โครงการการพัฒนาต่อยอดการติดต่อสื่อสารแบบภาพและเสียง

Video and Voice Communication on constraint condition

กวิน ธนิกกุล

สหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)
ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2560

Video and Voice Communication on constraint condition

KAWIN THANIKKUN

A COOPERATIVE EDUCATION SUBMITTED IN PARTIAL
FULFILLMENT OF THE REQUIREMENT FOR
THE DEGREE OF BACHELOR OF SCIENCE (COMPUTER SCIENCE)
DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF SCIENCE
KING MOMGKUT'S INSTITUTE OF TECHNOLOGIES LADKRABANG
ACADEMIC YEAR 2017

หัวข้อสหกิจศึกษา โครงการการพัฒนาต่อยอดการติดต่อสื่อสารแบบภาพและเสียง

Video and Voice Communication on constraint condition

ชื่อนักศึกษา นายกวิน ธนิกกุล รหัสนักศึกษา 58050207

ปริญญา วิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)

ภาควิชา วิทยาการคอมพิวเตอร์

ปีการศึกษา 2560

อาจารย์ที่ปรึกษา ผศ.กฤษฎา บุศรา

คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้สหกิจศึกษานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาตรีวิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์) ประจำปีการศึกษา 2560

คณะกรรมการการสอบ	ลายมือชื่อ
ผศ.กฤษฎา บุศรา ประธานกรรมการและอาจารย์ที่ปรึกษา	

หัวข้อสหกิจศึกษา โครงการการพัฒนาต่อยอดการติดต่อสื่อสารแบบภาพและเสียง

Video and Voice Communication on constraint condition

ชื่อนักศึกษา นายกวิน ธนิกกุล รหัสนักศึกษา 58050207

ปริญญา วิทยาศาสตรบัณฑิต (วิทยาการคอมพิวเตอร์)

ภาควิชา วิทยาการคอมพิวเตอร์

คณะ วิทยาศาสตร์

มหาวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (สจล.)

ปีการศึกษา 2560

อาจารย์ที่ปรึกษา ผศ.กฤษฎา บุศรา

บทคัดย่อ

ผู้พัฒนาได้พัฒนาแอปพลิเคชันเพื่อใช้ในการติดต่อสื่อสารแบบเสียง และติดต่อสื่อสารแบบวิดีโอ ทั้งแบบเดี่ยว (1:1) และแบบกลุ่ม (n:n) โดยทางผู้พัฒนาต้องการการติดต่อสื่อสารแบบ Peer to Peer เพื่อลดการทำงานของเซิร์ฟเวอร์ และเพื่อลดความล่าช้าในการติดต่อ (Latency) ของแอปพลิเคชัน โดย แอปพลิเคชันต้องรองรับแพลตฟอร์มทั้ง Windows, Android , IOS และ MacOS โดยทั้ง 4 แพลตฟอร์มต้องสามารถรองรับการทำงานใน Browser และการทำงานของแอปพลิเคชันต้องสามารถทำงานข้ามแพลตฟอร์มได้ (Cross Platform) เช่น แอปพลิเคชันบน Web Browser ต้องสามารถใช้งานร่วมกับ แอปพลิเคชันบน Desktop/Web Browser/Android/IOS เป็นต้น ซึ่งในการพัฒนาแอปพลิเคชันได้มีการใช้ HTML, CSS, JavaScript, NodeJS, PHP และ Electron โดยส่วนที่ Front-end ผู้พัฒนาได้ใช้ HTML, CSS, JavaScript และ Electron ในการพัฒนา และในส่วน Back-end ได้ใช้ JavaScript ,NodeJS และ PHPในการพัฒนา

คำสำคัญ : Voice Communication, Video communication, Web Application

Title Video and Voice Communication on constraint condition

Student Mr. Kawin Thanikkun Student ID 58050207

Degree Bachelor of Science (Computer Science)

Department Computer Science

Faculty Science

University King Mongkut's Institute of Technology Ladkrabang (KMITL)

Academic Year 2017

Advisor Asst. Prof. Kridsada Budsara

Abstract

We develop a Video and Voice communication application to communicate one to one (1:1) and many to many (n:n). Used peer to peer communication to decrease a workload of servers and decrease latency of communications. This Application must support multiple platforms such as Windows, Android, IOS and MacOS. Each and Every platform must support browsers and must work across platform (Cross platform) for example, an application on web browser must be able to communicate with an application on Windows/Android/IOS/MacOS. We used HTML, CSS, JavaScript, NodeJS, PHP and Electron to develop this Applications. For Front-end development we used HTML, CSS, JavaScript and Electron. For Back-end development we used JavaScript, NodeJS and PHP.

Keywords : Voice Communication, Video Communication, Web Application

กิตติกรรมประกาศ

ในการจัดทำสหกิจศึกษาเรื่องโครงการการพัฒนาต่อยอดการติดต่อสื่อสารแบบภาพและเสียง ได้สำเร็จลุล่วงไปได้ดีเนื่องจากผู้จัดทำได้รับการสนับสนุนจากผู้มีพระคุณหลายท่านดังนี้

ขอขอบคุณคณาจารย์ทุกท่านที่อบรมสั่งสอนและถ่ายทอดความรู้ให้ผู้พัฒนามาตลอด

ขอขอบพระคุณทางบริษัท Klickerlab Co.,Ltd. ที่ได้สนับสนุนการทำสหกิจในครั้งนี้และให้โอกาศผู้จัดทำได้ร่วมงาน รวมไปถึงพี่ๆที่คอยให้คำแนะนำและคอยสอนมาโดยตลอด

สุดท้ายนี้ขอขอบคุณบิดา มารดา น้อง เพื่อนๆ และครอบครัวที่คอยสนับสนุนและเป็นกำลังใจมาโดยตลอดการทำสหกิจศึกษาในครั้งนี้

กวิน ธนิกกุล

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ผู้พัฒนาต้องการพัฒนาแอปพลิเคชันเพื่อใช้ในการติดต่อสื่อสารทั้งการติดต่อสื่อสารแบบเสียงและติดต่อสื่อสารแบบวิดีโอ ทั้งแบบเดี่ยว (1:1) และแบบกลุ่ม (n:n) โดยทางผู้พัฒนาต้องการการติดต่อสื่อสารแบบ Peer to Peer เพื่อลดการทำงานของเซิร์ฟเวอร์ และเพื่อลดความล่าช้าในการติดต่อ (Latency) ของแอปพลิเคชัน โดยแอปพลิเคชันต้องรองรับแพลตฟอร์มทั้ง PC, Android , IOS และ Mac โดยทั้ง 4 แพลตฟอร์มต้องสามารถรองรับการทำงานใน Browser และการทำงานของแอปพลิเคชันต้องสามารถทำงานข้ามแพลตฟอร์มได้ (Cross Platform) เช่น แอปพลิเคชันบน Web Browser ต้องสามารถใช้งานร่วมกับ แอปพลิเคชันบน Desktop/Web Browser/Android/IOS เป็นต้น

1.2 วัตถุประสงค์ของสหกิจศึกษา

1. เพื่อติดต่อสื่อสารได้ทั้งแบบ PC/Mobile to Mobile/PC ผ่านระบบอินเทอร์เน็ต
2. เพื่อติดต่อสื่อสารแบบส่งข้อความเสียงแบบเดี่ยว (1:1) และแบบกลุ่มได้ (n:n)
3. เพื่อติดต่อสื่อสารแบบวิดีโอ แบบเดี่ยว (1:1) และแบบกลุ่มได้ (n:n)
4. เพื่อติดต่อสื่อสาร ผ่านผู้ให้บริการมือถือ ไม่น้อยกว่า 3 ราย โดยคุณภาพเสียงและความเร็วอยู่ที่ความเร็วอินเทอร์เน็ต
5. สามารถตรวจสอบการทำงานของระบบได้

1.3 ขอบเขตของสหกิจศึกษา

1. สามารถติดต่อสื่อสารแบบข้อความเสียงผ่านเว็บแอปพลิเคชันได้ทั้งแบบเดี่ยว และแบบกลุ่ม

2. สามารถติดต่อสื่อสารแบบข้อความเสียงผ่านPC แอปพลิเคชันได้ทั้งแบบเดี่ยว และแบบกลุ่ม

3. สามารถติดต่อสื่อสารแบบข้อความเสียงผ่าน Android (Web Browser) แอปพลิเคชันได้ทั้งแบบเดี่ยว และแบบกลุ่ม

4. สามารถติดต่อสื่อสารวิดีโอ ผ่านเว็บแอปพลิเคชันได้ทั้งแบบเดี่ยว และแบบกลุ่ม

5. สามารถติดต่อสื่อสารวิดีโอ ผ่าน PC แอปพลิเคชันได้ทั้งแบบเดี่ยว และแบบกลุ่ม

6. สามารถติดต่อสื่อสารวิดีโอ ผ่าน Android (Web Browser) แอปพลิเคชันได้ทั้งแบบเดี่ยว และแบบกลุ่ม

7. ผู้ใช้สามารถจัดการกลุ่มได้โดยประกอบด้วย สร้างกลุ่ม เพิ่มสมาชิกกลุ่ม ลบกลุ่ม

8. ผู้ใช้สามารถเพิ่ม/ลบ รายชื่อเพื่อนได้

9. สามารถส่งคำเชิญร่วมสนทนา โดยมีการแสดง push notification ได้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ใช้แอปพลิเคชันทั้งPCและเว็บแอปพลิเคชันในการติดต่อสื่อสาร

2. สามารถนำผลงานชิ้นนี้ไปต่อยอดในการพัฒนาต่อได้

3. ลดต้นทุนในการพัฒนา

1.5 ขั้นตอนและการดำเนินงาน

Voice / Video Communication	Month 1				Month 2				Month 3				Month 4				Month 5				Month 6				Month 7			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1. Study/Requirement																												
1.1 basic																												
1.2 study topic																												
1.3 requirement confirmation																												
2. Design																												
2.1 project planning																												
2.2 design																												
2.3 UI design																												
3. Development																												
3.1 client-side coding																												
3.2 server-side coding																												
3.3 unit test																												
4. Testing																												
5. Documentation																												
6. Deployment																												

รูปที่ 1.1 แผนการทำงาน

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

การพัฒนาเว็บแอปพลิเคชันและPCแอปพลิเคชัน ผู้พัฒนาได้ศึกษาและประยุกต์ใช้เทคโนโลยีที่เกี่ยวข้อง โดยสามารถแบ่งเป็นหัวข้อได้ดังนี้

2.1 ภาษา HTML

2.2 ภาษา JavaScript

2.3 WebRTC

2.4 NodeJS

2.5 Electron

2.6 MySQL

2.1 ภาษา HTML

HTML (Hypertext Markup Language) เป็นภาษาที่ใช้ในการแสดงผลบนเว็บบราวเซอร์ โดยสามารถนำเสนอข้อมูลที่มีทั้งอักษร เสียง ภาพ และวิดีโอ ไฟล์ของ HTML เป็น text file ทำให้สามารถเขียนได้ใน text editor ต่างๆ เช่น Notepad, EditPlus เป็นต้น การเขียนของภาษา HTML จะอยู่ในรูปของ แท็ก (Tag) ซึ่งจะเป็น <tag> Content </tag> โดยชื่อของแท็กนั้นๆจะอยู่ใน <> และ content ของแท็กนั้นๆจะอยู่ระหว่าง <> และ </> ภาษา HTML ได้รับการพัฒนาอย่างต่อเนื่องมาโดยตลอด เพื่อเพิ่มประสิทธิภาพ และทำให้รับรองความหลากหลายของ content ทำให้ภาษา HTML เป็นภาษาที่ใช้ใน Browser ในปัจจุบัน

โครงสร้างของภาษา HTML

HTML มีองค์ประกอบ 2 ส่วน คือ ส่วนที่เป็นเนื้อหาและส่วนทีเป็นคำสั่ง หรือแท็ก รูปแบบ พื้นฐานโครงสร้างของเอกสาร HTML ดังรูปแบบนี้

```
<HTML>
    <HEAD>
        <TITLE>ชื่อแสดงบน Title bar</TITLE>
    </HEAD>
    <BODY>.....</BODY>
<HTML>
```

คำสั่งเบื้องต้นของภาษา HTML คำสั่งของภาษา HTML หรือทีเราเรียกว่า แท็ก (Tag) เป็นส่วนที่จัดการเกี่ยวกับรูปแบบการจัดเอกสารเพื่อแสดงผลบนบราว์เซอร์โดยมีรูปแบบคำสั่งเบื้องต้น ดังนี้

1. HTML เป็นคำสั่งเริ่มต้นและสิ้นสุดของเอกสาร
2. HEAD ใช้กำหนดหัวของเอกสาร
3. TITLE ใช้ในการแสดงชื่อของเอกสารบน title bar ของ Browser
4. BODY เป็นส่วนเนื้อหาของเอกสาร

2.2 ภาษา JavaScript

JavaScript คือ ภาษาคอมพิวเตอร์สำหรับการเขียนโปรแกรมบนระบบอินเทอร์เน็ต ที่กำลังได้รับความนิยมอย่างสูง JavaScript เป็น ภาษาสคริปต์เชิงวัตถุ (ที่เรียกกันว่า "สคริปต์" (script) ซึ่งในการสร้างและพัฒนาเว็บไซต์ (ใช้ร่วมกับ HTML) เพื่อให้เว็บไซต์ของเราดูมีการเคลื่อนไหว สามารถตอบสนองผู้ใช้งานได้มากขึ้น ซึ่งมีวิธีการทำงานในลักษณะ "แปลความและดำเนินงานไปทีละคำสั่ง" (interpret) หรือเรียกว่า อ็อบเจ็กโอเรียลเต็ด (Object Oriented Programming) ที่มีเป้าหมายในการออกแบบและพัฒนาโปรแกรมบน Web Browser สำหรับผู้เขียนด้วยภาษา HTML สามารถทำงาน

ข้ามแพลตฟอร์มได้ โดยทำงานร่วมกับ ภาษา HTML และภาษา Java ได้ทั้งทางฝั่งไคลเอนต์ (Client) และ ทางฝั่งเซิร์ฟเวอร์ (Server)

เนื่องจาก JavaScript ช่วยให้ผู้พัฒนา สามารถสร้างเว็บเพจได้ตรงกับความต้องการ และมีความน่าสนใจมากขึ้น ประกอบกับเป็นภาษาเปิด (Open-source) ที่ทุกคนสามารถนำไปใช้ได้ ด้วยสังคมที่ใหญ่ขึ้นเรื่อยๆ และมี contributor ที่ช่วยกันสร้าง package ต่างๆและเครื่องมือให้กับ community ดังนั้นจึงได้รับความนิยมเป็นอย่างสูง มีการใช้งานอย่างกว้างขวาง รวมทั้งได้ถูกกำหนดให้เป็นมาตรฐานโดย ECMA การทำงานของ JavaScript จะต้องมีการแปลความคำสั่ง ซึ่งขั้นตอนนี้จะถูกจัดการโดยบราวเซอร์ (เรียกว่าเป็น client-side script) ดังนั้น JavaScript จึงสามารถทำงานได้ เฉพาะบนบราวเซอร์ที่สนับสนุน ซึ่งปัจจุบันบราวเซอร์เกือบทั้งหมดก็สนับสนุน JavaScript แล้ว อย่างไรก็ดี สิ่งที่ต้องระวังคือ JavaScript มีการพัฒนาเป็นเวอร์ชั่นใหม่ๆออกมาด้วย (ปัจจุบันคือรุ่น 1.5) ดังนั้น ถ้านำโค้ดของเวอร์ชั่นใหม่ ไปรันบนบราวเซอร์รุ่นเก่าที่ยังไม่สนับสนุน ก็อาจจะทำให้เกิดข้อผิดพลาดได้

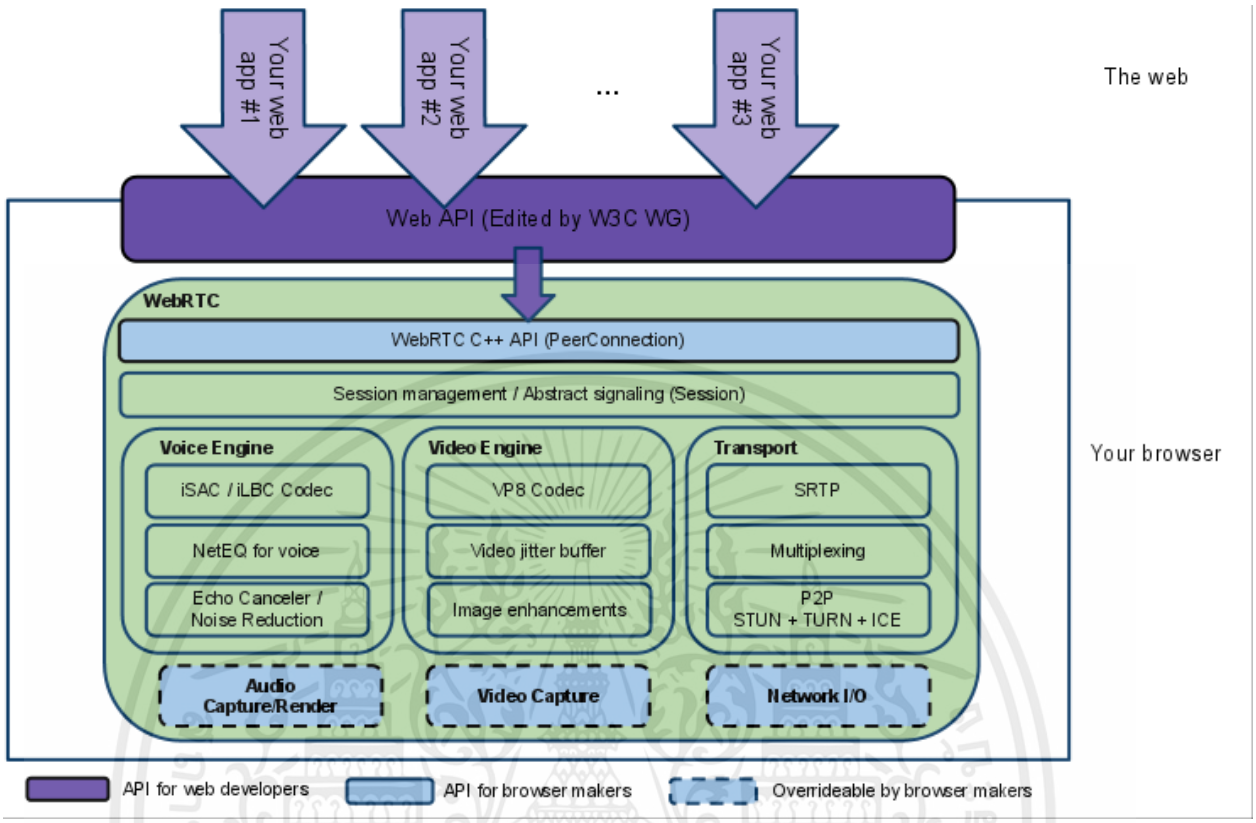
2.3 WebRTC

WebRTC (Web Real-Time Communication) เป็น Open Source Project ที่ใช้สำหรับทำให้ web Browser และ แอปพลิเคชันบนมือถือ ติดต่อกันได้แบบ Real Time โดยการเรียกใช้ผ่าน API อีกทั้งข้อมูลต่างๆได้โดยไม่จำเป็นต้องมีตัวกลางเข้ามาช่วย

WebRTC มีการใช้งาน Media Capture และ Stream API ไปพร้อมกัน ซึ่งทำให้สามารถ Capture Multimedia ใน Web Browser การติดต่อกันระหว่างผู้ใช้เป็การติดต่อแบบ Peer to Peer โดยไม่จำเป็นต้องใช้ plugs-in หรือ ตัวกลางเข้ามาช่วย

การติดต่อกันระหว่างผู้ใช้ ถูกสร้างขึ้นและถูกแสดงโดย RTCPeerConnection อินเตอร์เฟส เมื่อการติดต่อสำเร็จและสามารถเปิดใช้งานได้ Media Streams และ/หรือ Data Channels จะถูกเพิ่มลงไปในการติดต่อนั้น ๆ

Architecture diagram ด้านล่างแสดงโครงสร้างและหน้าที่ของ WebRTC



รูปที่ 2.1 โครงสร้างของ WebRTC

WebRTC ประกอบด้วยส่วนสำคัญ 2 ส่วน คือ Media Capture (getUserMedia) และ RTCPerrConnection

2.3.1 Media Capture (getUserMedia)

WebRTC ใช้ API ในการ Media Capture คือ getUserMedia หรือ navigator.getUserMedia (แบบใหม่จะใช้ navigator.MediaDevices.getUserMedia ซึ่งจะซัพพอร์ตการทำงานแบบ Promise) ซึ่ง API นี้ ซัพพอร์ตบราวเซอร์เป็นส่วนใหญ่ (ยกเว้น Internet Explorer) โดยก่อนการเรียกใช้จะต้องได้รับ Permission จากผู้ใช้งานก่อน หลังจากเรียกใช้แล้วจะได้ MediaStream ซึ่งจะประกอบได้ด้วย tracks ต่างๆ (tracks จะประกอบได้ด้วย Audio tracks ซึ่งคือ tracks เสียง และ Video tracks ซึ่งคือ tracks ของวิดีโอจาก webcam)

ตัวอย่างการใช้ getUserMedia

```
navigator.mediaDevices.getUserMedia(constraints)
```

constraints parameters จะประกอบด้วย Video และ Audio ซึ่งทั้ง 2 อย่างจะต้องระบุให้ชัดเจน เช่น

```
{ audio: true, video: true }
```

ซึ่ง true หมายถึงต้องการ tracks นั้นๆ หากเป็น false function จะไม่ return tracksนั้นๆกลับมา

Constraint video สามาถระบุขนาดได้เช่น

```
{
  audio: true,
  video: {
    width: { min: 1280 },
    height: { min: 720 }
  }
}
```

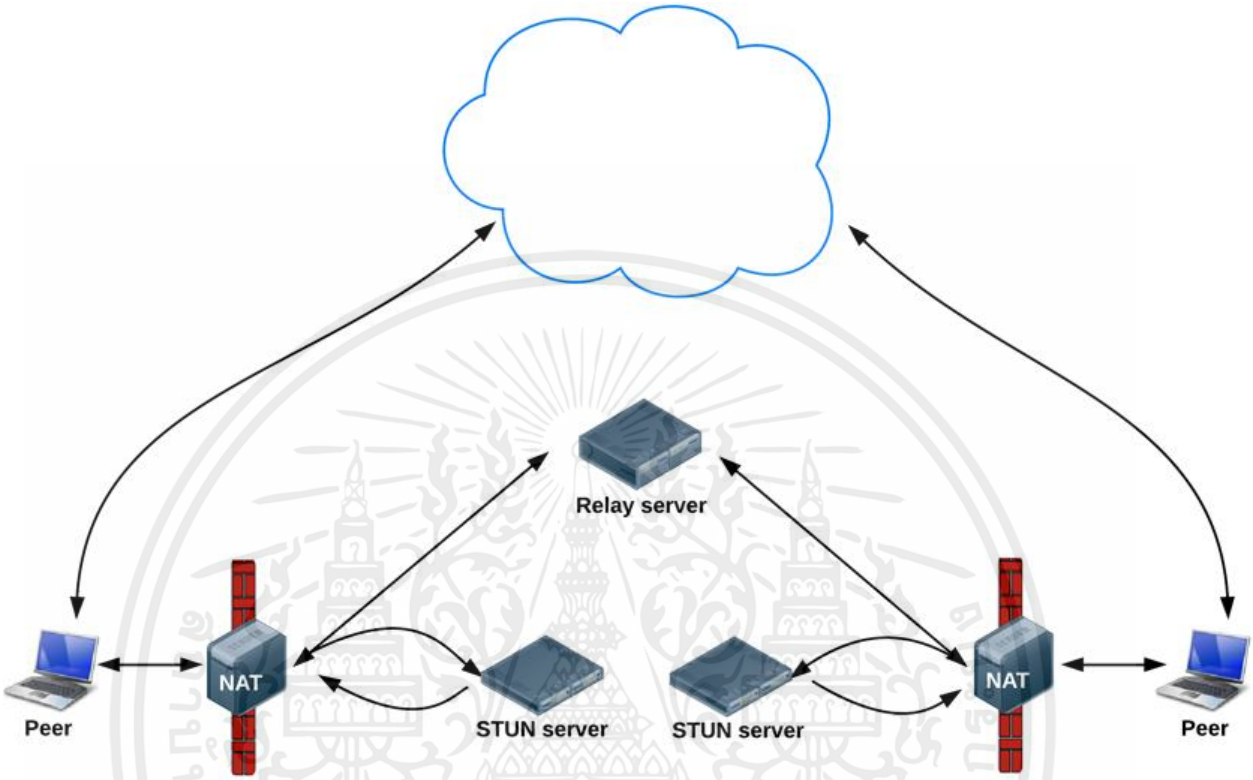
Constraint ข้างต้นหมายความว่าให้หา video ที่มีขนาดความกว้างอย่างน้อย1280 และควาวมยาวอย่างน้อย720

2.3.2 RTCPeerConnection

RTCPeerConnection เป็นส่วนที่จัดการการเชื่อมต่อกันระหว่างผู้ใช้ โดยมีขั้นตอนดังนี้

- การค้นหาผู้ใช้ และ การติดต่อ
- การส่งสัญญาณ (Signaling)
- NAT traversal
- Relay servers ในกรณีที่การติดต่อแบบ Peer to Peer ไม่สำเร็จ

ซึ่งตามทฤษฎีแล้ว การติดต่อแบบ Peer to Peer ไม่จำเป็นต้องผ่านตัวกลางใดๆเลย แต่ในความเป็นจริงนั้นจำเป็นต้องต้องทำ NAT traversal เพื่อให้ผู้ใช้ค้นพบกัน ดังนั้นจึงต้องมี Signaling server เข้ามาช่วยดังภาพด้านล่าง



รูปที่ 2.2 STUN server ช่วยใน NAT traversal

โดย NAT traversal จะใช้ STUN (Session Traversal Utilities for NAT) ซึ่งเป็น Protocol ของ WebRTC

2.3.3 ICE (Interactive Connectivity Establishment)

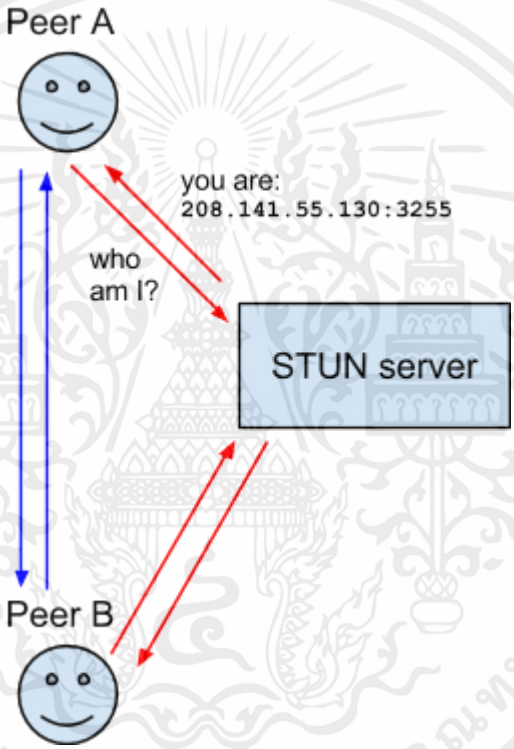
ICE (Interactive Connectivity Establishment) เป็น framework ที่ทำให้ Web Browser ติดต่อกับคนอื่นได้ มีหลายเหตุผลที่ทำให้การติดต่อโดยตรงแบบ Peer to Peer ไม่สามารถทำได้ เช่น การผ่าน Firewall ที่ป้องกันการเปิดการเชื่อมต่อ และความจำเป็นต้องมี Unique Address เมื่อไม่มี Public Address และจัดการการผ่านข้อมูลเมื่อไม่สามารถติดต่อโดยตรงได้

2.3.4 STUN (Session Traversal Utilities for NAT)

STUN (Session Traversal Utilities for NAT) คือ Protocol ของ WebRTC ที่ใช้ในการค้นพบ Public

Address และคาดการณ์หาข้อจำกัดใน Router ที่ทำให้เกิดการเชื่อมต่อ

ผู้ใช้งานจะส่งคำขอไปที่ STUN server ในอินเทอร์เน็ต และทาง server จะส่ง Public Address และบอกว่าผู้ใช้สามารถเข้าถึงได้หรือไม่ ดังรูปด้านล่าง

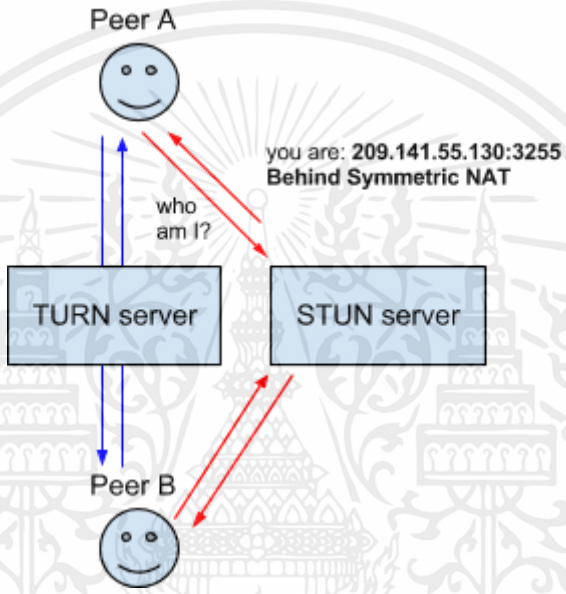


รูปที่ 2.3 การทำงานของ STUN

2.3.5 TURN (Traversal Using Relay around NAT)

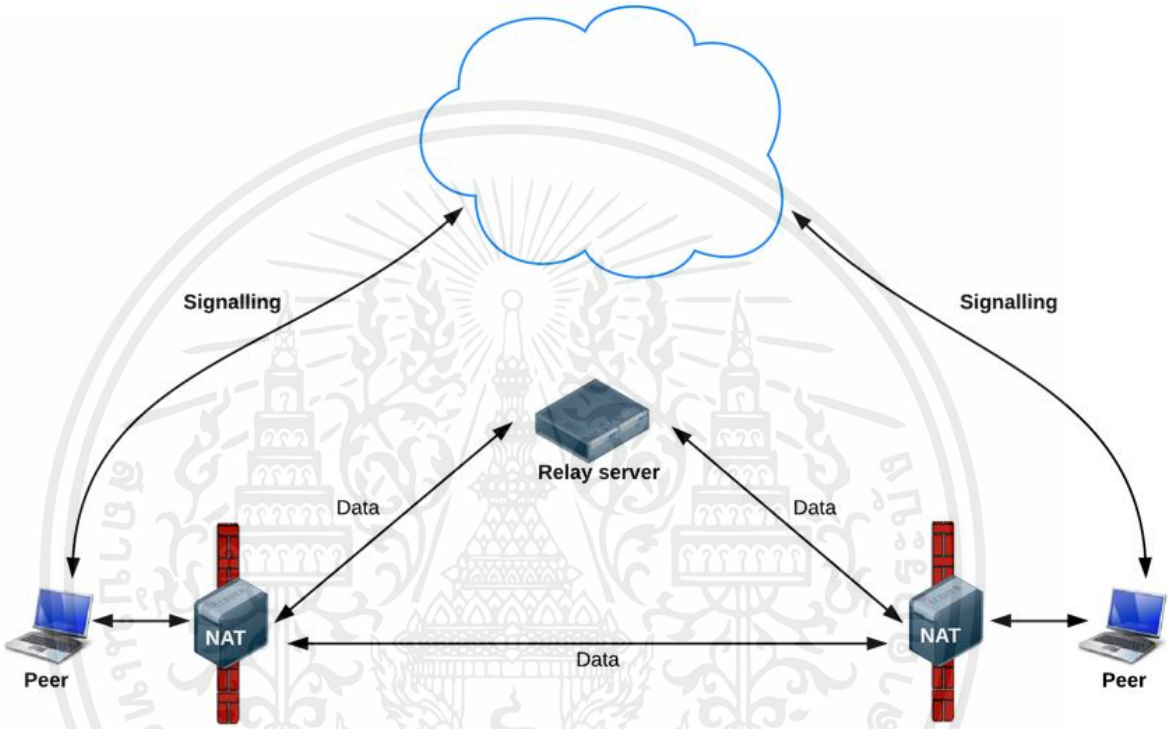
ในบาง Router เมื่อใช้ NAT ทำให้เกิดข้อจำกัดเรียกว่า “Symmetric NAT” หมายความว่า Router จะยอมรับการติดต่อจาก Peer ที่เคยติดต่อแล้วเท่านั้น

TURN (Traversal Using Relay around NAT) ช่วยในการข้าม Symmetric NAT โดยเปิดการติดต่อรหว่าง TURN server ละผ่านข้อมูลทั้งหมดไปยัง server ผู้ใช้จะสร้างการเชื่อมต่อระหว่าง TURN server และบอกผู้ใช้คนอื่นๆให้ส่ง packets ไปยัง server ซึ่งจะส่งกลับไปยังผู้ใช้ ดังรูปด้านล่าง



รูปที่ 2.4 การทำงานของ TURN

ตามขั้นตอนแล้ว ICE จะพยายามติดต่อโดยตรงระหว่างผู้ใช้โดยมีความล่าช้า (Latency) ให้น้อยที่สุดผ่าน UDP (User Datagram Protocol) ในขั้นตอนนี้ STUN server จะมีเพียงหน้าที่เดียว คือ การทำให้ผู้ใช้ที่อยู่หลัง NAT พบ Public Address และ port ถ้าไม่สามารถใช้ UDP ได้ ICE จะใช้ TCP (Transmission Control Protocol) ถ้าหากการติดต่อโดยตรงไม่สำเร็จจะใช้ตัวกลาง (TURN server) ดังรูปด้านล่าง



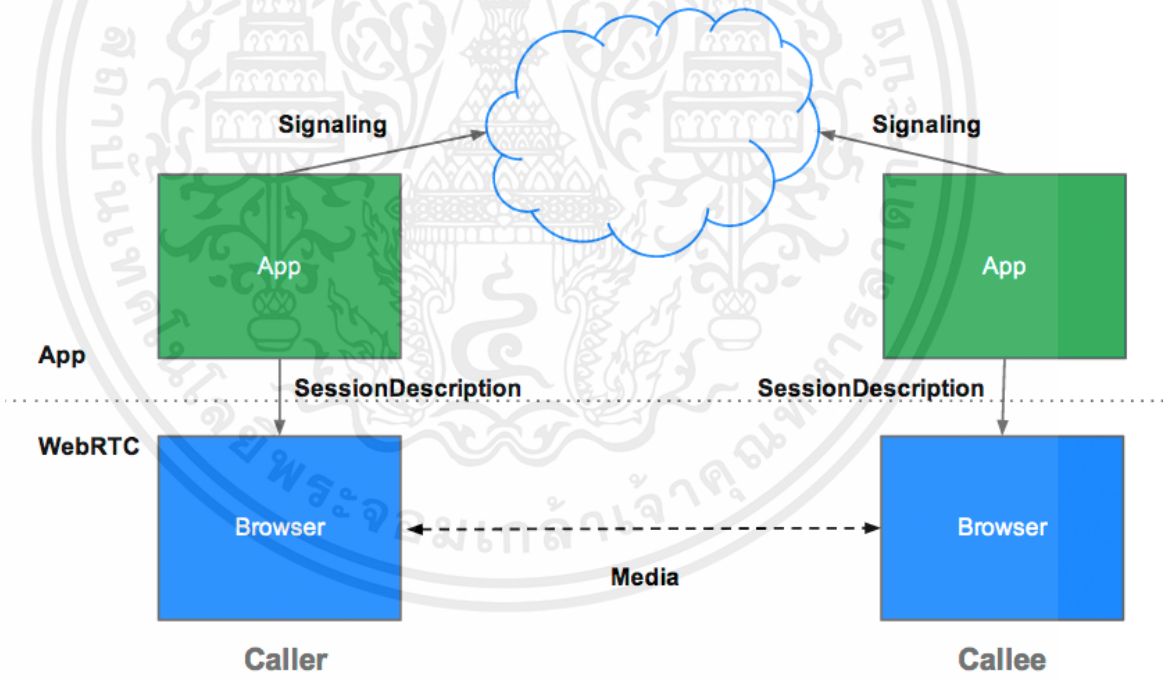
รูปที่ 2.5 WebRTC Data Pathway

2.3.6 Signaling Server

Signaling คือ หนึ่งในขั้นตอนการติดต่อระหว่างผู้ใช้เพื่อสร้าง call โดยผู้ใช้ต้องแลกเปลี่ยนข้อมูลดังนี้

- Session control message ใช้ในการเปิดและปิดการติดต่อนั้นๆ
- Error message ใช้ในการแจ้งความผิดพลาด
- Media metadata เช่น Code settings, Bandwidth และชนิดของ Media
- Key data เพื่อใช้ในการเปิดการติดต่อแบบ secure
- Network data เช่น IP ของ Host และ port

ขั้นตอนการ signaling ไม่ได้ถูกติดตั้งอยู่ใน WebRTC เนื่องจากลดการซ้ำและเพิ่ม compatibility โดย signaling ได้ถูกกล่าวไว้ใน JSEP (JavaScript Session Establishment Protocol) แล้ว



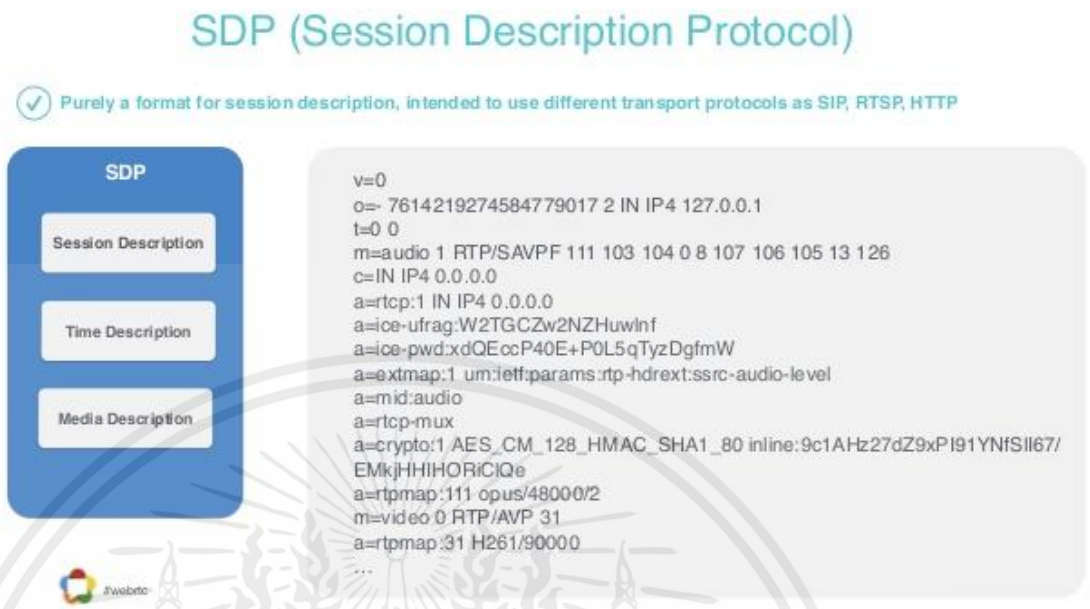
รูปที่ 2.6 JSEP's Signaling Architecture

2.3.7 Signaling และ RTCPeerConnection

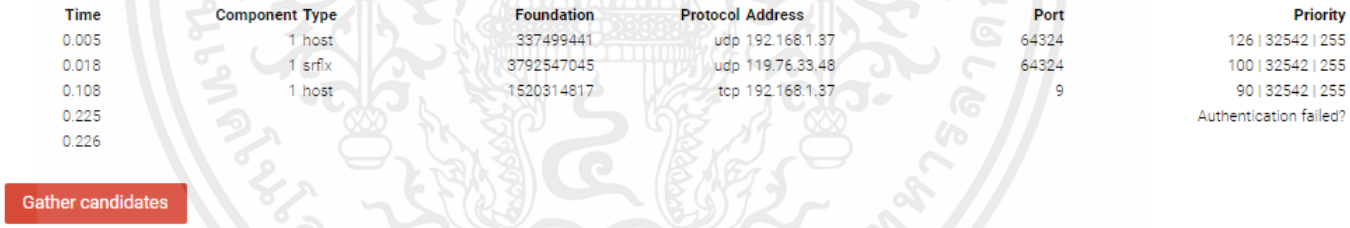
เมื่อนำ Signaling มารวมกับ WebRTC API จะทำให้ผู้ใช้สามารถแลกเปลี่ยนข้อมูลต่างๆกันได้

ตัวอย่างเช่น หาก A ต้องการติดต่อกับ B จะมีขั้นตอนดังนี้

- A สร้าง object WebRTCPeerConnection ด้วย function new RTCPeerConnection()
- A สร้าง offer (SDP Session Description) ด้วย function CreateOffer()
- A เรียกใช้ function setLocalDescription() หลังจากที่สร้าง offer แล้ว
- A ส่ง offer (Session Description ของ A) ให้ B ผ่าน Signaling server
- B สร้าง RTCPeerConnection และเรียก function setRemoteDescription() เพื่อนำข้อมูลของ A เข้ามา
- B เรียกใช้ function createAnswer() เพื่อให้ได้ Local Session Descriptionของตัวเองและ setLocalDescription() ของตนเอง
- B ส่ง answer (Session Description ของ B) ของตนเองผ่าน Signaling server
- A ได้รับข้อมูลของ B และเรียกใช้ function setRemoteDescription()
- เมื่อทำการส่ง Description เสร็จแล้ว Event candidate จะทำงานและส่งข้อมูล candidate ผ่าน Signaling server ให้อีกฝั่ง

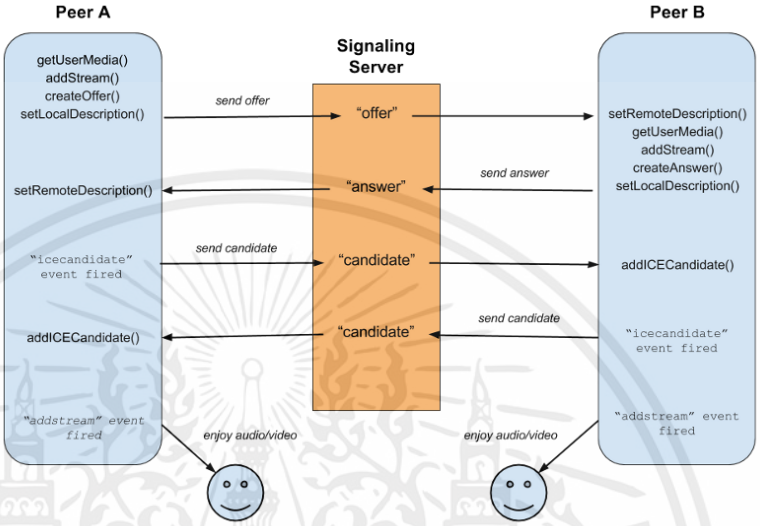


รูปที่ 2.7 ตัวอย่าง Session Description ของ WebRTC



รูปที่ 2.8 ตัวอย่าง ICE candidate ของ WebRTC

หากต้องการติดต่อกันโดยมีการส่งภาพ และเสียง ให้อีกฝั่งสามารถทำได้โดยการนำ getUserMedia มารวมกับ RTCPeerConnection ในขั้นตอนก่อนการสร้าง offer ให้กับอีกฝั่งโดยเมื่อเรียกใช้ function getUserMedia() แล้วเรียกใช้ function RTCConnection.AddStream(Stream) โดยมี Event Handler คือ onaddstream()



รูปที่ 2.9 ตัวอย่างขั้นตอนการทำงานของ RTCPeerConnection

2.4 NodeJS

NodeJS คือ Open Source, Cross-platform, JavaScript run-time environment ที่ทำงานโดยใช้ JavaScript NodeJS ทำให้ผู้พัฒนาสามารถใช้ JavaScript ในการเขียนโคดฝั่ง server ได้เพื่อให้เกิด dynamic web page ก่อนที่จะส่ง content ให้กับผู้ใช้งาน โดย NodeJS มีสโลแกนว่า “JavaScript Everywhere” ซึ่งเป็นการง่ายในการพัฒนาเนื่องจากผู้พัฒนาจะใช้เพียงภาษาเดียวในการพัฒนา ทั้งฝั่งผู้ใช้ และฝั่ง server

NodeJS มี architecture เป็น event-driven ซึ่งทำให้มีประสิทธิภาพในการทำงานแบบ asynchronous I/O, throughput และเพิ่มความสามารถในการ scale ใน web application ที่มี input และ output ที่สูง และเหมาะกับ web application แบบ realtime

1). V8

V8 เป็น open-source JavaScript Engine ของ Google ซึ่งเป็นส่วนทำงานหลักของ Google Chrome/Chromium Browsers โดยปรกติแล้ว JavaScript ใน Browser จะทำงานแบบ Interpret แต่ V8 จะแปลง JavaScript เป็นภาษาของเครื่องแล้วจึงทำงาน V8 ถูกเขียนด้วยภาษา C++

2). Libuv

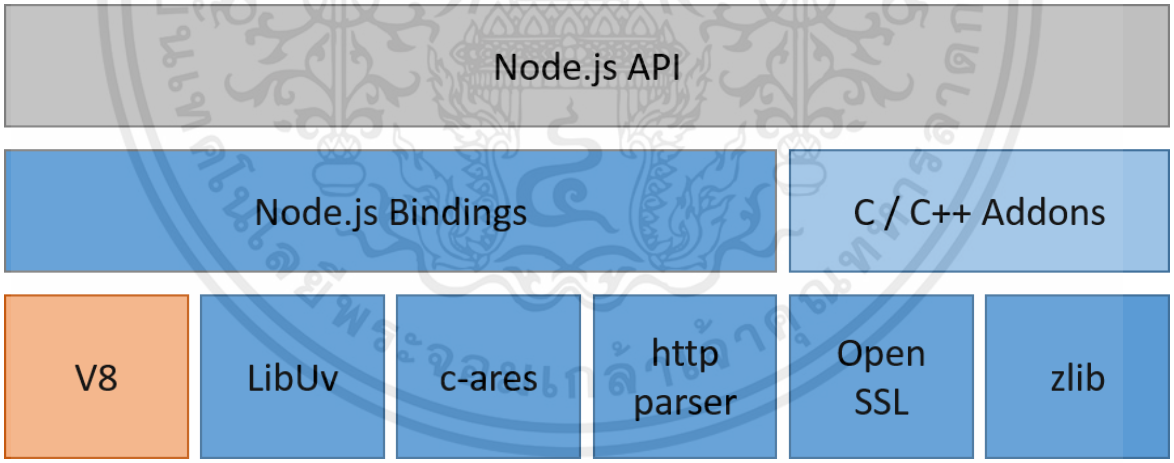
Libuv ถูกพัฒนาเพื่อให้การทำงานแบบ asynchronous I/O ที่รวมกับ asynchronous TCP และ UDP sockets, Event loop, file system read/write โดยถูกเขียนในภาษา C

3). Low level Components

ส่วนประกอบอื่นๆ เช่น c-ares, http parser, OpenSSL, zlib เป็นต้น

4). Binding

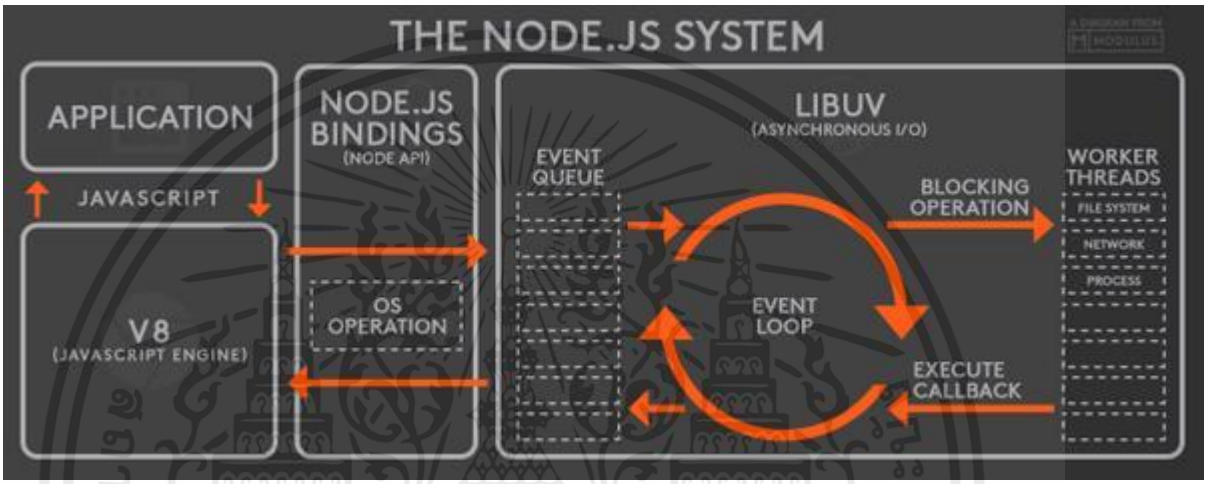
Binding คือ wrapper ที่เป็นส่วนกลางทำให้ภาษาต่างสามารถติดต่อกันได้ โดยจะเปลี่ยนภาษาหนึ่งไปเป็นภาษาหนึ่ง



รูปที่ 2.10 ส่วนประกอบของ NodeJS

NodeJS มีขั้นตอนการทำงานหลักๆคือ

- V8 compile โคดของ application
- โคดติดต่อกับ low level components ต่างๆ ผ่าน binding
- เมื่อ event เกิดขึ้นจะถึงเก็บไว้ใน event queue ตามลำดับที่เกิดขึ้น
- Event queue ส่ง function ไปให้ worker threads ทำงาน
- หากมี callback function จะถูกส่งจาก worker thread ไปยัง event queue



รูปที่ 2.11 ขั้นตอนการทำงานของ NodeJS

2.4.1 Socket.IO

Socket.IO เป็น JavaScript Library สำหรับการติดต่อสื่อสารแบบ realtime บน web application ทำให้สามารถเกิดการติดต่อสื่อสารแบบ2ทางระหว่าง ผู้ใช้ และ web server โดยแบ่งออกเป็น 2 ส่วนคือ library สำหรับรันบน Web Browser ของผู้ใช้ และ library ของ NodeJS สำหรับ web server อีกทั้ง Socket.IO เป็น Event-driven เช่นเดียวกับ NodeJS

Socket.IO โดยหลักแล้วใช้ WebSocket protocol และใช้ polling เป็น fallback option ถึงแม้ว่าจะสามารถใช้เป็นเพียงแค่ wrapper สำหรับ WebSocket แต่ Socket.IO มีลูกเล่นต่างๆ มากกว่า เช่น การกระจายข้อความไปยังหลาย socket, การเก็บข้อมูลต่างๆสำหรับแต่ละ socket และ asynchronous I/O

2.5 Electron

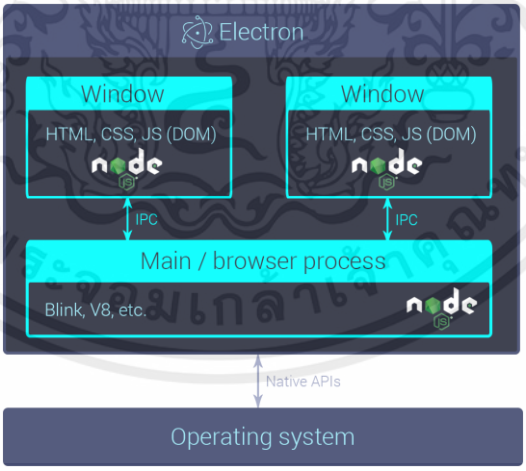
Electron (Atom shell) เป็น open-source framework ที่ถูกพัฒนาโดย GitHub ใช้ในการสร้าง Desktop GUI Applications โดยใช้ front end และ back end ของ web application (Web Technologies) ซึ่งคือ NodeJS runtime สำหรับ back end และ Chromium สำหรับ front end

Electron Architecture

ใน Electron process ที่รัน main script จะเรียกว่า Main Process ซึ่งเป็น process ที่แสดง GUI และสร้างหน้า web page ใน Electron จะมี main process เพียง1 process เท่านั้น

ในเมื่อ Electron ใช้ Chromium สำหรับแสดงหน้า web page แล้วดังนั้น Multi-process Architecture ของ Chromium ก็ถูกนำมาใช้ด้วยเช่นกัน โดยในแต่ละหน้า web page จะรัน process ของตัวเองเรียกว่า renderer process

โดยปรกติแล้วใน web Browser จะไม่ถูกอนุญาตให้เข้าถึง Native sources แต่ Electron สามารถใช้ NodeJS APIs ทำให้สามารถเกิดการติดต่อกันในระดับ low level ได้ทั้ง renderer process และ main process เช่น readfileSync หรือ readdirSync เป็นต้น และยังสามารถใช้ NodeJS modules ได้Electron สามารถติดต่อกันระหว่าง main process และ renderer process ผ่าน modules ของ ipcRenderer และ ipcMain สำหรับการส่งข้อมูลต่างๆ



รูปที่ 2.12 สถาปัตยกรรมของ Electron

2.6 MySQL

MySQL เป็น open-source Relational Database Management System (RDBMS) ซึ่งถูกเขียนด้วยภาษา C และ C++ ส่วนของ SQL parser เขียนด้วย yacc ซึ่ง MySQL สามารถทำงานได้บนหลาย platform เช่น Linux, MacOS, Windows เป็นต้น

MySQL สามารถ built และ install จาก source code ได้ โดดน Linux ส่วนใหญ่สามารถ download ผ่าน package management system และ install ได้ง่าย จึงทำให้ MySQL เป็นที่นิยมในกลุ่ม Developer หรือในองค์ขนาดเล็กถึงขนาดกลาง

บทที่ 3

วิธีการดำเนินงาน

3.1 การวางแผนและการเตรียมการ

ในการดำเนินงานและการจัดทำแอปพลิเคชันเพื่อการติดต่อสื่อสารแบบวิดีโอและแบบเสียง จำเป็นต้องมีการเตรียมการดังต่อไปนี้

3.1.1 ศึกษาข้อมูลเกี่ยวกับหัวข้อที่เกี่ยวข้อง

1. ศึกษา Web Technologies เนื่องจาก requirement ต้องการการติดต่อสื่อสารกัน โดยสามารถผ่าน Web browser ดังนั้นการศึกษา Web Technologies เช่น HTML, JavaScript จึงมีความจำเป็น อีกทั้งเทคโนโลยีที่นำมาใช้ในแอปพลิเคชันคือ WebRTC ซึ่งรองรับการทำงานบน Browser

2. ศึกษา WebRTC APIs ซึ่งเป็นส่วนในการทำงานหลักของแอปพลิเคชัน เป็นส่วนการทำงานของการติดต่อสื่อสารระหว่างผู้ใช้งาน และการแสดงภาพและเสียงของผู้ใช่ ซึ่งสามารถนำมาใช้ร่วมแอปพลิเคชันใน Android และ IOS ได้ การทำงานของ WebRTC จะพยายามให้ผู้ใช้ติดต่อกันแบบ Peer to Peer ซึ่งจะลดการล่าช้า (Latency) และลดการทำงานของ server รวมถึงการศึกษาความต้องการของ WebRTC เช่น STUN server, TURN server และ Signaling server

3. ศึกษา NodeJS และ Socket.IO เป็นส่วนการทำงานของ server โดย Socket.IO สนับสนุนการทำงานแบบ asynchronous I/O ซึ่งเหมาะสมสำหรับการติดต่อสื่อสาร อีกทั้งยังสามารถทำให้แบ่งเป็นห้องสนทนา (room) เพื่อรองรับการสนทนาแบบกลุ่ม (n:n) ได้

4. ศึกษา Electron เพื่อใช้ในการสร้าง Desktop Application โดยยังสามารถใช้ Web Technologies ต่างๆได้ โดยเฉพาะ WebRTC เนื่องจากเป็นส่วนหลักของแอปพลิเคชัน

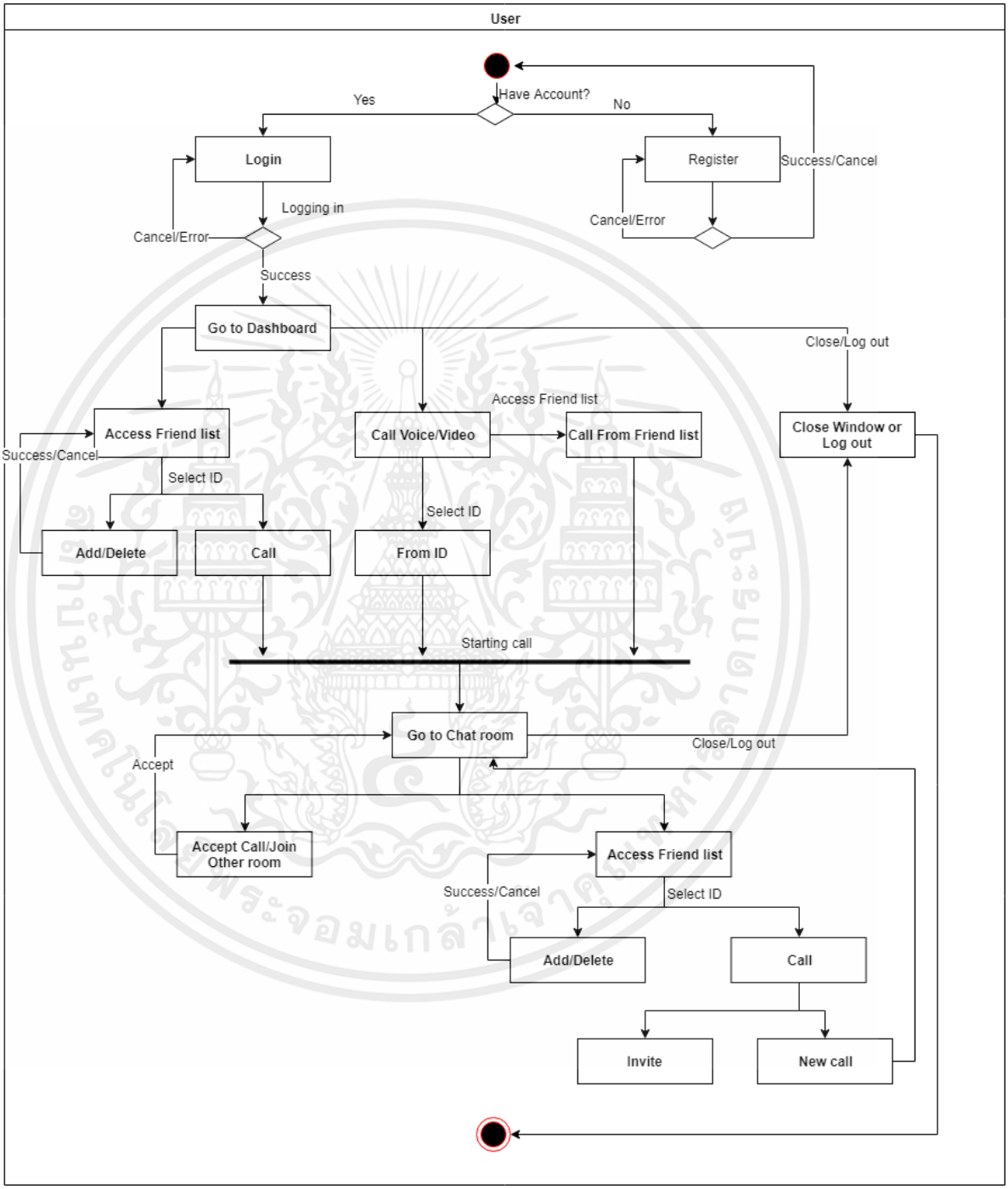
5. ศึกษาระบบฐานข้อมูล และ MySQL เพื่อใช้สำหรับเก็บข้อมูลของผู้ใช้งาน, กลุ่ม และรายชื่อเพื่อนของผู้ใช้

3.1.2 การออกแบบระบบ

1. ออกแบบการใช้งานของผู้ใช้ (Usecase diagram) เพื่อให้ทราบถึง functionการทำงานหลักที่ต้องมีในแอปพลิเคชัน โดยจะประกอบด้วยส่วนหลัก 4 ส่วนคือ Login, Voice chat, Video chat และ Friend list

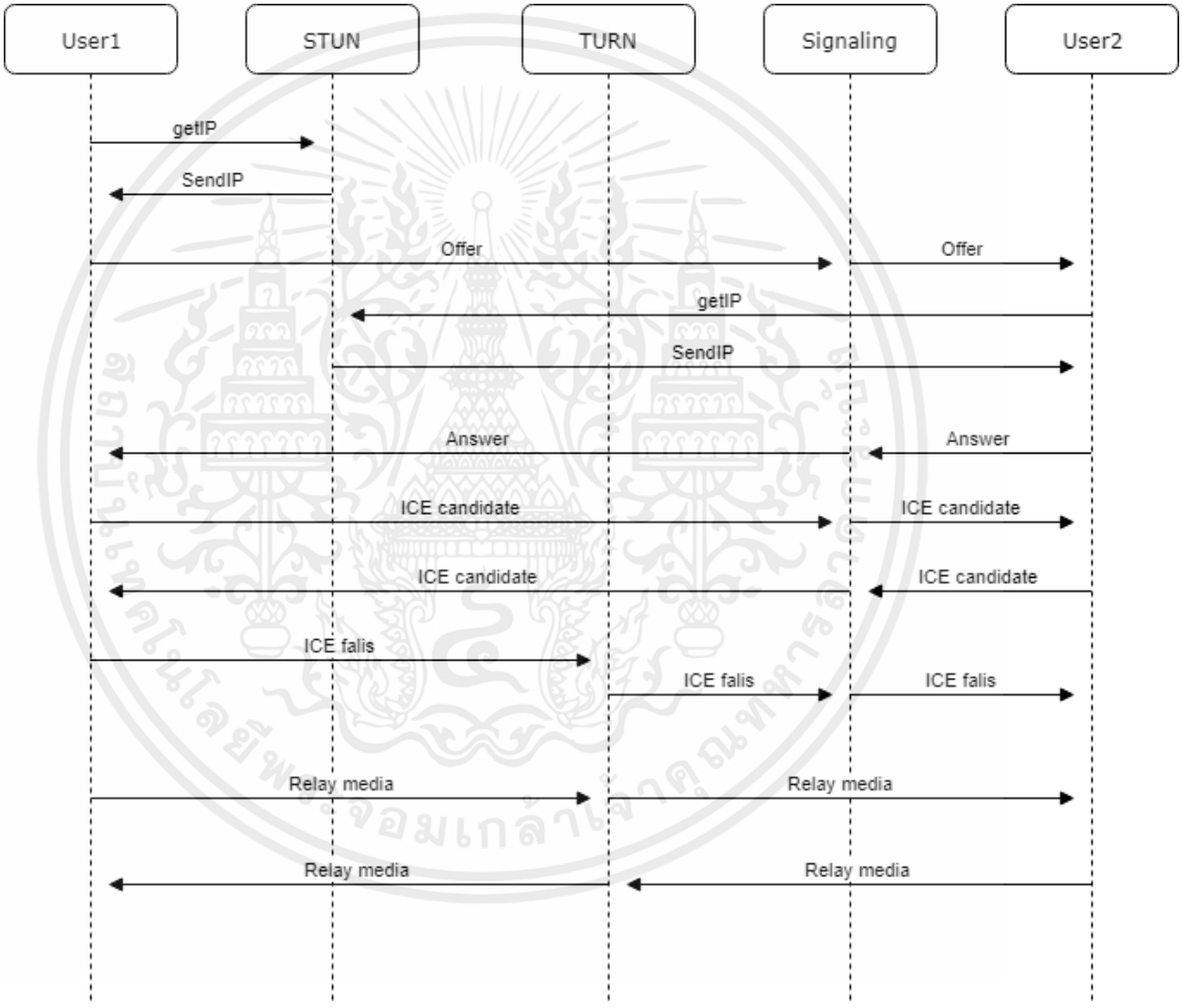
รูปที่ 3.1 Usecase Diagram ของ user

2. ออกแบบแผนภาพกิจกรรม (Activity Diagram) เพื่ออธิบายการไหลของการทำงาน (Workflow) และทราบถึงกระบวนการขั้นตอนที่ควรจะเป็นของแอปพลิเคชัน

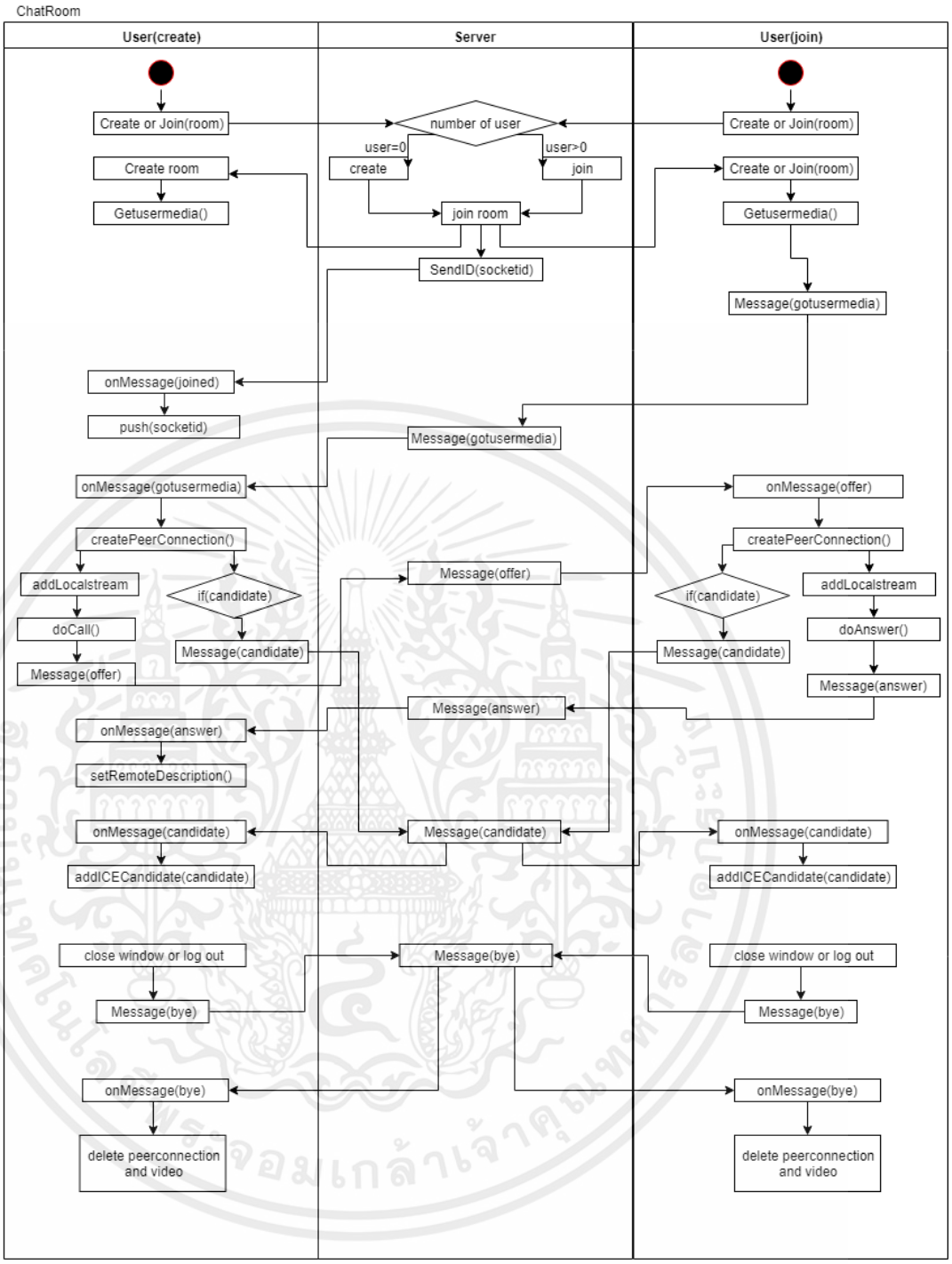


รูปที่ 3.2 Activity Diagram ของ user

3. ออกแบบการทำงานของ function หลักของแอปพลิเคชัน ซึ่งคือการทำ Voice/Video chat โดยมีผู้ใช้งาน2คน ประกอบด้วย เจ้าของห้อง (ผู้สร้างห้อง) และผู้เข้าร่วมห้อง ซึ่งในกรณีที่มีมากกว่า 2 คน คนที่เข้าร่วมใหม่ จะเป็นผู้เข้าร่วม และคนที่อยู่ในห้องอยู่แล้วจะเป็นเจ้าของห้อง โดยไม่มีการทำงานในส่วนการสร้างห้อง และจะมีการใช้ socketID จาก Socket.IO ของ Signaling server เพื่อใช้ในการระบุเป้าหมายในการส่งข้อความนั้นๆ



รูปที่ 3.3 Sequence Diagram การทำงานของ Voice/Video chat 1:1



รูปที่ 3.4 Workflow ของ function การทำงานของ Voice/Video chat 1:1

3.1 โดยการทำงานของ function หลัก Voice/Video chat 1:1 มีขั้นตอนการทำงานดังนี้

1. เจ้าของห้องได้เข้าห้อง หน้า web page จะทำการรับข้อมูลภาพและเสียงตามที่ผู้ใช้งานได้อนุญาต Browser ด้วย function getUserMedia()

2. ผู้ใช้อีกคนเข้าร่วมห้อง รับข้อมูลภาพและเสียง หลังจากนั้นจึงส่งข้อความไปให้ Signaling server

3. Signaling server ส่งข้อความให้เจ้าของห้องเพื่อให้เจ้าของห้องรู้ว่ามี ผู้ใช้งานอื่นเข้าร่วมห้องแล้ว และส่ง socketID ของผู้เข้าร่วม

4. เจ้าของห้องเก็บ socketID ของผู้เข้าร่วมไว้ใน array และสร้าง object RTCPeerConnection พร้อมกับ AddStream เข้าไปใน object ด้วย MediaStream ที่ได้จากขั้นตอนที่ 1 หลังจากที่สร้าง object เสร็จแล้วจะทำงาน function createOffer() พร้อม setLocalDescription และสร้าง Event Handler คือ onicecandidate เพื่อใช้ในการส่ง ICE candidate ให้อีกฝั่ง และ onaddstream จะทำงานเมื่อได้รับ stream จากอีกฝั่ง หลังจากนั้นจึงส่งข้อความ, socketID และ Session Description ไปที่ Signaling server

5. Signaling server ส่งข้อความ, socketID และ Session Description ไปยังผู้เข้าร่วม

6. เมื่อได้รับข้มความจาก Signaling server แล้วจะสร้าง object RTCPeerConnection ขึ้นมา AddStream ของตนเองที่ได้มาจากขั้นตอนที่ 2 สร้าง Event Handler คือ onicecandidate และ onaddstream ใช้งาน function setRemoteDescription ด้วย Session Description ที่ได้รับจาก Signaling server พร้อมทั้งใช้งาน function createAnswer() และ setLocalDescription หลังจากนั้นจึงส่งข้อความ, Session Description ไปยัง Signaling server

7. Signaling server ส่งข้อความไปยังเจ้าของห้อง

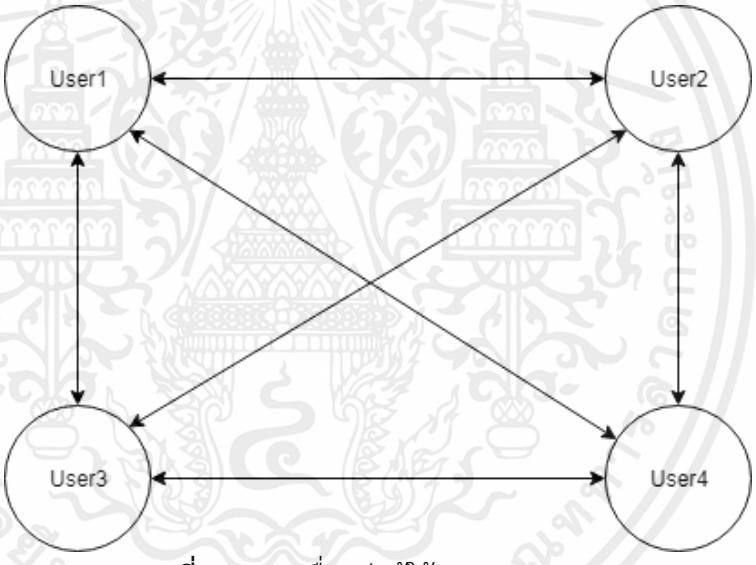
8. เจ้าของห้องนำ Session Description ที่ได้รับจาก Signaling server มา setRemoteDescription

9. เมื่อทั้ง 2 ฝั่งได้ setLocalDescription และ setRemoteDescription แล้ว Event Handler onicecadidate จะทำงานพร้อมส่ง ice candidate ไปให้ Signaling server เพื่อผ่านข้อความต่อไปให้กับอีกฝั่ง

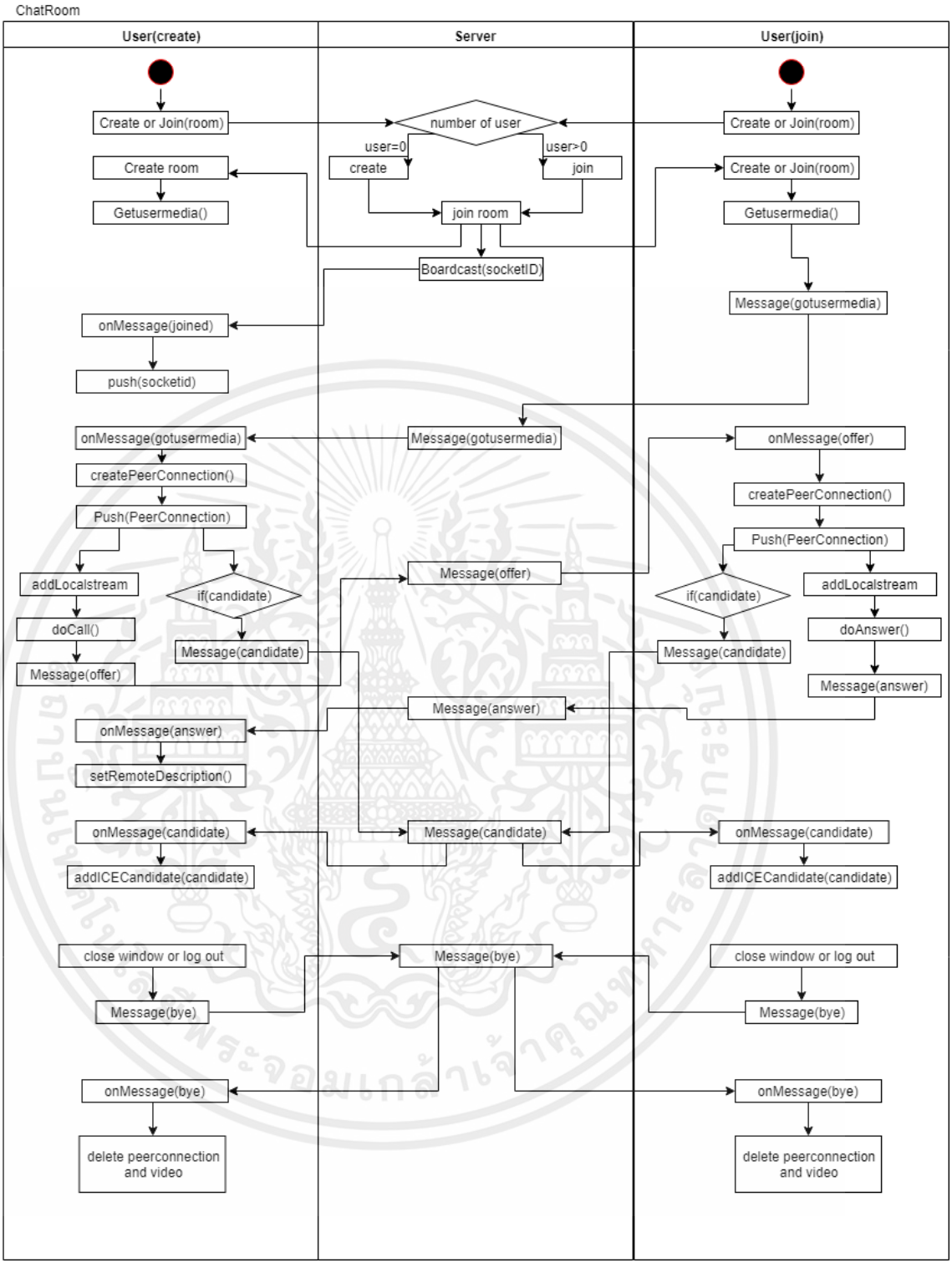
10. เมื่อทั้ง 2 ฝั่งติดต่อกันโดยไม่มีปัญหา Event handler onaddstream จะทำงาน หากมีปัญหาจะ ทำการ Relay media ทั้งหมดไปยัง TURN server เพื่อผ่าน media ไปให้อีกฝั่ง

3.2 การทำงานหลักของ function หลัก Voice/Video chat n:n

โดยหลักการทำงานของ function หลักใน Video/Voice chat แบบ n:n จะมีหลักการเชื่อมแต่ละผู้ใช้งานแบบ Mesh เพื่อลดการทำงานของ server ดังรูปด้านล่าง



รูปที่ 3.5 การเชื่อมต่อผู้ใช้งานแบบ Mesh



รูปที่ 3.6 Workflow ของ function หลักใน Voice/Video chat n:n

การทำงานหลักของ Voice/Video chat แบบ n:n จะมีการทำงานคล้ายกับการทำงานแบบ 1:1 โดยมีข้อแตกต่างกันดังนี้

1. เมื่อมีผู้เข้าร่วมเข้ามา Signaling server จะเปลี่ยนจากการส่งข้อความให้แค่เจ้าของห้องเป็น กระจายข้อความ (Broadcast) ไปยังทุกคนที่อยู่ในห้อง

2. เมื่อสร้าง object RTCPeerConnection ขึ้นมาแล้วจะเก็บไว้ใน array โดยไม่จำเป็นต้องเรียงลำดับของผู้ใช้

3. เมื่อได้รับ socketID จาก Signaling server แล้วจะเก็บ socketID นั้นไว้ใน array โดยไม่จำเป็นต้องเรียงตามลำดับของผู้ใช้

4. ออกแบบฐานข้อมูล การเก็บข้อมูลของแอปพลิเคชันนี้มี 4 ตาราง ประกอบด้วย

- User ซึ่งเก็บข้อมูลของผู้ใช้งาน คือ Account ชื่อของผู้ใช้งาน, Password รหัสของผู้ใช้งาน , IsOnline เก็บสถานะของผู้ใช้งานซึ่งจะเก็บเป็น binary number 0 คือไม่ออนไลน์ และ 1 คือออนไลน์อยู่, Last Activity เก็บการเคลื่อนไหวล่าสุดของผู้ใช้ และ Token ใช้สำหรับเก็บ token ของผู้ใช้เพื่อใช้ในการทำงานของระบบ push notification

- Friend เก็บข้อมูลรายชื่อเพื่อนของผู้ใช้ ซึ่งจะประกอบด้วย ชื่อผู้ใช้ทั้ง 2 คนที่เป็นเพื่อนกัน

- Group เก็บข้อมูลกลุ่มของผู้ใช้งานโดยประกอบด้วย Groupname ชื่อของกลุ่มที่ผู้ใช้งานตั้งขึ้น, GroupID หมายเลขของกลุ่มซึ่งจะเป็น Auto Increment, Creator เก็บ Account ของผู้ใช้งานที่เป็นคนสร้างกลุ่มขึ้น- Groupmember เก็บข้อมูลสมาชิกของกลุ่มนั้นๆ ประกอบด้วย GroupID หมายเลขของกลุ่มซึ่งมาจากตาราง Group และ Account เก็บสมาชิกของกลุ่ม มาจากตาราง User

รูปที่ 3.7 ภาพรวมของฐานข้อมูลของแอปพลิเคชัน

5.การออกแบบ User Interface ประกอบด้วย 4 ส่วนคือ Login, Dashboard และ Chat room

รูปที่ 3.8 User Interface ส่วน Login

รูปที่ 3.9 User Interface ส่วน Dashboard

รูปที่ 3.10 User Interface ส่วน Chat room

3.2 การพัฒนาและการติดตั้งระบบ

การพัฒนาแอปพลิเคชัน Web Browser และ Desktop

การพัฒนาแอปพลิเคชันบน Web Browser ใช้ Web Technologies ในการพัฒนาซึ่งประกอบด้วย HTML, CSS, JavaScript และ JQuery โดย

- HTML เป็นโครงของ document

- CSS ใช้ในการตกแต่ง จัดหน้า และจัด element ต่างๆ

- JavaScript ใช้ในการเขียน function การทำงานหลักของแอปพลิเคชัน โดยมีการเรียกใช้ APIs ต่างๆ เช่น WebRTC เพื่อใช้ในการติดต่อสื่อสาร, Socket.IO เพื่อใช้ในการติดต่อ server และใช้ในการส่งข้อความ

- JQuery ใช้สำหรับส่ง request ให้กับ server โดยใช้ AJAX และใช้สำหรับเพิ่มความสะดวกสบาย ความหลากหลายและลูกเล่นในการเขียนโคด

ในการพัฒนา Desktop Application ใช้เทคโนโลยีเช่นเดียวกับ Web Browser แต่ต่างกันโดย Desktop Application จะใช้ Electron ในการสร้าง GUI และสามารถใช้ modules ต่างๆจาก NodeJS เข้ามาร่วมในแอปพลิเคชันได้ ทำให้เพิ่มความสามารถของแอปพลิเคชันมากขึ้น ในการพัฒนาได้ใช้ HTML, CSS, JavaScript, JQuery และ NodeJS

บทที่ 4

ผลการดำเนินงาน

หมายเหตุ : เนื่องจากรายละเอียดนี้เป็นความลับเฉพาะของบริษัท จึงไม่สามารถเปิดเผยได้

บทที่ 5

บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

แอปพลิเคชันการติดต่อสื่อสารแบบภาพและเสียง ถูกพัฒนาโดยสามารถทำงานโดยมีการ Login, ระบบเพื่อนและกลุ่ม, การเพิ่มเพื่อน, การเพิ่มสมาชิกกลุ่ม, การลบสมาชิกออกจากกลุ่มโดยทำได้เฉพาะผู้สร้างกลุ่ม, การค้นหาเพื่อนและสมาชิกในกลุ่ม, การสนทนาแบบเดี่ยวและแบบกลุ่ม, การชวนเข้าร่วมสนทนา, การสนทนาแบบเสียงและแบบวิดีโอ โดยการทำงานสามารถทำงานได้โดยข้ามแพลตฟอร์ม (Cross Platform) ระหว่าง Web Browser/Desktop application และ Desktop application/Web Browser, ระบบ push notification เมื่อผู้ใช้ได้รับคำชวนเข้าห้องสนทนา โดยการทำงานทั้งหมดของแอปพลิเคชันสามารถใช้งานได้จริง และสามารถนำไปพัฒนาต่อได้

5.2 สรุปผลการทดสอบแอปพลิเคชัน

5.2.1 สรุปผลการทดสอบแอปพลิเคชันบน Web Browser

จากผลการทดสอบแอปพลิเคชันบน Web Browser สามารถสรุปได้ว่าการทำงานต่างๆ เช่น การเพิ่มเพื่อน, การเพิ่มสมาชิกกลุ่ม, การลบมสมาชิกกลุ่ม, การชวนเข้าห้องสนทนา สามารถทำงานได้อย่างถูกต้อง แต่มีข้อผิดพลาดคือ มี output ที่แสดงให้ผู้ใช้งานได้ไม่ถูกต้องซึ่งได้รับการแก้ไขใน version v1.02 แล้ว การทำงานในหน้า chat ซึ่งเป็นหน้าที่มีการทำงานหลักสามารถทำงานได้อย่างถูกต้อง ผู้ใช้งานสามารถสนทนาได้ทั้งแบบเดี่ยวและแบบกลุ่มโดยไม่เกิดข้อผิดพลาดขึ้น สามารถสนทนาทั้งแบบวิดีโอและเสียงได้ และสามารถใช้งานทั้งแบบภายในแพลตฟอร์มเดียวกันและแบบข้ามแพลตฟอร์มได้โดยไม่มีปัญหาใดๆ

5.2.2 สรุปผลการทดสอบแอปพลิเคชันบน Desktop

จากการทดสอบแอปพลิเคชันบน Desktop ได้ผลเช่นเดียวกับแอปพลิเคชันบน Web Browser โดยการทำงานต่างๆเช่น การเพิ่มสมาชิก, การเพิ่มสมาชิกกลุ่ม, การลบสมาชิกกลุ่ม , การค้นหาสมาชิกกลุ่ม, การชวนเข้าห้องสนทนา สามารถทำงานได้อย่างถูกต้อง แต่ยังเกิดข้อผิดพลาดเดียวกับ Web Browser คือ ยังมีบาง output ที่แสดงให้ผู้ใช้งานยังแสดงผลได้ไม่ถูกต้อง ซึ่งได้รับการแก้ไขใน version v08 แล้ว การทำงานในหน้า chat สามารถทำงานได้อย่างถูกต้อง ผู้ใช้งานสามารถสนทนาได้ทั้งแบบเดี่ยวและแบบกลุ่ม สามารถสนทนาได้ทั้งแบบวิดีโอและแบบเสียง และสามารถใช้งานทั้งแบบภายในแพลตฟอร์มเดียวกันและแบบข้ามแพลตฟอร์มโดยไม่มีปัญหาใดๆเกิดขึ้น

5.3 ข้อจำกัดในการพัฒนา

จากการพัฒนาแอปพลิเคชันที่ได้กล่าวมา ยังมีข้อที่ผู้พัฒนาเห็นว่าเป็นข้อจำกัดอยู่ดังนี้

1. เนื่องจากการติดต่อของผู้ใช้งานในหน้า chat นั้นเป็นการติดต่อแบบ Peer to Peer ถึงแม้ว่าการติดต่อจะรวดเร็วแต่มีข้อเสียคือ จะใช้ Bandwidth มากในการติดต่อ
2. เนื่องจากการติดต่อของผู้ใช้งานแบบกลุ่ม ใช้การรูปแบบการติดต่อแบบ Mesh คือ ผู้ใช้ทุกคนมีการติดต่อกันทั้งหมด ทำให้เมื่อมีการติดต่อมากยิ่งขึ้น จะทำให้การใช้ทรัพยากรและ Bandwidthมายิ่งขึ้นตามไปด้วย
3. ระบบ push notification เมื่อผู้ใช้ได้รับคำชวนมากกว่า 1 คำชวน จะทำให้การแสดงผลแสดงเพียงแค่1คำชวนเท่านั้น
4. เมื่อผู้ใช้งานเข้า/ออก ห้องสนทนาพร้อมกันในเวลาสั้นๆ จะทำให้เกิดข้อผิดพลาดในการทำงาน

5.4 ข้อเสนอแนะในการพัฒนาระบบ

จากข้อจำกัดที่ได้กล่าวมาข้างต้น ทางผู้พัฒนาเห็นว่าบางข้อจำกัดสามารถแก้ไขได้ด้วยการพัฒนาต่อดังนี้

1. จากข้อจำกัดในข้อ 2 สามารถแก้ไขได้โดยใช้การติดต่อในรูปแบบอื่น เช่นการติดต่อแบบ star หรือการติดต่อแบบผสมทั้ง mesh และ star หรือใช้ server เข้ามาช่วยในการส่งข้อมูล
2. จากข้อจำกัดในข้อ 3 สามารถแก้ไขได้โดยการใช้ระบบฐานข้อมูลเข้ามาช่วย
3. จากข้อจำกัดในข้อ 4 สามารถแก้ไขได้โดยให้ server จำกัดการเข้าห้องเช่น ให้ server จำกัดการเข้าในแต่ละห้อง เป็น 1การเข้าต่อ 2 วินาที เป็นต้น

เอกสารอ้างอิง

[1] **WebRTC**. [ออนไลน์]. สืบค้นจาก: https://webrtc.org/. เข้าถึงเมื่อวันที่ 28 มีนาคม 2561.

[2] Sam Dutton. 2557. **Getting Started with WebRTC.** [ออนไลน์]. สืบค้นจาก: https://www.html5rocks.com/en/tutorials/webrtc/basics/. เข้าถึงเมื่อวันที่ 2 เมษายน 2561.

[3] Eric Bidelman. 2559. **Capturing Video and Audio in HTML5.** [ออนไลน์]. สืบค้นจาก: https://www.html5rocks.com/en/tutorials/getusermedia/intro/. เข้าถึงเมื่อวันที่ 2 เมษายน 2561.

[4] **Realtime Communications with WebRTC.** [ออนไลน์]. สืบค้นจาก: https://codelabs.developers.google.com/codelabs/webrtc-web/. เข้าถึงเมื่อวันที่ 3 เมษายน 2561.

[5] Sam Dutton. 2556. **WebRTC in the real world: STUN, TURN and signaling.** [ออนไลน์]. สืบค้นจาก: https://www.html5rocks.com/en/tutorials/webrtc/infrastructure/. เข้าถึงเมื่อวันที่ 18 เมษายน 2561.

[6] Eric Shepherd. 2560. **WebRTC API.** [ออนไลน์]. สืบค้นจาก: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API. เข้าถึงเมื่อวันที่ 20 เมษายน 2561.

[7] Chad Hart. 2559. **getUserMedia resolutions III – constraints unleashed.** [ออนไลน์]. สืบค้นจาก: https://webrtchacks.com/getusermedia-resolutions-3/. เข้าถึงเมื่อวันที่ 23 เมษายน 2561.

[8] Muaz Khan. 2556. **WebRTC Experiment.** [ออนไลน์]. สืบค้นจาก: https://github.com/muaz-khan/WebRTC-Experiment**.** เข้าถึงเมื่อวันที่ 26 เมษายน 2561.

[9] **NodeJs tutorials**. [ออนไลน์]. สืบค้นจาก: https://www.tutorialspoint.com/nodejs/. เข้าถึงเมื่อวันที่ 30 เมษายน 2561.

[10] **Socket.IO tutorials.** [ออนไลน์]. สืบค้นจาก: https://www.tutorialspoint.com/socket.io/. เข้าถึงเมื่อวันที่ 1 พฤษภาคม 2561.

[11] **Electron**. [ออนไลน์]. สืบค้นจาก: https://electronjs.org/. เข้าถึงเมื่อวันที่ 29 พฤษภาคม 2561.

[12] **Electron Documentation**. [ออนไลน์]. สืบค้นจาก: https://electronjs.org/docs. เข้าถึงเมื่อวันที่ 29 พฤษภาคม 2561.

[13] **Electron Application Architecture**. [ออนไลน์]. สืบค้นจาก: https://electronjs.org/docs/tutorial/application-architecture. เข้าถึงเมื่อวันที่ 29 พฤษภาคม 2561.

[14] Cameron Nokes. 2559. **Deep dive into Electron's main and renderer processes.** [ออนไลน์]. สืบค้นจาก: https://medium.com/cameron-nokes/deep-dive-into-electrons-main-and-renderer-processes-7a9599d5c9e2. เข้าถึงเมื่อวันที่ 29 พฤษภาคม 2561.

[15] CodeDraken [นามแฝง]. 2560. **Electron Basics and Fundamentals.** [ออนไลน์]. สืบค้นจาก: https://codeburst.io/electron-basics-and-fundamentals-b85b23aa611d. เข้าถึงเมื่อวันที่ 30 พฤษภาคม 2561.

[16] Matt Gaunt. 2560. **Service Workers: An Introduction.** [ออนไลน์]. สืบค้นจาก: https://developers.google.com/web/fundamentals/primers/service-workers/. เข้าถึงเมื่อวันที่ 18 มิถุนายน 2561.

[17] **NodeJS MySQL**. [ออนไลน์]. สืบค้นจาก: https://www.w3schools.com/nodejs/nodejs_mysql.asp.

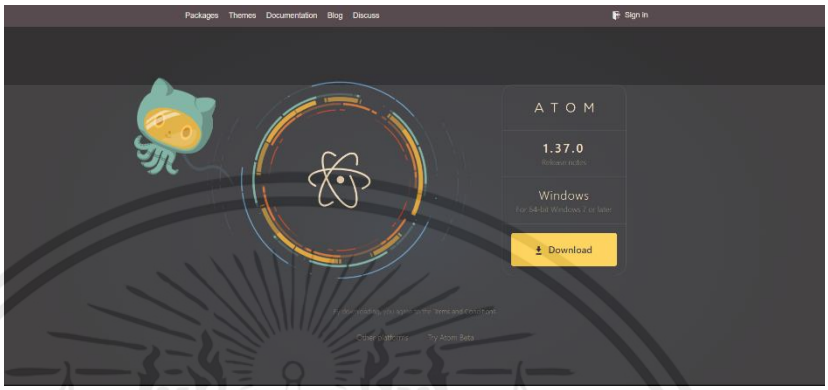
เข้าถึงเมื่อวันที่ 25 มิถุนายน 2561.

ภาคผนวก

ภาคผนวก ก

1. การติดตั้งโปรแกรม Atom Editor

1. เข้าเว็บไซต์ https://atom.io/ เพื่อดาวน์โหลด



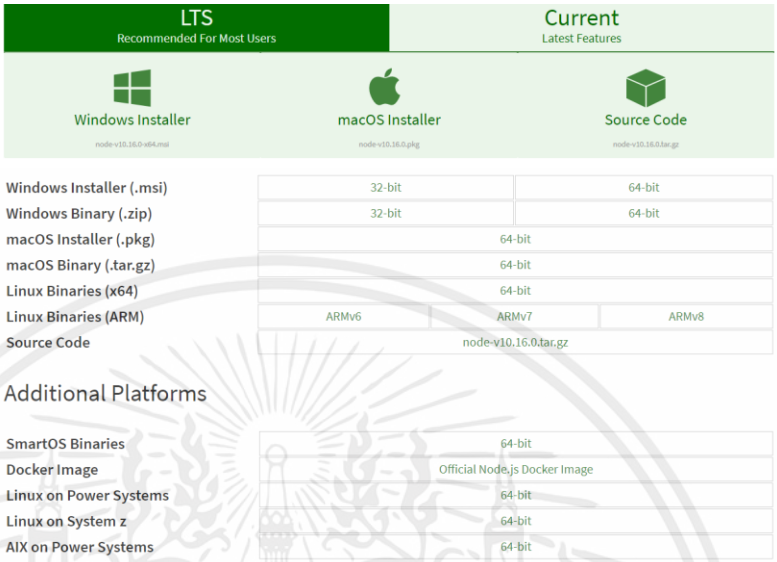
รูปที่ ก.1 หน้าเว็บไซต์สำหรับดาวน์โหลด Atom Editor

2. หลังจากดาวน์โหลดเสร็จ เปิด AtomSetup.exe ที่ดาวน์โหลดมา โดยหลังจากที่เปิดโปรแกรมจะทำการติดตั้งโดยอัตโนมัติ

รูปที่ ก.2 หน้าต่างของโปรแกรม Atom Editor

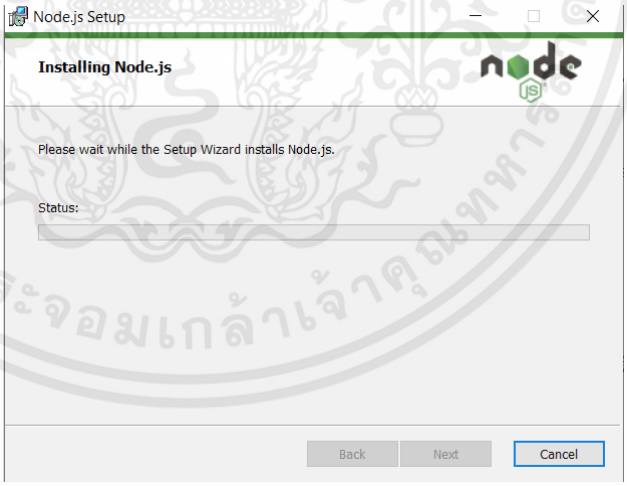
2. การติดตั้ง NodeJS

1. เข้าเว็บไซต์ https://nodejs.org/en/download เพื่อดาวน์โหลด โดยเลือดการดาวน์โหลดให้ตรงกับแพลตฟอร์มที่ใช้



รูปที่ ก.3 เว็บไซต์สำหรับดาวน์โหลดโปรแกรมติดตั้ง NodeJS

2. เปิดโปรแกรมติดตั้งที่ดาวน์โหลดมาเพื่อทำการติดตั้ง NodeJS



รูปที่ ก.4 หน้าต่างติดตั้ง NodeJS

3. การติดตั้ง Web server และ Signaling server

Web server และ Signaling ที่ใช้งานคือ NodeJS โดยใช้งานบน Linux เพื่อรองรับการทำงานทั้ง web server และ TURN server สำหรับ WebRTC และจำเป็นต้องทำให้ server ที่ใช้งานเป็น HTTPS เพื่อรับรองการทำงานของ WebRTC ด้วยเช่นกัน

1.การติดตั้ง NodeJS บน Linux สามารถติดตั้งผ่านทาง package manager system ได้ ตัวอย่างเช่นใน CentOS สามารใช้คำสั่ง

curl --silent --location https://rpm.nodesource.com/setup_10.x | sudo bash -

เพื่อใช้ในการ download package ของ NodeJS และสามารถใช้คำสั่ง sudo yum -y install nodejs ในการติดตั้ง NodeJS

2. การใช้งาน Socket.IO มีขั้นตอนดังนี้

- download package Socket.IO ผ่าน npm (Node Packager manager) ผ่านคำสั่ง

Npm install socket.io

- เมื่อติดตั้ง package เสร็จแล้วสามารถใช้งาน Socket.IO ได้บน server โดย

```
var httpserver =require('http');// createServer(app)
httpserver.createServer(function (req, res) {
   res.writeHead(301, { "Location": "https://" + req.headers['host'] + req.url });
   res.end();
}).listen(80);
var server = require('https').createServer(credentials, app);
var io = require('socket.io')(server);
```

รูปที่ ก.5 ตัวแปร io คือตัวแปรที่ได้จากการสร้าง Socket.IO บน server

4. การติดตั้ง TURN server และ MySQL

เนื่องจาก TURN server และ MySQL เป็น Open-source จึงสามารถติดตั้งได้โดยไม่มีข้อจำกัดใดๆ โดยมีวิธีติดตั้งดังนี้

1. download และติดตั้ง mysql openssl และ gcc ผ่าน package manager

yum install -y make gcc cc gcc-c++ wget

yum install -y openssl-devel libevent libevent-devel mysql-devel mysql-server

2. download และติดตั้ง libevent modules

Wgethttps://github.com/downloads/libevent/libevent/libevent-2.0.21-stable.tar.gz

tar xvfz libevent-2.0.21-stable.tar.gz

cd libevent-2.0.21-stable && ./configure

make && make install && cd ..

3. download และติดตั้ง TURN modules

Wget http://turnserver.open-sys.org/downloads/v3.2.3.8/turnserver-3.2.3.8.tar.gz

tar -xvzf turnserver-3.2.3.8.tar.gz

cd turnserver-3.2.3.8 && ./configure

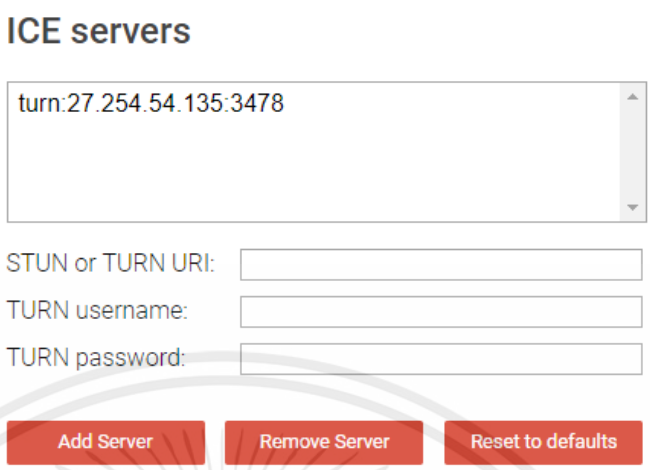
make && make install

4. ตั้งค่าต่างๆใน config.cnf ตามต้องการเช่น port, IP, fingerprint, lt-cred (Long-term Credentials mechanism) จำเป็นต่อการใช้ WebRTC APIs

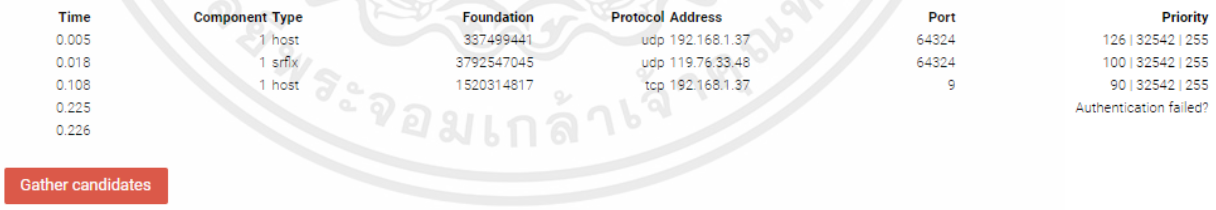
5. run TURN server ผ่านคำสั่ง turnserver &

6. สามารถตรวจสอบการทำงานของ turn server ผ่าน

https://webrtc.github.io/samples/src/content/peerconnection/trickle-ice/



รูปที่ ก.6 ตัวอย่างการใช้งานการตรวจสอบ TURN server



Time	Component Type	Foundation	Protocol Address	Port	Priority
0.005	1 host	337499441	udp 192.168.1.37	64324	126 | 32542 | 255
0.018	1 srflx	3792547045	udp 119.76.33.48	64324	100 | 32542 | 255
0.108	1 host	1520314817	tcp 192.168.1.37	9	90 | 32542 | 255
0.225					Authentication failed?
0.226					

Gather candidates

รูปที่ ก.7 ตัวอย่างผลการตรวจสอบ TURN server