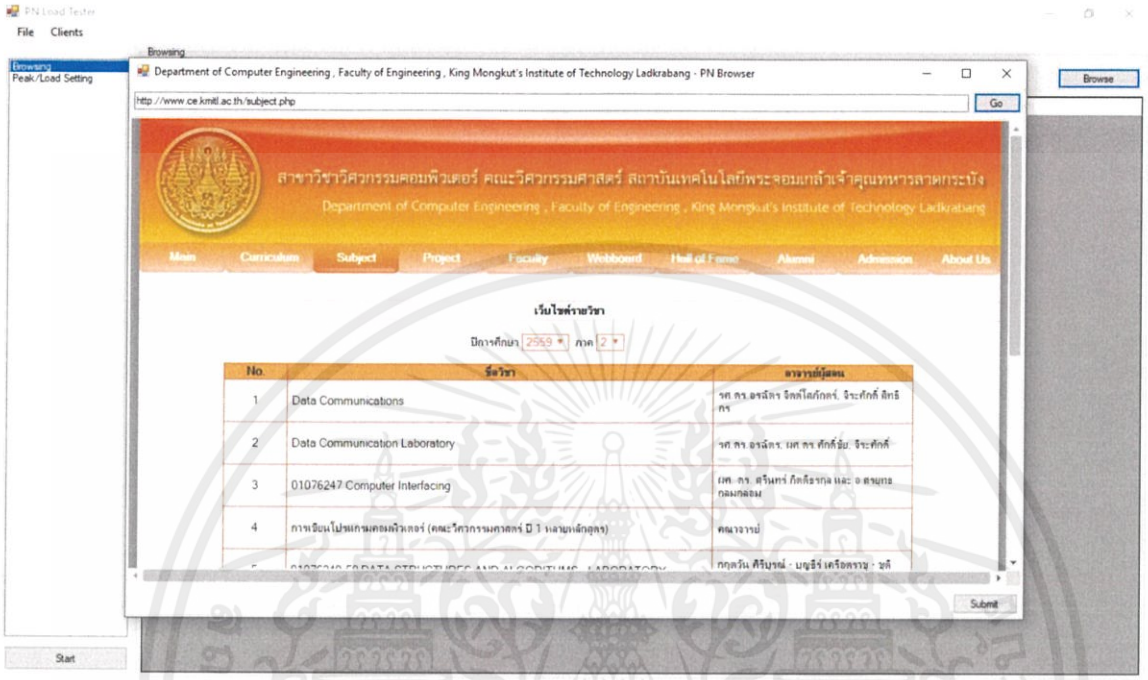
รูปที่ 3.12 หน้าต่างโปรแกรมเมื่อเปิดขึ้นมาบนเครื่อง Master

2) กำหนด Slave ที่ต้องการ (Set Slave) ดังรูปที่ 3.13

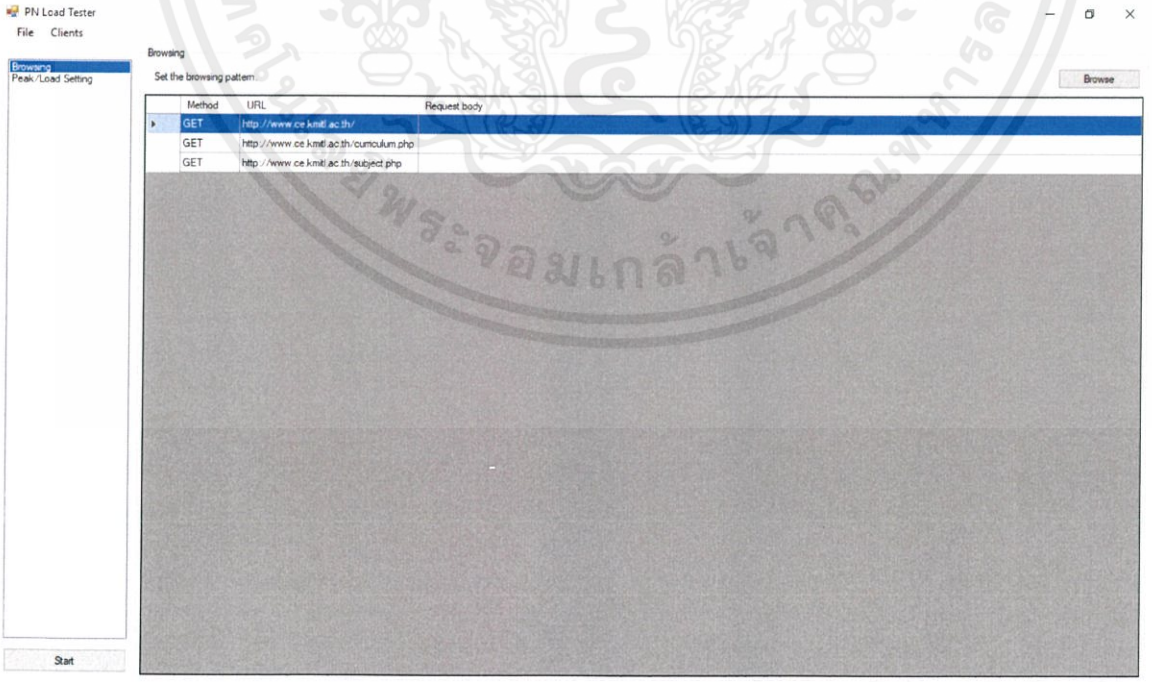


รูปที่ 3.13 การกำหนด Slave ที่ต้องการ

3) Brower สำหรับใส่ URL Website ที่ต้องการจะทดสอบ คลิกลิงค์ต่างๆ เพื่อจำลองการใช้งานของ User แล้วกด Submit ดังรูปที่ 3.14 โปรแกรมจะมีการเก็บค่า URL ของ Website ทุกลิงค์ที่ User คลิก ดังรูปที่ 3.15



รูปที่ 3.14 Browser สำหรับคลิก URL website



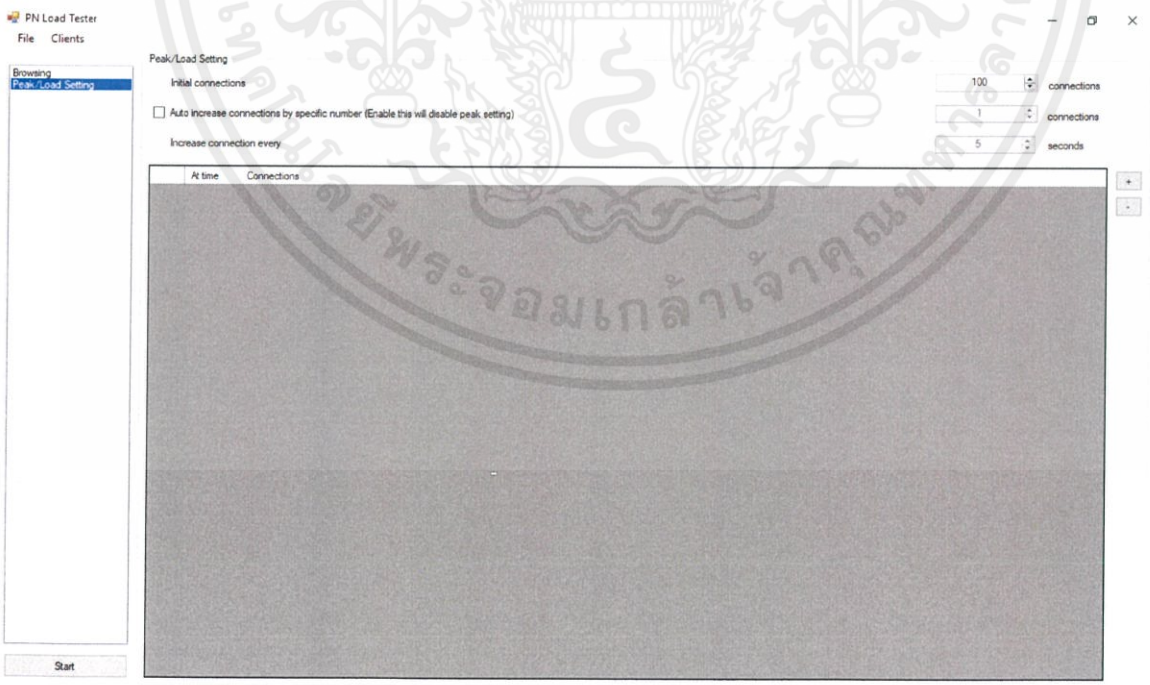
รูปที่ 3.15 Cache URL ที่เลือกไว้

4) กำหนดการทดสอบได้ 3 แบบ คือ

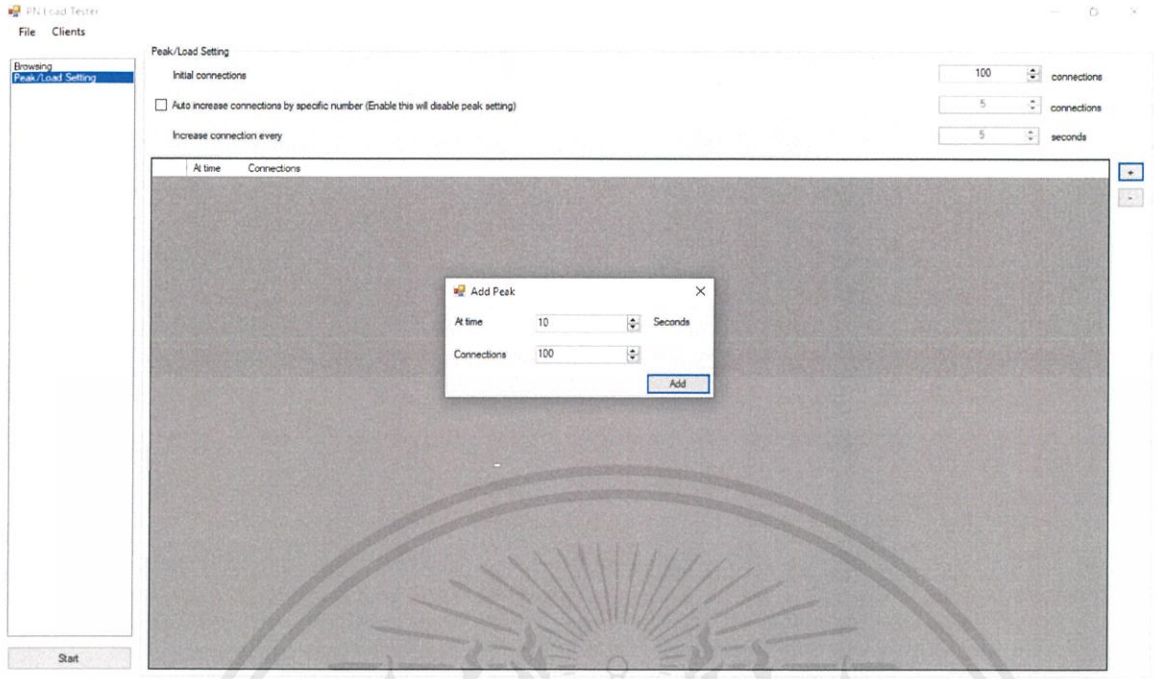
1. Constant Load: การทดสอบแบบนี้จะใช้ค่า Concurrent Connection ที่เป็นค่าคงที่ในการทดสอบ มีจุดประสงค์เพื่อจำลองจำนวน User ใช้งานคงที่ตลอด โดยวิธีการกรอกข้อมูล ให้กำหนดค่า Connection ที่ช่อง Initial Connection ดังรูปที่ 3.16

2. Peak Load: การทดสอบแบบนี้จะทำการเปลี่ยนค่า Concurrent Connection ให้ต่างกันมากๆ ในระยะเวลาหนึ่ง มีจุดประสงค์เพื่อจำลองจำนวน User คงที่ไป ณ จุดหนึ่ง แล้วเกิด Peak load จากการที่มีจำนวน User เข้าใช้เพิ่มขึ้นมาก ในระยะเวลาอันสั้น โดยวิธีการกรอกข้อมูล ให้กำหนดค่า Connection เริ่มต้นที่ช่อง Initial Connection จากนั้นคลิกเครื่องหมายบวกทางขวามือ ทำการกำหนดเวลาและจำนวน Connection ณ จุดที่ต้องการให้เกิด Peak Load ดังรูปที่ 3.17

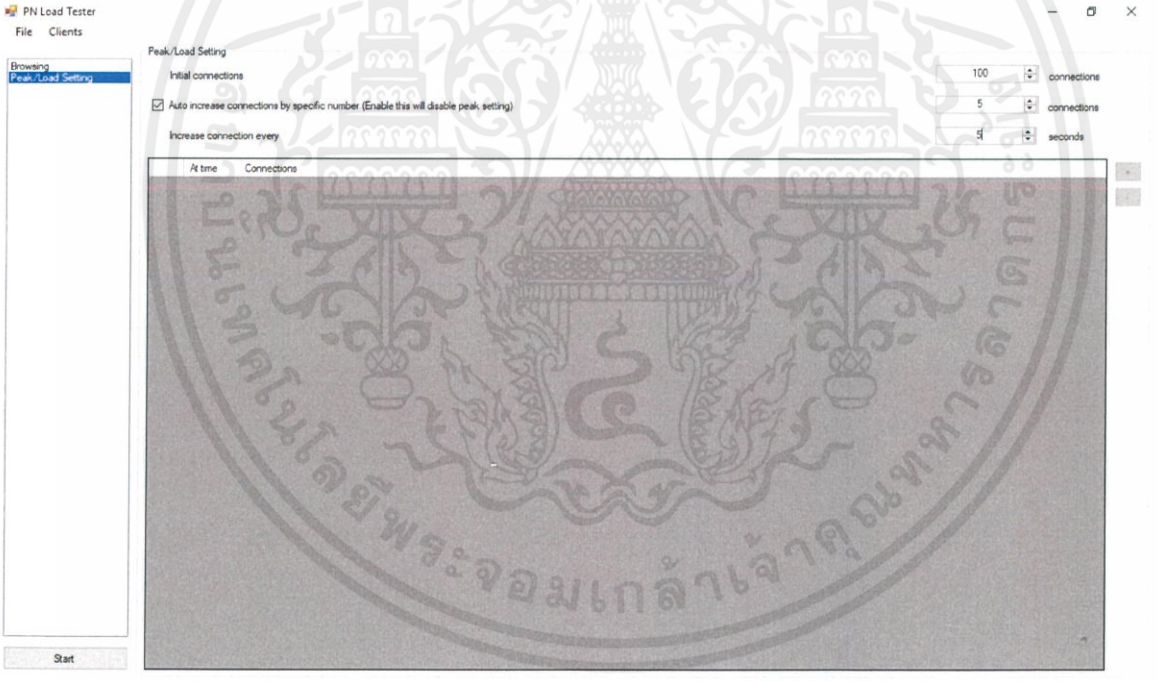
3. Increase Load: การทดสอบแบบนี้จะทำการเพิ่มค่า Concurrent Connection แบบคงที่ มีจุดประสงค์เพื่อจำลองจำนวน User ใช้งานเพิ่มขึ้นเรื่อยๆ แบบคงที่ โดยวิธีการกรอกข้อมูลให้กำหนดค่า Connection เริ่มต้นที่ช่อง Initial Connection และคลิกเครื่องหมายกาถูกหน้า Auto increase connection by specific number ทำการใส่ค่า connection ที่ต้องการให้เพิ่มขึ้นเรื่อยๆ และกำหนดเวลาทุกๆกี่วินาทีที่ต้องการให้จำนวน Connection เพิ่มขึ้น ดังรูปที่ 3.18



รูปที่ 3.16 การตั้งค่าแบบ Constant Load

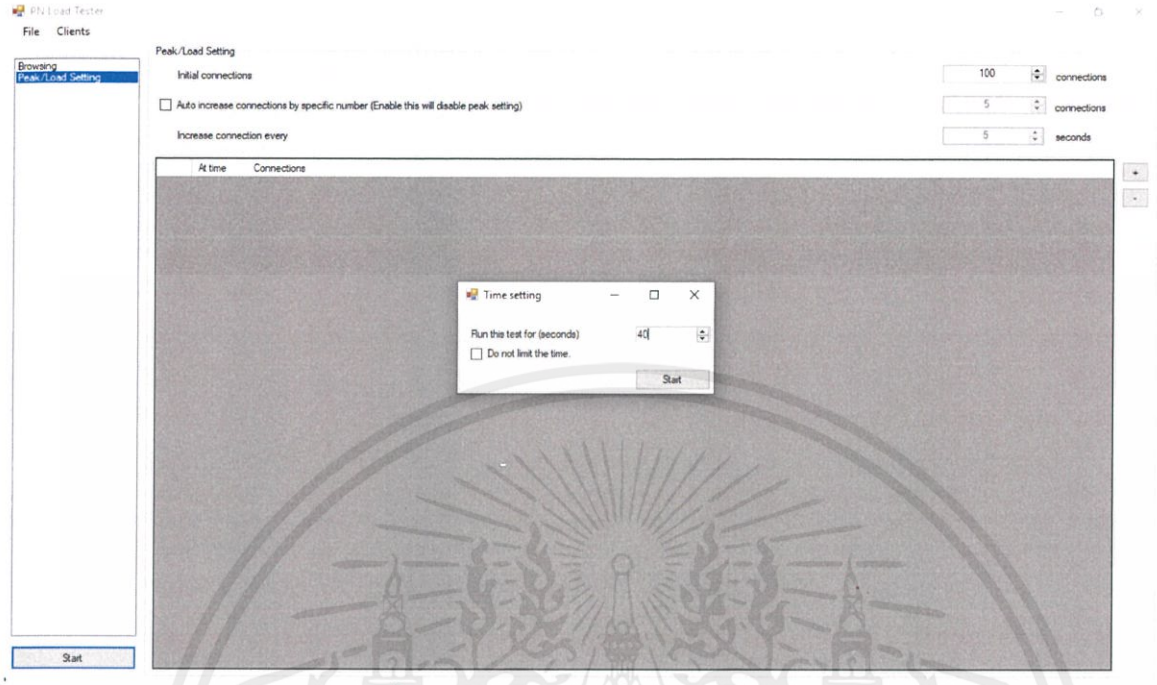


รูปที่ 3.17 การตั้งค่าแบบ Peak Load



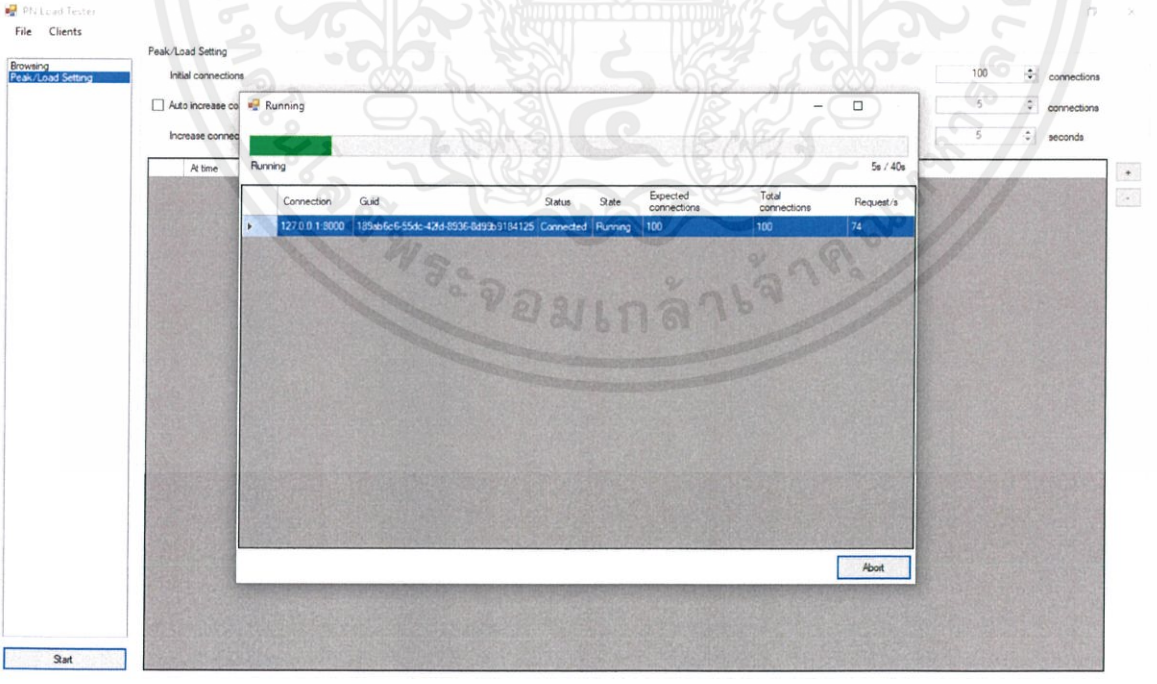
รูปที่ 3.18 การตั้งค่าแบบ Increase Load

5) กำหนดเวลาที่ใช้ในการทดสอบ ดังรูปที่ 3.19



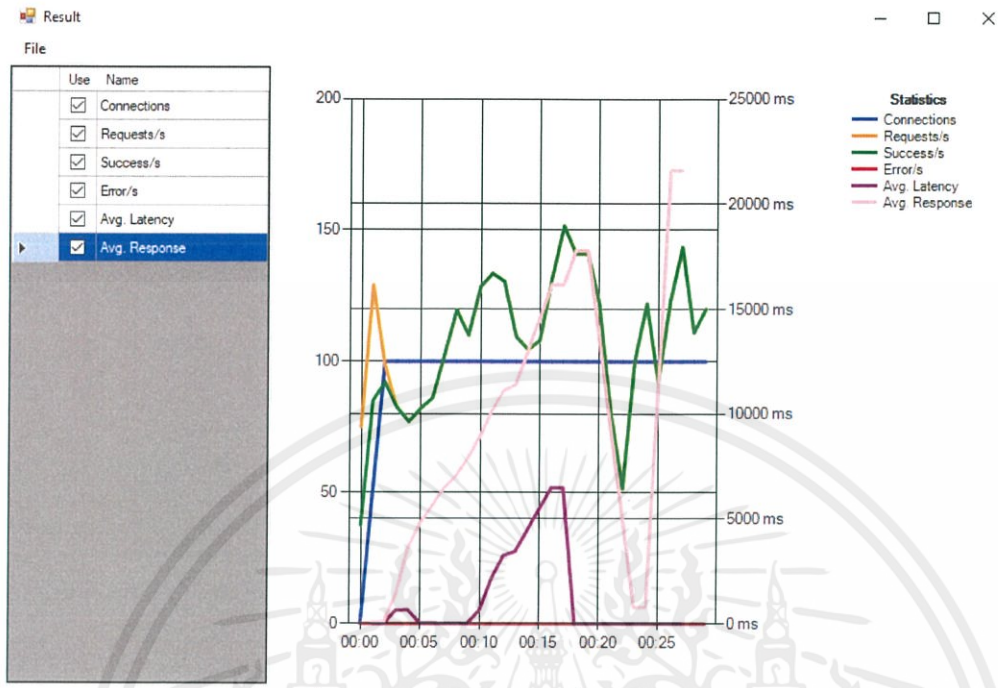
รูปที่ 3.19 กำหนดเวลาที่ใช้ในการทดสอบ

6) หน้าต่างแสดงผลระหว่างการทดสอบ ดังรูปที่ 3.20



รูปที่ 3.20 หน้าต่างแสดงผลระหว่างการทดสอบ

7) กราฟแสดงผลเมื่อทดสอบเสร็จ เป็นดังรูปที่ 3.15



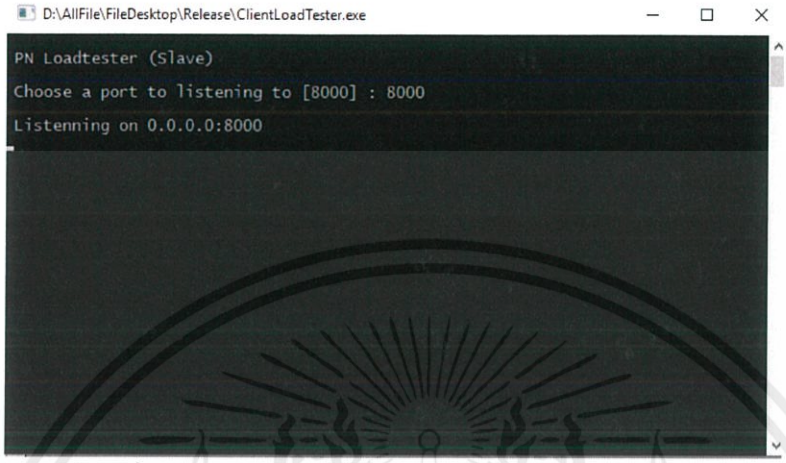
รูปที่ 3.21 กราฟแสดงผลเมื่อทดสอบเสร็จ

- กราฟ Connection ใช้เส้นสีน้ำเงิน อ่านค่าจากแกนตั้งซ้าย
- กราฟ Request/s ใช้เส้นสีเหลือง อ่านค่าจากแกนตั้งซ้าย
- กราฟ Success/s ใช้เส้นสีเขียว อ่านค่าจากแกนตั้งซ้าย
- กราฟ Error/s ใช้เส้นสีแดง อ่านค่าจากแกนตั้งซ้าย
- กราฟ Avg. Latency ใช้เส้นสีม่วง อ่านค่าจากแกนตั้งขวา
- กราฟ Avg. Response ใช้เส้นสีชมพู อ่านค่าจากแกนตั้งขวา

3.7.2 เมื่อเครื่องถูกกำหนดให้ทำหน้าที่เป็น Slave

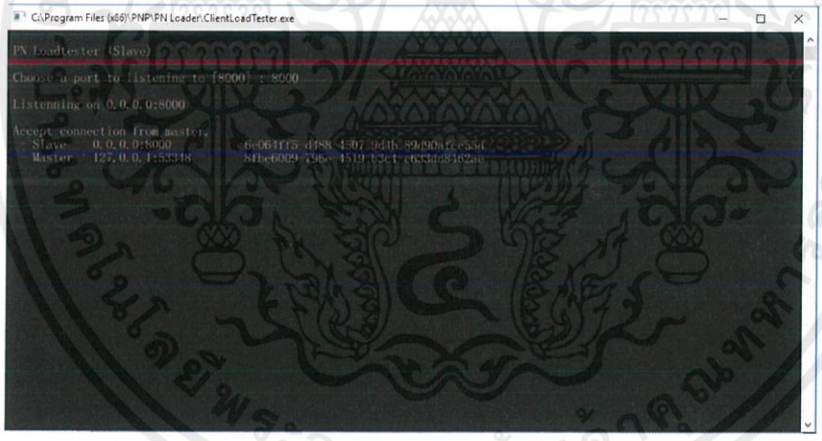
เครื่องที่เป็น Slave มี User Interface ของแต่ละขั้นตอนการใช้งานดังนี้

1) กำหนด port ที่ใช้ในการเชื่อมต่อกับ Master ดังรูปที่ 3.22



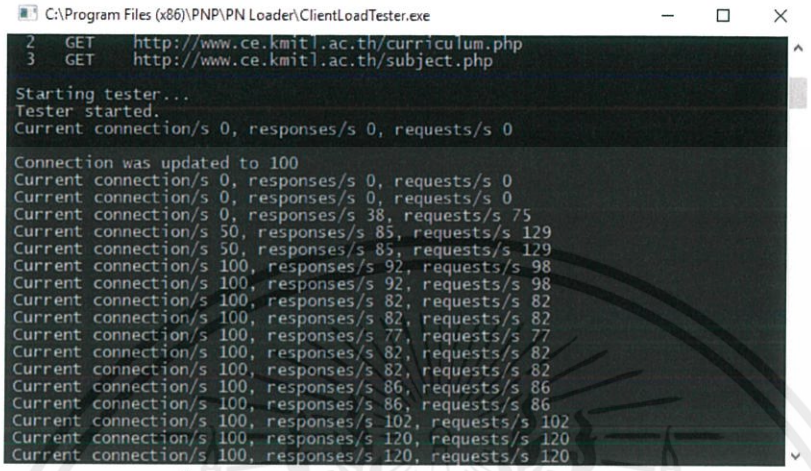
รูปที่ 3.22 กำหนด Port ที่ใช้ในการเชื่อมต่อกับ Master

2) หน้าต่างโปรแกรมเมื่อเครื่อง Master ทำการเชื่อมต่อมายังเครื่อง Slave ดังรูปที่ 3.23



รูปที่ 3.23 หน้าต่าง Slave เมื่อมีการเชื่อมต่อกับเครื่อง Master

3) หน้าต่างแสดงผลระหว่างการทดสอบบอกจำนวน Connection, Response/s และ Request/s แบบ Real time บนหน้าต่างดังรูปที่ 3.24 ซึ่งเราสามารถนำค่าที่แสดงไปเปรียบเทียบกับรูปผลลัพธ์กราฟ Master จากรูปที่ 3.21 ได้ว่ามีความถูกต้องหรือไม่

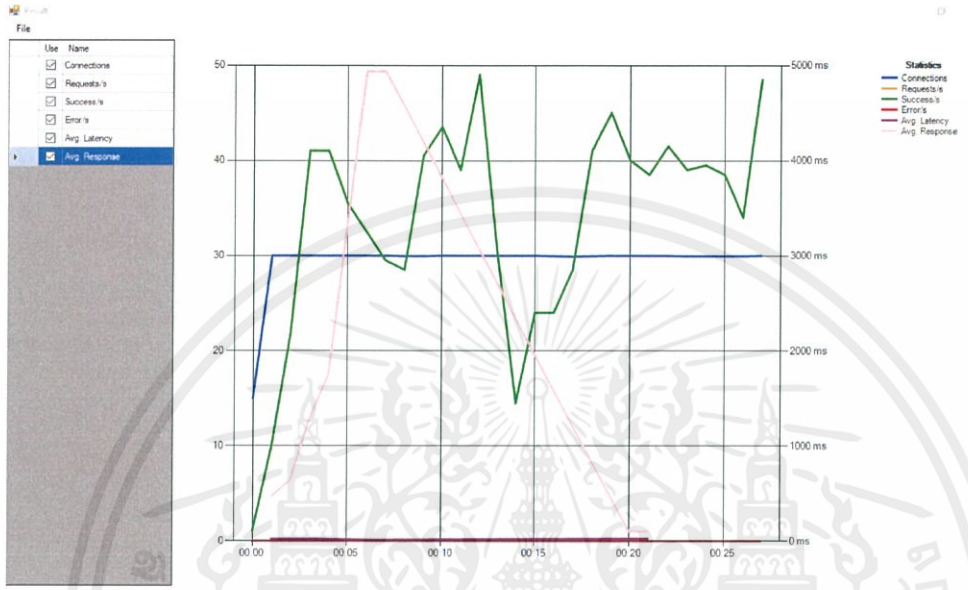


รูปที่ 3.24 หน้าต่างแสดงผลระหว่างการทดสอบ

3.8 รูปแบบของกราฟในกรณีต่าง ๆ

เมื่อทำการทดสอบเสร็จแล้ว เราจะได้ผลลัพธ์ในกราฟ ซึ่งกราฟที่คาดหวังสามารถแบ่งเป็นกรณีต่างๆ ได้ 2 แบบดังนี้

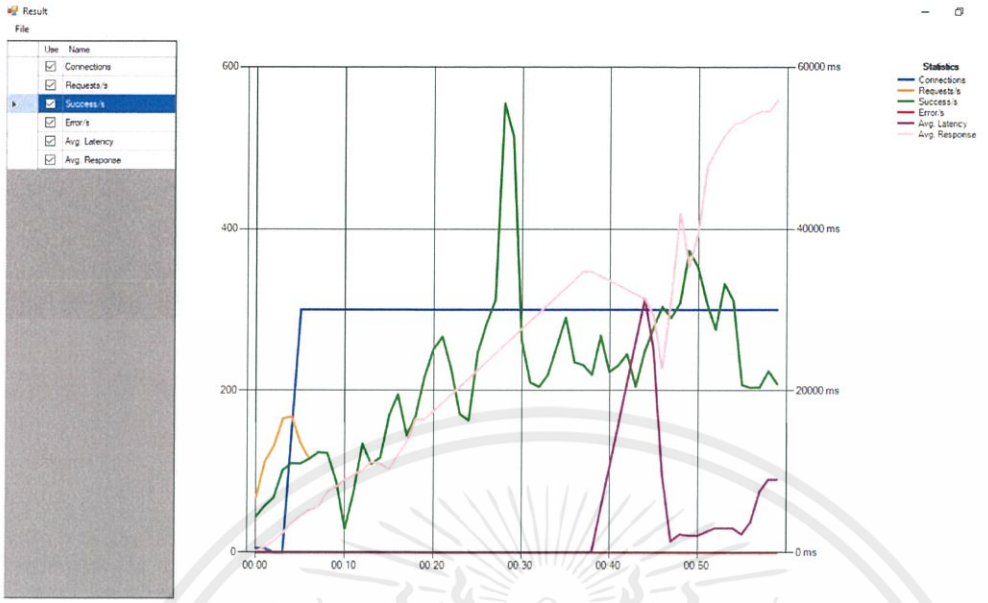
3.8.1 กรณีเครื่องทำงานปกติ



รูปที่ 3.25 กราฟแสดงผลเครื่องทำงานปกติ

เมื่อเครื่องคอมพิวเตอร์แม่ข่ายได้รับการร้องขอในจำนวนที่เหมาะสม ซึ่งเครื่องคอมพิวเตอร์แม่ข่ายสามารถให้บริการได้อย่างมีประสิทธิภาพ จำนวนการตอบสนองที่สำเร็จจะเท่ากับจำนวนการร้องขอ และไม่มีค่า Error และค่า Latency จะมีค่าน้อยในกรณีที่ระยะทางในการเชื่อมต่อสั้นหรือระบบเครือข่ายไม่มีปัญหาคอขวด จากรูปที่ 3.25 เป็นการทดสอบโดยกำหนดปริมาณการร้องขอในช่วงที่เครื่องคอมพิวเตอร์แม่ข่ายสามารถให้บริการได้ ผลลัพธ์ที่ได้จากการทดสอบจึงมีลักษณะเป็นค่าของ Request/s และค่าของ Success/s ทับเส้นกันตลอด แสดงว่าเมื่อทำการร้องขอไปที่เครื่องแม่ข่ายแล้วสามารถตอบสนองต่อการร้องขอได้ทุกครั้งตลอดการทดสอบ ส่วนค่าของ Error เป็น 0 ตลอดการทดสอบ แสดงว่าไม่มี Error เกิดขึ้นและ Latency มีค่าน้อยตลอด แสดงให้เห็นว่าการเชื่อมต่อค่อนข้างเสถียร ไม่มีช่วงที่ทำให้เกิดความล่าช้าในการส่งข้อมูลเกิดขึ้น ซึ่งสอดคล้องกับการทำงานของระบบคอมพิวเตอร์แม่ข่ายที่อธิบายแล้วข้างต้น

3.8.2 กรณีเครื่องทำงานเลยขีดจำกัด



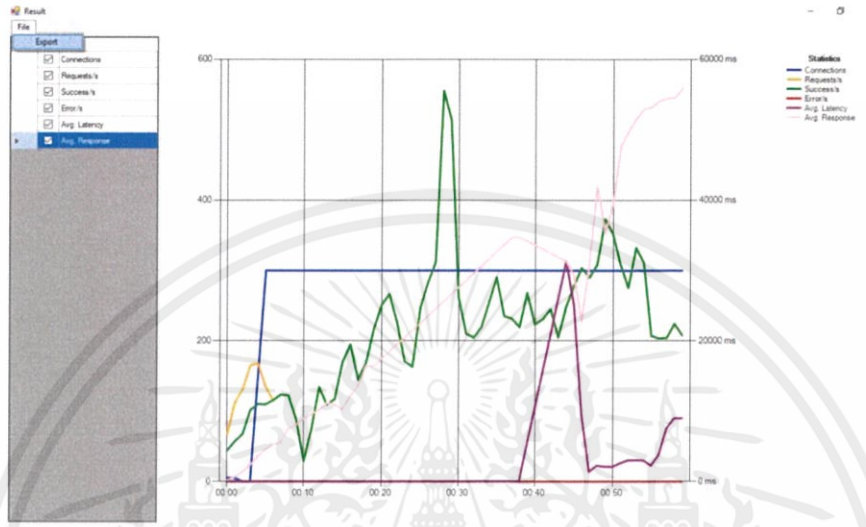
รูปที่ 3.26 กราฟแสดงผลเครื่องทำงานเลยขีดจำกัด

เมื่อเครื่องคอมพิวเตอร์แม่ข่ายได้รับการร้องขอในจำนวนที่เยอะมาก ทำให้เครื่องคอมพิวเตอร์แม่ข่ายไม่สามารถให้บริการได้อย่างมีประสิทธิภาพ จำนวนการตอบสนองที่สำเร็จอาจจะเท่ากับจำนวนการร้องขอและค่า Latency จะมีค่ามากขึ้นผิดปกติในช่วงเวลาหนึ่งจนถึงจุด Breakpoint แสดงให้เห็นถึงความล่าช้าในการส่งข้อมูลนั้นสูงขึ้นผิดปกติ เนื่องจากเครื่องคอมพิวเตอร์แม่ข่ายยังไม่มีการตอบกลับมายังเครื่องคอมพิวเตอร์ลูกข่าย หลังจากนั้นค่า Latency จะมีค่าลดลงอย่างเฉียบพลัน เนื่องจากเครื่องคอมพิวเตอร์แม่ข่ายมีการตอบกลับไปยังเครื่องคอมพิวเตอร์ลูกข่ายในรูปแบบของ Error ทำให้ค่าของความล่าช้าในการส่งข้อมูลนั้นลดลง จากรูปที่ 3.26 เป็นการทดสอบโดยกำหนดปริมาณการร้องขอในช่วงที่เครื่องคอมพิวเตอร์แม่ข่ายไม่สามารถให้บริการได้ ผลลัพธ์ที่ได้จากการทดสอบจึงมีลักษณะเป็นค่าของ Latency วินาทีที่ 38-44 เพิ่มมากขึ้นจาก 0 ไปถึงประมาณ 30000 ms แสดงให้เห็นถึงค่าของความล่าช้าในการส่งข้อมูลนั้นสูงขึ้นอย่างผิดปกติ เมื่อถึงวินาทีที่ 44 ซึ่งคาดว่าเป็นจุด Breakpoint ค่าของ Latency ลดลงอย่างเฉียบพลันจนเข้าใกล้ 0 แสดงให้เห็นว่าเครื่องคอมพิวเตอร์แม่ข่ายมีการตอบกลับ แต่ตอบกลับในรูปแบบของ Error ซึ่งการทดสอบนี้เครื่องคอมพิวเตอร์แม่ข่ายตอบกลับในรูปของ SQL Error ซึ่งสอดคล้องกับการทำงานของระบบคอมพิวเตอร์แม่ข่ายที่อธิบายแล้วข้างต้น

3.9 การบันทึกไฟล์กราฟผลการทดสอบ

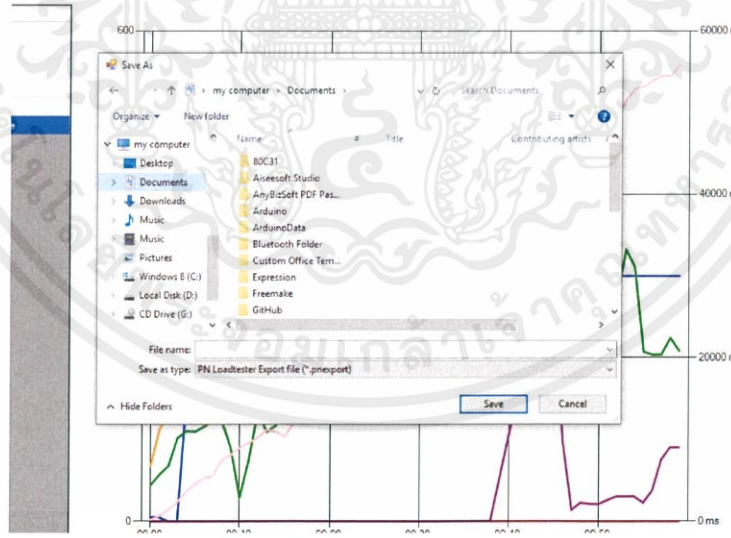
เมื่อทำการทดสอบเสร็จแล้วและต้องการเก็บผลลัพธ์ไว้ดูในภายหลัง ตัวโปรแกรมสามารถ Save ผลทดสอบเก็บไว้ในเครื่องได้ซึ่งมีขั้นตอนดังนี้

1) คลิก File >> Export ดังรูปที่ 3.27



รูปที่ 3.27 Save ผลการทดสอบ Graph ลงเครื่อง

2) ตั้งชื่อ File ซึ่งนามสกุลไฟล์ของกราฟเป็น .pnexport ดังรูปที่ 3.28



รูปที่ 3.28 ตั้งชื่อไฟล์ผลการทดสอบ

3.10 การวิเคราะห์ผลลัพธ์

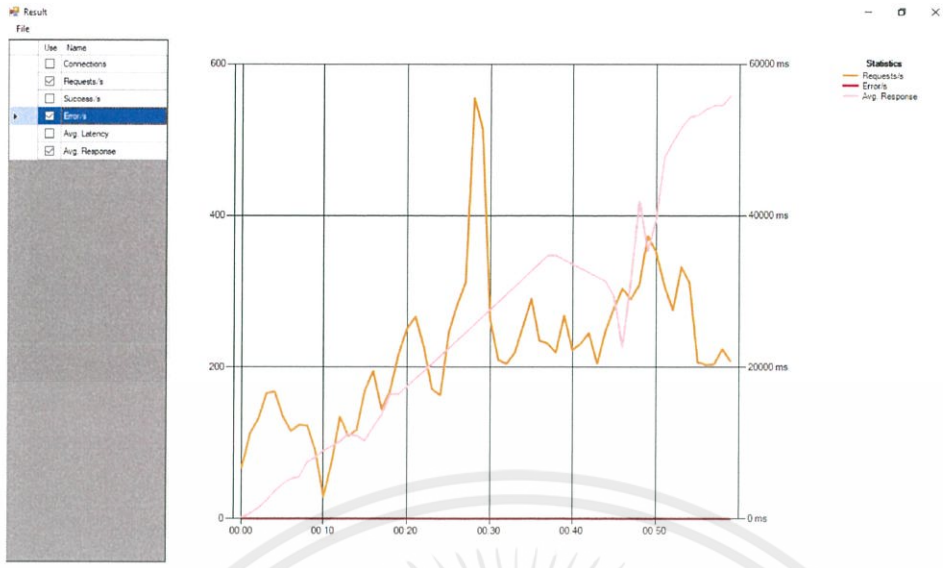
เราสามารถนำผลลัพธ์มาใช้ในการวิเคราะห์ เพื่อดูประสิทธิภาพของเป้าหมายระหว่างการทดสอบ โดยเราจะพิจารณาจาก 3 ข้อมูลหลักๆ ได้แก่

3.10.1 กราฟ Request/s และ Response Time และ Error/s (จุด Breakpoint)

เมื่อเริ่มการทำงานจะมีการส่งการร้องขอไปยังเครื่องคอมพิวเตอร์แม่ข่ายเป้าหมายที่ต้องการจะทดสอบ ค่าของ Request/s เป็นตัวบอกปริมาณงานที่ทำได้ในช่วงเวลาหนึ่ง ในการส่งข้อมูลควรจะมีค่ามากกว่า 0 Request/s ไม่ควรมีค่าต่ำจนเข้าใกล้ 0 Request/s ส่วนค่าของ Response Time เป็นค่าบอกเวลาที่เครื่องคอมพิวเตอร์แม่ข่ายเป้าหมายใช้ในการตอบสนองกลับมายังเครื่องคอมพิวเตอร์ลูกข่าย หากส่งได้ปกติค่าของ Response Time ควรอยู่ในช่วง 1-3000 ms และค่าของ Error/s เป็นค่าที่บอกว่าการทำงานนั้นมี Error เกิดขึ้นหรือไม่ เช่น Error 300 หรือ 400 เป็นต้น หากมีค่า Error/s มากกว่า 0 เป็นจำนวนมาก แสดงว่ามี Error เกิดขึ้นจำนวนมาก เมื่อนำทฤษฎีทั้งหมดมาวิเคราะห์จะได้ว่า

- ถ้าค่าของกราฟ Request/s มากกว่า 0 แต่ค่าของ Response Time อยู่ในช่วง 1-3000 ms และค่า Error/s ค่อนข้างเข้าใกล้ 0 หรือเป็น 0 แสดงว่ายังมีการถ่ายโอนข้อมูลได้ปกติ
- ถ้าค่าของกราฟ Request/s มากกว่า 0 แต่ค่าของ Response Time มากกว่า 3000 ms และค่า Errors/s ค่อนข้างเข้าใกล้ 0 หรือเป็น 0 แสดงว่าการถ่ายโอนข้อมูลใกล้ที่จะถึงจุด Breakpoint แล้ว เนื่องจากใช้เวลาในการตอบกลับมานาน แต่ไม่มี Error เกิดขึ้นแสดงว่าเครื่องคอมพิวเตอร์แม่ข่ายยังไม่มีการตอบกลับมา
- ถ้าค่าของกราฟ Request/s มากกว่า 0 แต่ค่าของ Response Time อยู่ในช่วง 1-3000 ms และค่า Errors/s มากกว่า 0 แสดงว่าเลยจุด Breakpoint ในการส่งข้อมูลแล้ว เนื่องจากใช้เวลาในการตอบกลับมาเร็ว แต่มี Error/s มากกว่า 0 เพราะส่งไปแล้วอาจจะขึ้น Error

ซึ่งจากกราฟในรูปที่ 3.29 มีการเลือกค่า Request/s, Error/s และ Avg.Response ทำให้สามารถสังเกตจุด Breakpoint ได้ง่ายขึ้น

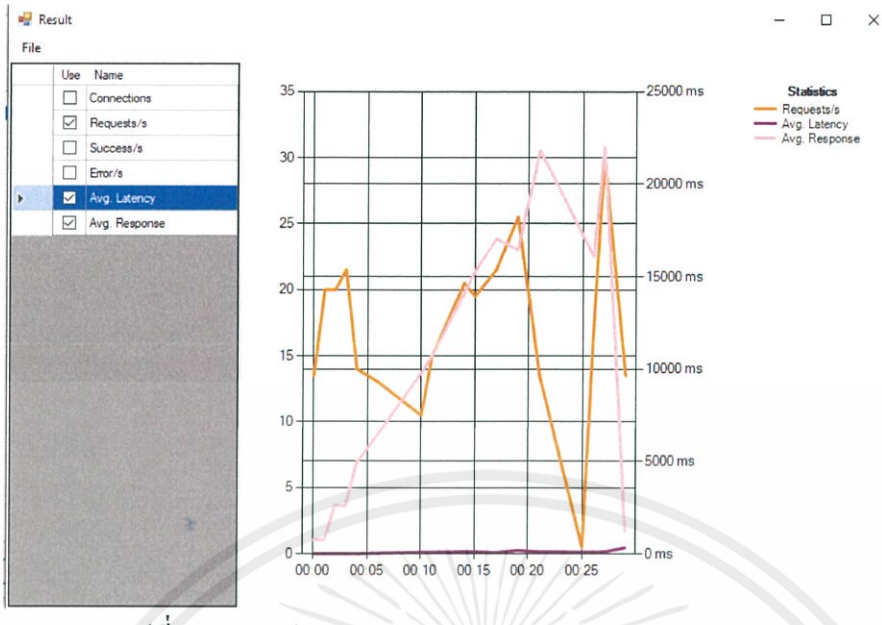


รูปที่ 3.29 กราฟ Request/s & Response Time & Error/s

3.10.2 กราฟ Request/s และ Response Time และ Latency (ปัญหา Network / Client)

เมื่อเริ่มการทำงานจะมีการส่งการร้องขอไปยังเครื่องคอมพิวเตอร์แม่ข่ายเป้าหมายที่ต้องการจะทดสอบ ค่า Latency คือค่าของความล่าช้าในการส่งข้อมูล ส่วนค่า Response Time คือค่าของเวลาที่ใช้ในการตอบสนอง

ซึ่งถ้าหากเครื่องคอมพิวเตอร์แม่ข่ายเป้าหมายมีระบบเครือข่ายที่มี Bandwidth มากและเครื่องคอมพิวเตอร์ลูกข่ายมีประสิทธิภาพสูง กราฟที่ได้จากการทดสอบจะมีค่า Request/s มากกว่า 0 ส่วนค่าของ Response Time จะอยู่ในช่วง 1-3000 ms และค่าของ Latency เข้าใกล้ 0 แสดงว่ายังมีการถ่ายโอนข้อมูลได้ปกติ แต่ถ้าหากมี Load จำนวนมากขึ้น กราฟที่ได้จากการทดสอบจะมีค่า Request/s มากกว่า 0 และค่าของ Response Time จะอยู่ในช่วง 1-3000 ms และค่าของ Latency มากกว่า 500 ms แสดงว่าการถ่ายโอนข้อมูลอาจจะมีปัญหาที่ Network แต่ในทางกลับกันหากค่า Request/s มากกว่า 0 และค่าของ Response Time มากกว่า 3000 ms แต่ค่าของ Latency เข้าใกล้ 0 แสดงว่าการถ่ายโอนข้อมูลอาจจะมีปัญหาที่ Client ซึ่งจากกราฟในรูปที่ 3.30 มีการเลือกค่า Request/s, Latency และ Avg.Response ทำให้สามารถสังเกตปัญหา Network / Client ได้ง่ายขึ้น

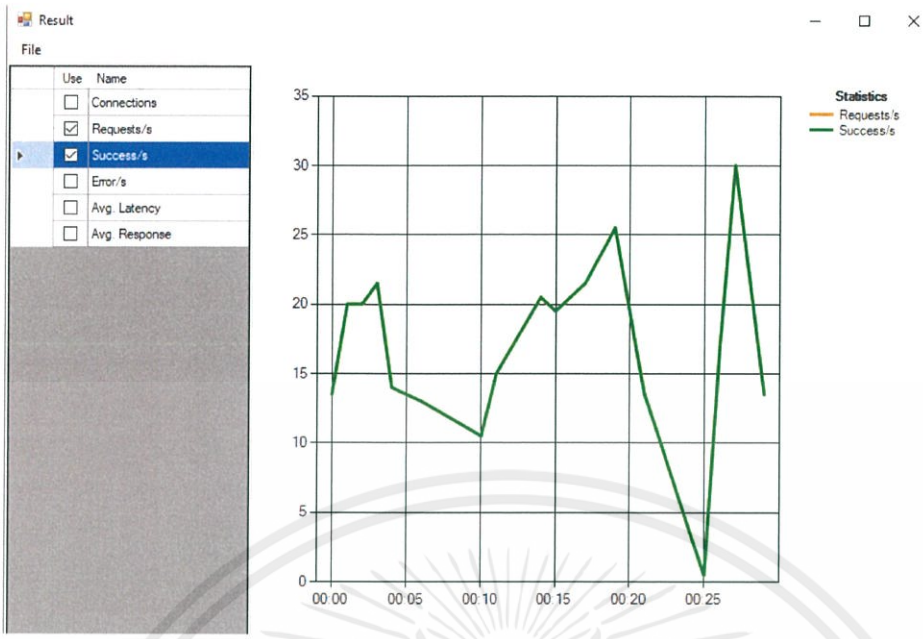


รูปที่ 3.30 กราฟ Request/s & Response Time & Latency

3.10.3 กราฟ Request/s และ Success/s (การส่งข้อมูลในภาพรวม)

เมื่อเริ่มการทำงานจะมีการส่งการร้องขอไปยังเครื่องคอมพิวเตอร์แม่ข่ายเป้าหมายที่ต้องการจะทดสอบ ค่า Request/s คือ ปริมาณการร้องขอที่ทำไปในช่วงเวลาหนึ่ง ส่วน Success/s คือปริมาณการร้องขอที่ส่งสำเสร็จในช่วงเวลาหนึ่ง หากส่งการร้องขอไปตามหลักควรจะส่งสำเร็จเช่นกัน

ถ้าค่าของกราฟ Request/s และ Success/s มากกว่า 0 แสดงว่ายังมีการส่งข้อมูลได้ปกติ เนื่องจากมีการส่งการร้องขอแล้วสำเร็จเป็นส่วนใหญ่หรือสำเร็จทั้งหมด แต่ถ้าค่าของ Request/s มากกว่า 0 ส่วนค่าของ Success/s มีค่าเข้าใกล้ 0 แสดงว่าการส่งข้อมูลอาจมีปัญหา เนื่องจากมีการส่งการร้องขอไปแล้วสำเร็จน้อยมาก ซึ่งจากกราฟในรูปที่ 3.31 มีการเลือกค่า Request/s และ Success/s ทำให้สังเกตได้ง่ายว่าระบบมีการส่งการร้องขอไปแล้วสำเร็จทั้งหมด เนื่องจากกราฟ Request/s ทับกับกราฟ Success/s ทั้งหมด



รูปที่ 3.31 กราฟ Request/s & Success/s

บทที่ 4

การทดลองและผลการทดลอง

ในส่วนการทดลองของการพัฒนาโปรแกรม Load Testing ผู้ทำโครงงานได้ออกแบบรูปแบบของการทดลองและขั้นตอนการทดลอง รวมถึงผลของการทดลองที่ได้ ซึ่งจะถูกอธิบายในบทนี้

4.1 การทดลอง

ในส่วนการทดลองนั้น ผู้ทำโครงงานจะใช้การจำลองสถานการณ์ สมมติการใช้งานของ User ในแบบต่างๆ โดยแบ่งออกเป็น 3 แบบได้แก่

1) **การทดลอง : เมื่อมี User เข้าใช้งานคงที่ (Constant Load)**

การทดลองนี้จะใช้ค่า Concurrent Connection ที่เป็นค่าคงที่ในการทดสอบ มีจุดประสงค์เพื่อวัดประสิทธิภาพของ Server เป้าหมาย ในสถานการณ์สมมติเมื่อมี User เข้าใช้งานคงที่ จากนั้นนำกราฟผลลัพธ์มาวิเคราะห์

2) **การทดลอง : เมื่อมี User เข้าใช้งานเพิ่มขึ้น/ลดลง เฉียบพลัน (Peak Load)**

การทดลองนี้จะใช้การเปลี่ยนค่า Concurrent Connection ให้เพิ่มขึ้น/ลดลง ภายในระยะเวลาสั้นๆ มีจุดประสงค์เพื่อวัดประสิทธิภาพของ Server เป้าหมาย ในสถานการณ์สมมติเมื่อมี User เข้าใช้งานคงที่ระยะเวลาหนึ่ง แล้วเพิ่มขึ้น/ลดลงเฉียบพลัน เนื่องจากมี User เข้าใช้เพิ่มขึ้น/ลดลงอย่างมากในระยะเวลาอันสั้น จากนั้นนำกราฟผลลัพธ์มาวิเคราะห์

3) **การทดลอง : เมื่อมี User เข้าใช้งานเพิ่มขึ้นอย่างต่อเนื่อง (Increase Load)**

การทดลองนี้จะใช้การเพิ่มค่า Concurrent Connection อย่างต่อเนื่อง มีจุดประสงค์เพื่อวัดประสิทธิภาพของ Server เป้าหมาย ในสถานการณ์สมมติเมื่อมี User เข้าใช้งานเพิ่มขึ้นอย่างต่อเนื่อง จากนั้นนำกราฟผลลัพธ์มาวิเคราะห์

4.2 ขั้นตอนการทดลองและผลการทดลอง

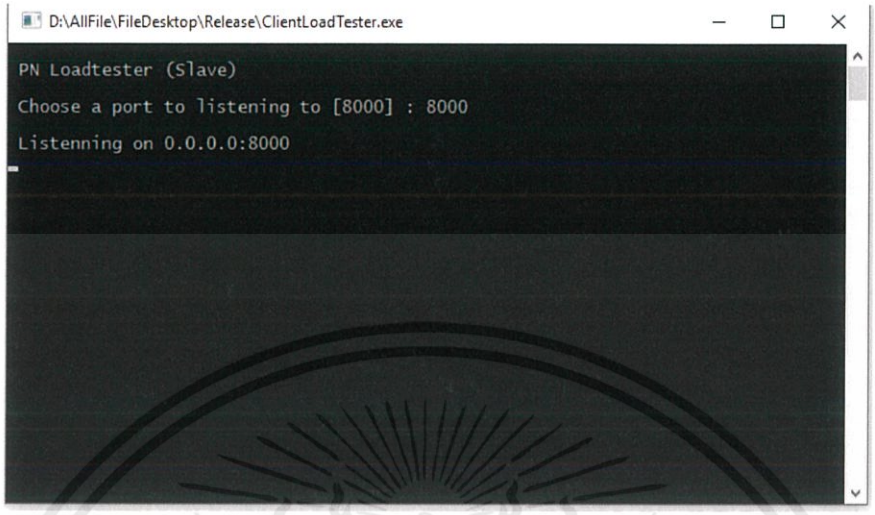
ขั้นตอนการทดลองหลักจะเป็นดังนี้

1) ติดตั้งโปรแกรมในเครื่องที่เป็น Master และเครื่อง Slave ซึ่งตัวติดตั้งจะเป็นตัวเดียวกัน เมื่อติดตั้งโปรแกรมเสร็จ จะมีไฟล์ทั้ง 2 ไฟล์ที่ Desktop คือ PN Loader (Master) และ PN Loader (Slave)

2) เครื่องที่เป็น Slave ทำการเปิดโปรแกรม PN Loader (Slave) จะแสดงผลในรูปแบบของ Command line ให้กำหนด Port ที่ใช้ในการเชื่อมต่อกับเครื่อง Master เมื่อใส่ Port เสร็จ จะรอการเชื่อมต่อจากเครื่อง Master
3) เครื่องที่เป็น Master ทำการเปิดโปรแกรม PN Loader (Master) ทำการเพิ่มเครื่องที่เป็น Slave โดยการใส่ IP เครื่อง Slave และ Port ที่ใช้ในการเชื่อมต่อ ซึ่งเป็น Port เดียวกันกับขั้นตอนที่ 2
4) เครื่องที่เป็น Master ทำการจำลองการใช้งานของ User โดยเข้า Website ที่ต้องการจะทดสอบแล้วคลิก URL จนเสร็จ เครื่อง Master จะเก็บ Cache URL ทั้งหมดที่เราคลิกไป
5) เครื่อง Master ทำการตั้งค่า Connection โดยสามารถเลือกการทดสอบได้ 3 แบบ คือ Constant Load, Peak load และ Increase load ดังหัวข้อที่ 4.1
6) เครื่อง Master เมื่อตั้งค่า Connection แล้ว ให้ตั้งค่าเวลาที่จะใช้ในการทดสอบ
7) เครื่อง Master เมื่อตั้งค่าเวลาทดสอบแล้ว จะเริ่มการทดสอบ ซึ่งระหว่างทดสอบจะมีหน้าต่างแสดงผลขึ้นมาระหว่างทดสอบบนเครื่อง Slave และเครื่อง Master
8) เมื่อการทดสอบเสร็จเรียบร้อย จะมีกราฟแสดงผลสรุปผลการทดสอบขึ้นบนเครื่อง Master สามารถกดเลือกว่าจะให้แสดงกราฟอะไรบ้าง
9) สามารถ Save ไฟล์กราฟของการทดสอบในครั้งนั้นได้ ซึ่งนามสกุลไฟล์จะเป็น .pnexport และสามารถเรียกดูในภายหลังได้โดยเปิดไฟล์จากโปรแกรม PN Loader (Master)

4.2.1 ขั้นตอนการทดสอบแบบ Constant Load

1) กำหนด Port ให้เครื่อง Slave

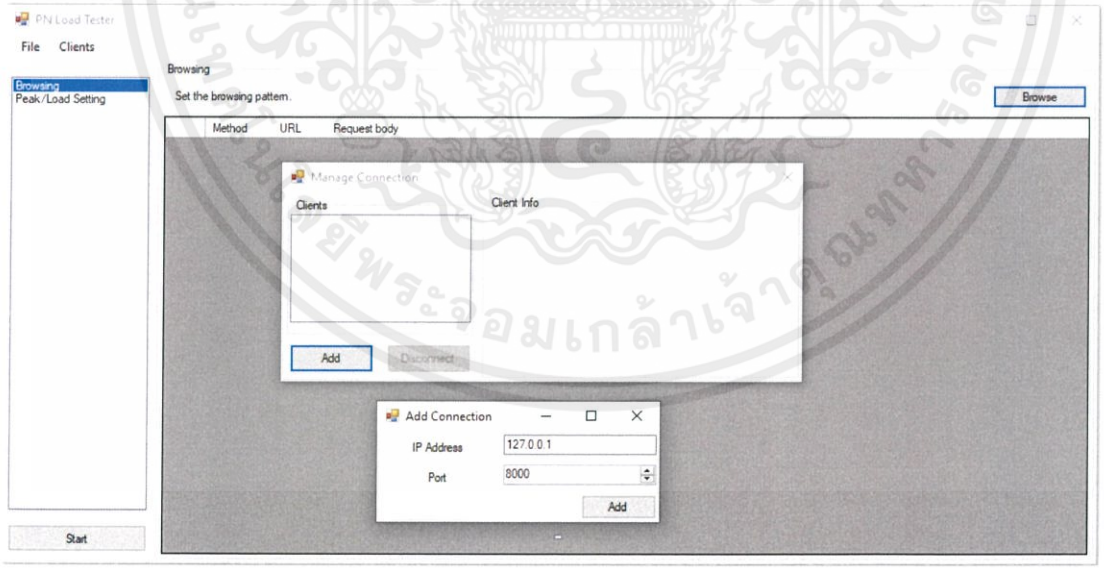


รูปที่ 4.1 กำหนด Port เครื่อง Slave

Slave Setting

- ให้กำหนด Port ที่ใช้เชื่อมต่อกับเครื่อง Master ดังรูปที่ 4.1
- ในการทดสอบครั้งนี้ใช้ Port 8000

2) Add Slave ที่เครื่อง Master



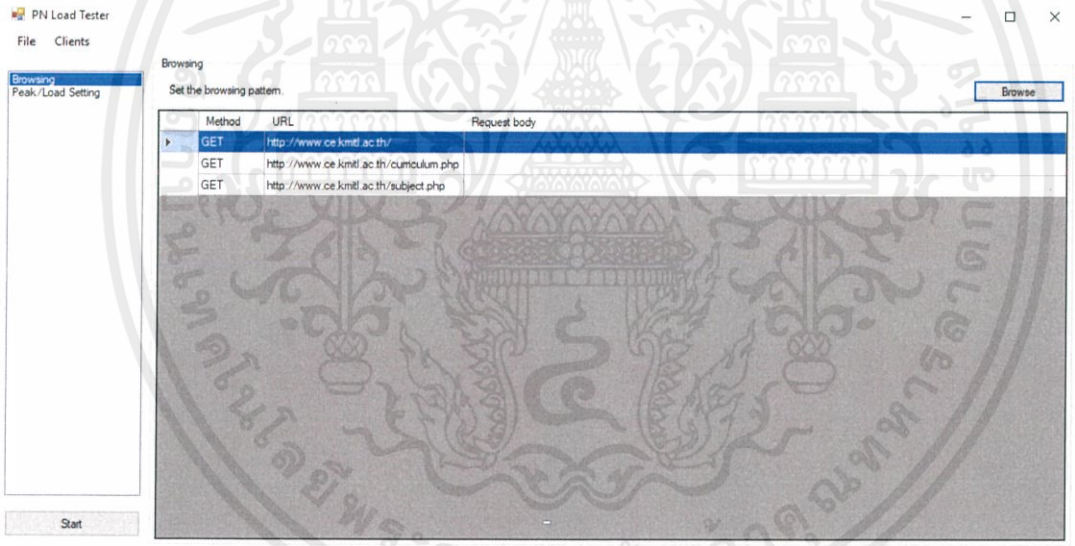
รูปที่ 4.2 Add เครื่อง Slave ที่เครื่อง Master

Master Setting

- คลิกที่ Client เลือก Manage Connection
- ใส่ IP Address และ Port ที่ใช้เชื่อมต่อกับเครื่อง Slave ดังรูปที่ 4.2

3) จำลองการใช้งาน User

รูปที่ 4.3 เข้า Website ที่ต้องการจะทดสอบ

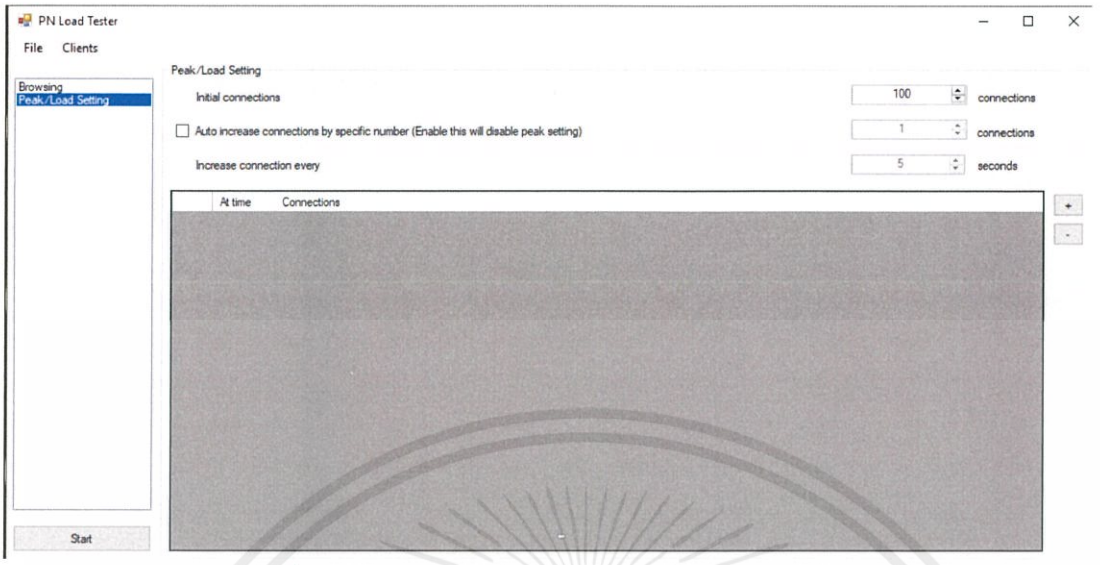


รูปที่ 4.4 โปรแกรมเก็บ Cache URL ทั้งหมด

Master Setting

- เข้าเว็บที่ต้องการจะทดสอบ ดังรูปที่ 4.3
- คลิก URL ต่างๆ เพื่อจำลองการใช้งาน
- ในโปรแกรมจะเก็บ Cache URL ที่ User คลิก ดังรูปที่ 4.4

4) กำหนดค่า Connection

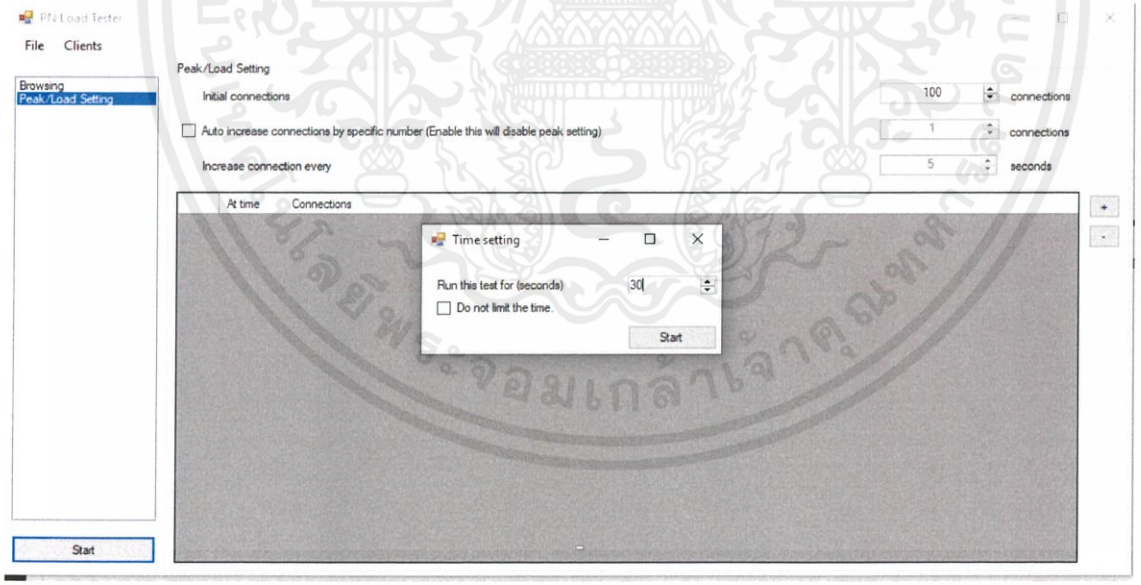


รูปที่ 4.5 กำหนดค่า Connection แบบ Constant Load

Master Setting

- คลิกที่ Peak/Load Setting
- กำหนดค่า Connection ที่ใช้ในการทดสอบ ในที่นี้กำหนดค่า Connection เป็น 100 ดังรูปที่ 4.5

5) กำหนดเวลาที่ใช้ในการทดสอบ

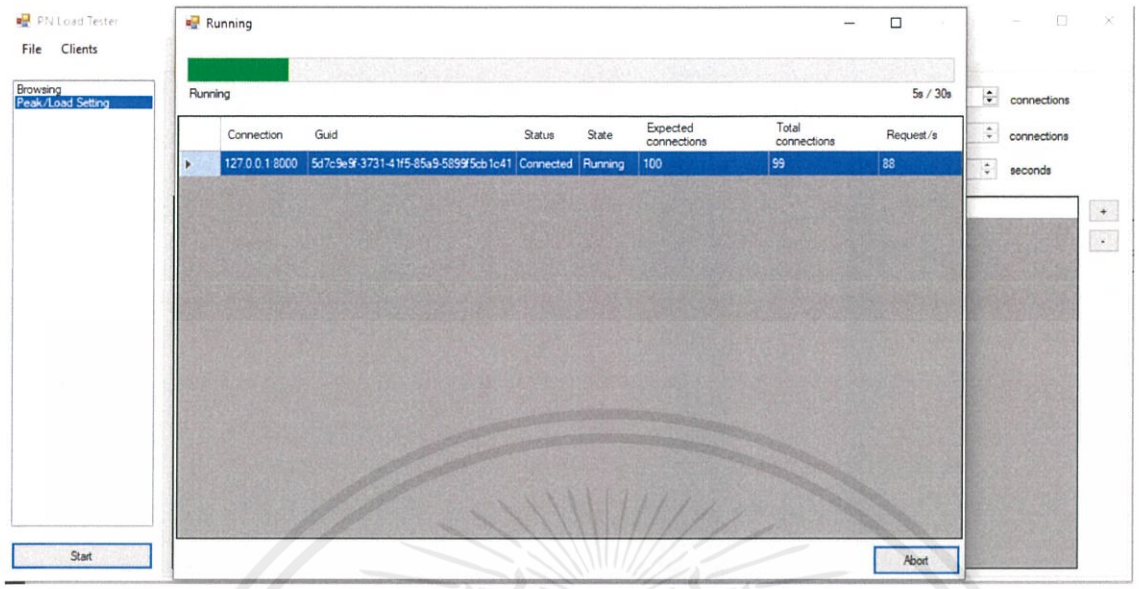


รูปที่ 4.6 กำหนดเวลาที่ใช้ในการทดสอบ

Master Setting

- กำหนดค่าเวลาที่ใช้ในการทดสอบ
- ในที่นี้กำหนดค่าเวลาเป็น 30 วินาที ดังรูปที่ 4.6

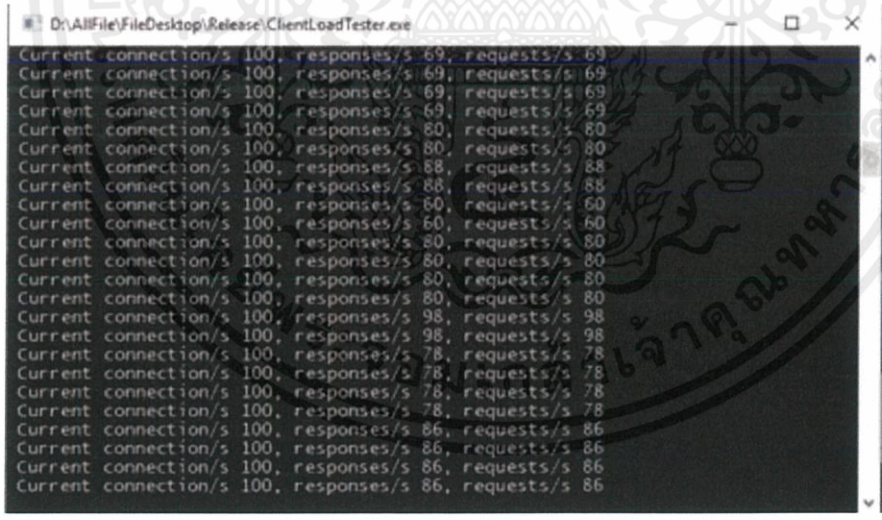
6) กด Start เริ่มการทดสอบ



รูปที่ 4.7 หน้าต่าง Process ระหว่างทดสอบของเครื่อง Master

Master Setting

- กด Start เพื่อเริ่มการทดสอบ
- เครื่อง Master จะแสดงหน้า Process ระหว่างการทดสอบ บอกค่า Request/s และ Connection แบบ Real time ดังรูปที่ 4.7

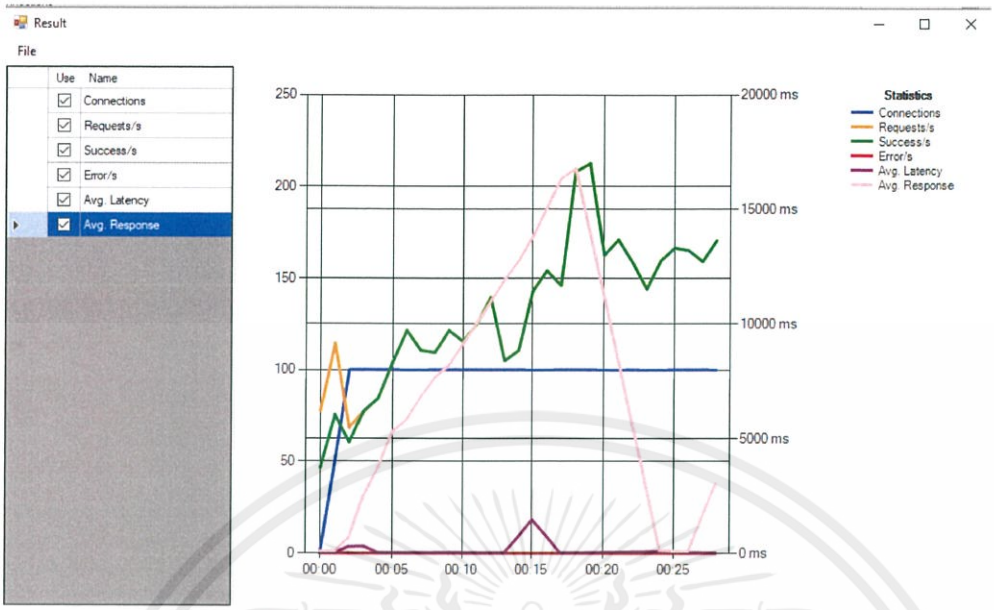


รูปที่ 4.8 หน้าต่าง Process ระหว่างทดสอบเครื่อง Slave

Slave Setting

- เครื่อง Slave จะแสดงหน้า Process ระหว่างการทดสอบ ดังรูปที่ 4.8

7) แสดงผลลัพธ์ของการทดสอบในรูปของกราฟ



รูปที่ 4.9 กราฟผลลัพธ์ของ Constant Load Test

Master Setting

- เมื่อทดสอบเสร็จจะแสดงผลออกมาในรูปของกราฟ ดังรูปที่ 4.9
- สามารถกดเลือกให้แสดงผลกราฟที่ต้องการ

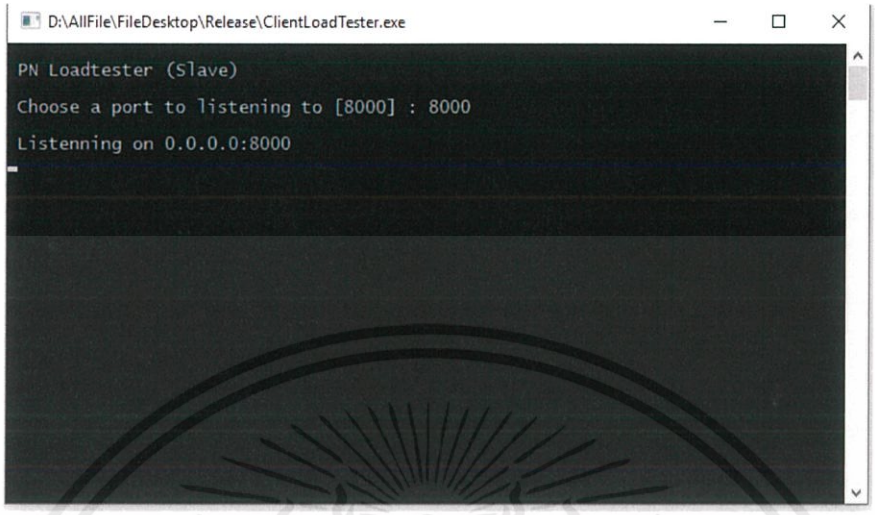
สรุปการทดสอบแบบ Constant Load ในครั้งนี้

จากกราฟรูปที่ 4.9 จะแสดงให้เห็นว่า Server สามารถทำงานได้ปกติ สามารถรองรับ 100 Connection ได้ โดยสามารถสรุปจากกราฟได้ดังนี้

- มีการ Prepare Connection ถึงวินาทีที่ 2 เมื่อถึงวินาทีที่ 3 ค่า Connection เป็น 100 ไปจนสิ้นสุดการทดสอบ เนื่องจากระบบมีการสร้างการเชื่อมต่อเพิ่มขึ้นจนถึงค่าการเชื่อมต่อตามที่กำหนด
- ช่วงวินาทีที่ 0-2 กราฟของค่า Request/s ยังไม่ทับกับเส้นค่าของ Success/s หลังจากนั้นวินาทีที่ 3 กราฟของค่า Request/s จะทับกับเส้นของค่า Success/s ไปตลอด แสดงว่าในช่วงเวลานี้ เมื่อส่ง Request ไปแล้ว Success ตลอดการทดสอบ
- ช่วงวินาทีที่ 2-5 และ 13-17 เกิด Error บางส่วน สาเหตุอาจจะมาจาก Network ที่ ISP ให้บริการอาจจะมีการใช้งานเยอะ ดูได้จาก Latency ที่ค่าสูงขึ้นตามในช่วงที่เกิด Error
- ช่วงวินาทีที่ 2-18 ค่าของ Response มีค่าสูงขึ้นแบบคงที่ แสดงว่ามีความเร็วในการตอบสนองช้าลงจนถึงวินาทีที่ 19 ค่าของ Response time ลดลงแบบคงที่เป็นเส้นตรงเช่นกัน แสดงว่ามีความเร็วในการตอบสนองเร็วขึ้น

4.2.2 ขั้นตอนการทดสอบแบบ Peak Load

1) กำหนด Port ให้เครื่อง Slave

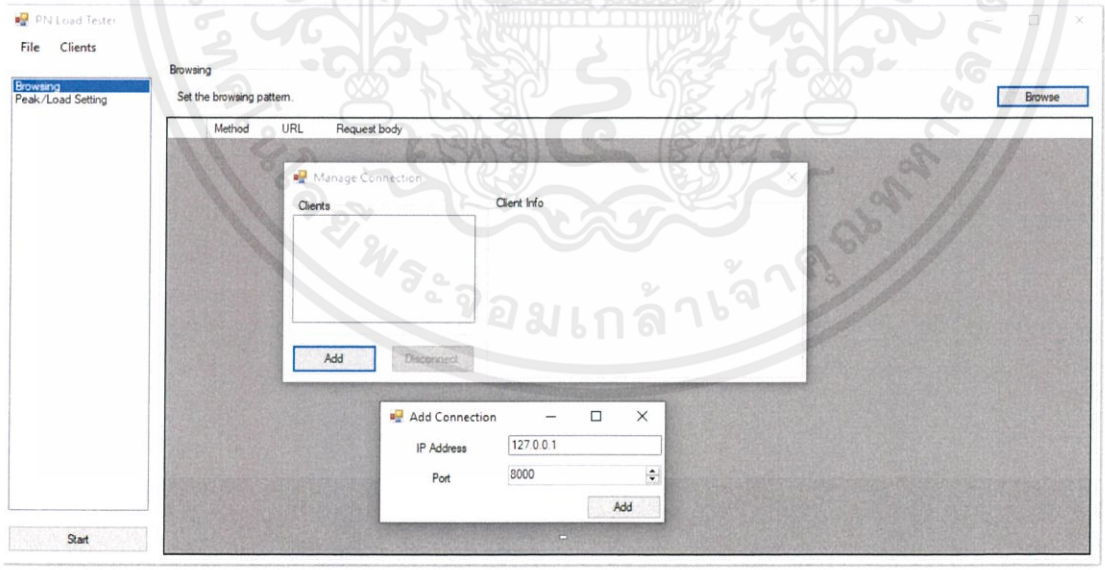


รูปที่ 4.10 กำหนด Port เครื่อง Slave

Slave Setting

- ให้กำหนด Port ที่ใช้เชื่อมต่อกับเครื่อง Master ดังรูปที่ 4.10
- ในการทดสอบครั้งนี้ใช้ Port 8000

2) Add Slave ที่เครื่อง Master



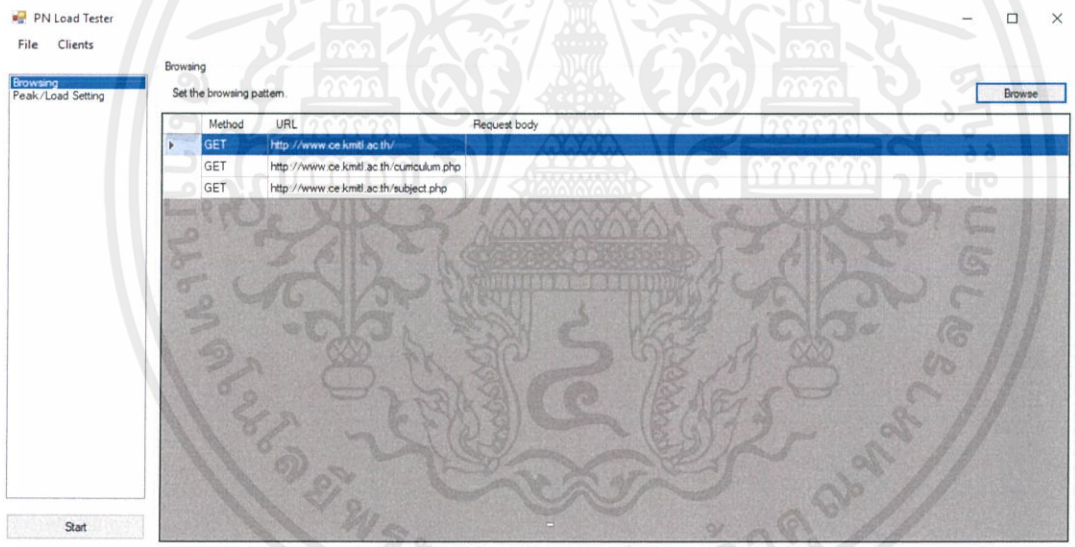
รูปที่ 4.11 Add เครื่อง Slave ที่เครื่อง Master

Master Setting

- คลิกที่ Client เลือก Manage Connection
- ใส่ IP Address และ Port ที่ใช้เชื่อมต่อกับเครื่อง Slave ดังรูปที่ 4.11

3) จำลองการใช้งาน User

รูปที่ 4.12 เข้า Website ที่ต้องการจะทดสอบ

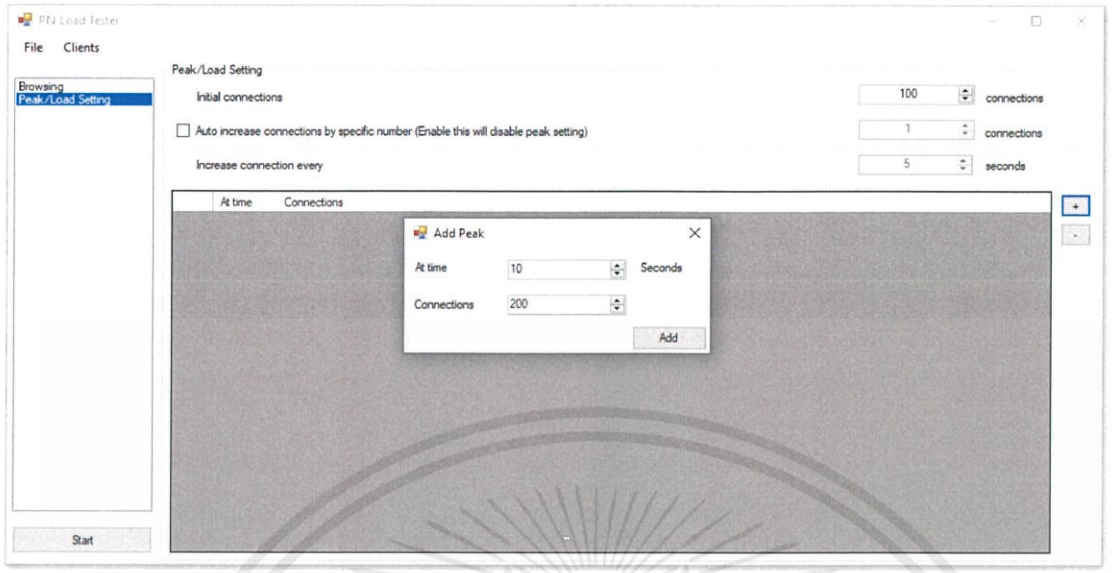


รูปที่ 4.13 โปรแกรมเก็บ Cache URL ทั้งหมด

Master Setting

- เข้าเว็บที่ต้องการจะทดสอบ ดังรูปที่ 4.12
- คลิก URL ต่างๆ เพื่อจำลองการใช้งาน
- ในโปรแกรมจะเก็บ Cache URL ที่ User คลิก ดังรูปที่ 4.13

4) กำหนดค่า Connection

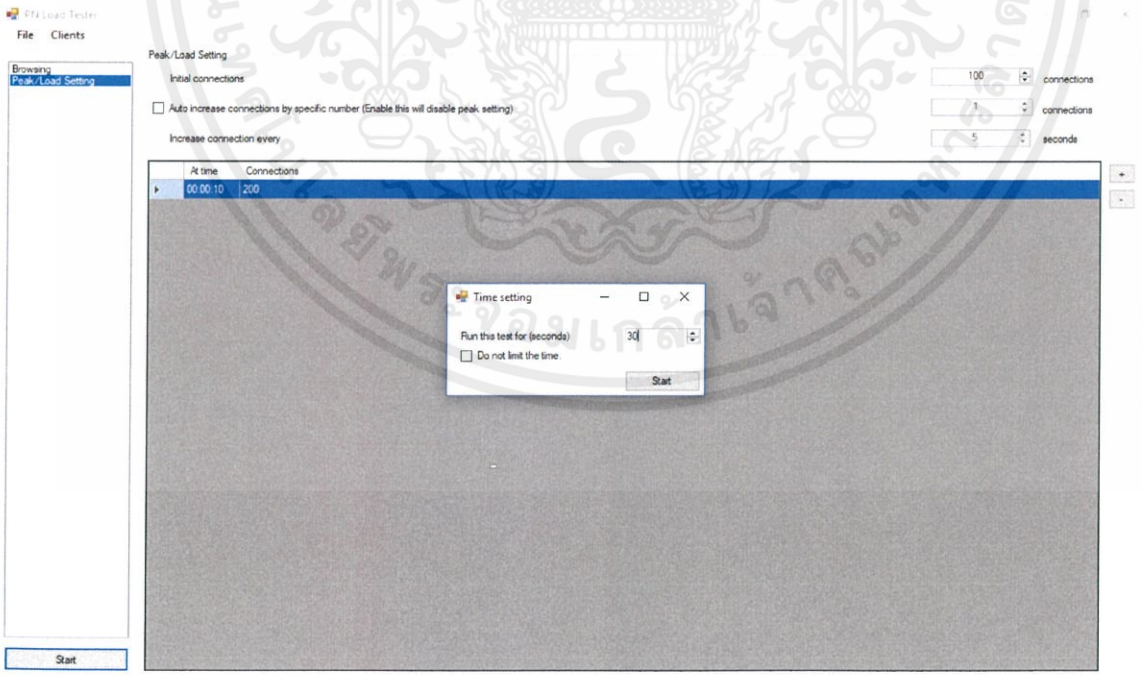


รูปที่ 4.14 กำหนดค่า Connection แบบ Peak Load

Master Setting

- กำหนดค่า Connection เริ่มต้น 100
- คลิกเครื่องหมาย + ทางด้านขวามือ กำหนด Peak Load ที่วินาทีที่ 10 เป็น 200 Connection ดังรูปที่ 4.14

5) กำหนดเวลาที่ใช้ในการทดสอบ

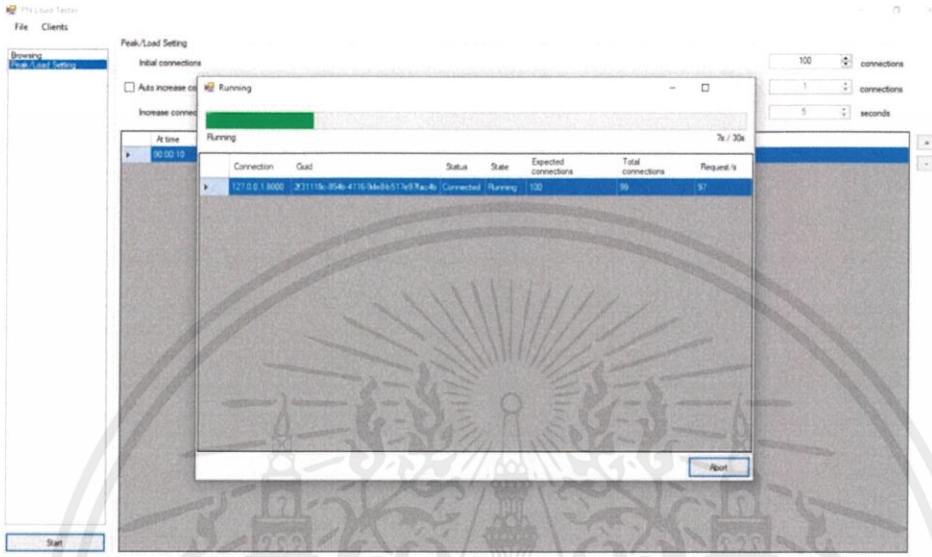


รูปที่ 4.15 กำหนดเวลาที่ใช้ในการทดสอบ

Master Setting

- กำหนดค่าเวลาที่ใช้ในการทดสอบ
- ในที่นี้กำหนดค่าเวลาเป็น 30 วินาที ดังรูปที่ 4.15

6) เริ่มการทดสอบ



รูปที่ 4.16 หน้าต่าง Process ระหว่างทดสอบของเครื่อง Master

Master Setting

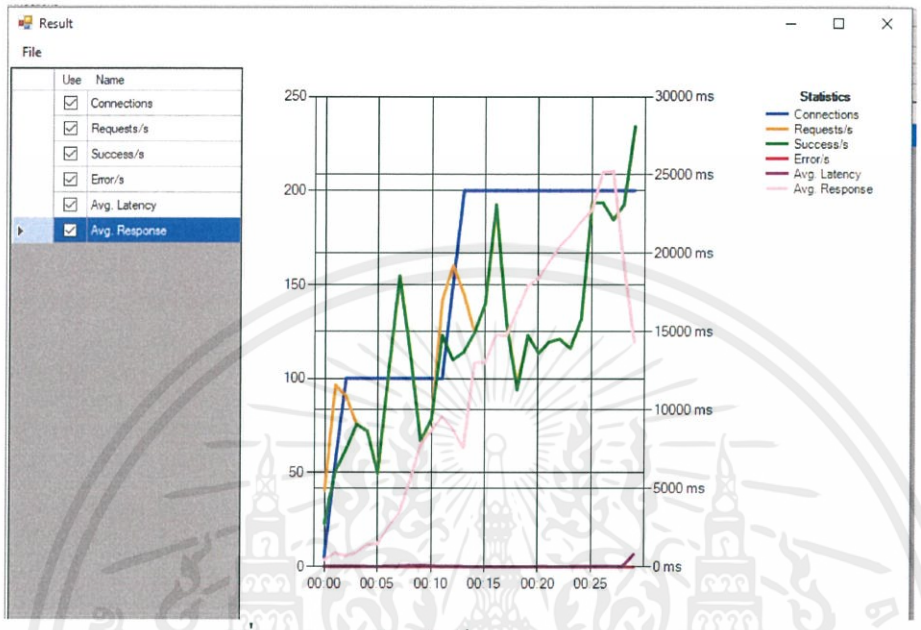
- กด Start เพื่อเริ่มการทดสอบ
- เครื่อง Master จะแสดงหน้า Process ระหว่างการทดสอบ ดังรูปที่ 4.16

รูปที่ 4.17 หน้าต่าง Process ระหว่างทดสอบเครื่อง Slave

Slave Setting

- เครื่อง Slave จะแสดงหน้า Process ระหว่างการทดสอบ ดังรูปที่ 4.17

7) แสดงผลลัพธ์ของการทดสอบในรูปของกราฟ



รูปที่ 4.18 กราฟผลลัพธ์ของ Peak Load Test

Master Setting

- เมื่อทดสอบเสร็จจะแสดงผลออกมาในรูปของกราฟ ดังรูปที่ 4.18
- สามารถกดเลือกให้แสดงผลกราฟที่ต้องการ

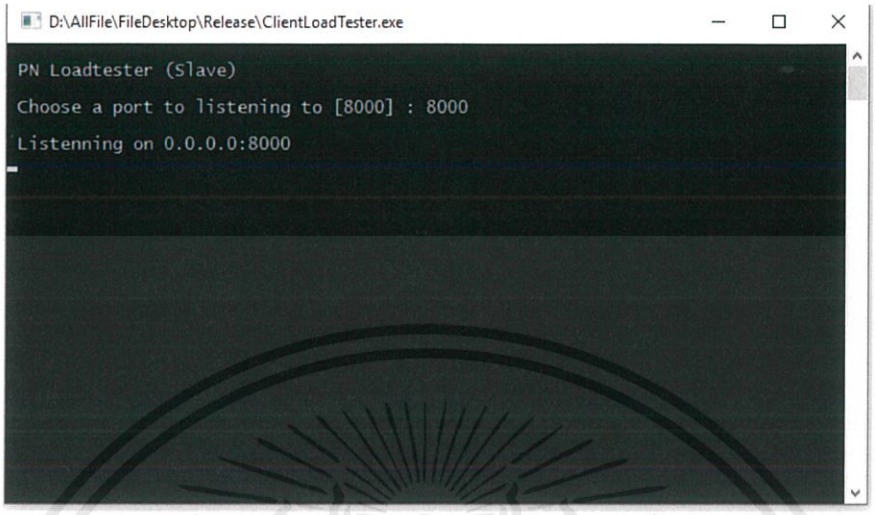
สรุปการทดสอบแบบ Peak load ในครั้งนี้

จากกราฟรูปที่ 4.18 จะแสดงให้เห็นว่า Server สามารถทำงานได้ปกติ สามารถรองรับ 100 Connection และ 200 Connectionได้ โดยสามารถสรุปจากกราฟได้ดังนี้

- มีการ Prepare Connection ถึงวินาทีที่ 2 เมื่อถึงวินาทีที่ 3 ค่า Connection เป็น 100 เนื่องจากระบบมีการสร้างการเชื่อมต่อเพิ่มขึ้นจนถึงค่าการเชื่อมต่อที่กำหนด เมื่อถึงวินาทีที่ 10 ตามที่ได้ทำการ Set ค่า Connection เป็น 200 ทำให้เกิดการ Peak load Connection ขึ้นไปเป็น 200 ที่วินาทีที่ 13 หลังจากนั้นเป็นค่า Connection เป็น 200 ตลอดจนการทดสอบสิ้นสุด
- ช่วงวินาทีที่ 0-2 กราฟของค่า Request/s ยังไม่ทับกับค่า Success/s สาเหตุอาจมาจากช่วงดังกล่าวระบบมีการสร้างการเชื่อมต่อเพิ่มขึ้นจนถึงค่าการเชื่อมต่อตามที่กำหนด หลังจากนั้นวินาทีที่ 3 กราฟของค่า Request/s จะทับกับเส้นของค่า Success/s ไปจนถึงวินาทีที่ 10 และวินาทีที่ 11-14 ซึ่งเป็นช่วง Peak Load กราฟของค่า Request/s ไม่ทับกับค่า Success/s แสดงว่าช่วง Peak load มีบาง Request ที่ส่งไปแล้วไม่ Success สาเหตุอาจมาจากช่วงดังกล่าวระบบมีการสร้างการเชื่อมต่อเพิ่มขึ้นจนถึงค่าการเชื่อมต่อตามที่กำหนด
- จากกราฟ ตลอดการทดสอบค่า Error เป็น 0 แสดงว่าตลอดการทดสอบไม่มี Error เกิดขึ้น
- กราฟของค่าของ Response Time โดยรวมแล้ว มีค่าเพิ่มขึ้นค่อนข้างเป็นเส้นตรง แสดงว่าระยะเวลาที่ใช้ในการตอบสนองนั้นเพิ่มขึ้นเรื่อยๆ แต่หลังจากวินาทีที่ 27 ค่าจะลดลง สาเหตุอาจมาจากใกล้หมดเวลาที่ใช้ในการทดสอบ ทำให้ระบบอาจมีการเตรียมความพร้อมสำหรับปิดการเชื่อมต่อ

4.2.3 ขั้นตอนการทดสอบแบบ Increase Load

1) กำหนด Port ให้เครื่อง Slave

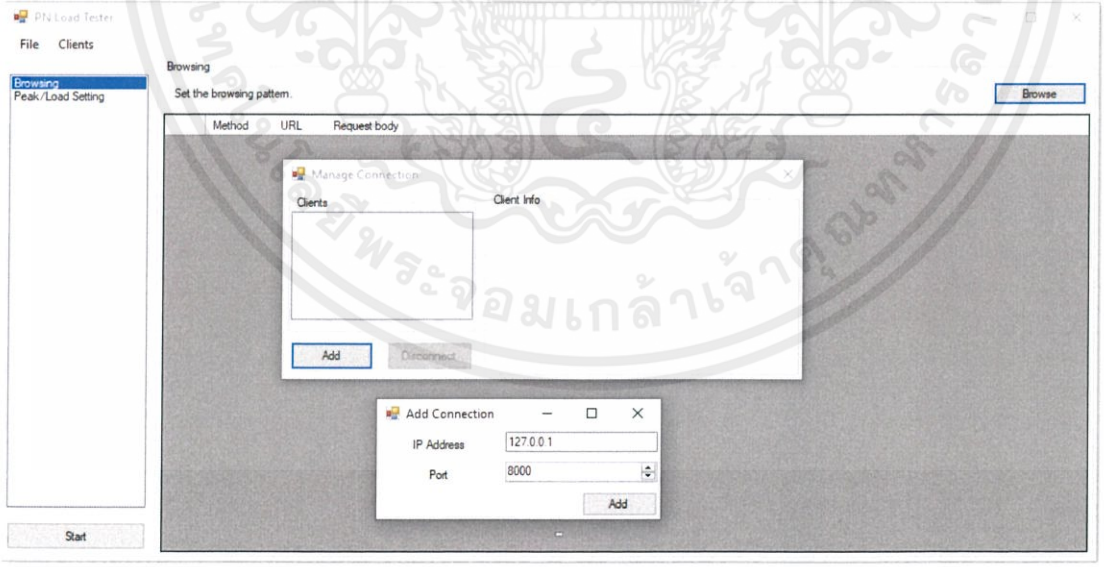


รูปที่ 4.19 กำหนด Port เครื่อง Slave

Slave Setting

- ให้กำหนด Port ที่ใช้เชื่อมต่อกับเครื่อง Master ดังรูปที่ 4.19
- ในการทดสอบครั้งนี้ใช้ Port 8000

2) Add Slave ที่เครื่อง Master



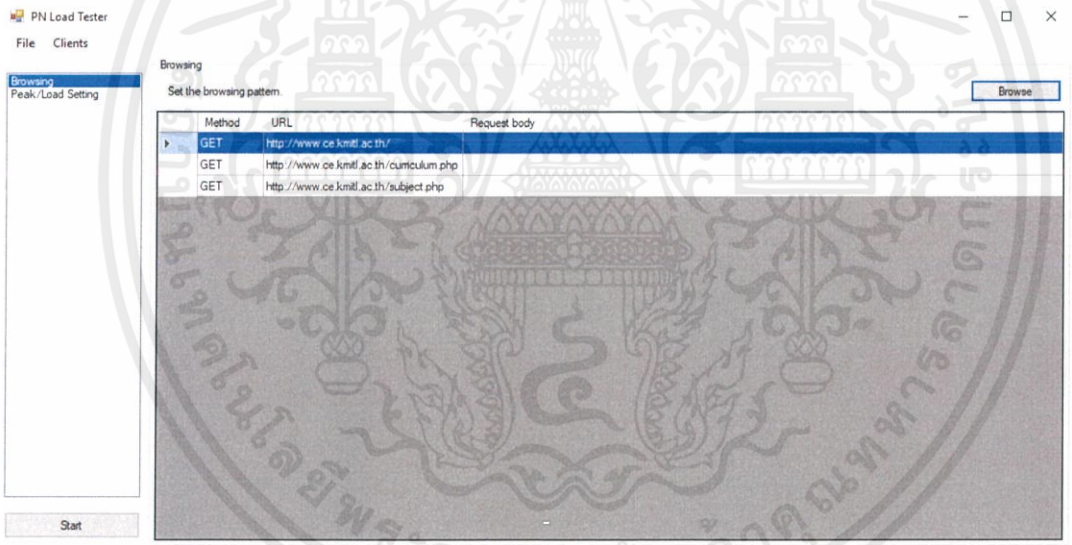
รูปที่ 4.20 Add เครื่อง Slave ที่เครื่อง Master

Master Setting

- คลิกที่ Client เลือก Manage Connection
- ใส่ IP Address และ Port ที่ใช้เชื่อมต่อกับเครื่อง Slave ดังรูปที่ 4.20

3) จำลองการใช้งาน User

รูปที่ 4.21 เข้า Website ที่ต้องการจะทดสอบ

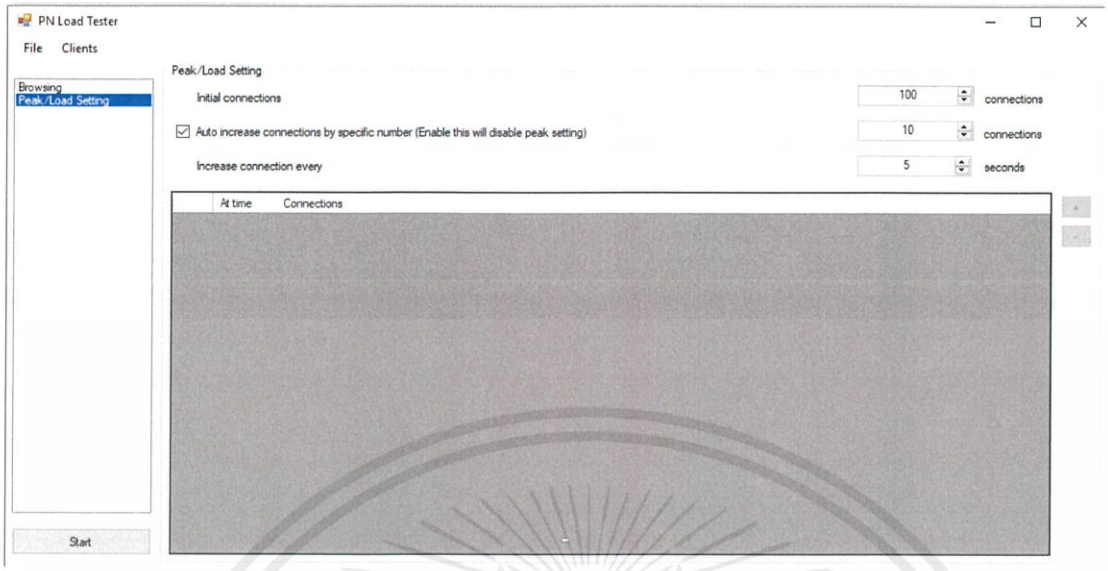


รูปที่ 4.22 โปรแกรมเก็บ Cache URL ทั้งหมด

Master Setting

- เข้าเว็บที่ต้องการจะทดสอบ ดังรูปที่ 4.21
- คลิก URL ต่างๆ เพื่อจำลองการใช้งาน
- ในโปรแกรมจะเก็บ Cache URL ที่ User คลิก ดังรูปที่ 4.22

4) กำหนดค่า Connection

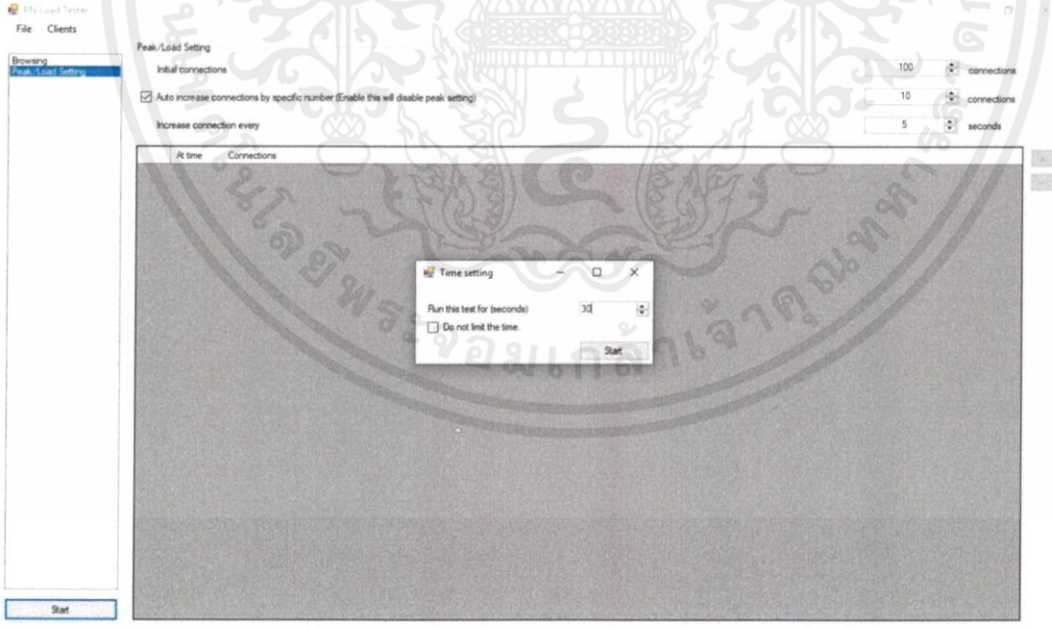


รูปที่ 4.23 กำหนดค่า Connection แบบ Increase Load

Master Setting

- กำหนดค่า Connection เริ่มต้นที่ 100
- ทำการเพิ่มค่า Connection ทีละ 10 connection ทุกๆ 5 วินาที ดังรูปที่ 4.23

5) กำหนดเวลาที่ใช้ในการทดสอบ

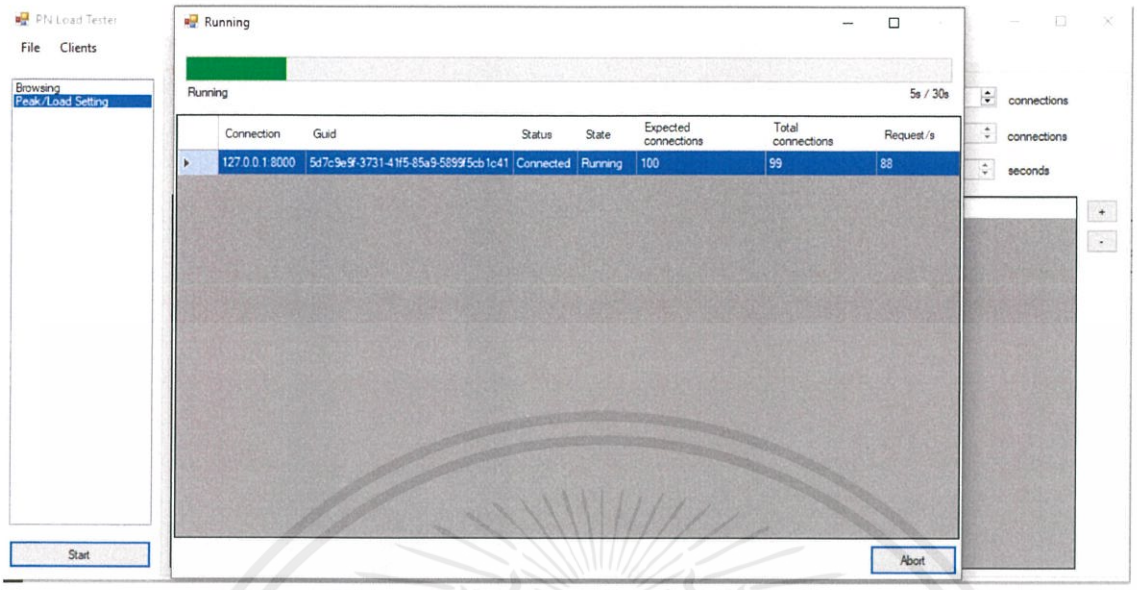


รูปที่ 4.24 กำหนดเวลาที่ใช้ในการทดสอบ

Master Setting

- กำหนดค่าเวลาที่ใช้ในการทดสอบ
- ในที่นี้กำหนดค่าเวลาเป็น 30 วินาที ดังรูปที่ 4.24

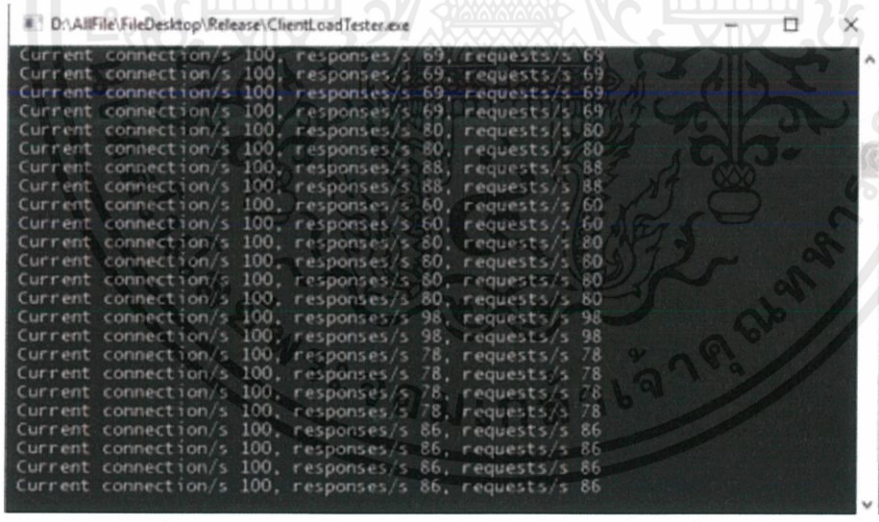
6) เริ่มการทดสอบ



รูปที่ 4.25 หน้าต่าง Process ระหว่างทดสอบของเครื่อง Master

Master Setting

- กด Start เพื่อเริ่มการทดสอบ
- เครื่อง Master จะแสดงหน้า Process ระหว่างการทดสอบ ดังรูปที่ 4.25

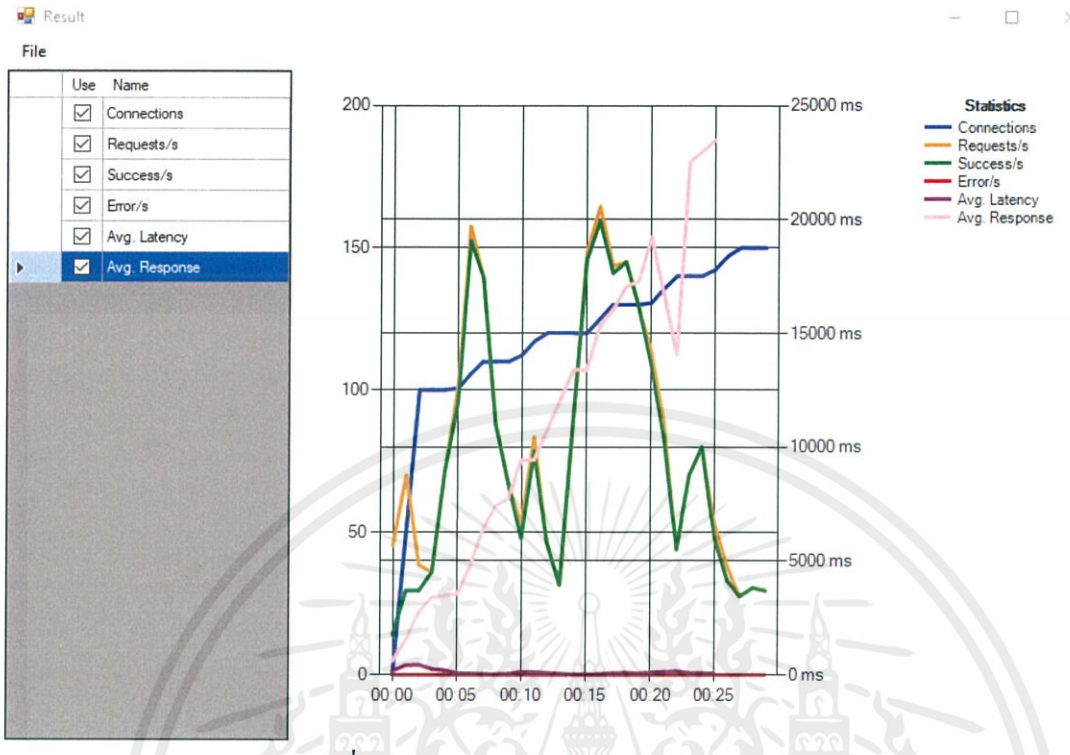


รูปที่ 4.26 หน้าต่าง Process ระหว่างทดสอบของเครื่อง Slave

Slave Setting

- เครื่อง Slave จะแสดงหน้า Process ระหว่างการทดสอบ ดังรูปที่ 4.26

7) แสดงผลลัพธ์ของการทดสอบในรูปของกราฟ



รูปที่ 4.27 กราฟผลลัพธ์ของ Increase Load

Master Setting

- เมื่อทดสอบเสร็จ จะแสดงผลออกมาในรูปของกราฟ ดังรูปที่ 4.27
- สามารถกดเลือกให้แสดงผลกราฟที่ต้องการ

สรุปการทดสอบแบบ Increase load ในครั้งนี้

จากกราฟรูปที่ 4.27 จะแสดงให้เห็นว่า Server สามารถทำงานได้ปกติ สามารถรองรับ 100 Connection เริ่มต้น และเพิ่ม 10 Connection ทุกๆ 5 วินาทีได้ โดยสามารถสรุปจากกราฟได้ดังนี้

- มีการ Prepare Connection ถึงวินาทีที่ 2 เมื่อถึงวินาทีที่ 3 ค่า Connection เป็น 100 เนื่องจากระบบมีการสร้างการเชื่อมต่อเพิ่มขึ้นจนถึงค่าการเชื่อมต่อตามที่กำหนด หลังจากนั้นวินาทีที่ 5 มีการเพิ่มค่า Connection อีก 10 Connection เป็น 110 ตามที่ได้กำหนดไว้ และวินาทีที่ 10 มีการเพิ่มค่า Connection อีก 10 Connection เป็น 120 ตามที่ได้กำหนดไว้ จะสังเกตเห็นว่ามีการเพิ่มค่า Connection จำนวน 10 Connection ทุกๆ 5 วินาที ตามที่ได้กำหนดไว้จนจบการทดสอบ
- ช่วงแรกค่าของกราฟ Request/s ยังไม่ทับกับเส้นค่าของ Success/s ไปจนถึงวินาทีที่ 3 เนื่องจากระบบมีการสร้างการเชื่อมต่อเพิ่มขึ้นจนถึงค่าการเชื่อมต่อตามที่กำหนด เมื่อถึงวินาทีที่ 4 ค่าของ Request/s จะทับกับเส้นค่าของ Success/s แสดงว่าตั้งแต่วินาทีที่ 4 เมื่อส่ง Request ไปแล้ว Success ตลอดการทดสอบ
- จากกราฟ ตลอดการทดสอบค่า Error เป็น 0 แสดงว่าตลอดการทดสอบไม่มี Error เกิดขึ้น
- ช่วงวินาทีที่ 1-20 ค่าของ Response Time มีค่าสูงขึ้นแบบค่อนข้างคงที่ เนื่องจากมีการตอบสนองช้าลงเรื่อยๆ จนถึงวินาทีที่ 20 ค่าของ Response Time ลดลงแบบคงที่เป็นเส้นตรงเช่นกัน เนื่องจากมีการตอบสนองได้เร็วขึ้น

บทที่ 5

บทสรุปและข้อเสนอแนะ

ในบทนี้จะกล่าวถึงบทสรุปและข้อเสนอแนะในการพัฒนาโปรแกรม Load Testing จากขอบเขตงานและวัตถุประสงค์ของโครงงาน สามารถสรุปผลการทำโครงงาน ได้ดังนี้

5.1 บทสรุป

สิ่งที่สามารถทำได้

1) พัฒนาโปรแกรม Load testing ให้สามารถแก้ไขและปรับค่าที่เกี่ยวข้องกับการทดสอบได้ – การปรับค่าหรือแก้ไขค่าที่ใช้ในการทดสอบนั้น ตัวโปรแกรมจะมีกล่องข้อความให้ User กรอกข้อมูลหรือค่าต่างๆที่ User ต้องการจะกำหนด โดยเมื่อทดสอบการทำงานจริง พบว่าข้อมูลที่ตัวทดสอบได้รับตรงกับข้อมูลที่ User กรอก

2) พัฒนาโปรแกรมให้สามารถทำการทดสอบในรูปแบบ Master-Slave ได้ – ตัวโปรแกรมถูกออกแบบให้ทำงานแบบ Master-Slave โดยจะมีเครื่องหนึ่งทำหน้าที่เป็น Master คอยสั่งการนอกเหนือจากนั้นจะเป็น Slave ที่คอยทำการทดสอบทั้งหมด โดยการเชื่อมต่อระหว่าง Master กับ Slave นั้นใช้การเชื่อมต่อแบบ socket โดย Master จะระบุ IP และ Port เฉพาะที่จะใช้ติดต่อกับ Slave และ Slave ก็จะถูกตั้งค่าให้รอการเชื่อมต่อจาก Port เฉพาะนั้นอย่างเดียว โดยเมื่อทดสอบการทำงานจริง พบว่าสามารถเชื่อมต่อกันได้สำเร็จ และสามารถส่งข้อมูลให้กันได้อย่างปกติ

3) สร้างตัวติดตั้งของโปรแกรม เมื่อติดตั้งเสร็จแล้ว สามารถใช้งานได้เป็นปกติ - เนื่องจากผู้ทำโครงงานใช้โปรแกรม Visual studio 2017 ในการพัฒนาโปรแกรม ซึ่งมีฟีเจอร์ช่วยสร้างตัว install ของโปรแกรม จึงทำให้สามารถสร้างตัว install ได้ง่ายขึ้น โดยเมื่อทำการทดสอบติดตั้งลงบนเครื่องจริง จากนั้นลองใช้โปรแกรม ปรากฏว่าไม่พบปัญหาที่แตกต่างจากตอนพัฒนาและสามารถใช้งานได้เป็นปกติ

4) จัดทำคู่มือการใช้งานระบบเป็นภาษาไทย - ประกอบไปด้วยเนื้อหาที่เกี่ยวข้องกับตัวโปรแกรม และ คำแนะนำของตัวโปรแกรม เช่น วิธีการติดตั้ง , วิธีการใช้งาน เป็นต้น

5) พัฒนาโปรแกรมให้สามารถแสดงผลในรูปแบบของกราฟ โดยสามารถแสดงค่าข้อมูลที่ใช้ในการวิเคราะห์ Performance – โดยตัวกราฟแสดงผลนั้น พัฒนาขึ้นโดยใช้ Newtonsoft เมื่อทดสอบการทำงานจริง พบว่ากราฟสามารถแสดงค่าต่างๆได้ตามปกติ

สิ่งที่ไม่สามารถทำได้

1) สามารถรองรับ User ได้สูงสุดจำนวน 200,000 User - ยังไม่สามารถทำได้เนื่องจากประสิทธิภาพของคอมพิวเตอร์ของผู้ทำโครงงานนั้นไม่เพียงพอ อีกทั้งยังขาดแคลนคอมพิวเตอร์ในด้านจำนวนเพื่อนำมาใช้เป็นตัว Slave ในการทดลอง ซึ่งปัจจุบันสามารถทำได้ 2000 Users จาก Slave 3 เครื่อง

5.2 ปัญหา อุปสรรค และแนวทางการแก้ไข

1) ไม่มี Library บางตัวที่จำเป็นในการสร้าง ทำให้ต้องสร้าง Library ขึ้นมาใช้เอง

2) ระบบบางส่วนของโปรแกรมมีความซับซ้อน บางครั้งการเพิ่มเติมฟีเจอร์จำเป็นจะต้องทำการออกแบบใหม่ในส่วนที่ทำไว้แล้ว จึงทำให้เกิดความล่าช้า

3) ในบางครั้ง Firewall อาจขัดขวางการทำงานของโปรแกรมทำให้ Slave กับ Master เชื่อมต่อกันไม่ได้

4) ประสิทธิภาพในการทดสอบขึ้นอยู่จำนวนและสเปคของเครื่อง Slave

5.3 แนวทางในการพัฒนาต่อ

1) ปรับปรุงให้ UI สวยงามและเข้าใจง่ายมากขึ้น

2) พัฒนาให้สามารถสั่งการและดูผลลัพธ์ (Master) บนอุปกรณ์ Smartphone หรือ Tablet ได้

3) Optimize โปรแกรมให้สามารถดึงประสิทธิภาพการทำงานของเครื่องให้ได้มากที่สุด

บรรณานุกรม

[1] Microsoft MSDN, "Visual C# Language," 21 July 2017. [ออนไลน์]. Available: https://msdn.microsoft.com/en-us/library/aa287558(v=vs.71).aspx.

[2] "C Sharp (programming language)," 22 July 2017. [ออนไลน์]. Available: https://en.wikipedia.org/wiki/C_Sharp_(programming_language).

[3] Webopedia, "C#," 22 July 2017. [ออนไลน์]. Available: http://www.webopedia.com/TERM/C/C_sharp.html.

[4] Beginners Heap, "Advantages of C# over other languages," 22 July 2017. [ออนไลน์]. Available: https://www.beginnersheap.com/c-vs-other-languages/.

[5] J. H. Godel, "Similarities and difference with C# and Other Languages," 22 July 2017. [ออนไลน์]. Available: http://www.c-sharpcorner.com/article/similarities-and-difference-with-C-Sharp-and-other-languages/.

[6] "What is a Socket?," 22 July 2017. [ออนไลน์]. Available: https://www.tutorialspoint.com/unix_sockets/what_is_socket.htm.

[7] อ. เกษมศิริ, เอกสารประกอบสอนวิชา 01074201 Network Programming, 2017.

[8] B. Mitchell, "An Overview of Socket Programming for Computer Networking," 22 July 2017. [ออนไลน์]. Available: https://www.lifewire.com/socket-programming-for-computer-networking-4056385.

[9] Laem, ".NET Framework คืออะไร มีที่มาและความสำคัญอย่างไร," 10 October 2016. [ออนไลน์]. Available: https://notebookspec.com/net-framework-คืออะไร-มีที่มาและควา/88056.

[10] wikipedia, ".NET_Framework," 22 July 2017. [ออนไลน์]. Available: https://en.wikipedia.org/wiki/.NET_Framework.

[11] Microsoft, "Overview of the .NET Framework," 20 July 2017. [ออนไลน์]. Available: https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview.

[12] "Transmission Control Protocol," 22 July 2017. [ออนไลน์]. Available: https://en.wikipedia.org/wiki/Transmission_Control_Protocol.

[13] ช. ทินกรสูติบุตร และ ทีมงาน Tnet , "ความรู้พื้นฐานเกี่ยวกับ โปรโตคอล TCP/IP," 21 July 2017. [ออนไลน์]. Available: http://www.tnetsecurity.com/content_basic/tcp_ip_knowledge.php.

[14] M. Rouse, "TCP/IP (Transmission Control Protocol/Internet Protocol)," 21 July 2017. [ออนไลน์]. Available: http://searchnetworking.techtarget.com/definition/TCP-IP.

[15] Computer Hope, "TCP/IP," 20 July 2017. [ออนไลน์]. Available: https://www.computerhope.com/jargon/t/tcpip.htm.

[16] S. Puisungnoen, "แนวทางการทำ Load Testing แบบง่ายๆ," 10 October 2016. [ออนไลน์]. Available: http://www.somkiat.cc/think-before-load-testing/.

[17] "Load testing," 10 October 2016. [ออนไลน์]. Available: https://en.wikipedia.org/wiki/Load_testing.

[18] "แบบจำลอง Master/Slave," 20 July 2017. [ออนไลน์]. Available: https://www.ibm.com/support/knowledgecenter/th/ssw_aix_71/com.ibm.aix.genprogc/master_slave_model.htm.

[19] "How to Perform Distributed Testing Using JMeter?," 10 October 2016. [ออนไลน์]. Available: http://www.360logica.com/blog/how-to-perform-distributed-testing-using-jmeter/.