วอยซ์สตรีมมิ่งของโทรศัพท์เคลื่อนที่บนระบบปฏิบัติการซิมเบียน

VOICE STREAMING ON SYMBIAN OS SMART PHONE

โดย

นายไพศิษฐ์ จรัลนามศิริ

นายอานนท์ กานต์วิศิษฏ์

อาจารย์ที่ปรึกษา

ผศ.ดร.สุรินทร์ กิตติธรกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง วอยซ์สตรีมมิ่งของโทรศัพท์เคลื่อนที่บนระบบปฏิบัติการซิมเบียน

VOICE STREAMING ON SYMBIAN OS SMART PHONE

คณะผู้จัดทำ นายไพศิษฐ์ จรัลนามศิริ รหัส 44010607

นายอานนท์ กานต์วิศิษฏ์ รหัส 44010612

..... อาจารย์ที่ปรึกษา

(ผศ.ดร.สุรินทร์ กิตติธรกุล)

วอยซ์สตรีมมิ่งของโทรศัพท์เคลื่อนที่บนระบบปฏิบัติการซิมเบียน

นายไพศิษฐ์ จรัลนามศิริ 44010607
นายอานนท์ กานต์วิศิษฎ์ 44010612
ผศ.ดร.สุรินทร์ กิตติธรกุล อาจารย์ที่ปรึกษา
ปีการศึกษา 2547

บทคัดย่อ

โครงงานนี้เป็นการพัฒนาโปรแกรมบนระบบปฏิบัติการซิมเบียน โดยโปรแกรมจะมีความสามารถในการทำวอยซ์สตรีมมิ่งผ่านเครือข่ายจีพีอาร์เอส โดยใช้การเปิดซ็อกเก็ต (Socket) ระหว่างโทรศัพท์เคลื่อนที่ 2 เครื่อง เครื่องหนึ่งทำหน้าที่เป็นเซิร์ฟเวอร์ (Server) คอยฟังการเชื่อมต่อ และอีกเครื่องหนึ่งทำหน้าที่เป็นไคลเอนท์ (Client) เชื่อมต่อไปยังเครื่องที่คอยฟังการเชื่อมต่ออยู่ โดยในการเปิดซ็อกเก็ตนั้นจะมีการนำเทคโนโลยี IP Multimedia Subsystem (IMS) เข้ามาช่วยทำให้โทรศัพท์เคลื่อนที่ทั้งสองเครื่องรู้หมายเลขไอพีแอดเดรส (IP Address) ของกันและกัน หลังจากที่ทำการเชื่อมต่อโทรศัพท์เคลื่อนที่สองเครื่องเข้าหากันได้แล้ว ก็จะเป็นส่วนของการส่งเสียง ซึ่งมีการนำเทคโนโลยี Voice over IP (VoIP) เข้ามาช่วย นอกจากนี้ยังมีส่วนของการแซมปลิ้งและส่วนของการ Codec อีกด้วย และเมื่อโครงงานนี้ประสบผลสำเร็จจะช่วยเพิ่มช่องทางการสื่อสารด้วยเสียงผ่านโทรศัพท์เคลื่อนที่แบบ Packet Switched Data ขึ้นมาจากเดิมที่มีเพียงแต่วิธี Circuit Switched Data ซึ่งจะช่วยให้มีการใช้ช่องทางการสื่อสารร่วมกันได้ นอกจากนี้ยังจะช่วยลดค่าใช้จ่ายที่เกิดจากการติดต่อสื่อสารแบบเดิมลงอีกด้วย

VOICE STREAMING ON SYMBIAN OS SMART PHONE

Mr. Paisit Jarunnamsiri
Mr. Aanon Kanwisit
Asst. Prof. Dr. Surin Kittitornkun Advisor
Academic Year 2004

ABSTRACT

This project is to develop an application on Symbian OS smart phone. The application can stream voice over a GPRS/EDGE network by creating a socket between two mobile phones. One mobile phone works as a server to listen for a connection. The other one works as a client to start connecting to server. In the socket openness, we use the IP Multimedia Subsystem (IMS) technology to help two mobile phones to know IP address of one another. After two mobile phones have established the connection, a stream of sampled, digitized and compressed audio will be sent along using Voice over IP (VoIP) technology. To a certain degree of success, this project has shown a method for voice communications in a Packet Switched Data instead of the Circuit Switched Data network. This method utilizes the bandwidth more efficiently among IP users and eventually reduces the cost.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงจะสำเร็จลุล่วงลงอย่างดีไม่ได้หากปราศจากคำแนะนำ คำปรึกษาและความดูแลเอาใจใส่จากหลายๆ ฝ่าย โดยเฉพาะ ผศ.ดร.สุรินทร์ กิตติธรกุล อาจารย์ที่ปรึกษาที่ให้โอกาสข้าพเจ้าได้ทำปริญญานิพนธ์ฉบับนี้ ทั้งยังคอยให้ความเอาใจใส่ ให้คำแนะนำต่างๆ และให้ความช่วยเหลือเสมอมา ต้องขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณห้องวิจัยโมบายล์คอมพิวติ้ง และภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้จัดเตรียมสิ่งอำนวยความสะดวกต่างๆ รวมถึงเครื่องมือในการพัฒนา ทำให้การวิจัยและพัฒนาโปรแกรมเป็นไปได้ด้วยความสะดวก รวดเร็วและทำให้โปรแกรมที่พัฒนาขึ้นมาสมบูรณ์ในที่สุด และที่จะขาดไม่ได้ก็คือพี่ๆ ปริญญาโท ที่คอยให้ความช่วยหลือและคอยตอบปัญหาของข้าพเจ้าอยู่เสมอ เพื่อนๆ น้องๆ ในห้องวิจัยโมบายล์คอมพิวติ้งที่คอยสร้างความคึกครื้นภายในห้องวิจัย และยังคอยเป็นกำลังใจเสมอมา

และสุดท้ายต้องขอขอบคุณบุคคลที่สำคัญที่สุดในชีวิตที่ทำให้ข้าพเจ้ามีวันนี้ นั่นคือ บิดา มารดา และบุคคลในครอบครัว อันเป็นที่เคารพรัก ซึ่งได้เลี้ยงดู คอยสั่งสอนข้าพเจ้ามาเป็นอย่างดี พร้อมให้โอกาสในการศึกษาอย่างเต็มที่ ทั้งยังให้กำลังใจและความรักเสมอมา ข้าพเจ้าขอกราบพระคุณมา ณ ที่นี้ด้วย

ไพศิษฐ์ จรัลนามศิริ
อานนท์ กานต์วิศิษฏ์

สารบัญ

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

เนื่องด้วยในปัจจุบันโทรศัพท์เคลื่อนที่ได้เข้ามามีบทบาทในชีวิตประจำวันของเรามากขึ้น ดังจะเห็นได้จากวิวัฒนาการของโทรศัพท์เคลื่อนที่ที่ได้มีการพัฒนาไปมาก จากอดีตที่มีการทำงานด้วยระบบอนาล็อก (Analog) มาสู่ยุคปัจจุบันที่ทำงานด้วยระบบดิจิตอล (Digital) หรือยุคของ 2จี (2G), 2.5จี (2.5G) และยังมีแนวโน้มจะก้าวไปสู่ยุค 3จี (3G) ในเวลาอันใกล้นี้ จึงทำให้โทรศัพท์เคลื่อนที่ไม่ได้เป็นแค่เพียงอุปกรณ์ที่ใช้สำหรับการสื่อสารด้วยเสียงเพียงอย่างเดียวเท่านั้น แต่ยังสามารถใช้การสื่อสารด้วยภาพเพิ่มเติมเข้าไปด้วย ช่วยให้เราสามารถสื่อสารและทำความเข้าใจกับคู่สนทนาของเราได้ดียิ่งขึ้น ในส่วนของเทคโนโลยีการสื่อสารบนโทรศัพท์เคลื่อนที่นั้นก็ได้มีการพัฒนาขึ้นอยู่ตลอดเวลา ไม่ว่าจะเป็นอินฟราเรด (Infrared), บลูทูธ (Bluetooth), จีพีอาร์เอส (GPRS: General Packet Radio Service) และเอดจ์ (EDGE: Enhanced Data Rates for Global Evolution) ทำให้ช่องทางที่ใช้ในการติดต่อสื่อสารมีมากขึ้น แต่ในปัจจุบันการติดต่อสื่อสารโดยใช้เสียงนั้นยังคงใช้ช่องทางการติดต่อสื่อสารแบบเดิมคือ Circuit Switched Data ซึ่งเป็นการจองช่องทางการสื่อสารไว้ตลอดเวลาจากโทรศัพท์เคลื่อนที่เครื่องหนึ่งไปยังอีกเครื่องหนึ่ง ทำให้มีการใช้ช่องสัญญาณ (Bandwidth) อย่างไม่เต็มประสิทธิภาพ เนื่องจากจะไม่มีผู้ใดสามารถเข้ามาใช้ช่องสัญญาณนั้นได้เลยจนกว่าจะสิ้นสุดการสื่อสาร ถึงแม้ว่าเราจะไม่ได้ส่งเสียงออกมาก็ตาม แทนที่จะใช้ช่องทางการติดต่อสื่อสารแบบ Packet Switched Data ซึ่งจะไม่มีการจองช่องทางในการสื่อสาร แต่จะใช้การแปลงสัญญาณเสียงเป็นข้อมูลในลักษณะของแพ็กเก็ต (Packet) แล้วส่งไปบนระบบเครือข่ายโทรศัพท์เคลื่อนที่ ทำให้มีการใช้ช่องสัญญาณอย่างเต็มประสิทธิภาพมากกว่า เพราะเราสามารถใช้ช่องสัญญาณร่วมกันได้ อีกทั้งยังช่วยลดภาระค่าใช้จ่ายในการติดต่อสื่อสารลงอีกด้วย โดยในที่นี้เราจะพูดถึงระบบจีพีอาร์เอสเท่านั้น

ระบบจีพีอาร์เอสเป็นช่องทางการติดต่อสื่อสารที่กำลังได้รับความนิยมเพิ่มขึ้นในปัจจุบัน โดยมีความสามารถในการรับส่งข้อมูลได้ทั้งภาพและเสียงในระยะไกลผ่านเครือข่ายโทรศัพท์เคลื่อนที่ มีความเร็วในการรับส่งข้อมูลอยู่ที่ 43.2 กิโลบิตต่อวินาที (Kbps) ทำให้มีการตอบสนองที่เป็นไปอย่างรวดเร็วทันต่อความต้องการของผู้ใช้ ทั้งในเรื่องค่าใช้จ่ายที่มีการคิดค่าใช้จ่ายตามปริมาณข้อมูลที่รับส่งจริงเป็นกิโลไบต์ (Kilobyte) หรือตามเวลาของการเชื่อมต่อเป็นวินาที ซึ่งเมื่อเทียบกับค่าใช้จ่ายในการโทรศัพท์ในปัจจุบันจะเห็นว่ามีค่าใช้จ่ายที่ถูกกว่าและยังไม่เป็นการเอาเปรียบผู้ใช้งานอีกด้วย

จากเหตุผลข้างต้นและแนวทางในการทำ Voice Over IP (VoIP) ทางคณะผู้จัดทำ จึงได้มีความคิดที่จะพัฒนาการติดต่อสื่อสารด้วยเสียงของโทรศัพท์เคลื่อนที่ไปเป็นแบบวอยซ์สตรีมมิ่ง (Voice Streaming) โดยมีพื้นฐานอยู่บนการใช้ช่องทางการสื่อสารแบบ Packet Switched Data ที่จะทำให้มีการใช้ช่องทางอย่างเต็มประสิทธิภาพ เพราะว่าเวลาที่เราไม่ได้ส่งเสียงหรือพูดคุย ผู้อื่นก็สามารถเข้ามาใช้ช่องทางของเราได้ โดยการติดต่อสื่อสารด้วยเสียงในลักษณะนี้เราจะใช้การติดต่อสื่อสารผ่านระบบจีพีอาร์เอสบน

เครือข่ายโทรศัพท์เคลื่อนที่ และมีลักษณะของการตอบสนองแบบเรียลไทม์ (Real Time) คือเมื่อฝ่ายใดฝ่ายหนึ่งพูดอีกฝ่ายจะได้ยินทันที นอกจากนี้ก่อนที่ระบบจะทำการส่งข้อมูลเสียงนั้นออกไปบนเครือข่ายโทรศัพท์เคลื่อนที่ ข้อมูลเสียงเหล่านั้นจะถูกบีบอัดให้มีขนาดเล็กลง โดยใช้เทคโนโลยีที่มีอยู่ เช่น การเข้ารหัสแบบต่างๆ เพื่อลดขนาดของข้อมูลและค่าใช้จ่ายที่จะเกิดขึ้น

1.2 วัตถุประสงค์

1. เพื่อศึกษาการทำวอยซ์สตรีมมิ่งของโทรศัพท์เคลื่อนที่บนระบบปฏิบัติการซิมเบียน (Symbian Operating System)
2. เพื่อพัฒนาและปรับปรุงการใช้ช่องทางในการติดต่อสื่อสารด้วยเสียงบนเครือข่ายโทรศัพท์เคลื่อนที่ให้มีประสิทธิภาพมากขึ้น
3. เพื่อเพิ่มความสามารถของระบบจีพีอาร์เอสบนเครือข่ายโทรศัพท์เคลื่อนที่ให้สามารถนำมาใช้ในการติดต่อสื่อสารด้วยเสียงได้
4. เพื่อลดภาระค่าใช้จ่ายที่จะเกิดขึ้นในการติดต่อสื่อสารด้วยเสียงบนเครือข่ายโทรศัพท์เคลื่อนที่

1.3 ผลที่คาดว่าจะได้รับ

1. ได้รับความรู้ ความเข้าใจเกี่ยวกับระบบปฏิบัติการซิมเบียน และการเขียนโปรแกรมบนระบบปฏิบัติการซิมเบียน
2. ได้รับความรู้ ความเข้าใจในเรื่องของการทำวอยซ์สตรีมมิ่งของโทรศัพท์เคลื่อนที่ระบบปฏิบัติการซิมเบียน
3. ได้รับความรู้ ความเข้าใจเรื่องการแซมปลิ้งสัญญาณเสียง และการบีบอัดข้อมูล
4. ได้รับความรู้ ความเข้าใจในเรื่องของการทำงานของระบบจีพีอาร์เอสว่ามีรูปแบบการทำงาน และมีหลักการอย่างไร
5. ได้รับความรู้ ความเข้าใจในเรื่องของเทคโนโลยี IP Multimedia Subsystem (IMS)

1.4 ขอบเขตของการพัฒนา

โครงงานนี้เป็นการพัฒนาแอปพลิเคชัน (Application) บนระบบปฏิบัติการซิมเบียน โดยใช้ภาษา C++ โดยที่แอปพลิเคชันที่พัฒนาขึ้นมานี้จะมีความสามารถในการทำวอยซ์สตรีมมิ่งผ่านระบบจีพีอาร์เอสบนเครือข่ายของโทรศัพท์เคลื่อนที่ เพื่อนำมาใช้ในการติดต่อสื่อสารด้วยเสียง โดยมีรายละเอียดต่างๆ ของการพัฒนาดังนี้

1. การติดต่อสื่อสารด้วยการทำวอยซ์สตรีมมิ่งนั้น จะต้องเป็นการสื่อสารแบบสองทาง (Duplex) คือต้องสามารถทำวอยซ์สตรีมมิ่งได้จากทั้งสองฝ่ายของคู่สนทนา
2. การติดต่อสื่อสารที่เกิดขึ้นระหว่างทั้งสองฝ่ายจะใช้การเปิดซ็อกเก็ต (Socket) ระหว่างโทรศัพท์เคลื่อนที่ทั้งสองเครื่อง โดยใช้เทคโนโลยี IMS ในการช่วยให้ทั้งสองฝ่ายรู้หมายเลขไอพีแอดเดรส (IP Address) ซึ่งกันและกัน

3. สัญญาณเสียงที่จะถูกส่งออกไปนั้นจะต้องผ่านการแซมปลิ้งด้วยอัตราที่กำหนด ผ่านการเข้ารหัสและการบีบอัด และถูกแปลงให้อยู่ในรูปของแพ็คเก็ตก่อน
4. เมื่อผู้รับได้รับแพ็คเก็ตข้อมูล จะต้องสามารถถอดรหัสและขยายข้อมูลออกมาเป็นสัญญาณเสียงที่ถูกต้องได้

1.5 วิธีการดำเนินงาน

สำหรับวิธีการดำเนินงานนั้นเราจะแยกการทำงานออกเป็นสองส่วน คือ ส่วนของการเปิดซ็อกเก็ตระหว่างโทรศัพท์เคลื่อนที่ทั้งสองเครื่อง และส่วนของการจัดการด้านเสียง โดยใช้การเขียนโปรแกรมภาษา C++ บนระบบปฏิบัติการซิมเบียน โดยมีรายละเอียดการดำเนินงานของแต่ละส่วนดังนี้

ส่วนของการเปิดซ็อกเก็ตระหว่างโทรศัพท์เคลื่อนที่ทั้งสองเครื่องนั้นเราจะอาศัยเทคโนโลยี IMS เข้ามาช่วยในการทำให้เครื่องโทรศัพท์ทั้งสองรู้หมายเลขไอพีแอดเดรสของกันและกัน ดังที่กล่าวไปแล้วข้างต้น หน้าที่สำคัญก็คือเราจะต้องสร้างการเชื่อมต่อระหว่างโทรศัพท์ทั้งสองเครื่องผ่านหมายเลขไอพีแอดเดรสที่ได้รับมาและหมายเลขพอร์ต (Port) ที่กำหนดให้ได้

สำหรับในส่วนของการจัดการด้านเสียงจะแบ่งเป็นส่วนย่อยๆ 4 ส่วน และมีรายละเอียดของการดำเนินงานในส่วนต่างๆ ดังนี้ คือ

1. ส่วนของการอัดเสียงและการแซมปลิ้งสัญญาณเสียงนั้น ซึ่งเราจะต้องทำให้เสียงที่อัดมาอยู่ในรูปแบบที่เรากำหนดไว้เพื่อที่จะนำไฟล์เสียงนั้นเข้าสู่ขั้นตอนต่อไป
2. ส่วนของการเข้ารหัสและบีบอัดข้อมูล ซึ่งเราจะนำไฟล์เสียงที่ได้จากส่วนแรกมาทำการเข้ารหัสและบีบอัดด้วยวิธีที่เรากำหนดไว้แล้วแปลงให้อยู่ในรูปของแพ็กเก็ต
3. ส่วนของการถอดรหัสและขยายข้อมูล หน้าที่ของเราก็คือ ทำให้แพ็กเก็ตที่รับมาแปลงกลับมาเป็นสัญญาณเสียงเดิม ซึ่งเราจำเป็นต้องใช้รูปแบบการถอดรหัสและขยายข้อมูลที่สัมพันธ์กับรูปแบบการเข้ารหัสและบีบอัดข้อมูลจากทางฝ่ายผู้ส่ง
4. ส่วนของการเล่นเสียงที่ได้รับมา ซึ่งในส่วนนี้เราจะต้องอาศัย API (Application Programming Interface) ที่มีให้ในการทำให้เสียงนั้นถูกเล่นออกมาให้ผู้รับได้ยินอย่างถูกต้อง

บทที่ 2
ระบบปฏิบัติการซิมเบียน (Symbian OS)

ระบบปฏิบัติการซิมเบียน เป็นระบบปฏิบัติการที่ออกแบบมาเพื่อรองรับเทคโนโลยีการสื่อสารแบบไร้สายบนโทรศัพท์เคลื่อนที่ ซึ่งสามารถที่จะทำงานได้โดยไม่ขึ้นต่อฮาร์ดแวร์ (Hardware) ของโทรศัพท์เคลื่อนที่ที่มีความแตกต่างกัน และสามารถจัดการกับการพัฒนาอย่างต่อเนื่องในอนาคตได้ ช่วยในด้านของการส่งข้อมูลของโทรศัพท์เคลื่อนที่ อีกทั้งยังเป็นระบบที่ใช้งานได้ง่าย มีความปลอดภัยสูง ช่วยประหยัดพลังงาน และใช้หน่วยความจำที่มีขนาดเล็ก เพื่อรองรับกับโทรศัพท์เคลื่อนที่ทั้งในปัจจุบันและอนาคต

ทั้งนี้ระบบปฏิบัติการซิมเบียนนั้นยังเป็นระบบปฏิบัติการเปิด ที่อนุญาตให้โปรแกรมเมอร์รวมถึงผู้ใช้สามารถพัฒนาซอฟท์แวร์ต่างๆ บนระบบปฏิบัติการซิมเบียนได้เอง เรียกได้ว่าในอนาคตจะมีแอพพลิเคชั่นมากมายที่ถูกสร้างขึ้นบนระบบปฏิบัติการซิมเบียน และยังส่งผลให้เป็นตลาดผลิตภัณฑ์ซอฟท์แวร์ที่ยิ่งใหญ่ในอนาคตอีกด้วย

2.1 ความเป็นมาและพัฒนาการของระบบปฏิบัติการซิมเบียน

ซิมเบียน คือ บริษัทผลิตซอฟท์แวร์ (Software) ที่เป็นผู้นำในด้านการผลิตซอฟท์แวร์ที่รองรับเทคโนโลยีการสื่อสารแบบไร้สาย (Wireless) ซึ่งผลิตภัณฑ์ที่ได้รับการยอมรับจากบริษัทคู่ค้า ก็คือระบบปฏิบัติการซิมเบียนนั่นเอง

ระบบปฏิบัติการซิมเบียนเกิดขึ้นในเดือนมิถุนายน ปี ค.ศ.1998 ซึ่งในตอนนั้นมีพันธมิตรร่วมกัน 4 ราย คือ Ericsson, Motorola, Nokia และ Psion มีสำนักงานใหญ่อยู่ที่ประเทศอังกฤษ

ถัดมาในปี ค.ศ. 1999 ซิมเบียนก็ได้พันธมิตรรายใหม่นั่นคือ Matsushita (Panasonic) ในปี ค.ศ. 2000 พันธมิตรของซิมเบียน ก็มากขึ้นอีกโดยมีการจับมือกับ Sony, Sanyo และ Kenwood รวมถึงการเปิดตัวโทรศัพท์ที่ใช้ระบบปฏิบัติการซิมเบียน เครื่องแรกในรูปแบบของสมาร์ทโฟนนั่นก็คือ Ericsson R380 ซึ่งมีคุณสมบัติการใช้งานที่มากมาย และยังมีทำงานด้วยหน้าจอระบบสัมผัสอีกด้วย

ในปี 2001 Nokia ที่เคยใช้ระบบปฏิบัติการ GEOS (Graphical Environment Operating System) ได้เปลี่ยนมาใช้ระบบปฏิบัติการซิมเบียนแทนระบบปฏิบัติการเดิม โดยทาง Nokia ได้พัฒนาคอมมิวนิเคเตอร์ (Communicator) รุ่นใหม่ออกมาคือ Nokia 9210 และโทรศัพท์ที่ใช้แพลทฟอร์ม (Platform) Series 60 รุ่นแรกคือ Nokia 7650 ซึ่งระบบปฏิบัติการที่ใช้นั้นแตกต่างจาก Ericsson R380 เนื่องจากระบบปฏิบัติการซิมเบียนของ Nokia 9210 และ Nokia 7650 นั้นเป็นระบบปฏิบัติการเปิด (Open Symbian OS) คือสามารถที่จะนำโปรแกรมอื่นที่สามารถทำงานบนระบบปฏิบัติการซิมเบียนมาลงเพิ่มในเครื่องได้

ในปี 2002 ทางซิมเบียนได้มีการพัฒนาระบบปฏิบัติการรุ่นใหม่ที่เรียกว่าระบบปฏิบัติการซิมเบียนเวอร์ชัน 7.0 และทาง Sony Ericsson ก็เข้ามาเป็นพันธมิตรและเป็นหุ้นส่วนรายใหญ่ รวมทั้งทาง Sony Ericsson ก็เปิดตัวสมาร์ทโฟนรุ่นล่าสุด คือ Sony Ericsson P800 และในปี 2003 ก็ได้มีการพัฒนา

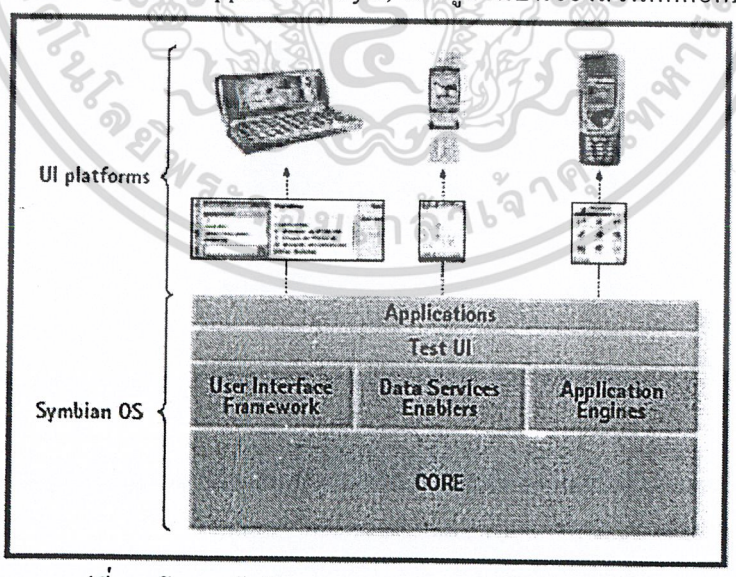
ระบบปฏิบัติการต่อจากเวอร์ชัน 7.0 เป็นเวอร์ชัน 7.0s โดยได้เพิ่มส่วนของการรองรับระบบโทรศัพท์แบบดับบลิวซีดีเอ็มเอ (WCDMA: Wideband Code Division Multiple Access) และได้มีการปรับปรุงการรองรับ Java เป็น MIDP 2.0

และในปี 2004 ทางซิมเบียนก็ได้ทำการเปิดตัวระบบปฏิบัติการซิมเบียนเวอร์ชันใหม่ล่าสุดนั่นก็คือ เวอร์ชัน8.0 ซึ่งระบบปฏิบัติการตัวนี้ได้ถูกสร้างขึ้นมาเพื่อพยายามลดค่าใช้จ่ายที่เกิดขึ้นจากการพัฒนาโทรศัพท์เคลื่อนที่ในอนาคต

2.2 โครงสร้างของระบบปฏิบัติการซิมเบียน

ในส่วนโครงสร้างของระบบปฏิบัติการซิมเบียนประกอบไปด้วยส่วนต่างๆ ที่สำคัญดังนี้

1. แกนหลัก (Core) ของระบบปฏิบัติการซิมเบียน เป็นแกนหลักในการทำงานของอุปกรณ์ เช่น เคอร์เนล (Kernel) ไฟล์เซิร์ฟเวอร์ (File server) การจัดการด้านหน่วยความจำ (Memory management) และไดรเวอร์ของอุปกรณ์ (Device driver) ซึ่งส่วนประกอบต่างๆ นั้นสามารถที่จะเพิ่มหรือลดได้ขึ้นอยู่กับอุปกรณ์แต่ละชนิด
2. ชั้นของระบบ (System Layer) เป็นส่วนที่จัดการเกี่ยวกับการคำนวณ การติดต่อสื่อสาร เช่น TCP/IP, IMAP4, SMS และ การจัดการฐานข้อมูล
3. แอพพลิเคชันเอนจิน (Application Engine) ซึ่งอยู่บนชั้นของระบบ ทำให้นักพัฒนาโปรแกรมสามารถสร้างส่วนติดต่อกับผู้ใช้ (User Interface) ติดต่อกับข้อมูลได้
4. ส่วนติดต่อกับผู้ใช้ โดยผู้ผลิตแต่ละรายสามารถที่จะสร้างขึ้นได้ เช่น แพลทฟอร์ม Series 60 ของบริษัท Nokia
5. ชั้นของแอพพลิเคชัน (Application Layer) ซึ่งอยู่ส่วนบนของส่วนติดต่อกับผู้ใช้



รูปที่ 2-1 โครงสร้างโดยรวมของระบบปฏิบัติการซิมเบียน

2.3 โครงสร้างของไคลเอนท์ – เซิร์ฟเวอร์ (Client – Server)

การทำงานแบบไคลเอนท์ – เซิร์ฟเวอร์เป็นที่รู้จักกันดีในการติดต่อสื่อสารของซอร์ฟแวร์ ในระบบปฏิบัติการซิมเบียน ส่วนของโปรแกรมไคลเอนท์จะมีส่วนติดต่อกับผู้ใช้ที่ติดต่อกับโปรแกรมเซิร์ฟเวอร์ โดยสามารถติดต่อได้ทางอินเตอร์เฟส (Interface) ที่โปรแกรมเซิร์ฟเวอร์ได้สร้างไว้ให้ หลักของการทำงานก็คือ ไคลเอนท์จะร้องขอบริการจากเซิร์ฟเวอร์ ซึ่งเซิร์ฟเวอร์จะตอบสนองตามที่ร้องขอ

2.4 การจัดการกับเหตุการณ์ (Event management)

การจัดการกับเหตุการณ์เป็นส่วนสำคัญที่จะต้องพิจารณา ในระบบปฏิบัติการซิมเบียน ถูกสร้างขึ้นจากการแบ่งปันเวลาที่มีพื้นฐานอยู่บนเหตุการณ์ (Event-base Time Sharing) ในซิงเกิลเทรด (Single thread) แต่อย่างไรก็ตามก็มีวิธีการที่ทำให้เป็นแอพพลิเคชันแบบมัลติเทรด (Multi-thread application) ที่ถูกออกแบบในลักษณะเชิงวัตถุ (Object oriented) เนื่องจากระบบปฏิบัติการซิมเบียน สร้างขึ้นในลักษณะเชิงวัตถุทำให้สามารถทำการแก้ไขได้ง่ายสำหรับชนิดของฮาร์ดแวร์ที่ต่างกัน และอนุญาตให้ผู้ผลิตต่างๆ สามารถที่จะเพิ่มหรือลดคอมโพเนนท์ (Component) ต่างๆ ได้ทำให้เหมาะกับความต้องการของลูกค้ามากขึ้น ซึ่งการที่มีความยืดหยุ่นนี้ในการออกแบบทำให้สามารถออกแบบส่วนติดต่อกับผู้ใช้ในอุปกรณ์ชนิดต่างกัน แต่ทำงานในระบบปฏิบัติการเหมือนกันได้ ระบบปฏิบัติการซิมเบียน เป็นแพลตฟอร์มที่เสถียรภาพสามารถที่จะรองรับเทคโนโลยีใหม่ๆ ได้ เช่น SyncML, บลูทูธและการส่งข้อความแบบมัลติมีเดีย (Multimedia Messaging)

2.5 การจัดการเกี่ยวกับพลังงาน (Power management)

การจัดการเกี่ยวกับพลังงานถูกสร้างขึ้นในเคอร์เนลของระบบปฏิบัติการซิมเบียน การออกแบบช่วยทำให้การใช้พลังงานของโปรเซสเซอร์ (Processor) และอุปกรณ์ต่างๆ เป็นไปอย่างมีประสิทธิภาพ เมื่ออุปกรณ์ชนิดใดไม่ถูกใช้งานก็จะถูกปิดโดยระบบ การที่มีระบบการจัดการพลังงานที่ดีเช่นนี้เองทำให้สามารถยืดอายุการใช้งานของแบตเตอร์รี่ที่มีขนาดเล็กและมีอยู่อย่างจำกัดได้

2.6 ความทนทานและความน่าเชื่อถือ (Robustness and Reliability)

ระบบปฏิบัติการซิมเบียน ประสบความสำเร็จในด้านประสิทธิภาพในระบบโทรศัพท์เคลื่อนที่ โดยอุปกรณ์ไม่ควรที่จะสูญเสียข้อมูล ระบบล่ม หรือมีการรีบูต สิ่งที่ทำให้ระบบปฏิบัติการซิมเบียนประสบความสำเร็จ คือ

1. แต่ละกระบวนการทำงานอยู่บนพื้นที่แอดเดรส (Address space) ที่มีการป้องกัน ทำให้แอพพลิเคชันใดๆ ไม่สามารถเขียนข้อมูลทับในพื้นที่แอดเดรสของแอพพลิเคชันอื่นได้
2. เคอร์เนลทำงานอยู่บนพื้นที่แอดเดรสที่มีการป้องกันจึงทำให้ข้อผิดพลาดของโปรแกรมต่างๆ ไม่สามารถเขียนข้อมูลทับลงในส่วนสแต็กหรือหน่วยความจำฮีพ (Heap) ของเคอร์เนลได้

2.7 การจัดการหน่วยความจำ (Memory management)

ในอุปกรณ์เคลื่อนที่การจัดการหน่วยความจำมีความจำเป็นอย่างยิ่ง เนื่องจากหน่วยความจำมีขนาดจำกัด ระบบปฏิบัติการซิมเบียนก็เช่นกัน ดังนั้นระบบจะมองว่าหน่วยความจำมีขนาดจำกัดและจะใช้หน่วยความจำให้น้อยที่สุดในการทำงานแต่ละครั้ง การใช้หน่วยความจำน้อยนี้ทำให้การใช้พลังงานลดลงด้วย

2.8 การทำงานแบบมัลติทาสกิ้ง (Multitasking)

แอพพลิเคชันต่างๆ ในระบบปฏิบัติการซิมเบียนทำงานอยู่ต่างกระบวนการกัน จึงสามารถทำให้หลายแอพพลิเคชันทำงานได้พร้อมกัน ตัวอย่างเช่น เมื่อผู้ใช้ต้องการตรวจสอบรายการประจำวันและรับสายโทรเข้า ระบบต้องสามารถที่จะสลับการทำงานระหว่างสองกระบวนการได้อย่างรวดเร็ว ผู้ใช้สามารถดูรายการประจำวันขณะที่มีการโทรได้ ซึ่งในปัจจุบันโทรศัพท์เป็นสิ่งที่ให้ข้อมูลอย่างมาก ดังนั้นความสามารถนี้จึงเป็นสิ่งที่สำคัญ

2.9 แอพพลิเคชันเฟรมเวอร์ค (Application Framework)

การทำงานของแอพลิเคชันที่ใช้ระบบปฏิบัติการซิมเบียน บนอีมูเลเตอร์ (Emulator) ที่อยู่บนระบบปฏิบัติการวินโดวส์ (Windows) (ในทางเทคนิคเรียกว่า "WINS" ที่ทำงานแบบกระบวนการเดี่ยว) และอุปกรณ์จริง อย่างเช่น Nokia 3650 จะมีความแตกต่างกันอยู่เล็กน้อย ซึ่งความแตกต่างทำให้ผู้พัฒนาโปรแกรมนั้นมีปัญหากับความไม่สัมพันธ์กันระหว่างอีมูเลเตอร์กับอุปกรณ์จริง โดยเฉพาะผู้พัฒนาโปรแกรมที่เป็นโปรแกรมที่มีความซับซ้อนมาก จะเป็นห่วงเกี่ยวกับความแตกต่างระหว่าง WINS กับอุปกรณ์จริง

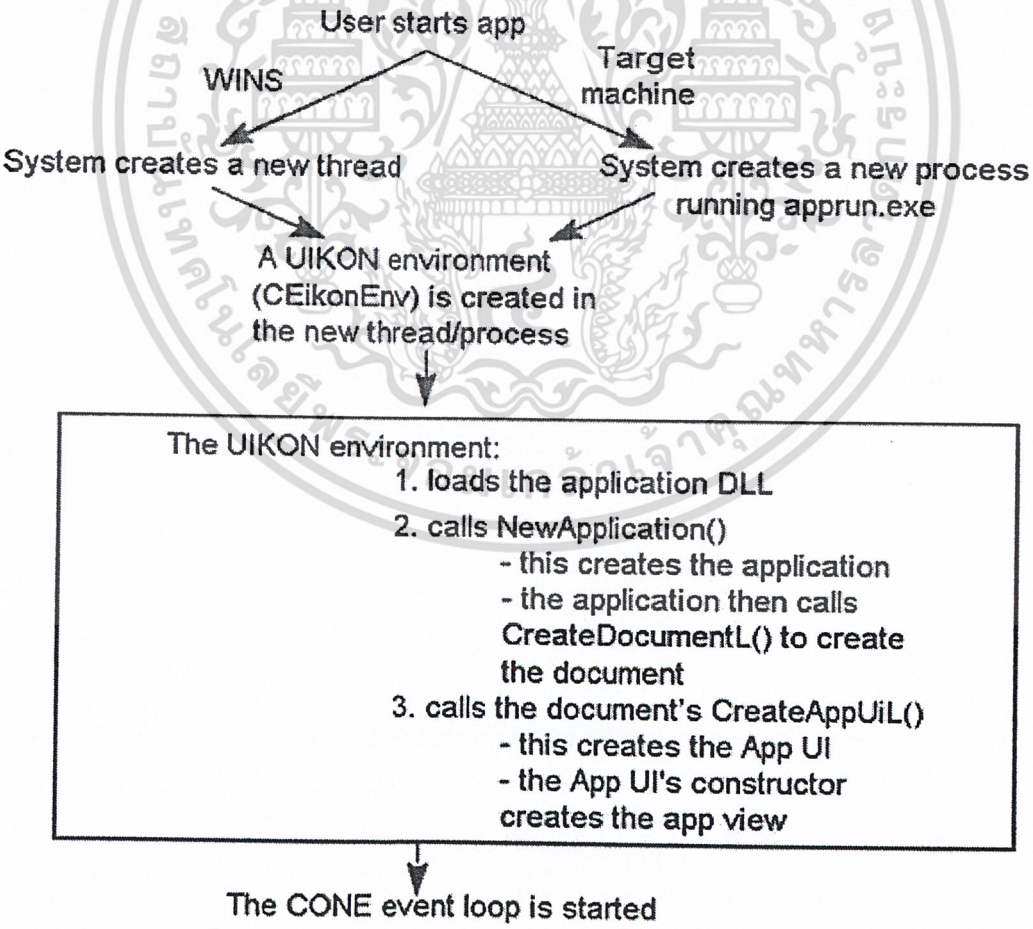
กุญแจหลักสำคัญของแอพพลิเคชันเฟรมเวอร์คนี้คือ UIKON ซึ่ง UIKON ก็คือ มาตรฐานของเฟรมเวอร์คที่ทุกๆ อุปกรณ์ซึ่งทำงานอยู่บนระบบระบบปฏิบัติการซิมเบียนต่างมีเหมือนกัน UIKON ไม่เพียงแต่จะมีเฟรมเวอร์คที่จัดเตรียมไว้สำหรับการทำงานของแอพพลิเคชันเท่านั้น แต่ยังมีส่วนที่สนับสนุนการทำงานของคอมโพเนนท์สำหรับการควบคุมทั่วๆ ไปด้วย เช่น กล่องข้อความ (dialog box) ตัวแก้ไขตัวเลข (number editor) และตัวแก้ไขวันที่ (date editor) ซึ่งกลุ่มของแอพพลิเคชันจะสามารถใช้ประโยชน์ได้ขณะทำงาน

แอพพลิเคชันบนระบบระบบปฏิบัติการซิมเบียน ประกอบด้วย 4 คอมโพเนนท์ ซึ่งแต่และส่วนมีความเกี่ยวข้องกับคลาส (Class) ใน UIKON เฟรมเวอร์ค ได้แก่

1. แอพพลิเคชันเชลล์ (Application shell) ซึ่งได้รับการถ่ายทอดคุณสมบัติมาจาก CEikApplication และคลาสนี้จะถูกสร้างขึ้นตอนแรกสุดโดยเฟรมเวอร์ค ทันทีที่ถูกสร้างขึ้นมันจะมีหน้าที่รับผิดชอบต่อการกำหนดค่าเริ่มต้นต่างๆ ให้กับโค้ด (Code) ส่วนที่เหลืออยู่
2. ดอคคิวเมนท์ (Document) ซึ่งได้รับการถ่ายทอดคุณสมบัติมาจาก CEikDocument ส่วนนี้จะทำให้เข้าใจผิดเล็กน้อยในเรื่องที่ว่า ไม่ใช่ทุกแอพพลิเคชันต้องมีดอคคิวเมนท์ที่จะต้องมาเกี่ยวข้องกับมัน เหมือนกับที่แอพพลิเคชันเหล่านี้จะต้องเกี่ยวข้องกับผู้ใช้ตัวอย่างเช่น

โปรแกรม Notepad ที่เป็นโปรแกรมเกี่ยวข้องกับดอคคิวเมนท์โดยตรงอย่างชัดเจน ในขณะที่โปรแกรม เช่น นาฬิกา นั้นจะไม่ถูกกำหนดความสามารถในการจัดการดอคคิวเมนท์ทั้งการสร้าง การเปิดเอกสารและการแก้ไขดอคคิวเมนท์ใดๆ ในความเป็นจริงทุกๆ แอพพลิเคชันจะต้องมีคลาส CEikDocument ที่เฟรมเวอร์คจะต้องการสร้างขึ้นมา

3. ส่วนติดต่อผู้ใช้งานของแอพพลิเคชัน (App UI) ซึ่งได้รับการถ่ายทอดคุณสมบัติมาจาก CEikAppUi ซึ่งคลาสนี้จัดเตรียมหน้าที่การทำงานหลักสำหรับทุกๆ แอพพลิเคชัน เช่น การรับมือกับเหตุการณ์ (event handing) การควบคุมเหตุการณ์ (event control) การสร้างการควบคุม การเข้าถึงการเรียกของระบบ (system call) ที่เป็นประโยชน์หลายๆ ตัว เป็นต้น
4. วิว (View) ส่วนนี้คือส่วนที่ผู้ใช้สามารถมองเห็นได้จริงในหน้าจอของอุปกรณ์ มันสามารถใช้งานสำหรับการแสดงข้อมูลอย่างง่าย หรือรวบรวมข้อมูลจากผู้ใช้ในแอพพลิเคชันที่ซับซ้อนมากขึ้น ตัวอย่างเช่น ในแอพพลิเคชันประเภทการแก้ไขข้อความ ตัวหนังสือที่ถูกพิมพ์นั้น ก็คือการควบคุมมาตรฐานซึ่งถูกจัดไว้โดย UIKON โดยจัดเก็บอยู่ในส่วนของวิว ในขณะที่ทุกแอพพลิเคชันจะมีอยู่หนึ่งวิวโดยปกติ แต่แอพพลิเคชันที่มีความซับซ้อนมากสามารถมีได้หลายวิว



รูปที่ 2-2 แอพพลิเคชันเฟรมเวอร์คของระบบปฏิบัติการซิมเบียน

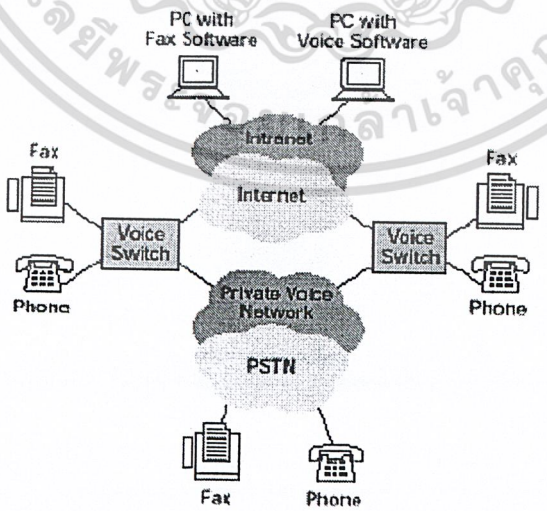
บทที่ 3

Voice over IP และซ็อกเก็ต (Socket)

3.1 Voice over IP (VoIP)

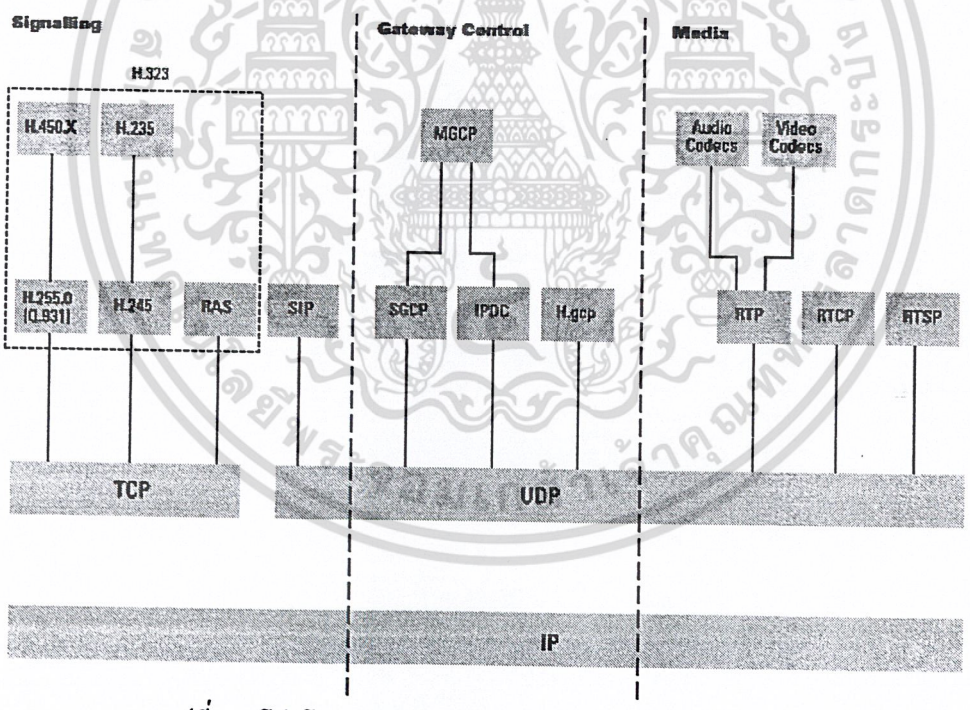
VoIP คือ มาตรฐานที่พัฒนาขึ้นมาสำหรับส่งสัญญาณเสียงบนอินเตอร์เน็ตที่มีพื้นฐานอยู่บนหมายเลขไอพี (IP based Internet) การส่งเสียงบนเครือข่ายอินเตอร์เน็ตเป็นระบบที่นำสัญญาณข้อมูลเสียงมาบรรจุลงเป็นแพ็กเก็ตไอพีแล้วส่งไป โดยที่เราเตอร์ (Router) มีวิธีการปรับตัวเพื่อรับสัญญาณแพ็กเก็ตและยังแก้ปัญหาบางอย่างให้ เช่น การบีบอัดสัญญาณเสียงให้มีขนาดเล็กลง การแก้ปัญหาเมื่อมีแพ็กเก็ตสูญหาย หรือได้มาล่าช้า

ระบบ VoIP เป็นระบบที่นำสัญญาณเสียงที่ผ่านการดิจิไตซ์ (Digitize) โดยหนึ่งช่องเสียงเมื่อแปลงเป็นข้อมูลจะมีขนาด 64 กิโลบิตต่อวินาที การนำข้อมูลเสียงขนาด 64 กิโลบิตต่อวินาทีนี้ไปใช้จำเป็นต้องนำมาบีบอัดก่อน โดยทั่วไปจะเหลือประมาณ 10 กิโลบิตต่อวินาทีต่อช่องสัญญาณเสียง แล้วจึงบรรจุลงในไอพีแพ็กเก็ต เพื่อส่งผ่านทางเครือข่ายอินเตอร์เน็ต การสื่อสารผ่านทางเครือข่ายอินเตอร์เน็ตต้องมีเราเตอร์ที่ทำหน้าที่พิเศษเพื่อประกันคุณภาพของช่องสัญญาณเพื่อให้ข้อมูลไปถึงปลายทางหรือกลับมาได้อย่างถูกต้อง และอาจมีการให้สิทธิพิเศษก่อนแพ็กเก็ตไอพีอื่น เพื่อการให้บริการที่ทำให้เสียงมีคุณภาพ นอกจากนี้ VoIP ยังถูกนิยามให้เป็นความสามารถในการใช้โทรศัพท์ให้ทำทุกอย่างที่เราสามารถทำได้ในปัจจุบันด้วยโครงข่ายโทรศัพท์สาธารณะ (PSTN: Public Switched Telephone Network) และส่งสำเนาเอกสารบนเครือข่ายข้อมูลที่มีพื้นฐานอยู่บนเครือข่ายอินเตอร์เน็ตโดยการวัดประสิทธิภาพ (QoS: Quality of Service) ของระบบสื่อสารซึ่งแสดงถึงคุณภาพในการส่งและความสะดวกในการใช้บริการที่เหมาะสมและราคากับผลประโยชน์ที่คุ้มค่ากว่า



รูปที่ 3-1 โครงสร้างของ VoIP

นอกจากนี้มาตรฐาน VoIP ยังรวมการสื่อสารแบบเรียลไทม์ (Real time) และไม่เรียลไทม์ (Non-real time) เข้าด้วยกัน การส่งข้อความเสียงและแฟกซ์ใช้วิธีการเดียวกับการเรียกโทรศัพท์ธรรมดา แต่ไม่จำเป็นต้องมี QoS ระดับเดียวกันสำหรับในกรณีที่ต้องการคุณสมบัติแบบเรียลไทม์ ดังนั้นจึงต้องมีการพัฒนาโปรโตคอลที่สามารถให้คุณสมบัติดังกล่าวในการส่งข้อมูล และที่สำคัญ VoIP ต้องการฟังก์ชันในการสร้างหรือสิ้นสุดการเรียก และหาตำแหน่งที่อยู่ของผู้ใช้ที่ถูกเรียก รวมทั้งฟังก์ชันที่ช่วยให้สามารถให้บริการได้เช่นเดียวกับในระบบโทรศัพท์ ซึ่งในชุดโปรโตคอล TCP/IP (Transport Control Protocol/ Internet Protocol) นั้นไม่มีโปรโตคอลที่ให้ฟังก์ชันดังกล่าว ดังนั้นจึงต้องมีการพัฒนาโปรโตคอลขึ้นใหม่เพื่อรองรับ VoIP ในปัจจุบัน โปรโตคอลสำหรับ VoIP มีอยู่ 2 มาตรฐาน คือ H.323 และ SIP (Session Initial Protocol) โดยโปรโตคอล H.323 เป็นโปรโตคอลที่พัฒนาโดย ITU-T (International Telecommunications Union - Telecommunications Standardization Sector) ส่วนโปรโตคอล SIP ถูกพัฒนาโดย IETF (Internet Engineering Task Force) โปรโตคอลทั้งสองมีหน้าที่หลักในการสร้างหรือสิ้นสุดการเรียก และการเปลี่ยนแปลงรายละเอียดของการเรียกระหว่างผู้ใช้ VoIP รวมทั้งยังสามารถให้ฟังก์ชันเพิ่มเติมอื่นๆ โปรโตคอลทั้งสองเป็น โปรโตคอลสำหรับ VoIP ซึ่งใช้บริการชุดโปรโตคอล TCP/IP ในชั้นต่ำกว่า และสามารถใช้งานร่วมกับโปรโตคอลอื่น เพื่อให้เกิดการบริการที่มีคุณภาพมากขึ้น



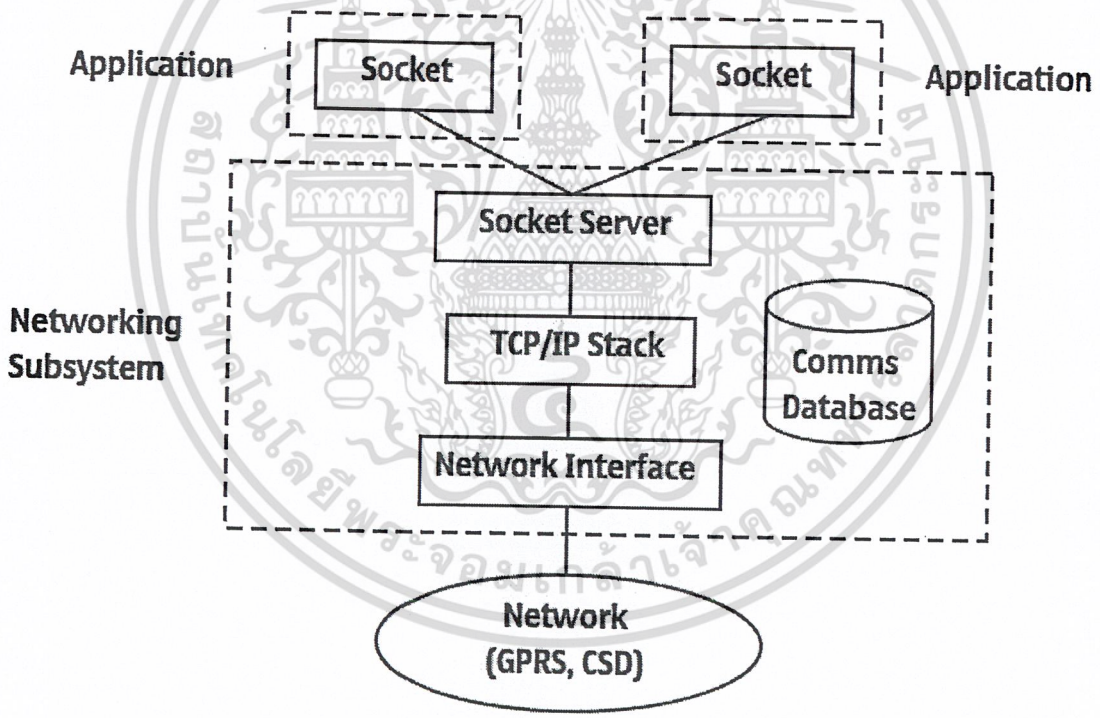
รูปที่ 3-2 โปรโตคอลสแต็กของ VoIP (VoIP Protocol Stack)

โปรโตคอลสแต็กสำหรับ VoIP จะเป็นดังรูปที่ 2-2 จะเห็นว่าทั้งโปรโตคอล H.323 และ SIP เป็นโปรโตคอลในชั้นแอปพลิเคชัน (Application Layer) และใช้บริการของโปรโตคอลในชั้นที่ต่ำกว่า โดยที่ SIP สามารถใช้ได้ทั้ง UDP และ TCP ส่วน H.323 ใช้ TCP เท่านั้น แต่เนื่องจากว่าฟังก์ชันของ H.323 และ SIP มีขอบเขตจำกัด ดังนั้นจึงได้นำโปรโตคอลอื่นมาช่วยในการทำงาน ซึ่งได้แก่ RTSP (Real-time

Streaming Protocol) RSVP (Resource Reservation Protocol) และ RTP/RTCP (Real-time Transport Protocol / Real-time Transport Control Protocol) ซึ่งทำให้ VoIP สามารถให้บริการได้อย่างมีประสิทธิภาพมากขึ้น โปรโตคอลดังกล่าวเป็นโปรโตคอลในชั้นแอปพลิเคชันซึ่งทำงานอยู่บนชุดโปรโตคอล TCP/IP โปรโตคอลเหล่านี้ไม่ได้เป็นโปรโตคอลเฉพาะสำหรับ VoIP ต่างกับโปรโตคอล H.323 และ SIP ซึ่งเป็นโปรโตคอลหลักสำหรับ VoIP

3.2 ซ็อกเก็ต (Socket)

ซ็อกเก็ต API ได้ถูกออกแบบมาเพื่อการติดต่อแบบ TCP/IP บน BSD UNIX และต่อมากลับกลายมาเป็นมาตรฐาน API ในการทำอะไรก็ตามที่ทำข้ามแพลทฟอร์ม รวมทั้งยูนิกซ์ (UNIX) วินโดวส์ และระบบปฏิบัติการอื่นๆ รวมทั้งซิมเบียนด้วย ตั้งแต่ที่มีการใช้ซ็อกเก็ตมานั้น ไม่ว่าจะเป็นแพลทฟอร์มไหนก็จะมีขั้นตอนการใช้งานที่ค่อนข้างใกล้เคียงกัน ซึ่งในที่นี้จะอธิบายถึงการใช้ซ็อกเก็ตในระบบปฏิบัติการซิมเบียน



รูปที่ 3-3 สถาปัตยกรรมเครือข่าย (Network Architecture) ของระบบปฏิบัติการซิมเบียน

ในระบบปฏิบัติการซิมเบียนได้มีการเตรียมซ็อกเก็ต API ชื่อ RSocket ซึ่งคลาสนี้จำเป็นที่ต้องใช้ในการเชื่อมต่อ การส่งข้อมูลหรือการรับข้อมูลผ่านการติดต่อแบบ TCP/IP ซึ่ง API ตัวนี้มีความอิสระที่จะใช้ในการเชื่อมต่อ โดยที่สามารถสร้างการติดต่อได้ทั้งการติดต่อผ่านอินฟราเรด บลูทูธ และจีพีอาร์เอส

นอกจากนั้นซ็อกเก็ตจะมีการติดต่อแบบคอนเน็คชันเลส (Connectionless) หรือคอนเน็คชันโอเรียนเต็ด (Connection-Oriented) ซึ่งการติดต่อแบบคอนเน็คชันเลสซ็อกเก็ตจะเป็นการส่งแพ็คเก็ตไปสู่อีกเครื่องหนึ่ง ซึ่งจะมีขั้นตอนการทำงานดังนี้

1. เวลาที่ต้องการส่งแพ็คเก็ตในคอนเน็คชันเลสซ็อกเก็ตจะต้องมีการใส่ที่อยู่ของผู้รับ
2. เวลาที่ต้องการรับแพ็คเก็ตในคอนเน็คชันเลสซ็อกเก็ตจะต้องมีการใส่ที่อยู่ของผู้ส่ง
3. คอนเน็คชันเลสซ็อกเก็ตจะไม่มีการรับประกันว่า แพ็คเก็ตที่ส่งมาจะเหมือนกันกับที่ผู้ส่งทำการส่ง ซึ่งจะเหมือนกับการที่เราส่งจดหมายหาเพื่อน แต่ละจดหมายต้องการที่อยู่ แต่เราจะไม่ทราบอย่างแท้จริงว่าจดหมายได้ถึงแล้ว

แต่ในส่วนของคอนเน็คชันโอเรียนเต็ดซ็อกเก็ตจะเหมือนกับการโทรศัพท์ไปหาเพื่อน ซึ่งจะสามารถรับประกันได้ ถ้าได้ยินในสิ่งที่พูดอย่างแท้จริง ก็จะสามารถตอบกลับมาได้

โครงสร้างของซ็อกเก็ต API จะประกอบด้วย

1. ซ็อกเก็ตเซิร์ฟเวอร์เพื่อทำการแบ่งปันและการเข้าถึงของโปรโตคอลของซ็อกเก็ต
2. ไคลเอนท์ไซด์ API (Client-side API) เพื่อทำการควบคุมการเข้าถึงเซิร์ฟเวอร์นี้
3. ระบบของเซิร์ฟเวอร์ปลั๊กอิน (Sever Plug-ins) เพื่อให้ทราบถึงโปรโตคอลโมดูล (Protocol Modules)

ซ็อกเก็ตจะถูกแสดงให้เห็นการติดต่อออกเป็นสองส่วน โดยที่ส่วนแรกจะเป็นส่วนของเซิร์ฟเวอร์และอีกส่วนหนึ่งจะแสดงในส่วนของไคลเอนท์ ซึ่งข้อแตกต่างของเซิร์ฟเวอร์กับไคลเอนท์ คือ เซิร์ฟเวอร์จะรอการเข้ามาของการติดต่อ ส่วนไคลเอนท์จะเป็นตัวร้องขอเข้าไปยังเซิร์ฟเวอร์

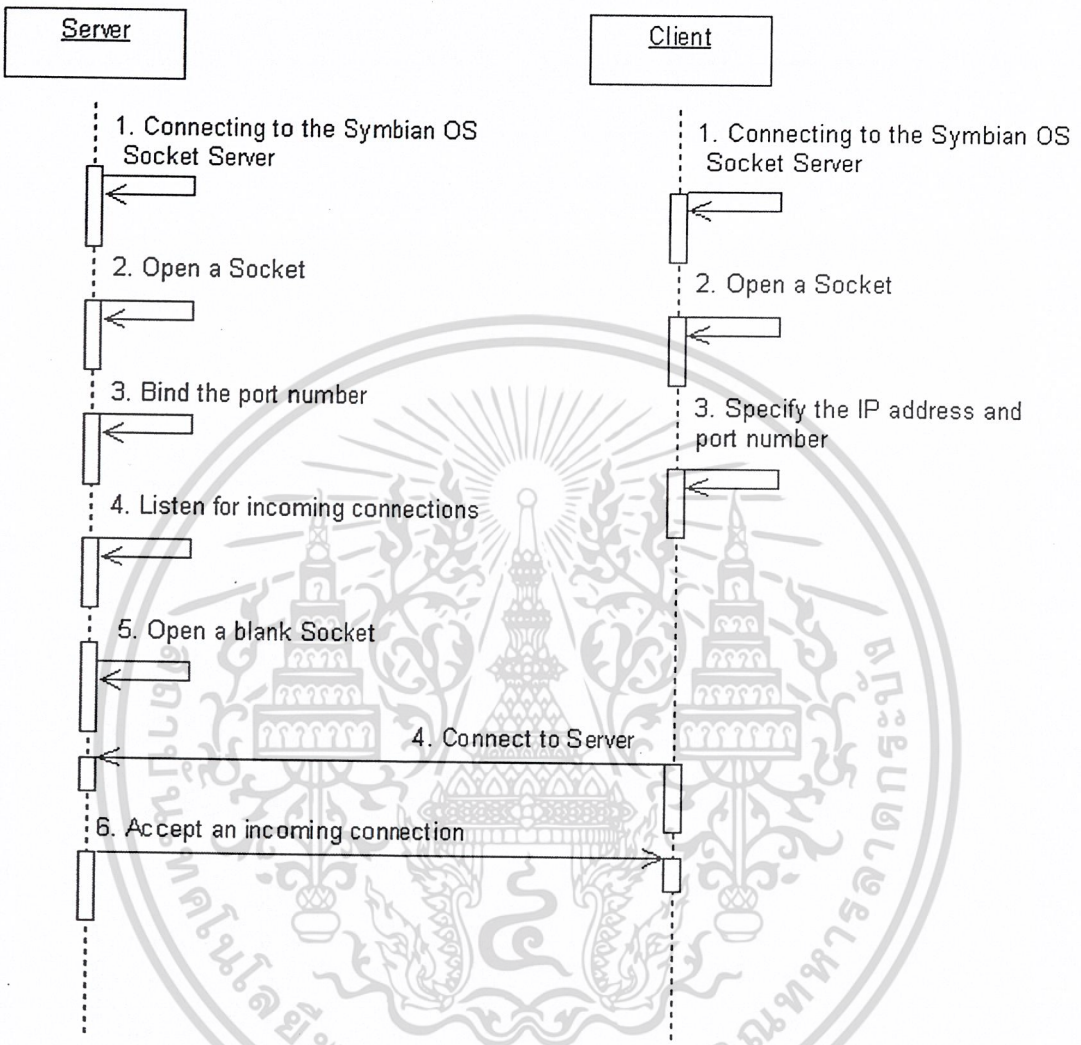
ในการสร้างการเชื่อมต่อระหว่างไคลเอนท์และเซิร์ฟเวอร์นั้น ไคลเอนท์จำเป็นที่ต้องรู้หมายเลขไอพีแอดเดรสของเซิร์ฟเวอร์ และหมายเลขพอร์ตที่เซิร์ฟเวอร์ทำการเปิดเพื่อรอการเข้ามาติดต่อ โดยในส่วนของการรอการเข้ามาของการติดต่อของซ็อกเก็ตส่วนเซิร์ฟเวอร์มีขั้นตอนต่างๆ ดังนี้

1. ทำการติดต่อไปยังซ็อกเก็ตเซิร์ฟเวอร์ของระบบปฏิบัติการซิมเบียน
2. ทำการเปิดซ็อกเก็ต
3. กำหนดหมายเลขพอร์ตที่รอการเชื่อมต่อ
4. รอการเข้ามาติดต่อของไคลเอนท์
5. ทำการเปิดซ็อกเก็ตอีกอันหนึ่งเพื่อใช้ในการส่งหรือรับข้อมูล
6. เมื่อได้รับการเข้ามาติดต่อของไคลเอนท์แล้ว การสร้างการเชื่อมต่อจึงสำเร็จและทำให้ส่งหรือรับข้อมูลได้

ในส่วนต่อมาจะพูดถึงในการที่ซ็อกเก็ตส่วนไคลเอนท์สามารถร้องขอเข้าไปที่เซิฟเวอร์ มีขั้นตอนต่างๆ ดังนี้

1. ทำการติดต่อไปยังซ็อกเก็ตเซิร์ฟเวอร์ของระบบปฏิบัติการซิมเบียน
2. ทำการเปิดซ็อกเก็ต
3. กำหนดหมายเลขพอร์ตและหมายเลขไอพีแอดเดรสของเซิร์ฟเวอร์

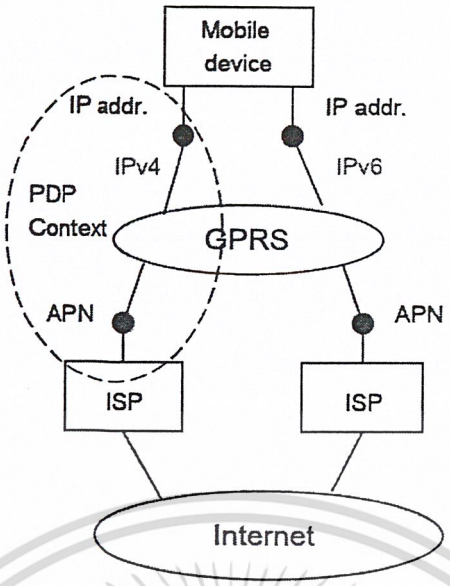
4. ทำการร้องขอเข้าไปที่เซิร์ฟเวอร์ เมื่อเซิร์ฟเวอร์ยอมรับการเชื่อมต่อ จึงจะสามารถที่จะส่งหรือรับข้อมูลได้



รูปที่ 3-4 ขั้นตอนการสร้างการเชื่อมต่อระหว่างไคลเอนท์และเซิร์ฟเวอร์

โดยในการรับหรือส่งข้อมูลระหว่างไคลเอนท์กับเซิร์ฟเวอร์นั้น จะต้องทำกันคนละซ็อกเก็ต เพราะแต่ละซ็อกเก็ตจะสามารถทำได้แค่รับหรือส่งข้อมูลเพียงอย่างเดียว ดังนั้นถ้าต้องการทั้งรับและส่งในเวลาเดียวกัน จึงจำเป็นที่ต้องใช้ซ็อกเก็ต 2 ตัว และหลังจากทำการเชื่อมต่อเสร็จแล้ว ต้องมีการปิดการใช้งานซ็อกเก็ตด้วย

ในทุกวันนี้หมายเลขไอพีแอดเดรสที่ใช้กันอยู่เป็นแบบ IPV4 โดยที่หมายเลขไอพีแอดเดรสมีขนาด 32 บิต ซึ่งหมายเลขไอพีแอดเดรสที่ใช้อาจจะไม่คงทนหรือไม่เพียงพอต่ออุปกรณ์ที่มีเพิ่มมากขึ้นในอนาคต ดังนั้น IPV6 จึงถูกเพิ่มเข้ามาในซ็อกเก็ต API และ IPV6 ได้มีการปรับปรุงโครงสร้างโปรโตคอลในส่วนของหมายเลขไอพีแอดเดรสให้มีขนาด 48 บิต เพื่อรองรับอุปกรณ์ที่เพิ่มมากขึ้นในอนาคต จึงทำให้ซ็อกเก็ต API รองรับทั้ง IPV4 และ IPV6 ดังรูป



รูปที่ 3-5 แสดงการรองรับการใช้งานของ IPV4 และ IPV6

บทที่ 4
การเข้ารหัส (Encoding)

4.1 การเข้ารหัสของสัญญาณเสียง (Audio Encoding)

โดยทั่วไปสัญญาณเสียงจะถูกแสดงในรูปของผลรวมของคลื่นที่มีความถี่และความกว้างที่แตกต่างกัน อีกนัยหนึ่งก็คือ สัญญาณเสียงใดๆ จะอยู่ในรูปแบบของคลื่นแอมพลิจูดซึ่งเป็นฟังก์ชันของเวลา สัญญาณเสียงจะถูกเปลี่ยนไปอยู่ในรูปดิจิตอลเมื่อใช้ความถี่ในการแซมปลิ้งที่เพียงพอ สำหรับสัญญาณเสียงทั่วๆ ไปและเสียงดนตรีจะใช้ความถี่ในการแซมปลิ้งเท่ากับ 44.1 กิโลเฮิร์ซ (Kilohertz) เพื่อจะได้คุณภาพเสียงที่ดี สำหรับเสียงพูดของคนจะใช้ความถี่ในการแซมปลิ้งเท่ากับ 8 กิโลเฮิร์ซ โดยทั่วไปแล้วการแซมปลิ้งหนึ่งครั้งสามารถถูกแสดงด้วยข้อมูลขนาด 16 บิต

สัญญาณเสียงที่อยู่ในรูปดิจิตอลจะถูกบีบอัดให้มีขนาดเล็กลงได้หลายวิธี ซึ่งใช้การเข้ารหัสแบบง่ายๆ คือ การปรับเปลี่ยนขนาดของขั้นเพื่อประเมินค่าเสียงที่ได้จากการแซมปลิ้ง ซึ่งวิธีนี้ถูกใช้ในมาตรฐานการเข้ารหัสของสัญญาณเสียงแบบ IMA ADPCM โดยที่จะแทนการแซมปลิ้งหนึ่งครั้งด้วยข้อมูลขนาด 4 บิต ดังนั้นถ้าใช้ความถี่ในการแซมปลิ้งเท่ากับ 8 กิโลเฮิร์ซ วิธี IMA ADPCM จะเข้ารหัสด้วยความเร็ว 32 กิโลบิตต่อวินาที นอกจากนี้ก็ยังมีการเข้ารหัสแบบง่ายๆ อีกแบบ คือ A-law PCM ซึ่งใช้การประเมินค่าโดยใช้ล็อกการิทึม (Logarithm) และใช้ข้อมูล 8 บิตต่อการแซมปลิ้งหนึ่งครั้ง

การเข้ารหัสที่มีความก้าวหน้าของสัญญาณเสียงนั้นจะใช้ประโยชน์ของรูปแบบไซโคอคูสติกของมนุษย์ (the human psychoacoustic model) ส่วนประกอบบางส่วนของสัญญาณเสียงนั้นเป็นสัญญาณเสียงที่มนุษย์เราไม่สามารถได้ยิน เวลาทำการเข้ารหัสนั้นสัญญาณเสียงเหล่านี้จะถูกตัดทิ้งหรือถูกทำการบีบอัด โดยทั่วไปวิธีการเข้ารหัสที่มีความก้าวหน้าของสัญญาณเสียงจะถูกแบ่งออกเป็น การเข้ารหัสของสัญญาณเสียงแบบทั่วไป (generic audio encoding) และเทคนิคการเข้ารหัสของเสียงพูด (speech encoding techniques) ซึ่งการเข้ารหัสของสัญญาณเสียงแบบทั่วไปใช้ได้ดีกับเสียงเพลง และเสียงที่มีลักษณะเหมือนกับเสียงพูดของมนุษย์ ส่วนเทคนิคการเข้ารหัสของเสียงพูดจะมุ่งไปใช้กับเสียงพูดของมนุษย์เท่านั้น ซึ่งการใช้วิธีนี้จะไม่ดีเมื่อใช้ในการเข้ารหัสเสียงเพลง

4.2 การเข้ารหัสของเสียงพูด (Speech Encoding)

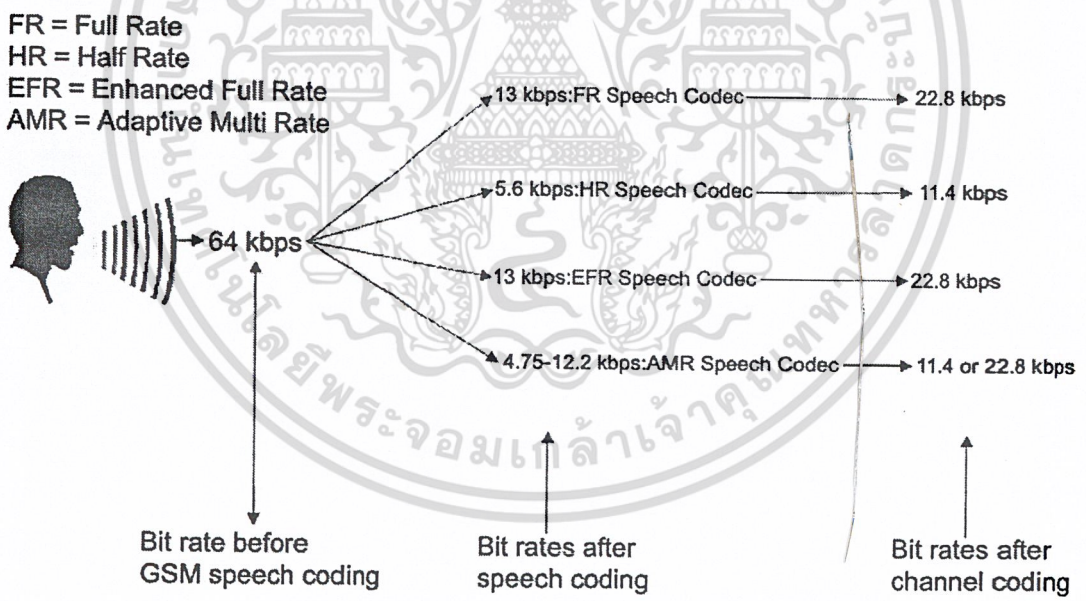
การเข้ารหัสเสียงพูดของมนุษย์ จะหมายถึงการบีบอัดเสียงพูดบนโทรศัพท์เคลื่อนที่ก่อนที่ทำการส่งเสียงพูดไปยังปลายทาง การบีบอัดถูกทำให้สำเร็จโดยรูปแบบลักษณะพิเศษของเสียงพูดทางคณิตศาสตร์ โดยที่เวลาส่งไปในเครือข่ายจะถูกทำให้เป็นข้อมูล ซึ่งข้อมูลเหล่านี้จำเป็นที่ต้องถอดรหัสก่อนจึงจะสามารถได้รับฟังได้ ในการเข้ารหัสทำให้ระบบเครือข่ายของโทรศัพท์เคลื่อนที่ถูกใช้ประโยชน์ได้มากขึ้น

โดยทั่วไปแล้วจะเข้ารหัสด้วยบิตเรท (Bit rate) ที่น้อย จึงทำให้ได้สัญญาณเสียงที่มีคุณภาพที่ไม่ดี แต่อย่างไรก็ตาม เมื่อใช้การเข้ารหัสที่มีความก้าวหน้าและซับซ้อนแล้วจะได้สัญญาณเสียงที่คุณภาพที่ดีขึ้น ยกตัวอย่างเช่น การเข้ารหัสแบบเอนฮานซ์ฟูลเรท (enhanced full rate) และการเข้ารหัสแบบฟูลเรท (full rate) ซึ่งมีค่าบิตเรทที่เท่ากัน แต่เนื่องจากการเข้ารหัสแบบเอนฮานซ์ฟูลเรทใช้เทคโนโลยีในการเข้ารหัสที่ก้าวหน้ากว่าจึงทำให้ได้สัญญาณเสียงที่ดีกว่าการเข้ารหัสแบบฟูลเรท

ในการเข้ารหัสเสียงพูดมีความจำเป็นที่ต้องป้องกันข้อผิดพลาดในแอร์อินเตอร์เฟส (Air interface) ซึ่งจำนวนการแก้ไขข้อผิดพลาด (error correction) ที่ถูกใช้นั้นจะส่งผลกระทบต่อคุณภาพเสียง เช่น ยิ่งเพิ่มจำนวนการแก้ไขข้อผิดพลาดมากเท่าไหร่ ตัวเข้าและถอดรหัสก็สามารถดำเนินการในเงื่อนไขที่แย่ได้ดีขึ้นเท่านั้น

ผลรวมของบิตเรทที่ถูกใช้สำหรับการเชื่อมต่อเพื่อส่งเสียงในระบบจีเอสเอ็ม (GSM: Global System for Mobile Communications) จะถูกแบ่งออกเป็นเสียงที่ถูกเข้ารหัสและตัวแก้ไขหรือตัวควบคุมข้อผิดพลาด เพราะฉะนั้นถ้าต้องการเพิ่มบิตเรทของการเข้ารหัสเสียงจะต้องใช้การแก้ไขข้อผิดพลาดที่น้อยลงและประสิทธิภาพการทำงานในเงื่อนไขที่แย่ก็จะแย่ลงด้วย

ในระบบ GSM มีการกำหนดมาตรฐานการเข้ารหัสเสียงพูดของมนุษย์อยู่ 4 แบบ ดังรูป

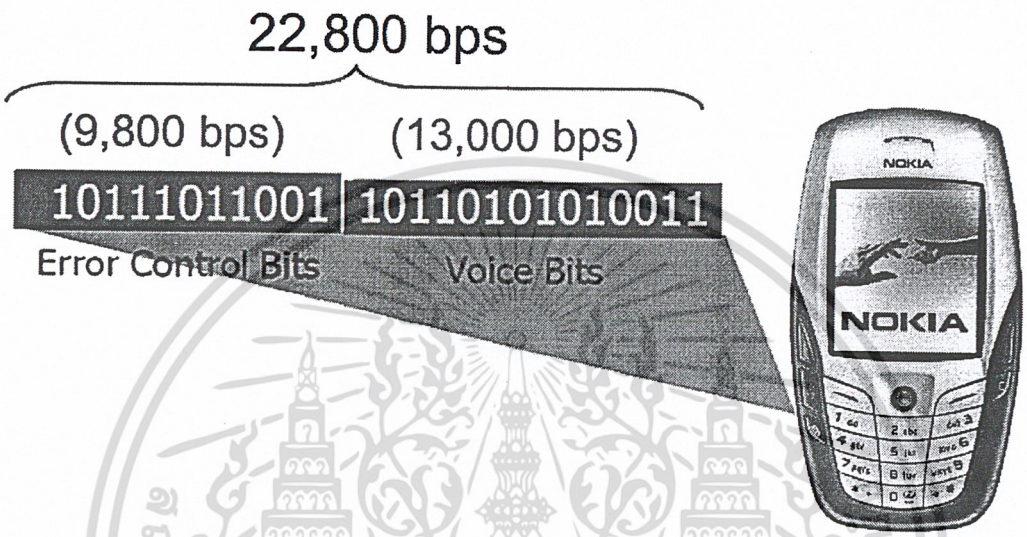


รูปที่ 4-1 มาตรฐานการเข้ารหัสเสียงพูดของมนุษย์

4.2.1 การเข้ารหัสแบบฟูลเรท (FR: Full Rate)

การเข้ารหัสแบบฟูลเรทซึ่งเป็นการเข้ารหัสอันดับแรกที่ถูกกำหนดสำหรับระบบจีเอสเอ็ม การเข้ารหัสแบบนี้มีในโทรศัพท์เคลื่อนที่ระบบจีเอสเอ็มมาตั้งแต่ ค.ศ. 1992

หนึ่งช่องทางของเครือข่ายจีเอสเอ็มแบบฟูลเรทจะมีบิตเรทเท่ากับ 22.8 กิโลบิตต่อวินาที ซึ่งส่วนประกอบของบิตเรทจะถูกใช้สำหรับเสียงพูดที่ถูกเข้ารหัสและส่วนของการควบคุมความผิดพลาด



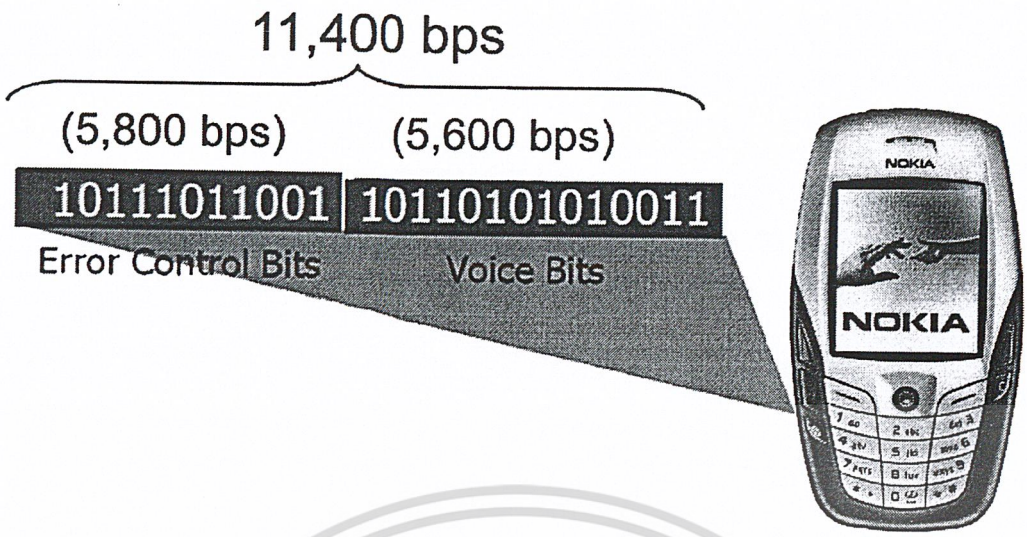
รูปที่ 4-2 แพ็กเก็ตของการเข้ารหัสแบบฟูลเรท

4.2.2 การเข้ารหัสแบบฮาล์ฟเรท (HR: Half Rate)

การเข้ารหัสแบบฮาล์ฟเรทซึ่งเป็นการเข้ารหัสอันดับสองที่ถูกกำหนดสำหรับระบบจีเอสเอ็ม การเข้ารหัสแบบนี้จะถูกกำหนดให้เพิ่มความจุให้กับระบบจีเอสเอ็ม มันสามารถทำได้โดยใช้ช่องทางแบบฮาล์ฟเรทและการเข้ารหัสเสียงพูดแบบที่มีบิตเรทต่ำ ด้วยเหตุนี้ทำให้ใช้งานหนึ่งช่องทางได้ถึงสองเท่าจากเดิมในกรณีที่ดีที่สุด ซึ่งโทรศัพท์เคลื่อนที่เครื่องแรกที่สนับสนุนการเข้ารหัสแบบฮาล์ฟเรทมีขึ้นใน ค.ศ. 1997

การเข้ารหัสแบบฮาล์ฟเรทมีความก้าวหน้ากว่าการเข้ารหัสแบบฟูลเรทและยังให้คุณภาพใกล้เคียงกับการเข้ารหัสแบบฟูลเรทอีกด้วย

หนึ่งช่องทางของเครือข่ายจีเอสเอ็มแบบฮาล์ฟเรทจะมีบิตเรทเท่ากับ 11.4 กิโลบิตต่อวินาที ซึ่งมีบิตเรทเป็นครึ่งหนึ่งของช่องทางแบบฟูลเรท ซึ่งส่วนประกอบของบิตเรทจะถูกใช้สำหรับเสียงพูดที่ถูกเข้ารหัสและส่วนของการควบคุมความผิดพลาดเช่นเดียวกับแบบฟูลเรท

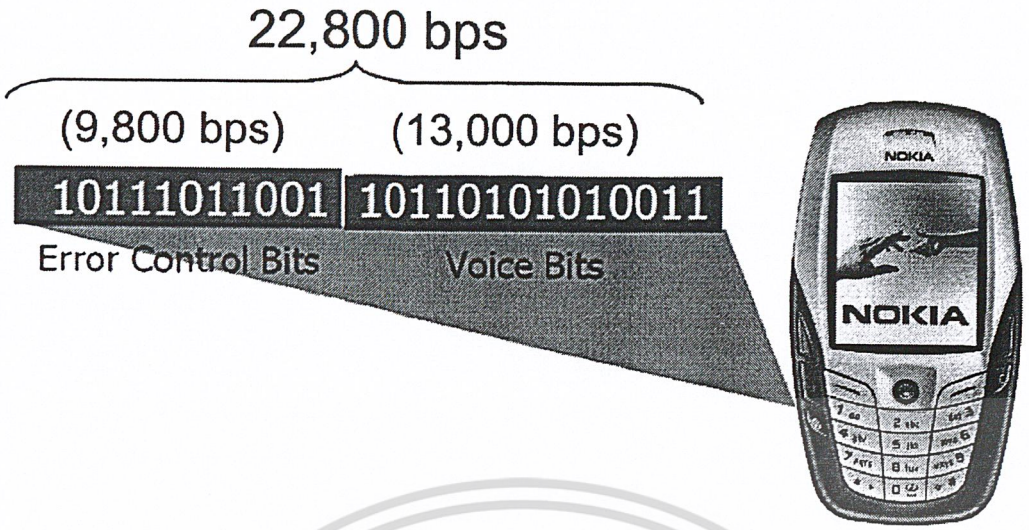


รูปที่ 4-3 แพ็กเก็ตของการเข้ารหัสแบบฮาล์ฟเรท

4.2.3 การเข้ารหัสแบบเอนฮานซ์ฟูลเรท (EFR: Enhanced Full Rate)

การเข้ารหัสแบบเอนฮานซ์ฟูลเรท ซึ่งเป็นการเข้ารหัสอันดับสามที่ถูกกำหนดสำหรับระบบจีเอสเอ็ม การเข้ารหัสแบบนี้ถูกกำหนดให้พัฒนาในเรื่องคุณภาพของเสียงบนเครือข่ายจีเอสเอ็ม ซึ่งการเข้ารหัสแบบเอนฮานซ์ฟูลเรททำให้คุณภาพเสียงที่ได้มีความชัดเจนมากกว่าการเข้ารหัสแบบฟูลเรท หรือการเข้ารหัสแบบฮาล์ฟเรท การเข้ารหัสแบบเอนฮานซ์ฟูลเรทนี้ทำได้ดีกว่าเนื่องจากใช้การเข้ารหัสที่มีอัลกอริทึม (Algorithm) ที่มีความยุ่งยากและซับซ้อนมากกว่าการเข้ารหัสแบบฟูลเรท โดยที่โทรศัพท์เคลื่อนที่เครื่องแรกที่สนับสนุนการเข้ารหัสเอนฮานซ์ฟูลเรทมีขึ้นใน ค.ศ. 1997

เนื่องจากการเข้ารหัสแบบเอนฮานซ์ฟูลเรทเหมือนการเข้ารหัสแบบฟูลเรท ต่างกันตรงที่ใช้อัลกอริทึมที่มีความยุ่งยากและซับซ้อนกว่า ดังนั้นหนึ่งช่องทางของเครือข่ายจีเอสเอ็มแบบเอนฮานซ์ฟูลเรทจะมีบิตเรทเท่ากับแบบฟูลเรทนั่นคือ 22.8 กิโลบิตต่อวินาที ซึ่งส่วนประกอบของบิตเรทจะถูกใช้สำหรับเสียงพูดที่ถูกเข้ารหัสและส่วนของการควบคุมความผิดพลาดเหมือนสองแบบก่อนหน้านี้



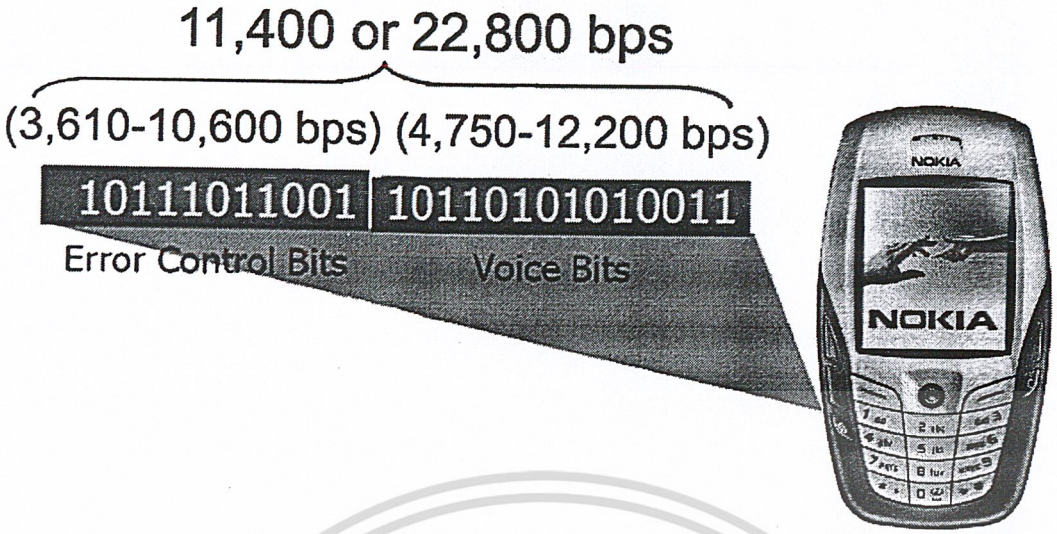
รูปที่ 4-4 แพ็กเก็ตของการเข้ารหัสแบบเอนฮานซ์ฟูลเรท

4.2.4 การเข้ารหัสแบบเอเอ็มอาร์ (AMR: Adaptive Multi-Rate)

การเข้ารหัสแบบเอเอ็มอาร์ เป็นการเข้ารหัสเสียงพูดอันดับที่สี่ที่ถูกกำหนดสำหรับระบบจีเอสเอ็ม และถูกแสดงว่าเป็นการเข้ารหัสที่มีความก้าวหน้าที่สุดจากทั้งสี่แบบ ซึ่งการเข้ารหัสแบบนี้ได้รวมข้อดีการเข้ารหัสแบบเอนฮานซ์ฟูลเรทและแบบฮาล์ฟเรทเข้าด้วยกัน ซึ่งเป็นการพัฒนาในด้านคุณภาพของเสียงและความจุนั่นเอง

การเข้ารหัสแบบเอเอ็มอาร์ได้รับความสำเร็จ เนื่องจากการเปลี่ยนแปลงการกำหนดค่าบิตเรทได้แบบไดนามิก (Dynamic) ระหว่างเสียงพูดที่ถูกเข้ารหัสและช่องสัญญาณ (channel) ด้วยวิธีนี้จึงให้คุณภาพเสียงพูดในช่องสัญญาณวิทยุในเงื่อนไขที่แตกต่างกันมีคุณภาพดีที่สุด โดยขึ้นกับสภาพแวดล้อมของช่องสัญญาณนั้นๆ เช่น เมื่อสภาพแวดล้อมของช่องสัญญาณดีแล้ว จะใช้จำนวนบิตที่ใช้สำหรับการตรวจสอบและแก้ไขความผิดพลาดลดลงและเพิ่มจำนวนบิตที่ใช้สำหรับเสียงพูดที่ถูกเข้ารหัสแทน แต่เมื่อสภาพแวดล้อมของช่องสัญญาณแย่แล้ว จะใช้จำนวนบิตที่ใช้สำหรับการตรวจสอบและแก้ไขความผิดพลาดมากขึ้นและลดจำนวนบิตที่ใช้สำหรับเสียงที่ถูกเข้ารหัสลง

การเข้ารหัสแบบเอเอ็มอาร์ขึ้นกับสภาพแวดล้อมของช่องสัญญาณ ดังนั้นหนึ่งช่องทางของเครือข่ายจีเอสเอ็มแบบเอเอ็มอาร์จะมีการเปลี่ยนแปลงการกำหนดค่าบิตเรทแบบไดนามิก ซึ่งจะเหมือนกับการใช้ช่องทางแบบฟูลเรทด้วยบิตเรทเท่ากับ 22.8 กิโลบิตต่อวินาที หรือจะเหมือนกับการใช้ช่องทางแบบฮาล์ฟเรทด้วยบิตเรทเท่ากับ 11.4 กิโลบิตต่อวินาที ซึ่งส่วนประกอบของบิตเรทจะถูกใช้สำหรับเสียงพูดที่ถูกเข้ารหัสและส่วนของการควบคุมความผิดพลาดเช่นเดียวกัน



รูปที่ 4-5 แพ็กเก็ตของการเข้ารหัสแบบเอเอ็มอาร์

การเข้ารหัสแบบเอเอ็มอาร์นั้นถูกพัฒนาโดย ETSI (European Telecommunications Standards Institute) ซึ่งการเข้ารหัสแบบนี้มีอยู่ 8 โหมด โดยมีบิตเรทตั้งแต่ 4.75 ถึง 12.2 กิโลบิตต่อวินาที (4.75, 5.15, 5.9, 6.7, 7.4, 7.95, 10.2 และ 12.2 กิโลบิตต่อวินาที) สำหรับบางโหมดนั้นจะเหมือนโหมดการเข้ารหัสแบบวอยซ์เทเลโฟนนี่ (voice telephony codec) ที่กำหนดไว้ก่อนแล้ว แต่ใช้สำหรับมาตรฐานอื่นๆ ยกตัวอย่างเช่น การเข้ารหัสแบบเอเอ็มอาร์ที่มีบิตเรทเท่ากับ 12.2 กิโลบิตต่อวินาที จะเป็นการเข้ารหัสเสียงพูดของมนุษย์เช่นเดียวกับการเข้ารหัสแบบเอนฮานซ์ฟูลเรท ซึ่งการเข้ารหัสแบบเอเอ็มอาร์เป็นการเข้ารหัสแบบโมโน (mono) ซึ่งบางทีเรียกว่าเอเอ็มอาร์ฟูลเรท และการเข้ารหัสแบบเอเอ็มอาร์ที่มีบิตเรทเท่ากับ 4.75 กิโลบิตต่อวินาที บางทีเรียกว่าเอเอ็มอาร์ฮาล์ฟเรท

ผลดีของการใช้การเข้ารหัสแบบเอเอ็มอาร์ คือ

1. คุณภาพของเสียงพูดดีกว่าการเข้ารหัสแบบเอนฮานซ์ฟูลเรทหรือฮาล์ฟเรท
2. ปรับปรุงความครอบคลุม
3. ทำให้คุณภาพของเสียงระหว่างเครือข่ายจีเอสเอ็มและดับบลิวซีดีเอ็มเอตรงกัน
4. รักษาความสามารถของฮาร์ดแวร์ที่ทำงานในแบบเดียวกับการเข้ารหัสแบบฮาล์ฟเรท
5. รักษาความสามารถของการพัฒนาความจุในทางเดียวกับการเข้ารหัสแบบฮาล์ฟเรท

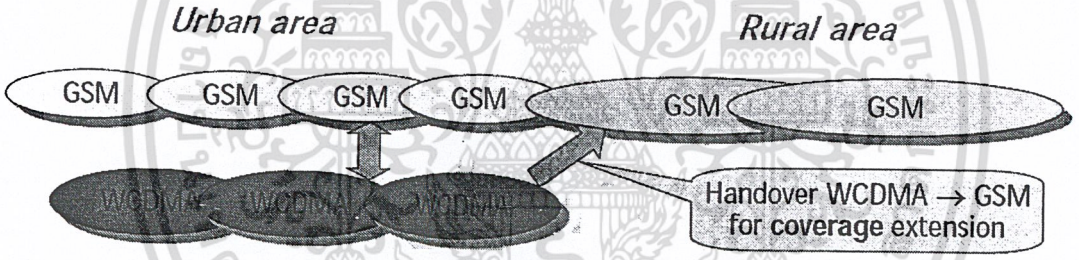
ความแตกต่างที่เห็นได้ชัดของคุณภาพของเสียงพูดที่ใช้การเข้ารหัสแบบเอเอ็มอาร์และฮาล์ฟเรท ซึ่งใช้คะแนนความเห็นเฉลี่ย (MOS: Mean Opinion Score) ที่มีสเกล (scale) 1 ถึง 5 ในการประเมินค่าของการเข้ารหัส

1. การเข้ารหัสแบบเอเอ็มอาร์ได้ค่าคะแนนความเห็นเฉลี่ยเท่ากับ 4 ซึ่งถือว่าคุณภาพเสียงที่ได้อยู่ในระดับที่ดีทีเดียว
2. การเข้ารหัสแบบฮาล์ฟเรทได้ค่าคะแนนความเห็นเฉลี่ยเท่ากับ 3.2 ซึ่งถือว่าคุณภาพเสียงที่ได้อยู่ในระดับที่ดีพอใช้

3. การเข้ารหัสแบบใดก็ตามที่มีค่าคะแนนความเห็นเฉลี่ยน้อยกว่า 3.2 ถือว่าคุณภาพเสียงที่ได้อยู่ในระดับที่แย่มาก

นอกจากนี้ในปัจจุบันโทรศัพท์เคลื่อนที่ในระบบจีเอสเอ็มยังสนับสนุนการรวมการเข้ารหัสหลายแบบเข้าด้วยกันอีกด้วยเช่น การรวมกันระหว่างการเข้ารหัสแบบเอนฮานซ์ฟูลเรทและฮาล์ฟเรท การรวมกันระหว่างการเข้ารหัสแบบเอนฮานซ์ฟูลเรท ฮาล์ฟเรทและเอเอ็มอาร์ และในอนาคตการเข้ารหัสแบบฮาล์ฟเรทก็จะถูกตัดออกจากโทรศัพท์เคลื่อนที่ในระบบจีเอสเอ็มอีกด้วย

ในระบบดับบลิวซีดีเอ็มเอการเข้ารหัสแบบเอเอ็มอาร์เป็นการเข้ารหัสสัญญาณเสียงอันเดียวที่เป็นมาตรฐาน ดังนั้นโทรศัพท์เคลื่อนที่ระบบดับบลิวซีดีเอ็มเอทั้งหมดจะสนับสนุนการเข้ารหัสแบบเอเอ็มอาร์เท่านั้น ซึ่งการเข้ารหัสแบบเอเอ็มอาร์ทำให้คุณภาพของเสียงระหว่างเครือข่ายจีเอสเอ็มและดับบลิวซีดีเอ็มเอตรงกัน เนื่องด้วยเครือข่ายดับบลิวซีดีเอ็มเอจะไม่หาใช้ได้ง่ายโดยทั่วไปในทุกๆ พื้นที่ ดังนั้นเวลาแฮนด์โอเวอร์ (Handover) ระหว่างเครือข่ายจีเอสเอ็มและดับบลิวซีดีเอ็มเอจึงเกิดขึ้นเมื่อผู้ใช้มีการเคลื่อนที่ มีแค่การเข้ารหัสแบบเอเอ็มอาร์ที่ถูกใช้ในเครือข่ายดับบลิวซีดีเอ็มเอซึ่งจะให้คุณภาพของเสียงตรงกับที่ถูกออกแบบมาจากผู้ใช้แบบดูอัลโหมด (Dual mode) ถ้าการเข้ารหัสแบบเอเอ็มอาร์ถูกใช้เหมือนกันบนเครือข่ายจีเอสเอ็มจึงทำให้คุณภาพเสียงหลังจากเกิดแฮนด์โอเวอร์เหมือนกัน

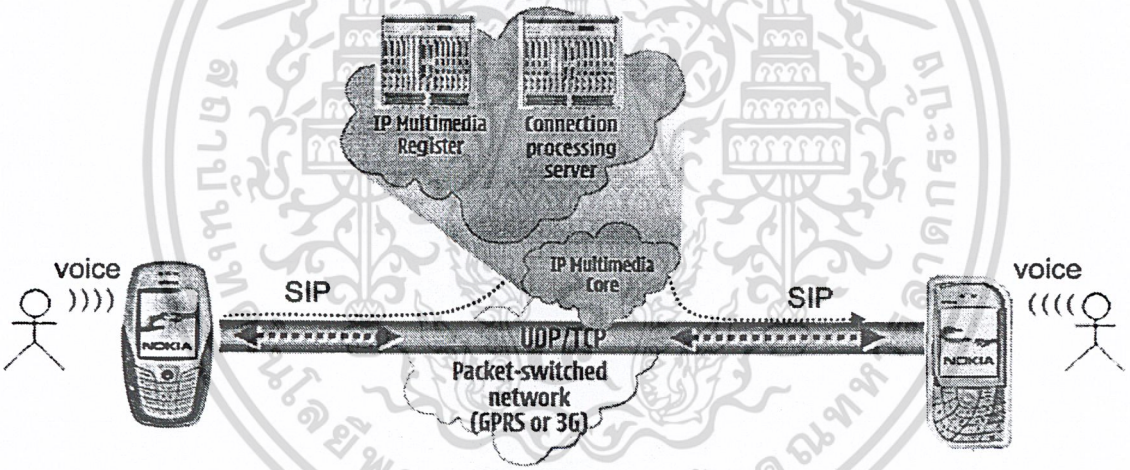


รูปที่ 4-6 การแฮนด์โอเวอร์ระหว่างระบบจีเอสเอ็มและดับบลิวซีดีเอ็มเอ

บทที่ 5
รายละเอียดของโครงงาน

5.1 ภาพรวมและองค์ประกอบหลักของโครงงาน

โทรศัพท์เคลื่อนที่ในยุค 2.5จี และ 3จี มุ่งเน้นการติดต่อสื่อสารข้อมูลข่าวสารทั้งภาพและเสียง ในรูปของสัญญาณดิจิตอลผ่านระบบจีพีอาร์เอส ทำให้ผู้ใช้สามารถสื่อสารกับคู่สนทนา และรับรู้ข้อมูลข่าวสารต่างๆ ในอินเตอร์เน็ตขณะที่ผู้ใช้เดินทาง จึงได้มีความคิดที่จะพัฒนาการติดต่อสื่อสารด้วยเสียงของโทรศัพท์เคลื่อนที่ไปเป็นแบบ VoIP ซึ่งจะทำให้มีการใช้ช่องทางการสื่อสารได้อย่างเต็มประสิทธิภาพ โดยการติดต่อสื่อสารด้วยเสียงแบบนี้ เราจะทำผ่านระบบจีพีอาร์เอสบนเครือข่ายของโทรศัพท์เคลื่อนที่ โดยข้อมูลเสียงที่ส่งจะมีลักษณะเช่นเดียวกับระบบเรียลไทม์ ซึ่งก่อนที่เราจะทำการส่งข้อมูลเสียงนั้น จะมีการบีบอัดข้อมูลเข้ามาเกี่ยวข้อง เพื่อแก้ไขข้อจำกัดในเรื่องของความเร็วและลดค่าใช้จ่ายที่ต้องใช้



รูปที่ 5-1 ภาพรวมของการทำงานของแอพพลิเคชัน

จากภาพรวมของการทำงานของแอพพลิเคชันในรูปที่ 5-1 สามารถอธิบายถึงการทำงานแบบเป็นขั้นตอนได้ดังต่อไปนี้

1. ทั้งสองฝ่ายจะต้องทำการเปิดซ็อกเก็ตเพื่อสร้างการเชื่อมต่อถึงกัน โดยฝ่ายหนึ่งทำหน้าที่เป็นเซิร์ฟเวอร์คอยฟังการเชื่อมต่อ ส่วนอีกฝ่ายหนึ่งทำหน้าที่เป็นไคลเอนท์เพื่อสร้างการเชื่อมต่อไปยังหมายเลขไอพีแอดเดรสของฝ่ายเซิร์ฟเวอร์และพอร์ทที่ถูกกำหนดไว้
2. ผู้ส่งจะทำการอัดเสียงพูดและทำการแซมปลิ้งสัญญาณเสียงพูดนั้น

3. หลังจากทำการอัดและแซมปลิ้งแล้ว ก็จะนำสัญญาณเสียงที่ได้มาผ่านการเข้ารหัสและบีบอัดให้มีขนาดเล็กลง และทำการส่งแปลงเป็นแพ็กเก็ตส่งไปยังผู้รับ
4. เมื่อฝ่ายผู้รับได้รับแพ็กเก็ตแล้ว ก็จะทำการถอดรหัสและขยายข้อมูลในแพ็กเก็ตออกมาเก็บไว้ในบัฟเฟอร์ เพื่อทำการเล่นเสียงจากบัฟเฟอร์นั้นออกมาให้ผู้รับได้ยิน

5.2 คุณสมบัติหลักของโครงงาน

คุณสมบัติหลักของโครงงาน จะเป็นความสามารถที่แอพพลิเคชันที่ออกแบบขึ้นมาสามารถทำได้ โดยมีรายละเอียดในส่วนต่างๆ ดังนี้

1. แอพพลิเคชันจะมีความสามารถในการสร้างการเชื่อมต่อโดยการเปิดซ็อกเก็ตกับโทรศัพท์เคลื่อนที่ของคู่สนทนาได้
2. แอพพลิเคชันมีความสามารถในการที่จะทำให้โทรศัพท์เคลื่อนที่ทั้งสองเครื่องที่เกิดการเชื่อมต่อถึงกัน สามารถทำหน้าที่เป็นได้ทั้งผู้ส่งและผู้รับข้อมูล
3. แอพพลิเคชันมีความสามารถในการอัดและแซมปลิ้งเสียงพูด และสามารถเล่นเสียงนั้นออกมาอย่างถูกต้อง
4. แอพพลิเคชันสามารถเข้ารหัสและบีบอีดเสียงที่ถูกอัดเข้ามาให้มีขนาดเล็กลงเพื่อที่จะให้เกิดความรวดเร็วในการส่งข้อมูลไปยังผู้รับได้

5.3 ลำดับขั้นตอนของการพัฒนา

สำหรับลำดับขั้นตอนของการพัฒนานั้น ทั้งหมดเป็นส่วนของการเขียนโปรแกรมเพื่อสร้างแอพพลิเคชันขึ้นมาโดยแบ่งออกเป็นส่วนต่างๆ ซึ่งมีลำดับการพัฒนาดังนี้ คือ

1. พัฒนาในส่วนของการเปิดซ็อกเก็ตเพื่อสร้างการเชื่อมต่อระหว่างคู่สนทนา เพื่อที่จะให้เกิดการรับส่งข้อมูลถึงกันได้
2. พัฒนาในส่วนของการอัดเสียงพูด โดยรับสัญญาณเสียงเข้ามาทำการแซมปลิ้งและบันทึกเก็บเป็นไฟล์ไว้ที่เครื่อง
3. พัฒนาในส่วนของการเข้ารหัสและบีบอัดสัญญาณเสียงที่ได้จากการอัดเสียงพูดเข้ามา เพื่อให้ข้อมูลเสียงนั้นมีขนาดเล็กลง
4. พัฒนาในส่วนของการถอดรหัสและขยายสัญญาณเสียงที่เกิดจากการเข้ารหัสและบีบอัดเพื่อให้สัญญาณเสียงกลับมาอยู่ในรูปเดิมที่สามารถเล่นได้
5. พัฒนาในส่วนของการเล่นเสียงที่ได้มาจากการถอดรหัสและขยายสัญญาณเสียง เพื่อให้ฝ่ายผู้รับได้ยินสิ่งที่ฝ่ายผู้ส่งพูดและส่งมาให้
6. ตรวจสอบแก้ไขให้การทำงานของแอพพลิเคชันทั้งหมดให้มีความถูกต้องและแม่นยำมากขึ้น โดยจะต้องพยายามทำให้แอพพลิชันตอบสนองการทำงานได้อย่างรวดเร็ว

บทที่ 6
การวิเคราะห์และออกแบบ

6.1 การวิเคราะห์และการออกแบบระบบ

การพัฒนาของเราจะใช้ภาษา C++ เป็นหลักในการพัฒนาเพราะได้มี API มาให้เหมาะสมแก่การพัฒนา ขั้นตอนการพัฒนาจะเป็นการทดลองทำบนเครื่องคอมพิวเตอร์ตั้งโต๊ะก่อนเพื่อทำการพัฒนาให้แน่ใจว่าสามารถแสดงผลได้ โดยจำลองการทำงานของโทรศัพท์เคลื่อนที่และทำการดีบัก (Debug) โปรแกรมในอีมูเลเตอร์ (Emulator) ของ Nokia Series60 ที่ทำงานบนระบบปฏิบัติการ ซิมเบียน ส่วนการพัฒนาส่วนติดต่อ Socket นั้น จำเป็นต้องใช้โทรศัพท์เคลื่อนที่ในการทดลองรับส่งข้อมูลจริงเพื่อทดสอบการทำงานของโปรแกรมที่ถูกต้อง

ในการวิเคราะห์ ออกแบบ และเขียนโปรแกรมตามหลักการของออบเจ็คต์ โดยใช้ UML (Unified Modeling Language) ซึ่งเป็นการวิเคราะห์และออกแบบระบบโดยการสร้างโมเดลขึ้นมาเพื่อจำลองออบเจ็คต์ต่างๆที่มีอยู่ในระบบ และแสดงความสัมพันธ์ระหว่างออบเจ็คต์เหล่านั้น ใช้ UML ที่ประกอบด้วย 3 ไดอะแกรมดังนี้

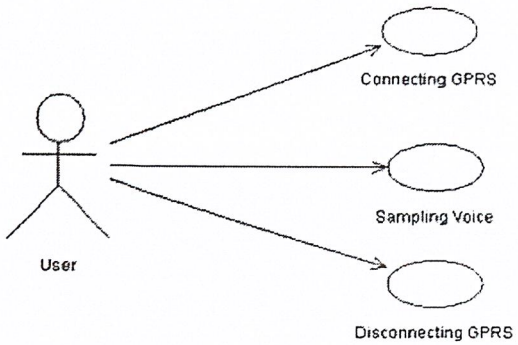
1. Use Case Diagram
2. Sequence Diagram
3. State Diagram

ในการแสดงรายละเอียดการทำงานของแอปพลิเคชันในแต่ละไดอะแกรมนั้นจะแสดงเฉพาะส่วนที่สำคัญเท่านั้น ดังแสดงต่อไปนี้

6.1.1 Use Case Diagram

เป็นไดอะแกรมที่แสดงฟังก์ชันการทำงานที่สำคัญของแอปพลิเคชันที่ยูสเซอร์สามารถเข้ามาใช้บริการได้โดยจะมี ฟังก์ชันที่สำคัญอยู่ 3 ฟังก์ชันคือ

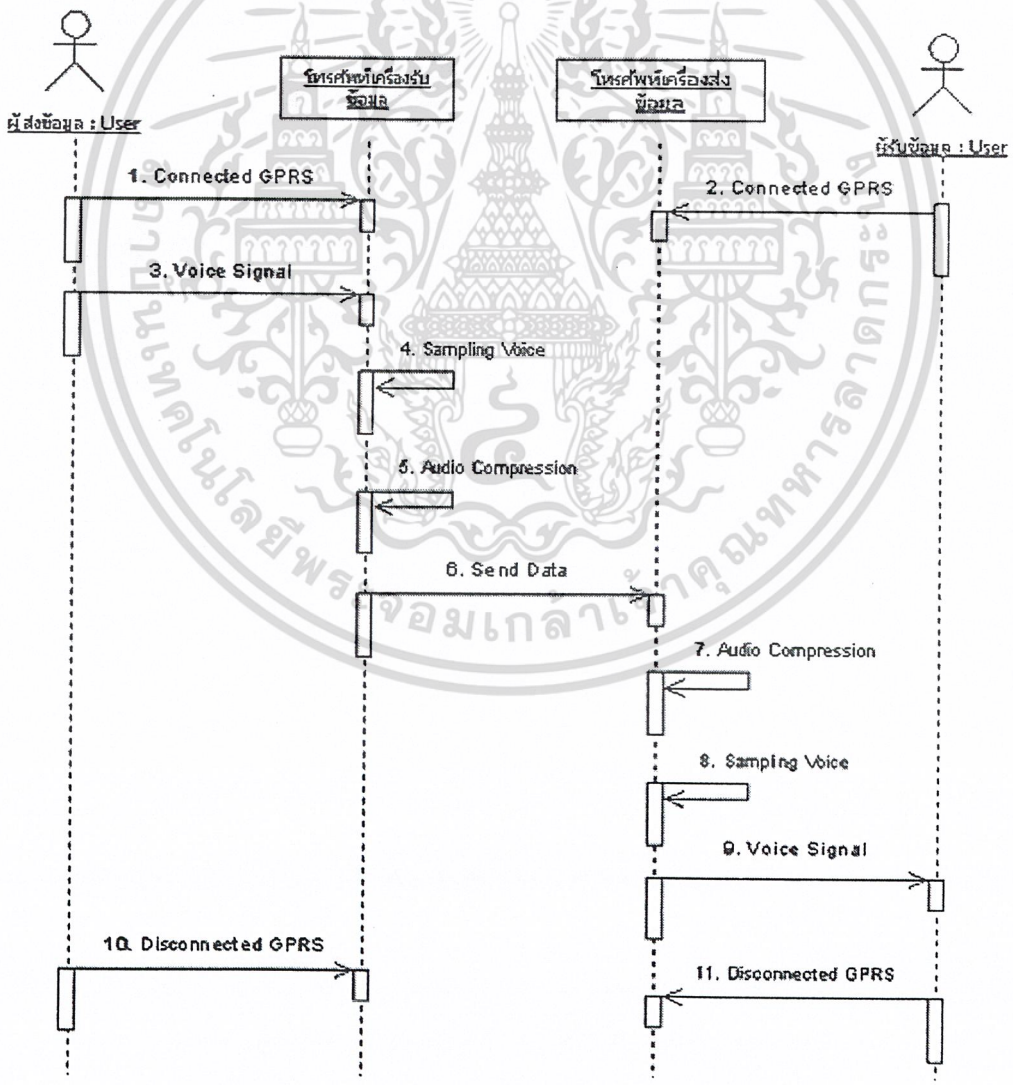
1. Connecting GPRS เป็นฟังก์ชันที่ใช้ในทำการเปิดรับการติดต่อผ่านระบบจีพีอาร์เอส
2. Sampling Voice เป็นฟังก์ชันที่ใช้ในทำการแซมปิ้งสัญญาณเสียง
3. Disconnecting GPRS เป็นฟังก์ชันที่ใช้ในทำการปิดรับการติดต่อผ่านระบบจีพีอาร์เอส



รูปที่ 6-1 Use Case Diagram ของโปรแกรม

6.1.2 Sequence Diagram

เป็นไดอะแกรมที่แสดงลำดับการทำงานภายใน Use Case Diagram อย่างมีลำดับขั้นตอน และการทำงานร่วมกันของอ็อบเจ็กต์ต่างๆ



รูปที่ 6-2 การทำงานระหว่างโทรศัพท์เครื่องรับและโทรศัพท์เครื่องส่ง

ลำดับการทำงานระหว่างโทรศัพท์เคลื่อนที่เครื่องรับและเครื่องส่งมีขั้นตอนดังนี้ คือ

1. เมื่อผู้ใช้ต้องการส่งข้อมูลจะเลือกให้โทรศัพท์เคลื่อนที่เป็นเครื่องส่ง มีการกำหนดค่าต่างๆในการติดต่อผ่านระบบจีพีอาร์เอส
2. ผู้รับข้อมูลก็ต้องติดต่อผ่านระบบจีพีอาร์เอสเช่นกัน
3. โทรศัพท์เครื่องส่งข้อมูลจะทำการรับสัญญาณเสียงจากผู้ส่ง
4. โทรศัพท์เครื่องส่งจะทำการแซมปิ้งสัญญาณเสียง
5. โทรศัพท์เครื่องส่งจะทำการเข้ารหัสและบีบอัดข้อมูล
6. โทรศัพท์เครื่องส่งจะทำการส่งข้อมูล
7. โทรศัพท์เครื่องรับจะรับข้อมูล แล้วมาทำการถอดรหัสและขยายข้อมูล
8. โทรศัพท์เครื่องรับจะทำการแซมปิ้งกลับมาเป็นสัญญาณเสียง
9. โทรศัพท์เครื่องรับส่งสัญญาณเสียงออกมา
10. เมื่อไม่ต้องการส่งสัญญาณเสียง ก็ทำการตัดการเชื่อมต่อของเครื่องผู้ส่ง
11. เครื่องผู้รับทำการตัดการเชื่อมต่อ

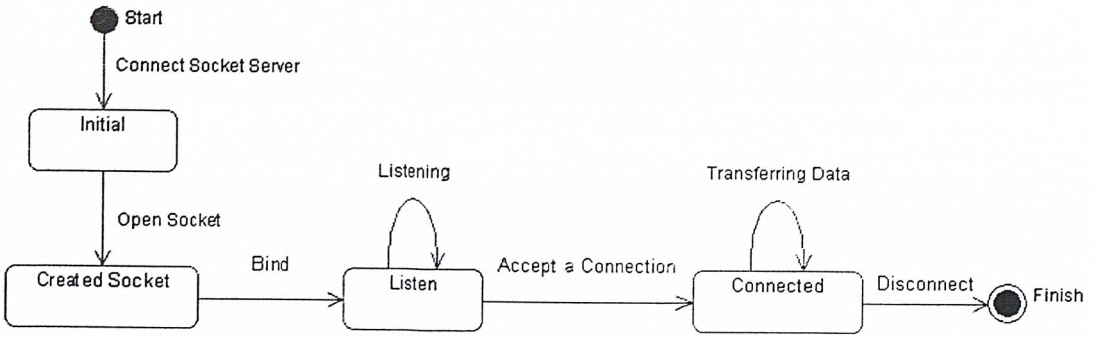
6.1.3 State Diagram

เป็นไดอะแกรมที่อธิบายพฤติกรรมของซอฟต์แวร์และการเปลี่ยนแปลงสถานะเมื่อมีเหตุการณ์ต่างๆ เกิดขึ้น ซึ่งในแอปพลิเคชันนี้มีซอฟต์แวร์อยู่ 2 ส่วนที่สามารถแสดงเสตทไดอะแกรมดังต่อไปนี้ คือ

6.1.3.1 State Diagram ของเซิร์ฟเวอร์

ส่วนของเซิร์ฟเวอร์จะมี 4 สถานะ ดังนี้

1. Initial State : เป็นส่วนที่จะทำการติดต่อไปยังซ็อกเก็ตเซิร์ฟเวอร์ของระบบปฏิบัติการซิมเบียน เพื่อทำการตั้งค่าเริ่มต้นต่างๆ ให้สามารถใช้ซ็อกเก็ต API ได้
2. Created Socket State : เป็นการเปิดซ็อกเก็ต โดยมีการกำหนดค่าในการเชื่อมต่อต่างๆ เช่น การกำหนดชนิดซ็อกเก็ตและโปรโตคอลที่ใช้
3. Listen State : เป็นการรอการเข้ามาติดต่อของไคลเอนท์ โดยที่ต้องมีการกำหนดหมายเลขพอร์ทก่อน
4. Connected State : เป็นสถานะที่ทำการเชื่อมต่อกับไคลเอนท์เรียบร้อยแล้ว ในสถานะนี้เซิร์ฟเวอร์และไคลเอนท์จะสามารถส่งข้อมูลถึงกันได้

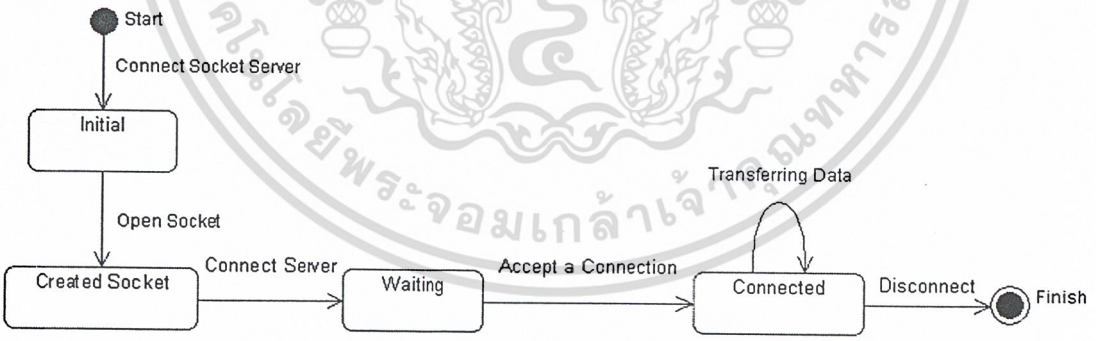


รูปที่ 6-3 สถานะการทำงานฝั่งเซิร์ฟเวอร์

6.1.3.2 State Diagram ของไคลเอนท์

ส่วนของไคลเอนท์จะมี 4 สถานะ ดังนี้

1. Initial State : เป็นส่วนที่จะทำการติดต่อไปยังซ็อกเก็ตเซิร์ฟเวอร์ของระบบปฏิบัติการซิมเบียน เพื่อทำการตั้งค่าเริ่มต้นต่างๆ ให้สามารถใช้ซ็อกเก็ต API ได้
2. Created Socket State : เป็นการเปิดซ็อกเก็ต โดยมีการกำหนดค่าในการเชื่อมต่อต่างๆ เช่น การกำหนดชนิดซ็อกเก็ตและโปรโตคอลที่ใช้
3. Waiting State : เป็นการเข้าไปติดต่อกับเซิฟเวอร์ โดยที่ต้องมีการกำหนดหมายเลขไอพีแอดเดรสและหมายเลขพอร์ท
4. Connected State : เป็นสถานะที่ทำการเชื่อมต่อกับเซิฟเวอร์เรียบร้อยแล้ว ในสถานะนี้เซิร์ฟเวอร์และไคลเอนท์จะสามารถส่งข้อมูลถึงกันได้



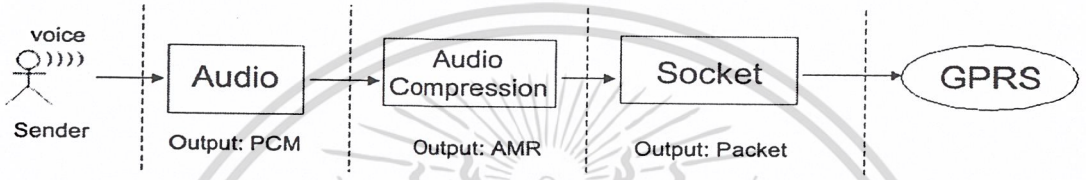
รูปที่ 6-4 สถานะการทำงานฝั่งไคลเอนท์

6.2 โครงสร้างของโปรแกรม

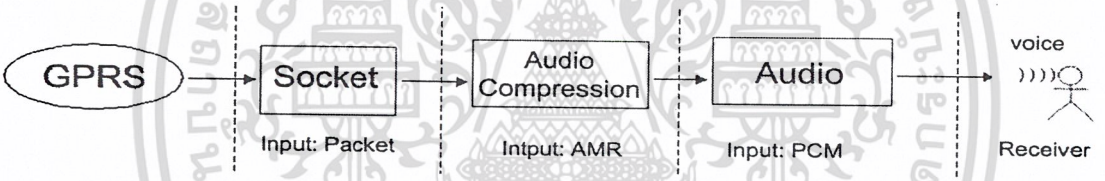
โปรแกรมที่พัฒนาขึ้นนั้นเป็นโปรแกรมที่รองรับการทำงานบนโทรศัพท์เคลื่อนที่ของบริษัท Nokia รุ่น 6600 บนระบบปฏิบัติการ Symbian OS Series 60 เวอร์ชัน 2.0 ซึ่งการทำงานของโปรแกรมนั้นเป็นการเชื่อมต่อกันของโทรศัพท์เคลื่อนที่สองเครื่องผ่านระบบจีพีอาร์เอส

โครงสร้างของโปรแกรมนั้นมีส่วนประกอบอยู่ 3 ส่วนหลัก โดยแต่ละส่วนนั้นมีหน้าที่ในการรับผิดชอบที่แตกต่างกันออกไปดังนี้ คือ

1. Audio เป็นส่วนที่ทำหน้าที่เกี่ยวกับการอัดเสียงพูดและทำการแซมปลิ้งสัญญาณเสียงพูด
2. Audio Compression เป็นส่วนที่ทำหน้าที่เกี่ยวกับการเข้ารหัสและบีบอัดข้อมูลที่ได้มาจากการแซมปิ้ง หรือทำการถอดรหัสและขยายข้อมูลที่ถูกส่งผ่านทางระบบจีพีอาร์เอส
3. Socket เป็นส่วนที่มีหน้าที่เกี่ยวกับการกำหนดค่าการส่งข้อมูลผ่านทางระบบจีพีอาร์เอส การประกาศให้อุปกรณ์อื่นรู้ว่ามีบริการอยู่ การรอรับการเชื่อมต่อจากอุปกรณ์ที่ขอการเชื่อมต่อ การให้สิทธิกับอุปกรณ์ที่ขอเชื่อมต่อ และการส่งข้อมูลให้อุปกรณ์ที่ขอเชื่อมต่อ



รูปที่ 6-5 รูปแบบการทำงานของโปรแกรมเมื่อส่งเสียง



รูปที่ 6-6 รูปแบบการทำงานของโปรแกรมเมื่อรับเสียง

6.3 เครื่องมือที่ใช้ในการพัฒนา

1. Microsoft visual C++ อีดิทเตอร์สำหรับใช้เขียน ภาษา C++
2. อีมูเลเตอร์สำหรับทดลองการแสดงผลบนโทรศัพท์เคลื่อนที่โดยที่ไม่ต้องใช้มือถือจริงมาทดลอง
3. โทรศัพท์เคลื่อนที่ Nokia รุ่น 6600
4. Series 60 SDK 2.0 สำหรับ Symbian OS เป็นโปรแกรมที่ใช้พัฒนาแอพพลิเคชันบน Symbian OS โดยใช้ภาษา C++ ซึ่งสามารถเข้ากันได้กับโทรศัพท์เคลื่อนที่ของ Nokia รุ่น 6600
5. Nokia SIP Plug-In เวอร์ชั่น 1.0 สำหรับ Visual C++
6. Nokia Ethernet Plug-In สำหรับการจำลองให้อีมูเลเตอร์เชื่อมต่ออินเตอร์เน็ตได้

6.4 แผนการดำเนินงาน

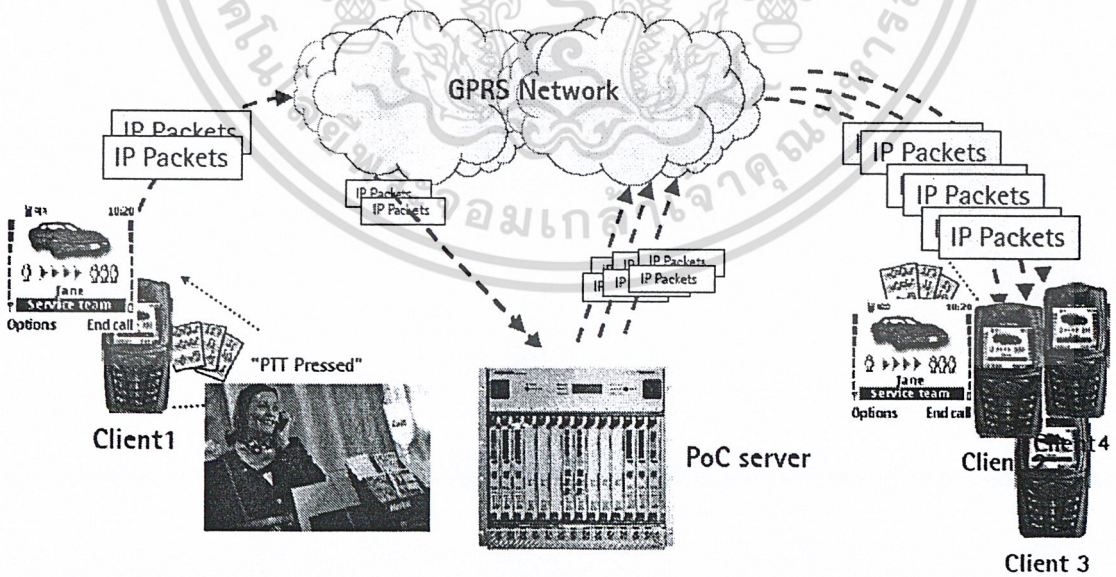
การดำเนินงานในภาคเรียนที่ 1 มีดังตารางต่อไปนี้

ID	Task Name	Start	Finish
1	สำรวจและศึกษาผลงานที่ใกล้เคียงทั้งภายในและภายนอกประเทศ	7/6/2004	10/9/2004
2	กำหนดขอบเขตและคุณสมบัติของโปรเจคโดยละเอียด ครบถ้วน และชัดเจน	14/6/2004	25/6/2004
3	ศึกษาความรู้และเทคโนโลยีที่จำเป็นต้องใช้ประกอบการทำโปรเจค	21/6/2004	10/9/2004
4	วางโครงสร้างหลักและย่อยของโปรเจคพร้อมแบ่งความรับผิดชอบ	28/6/2004	12/7/2004
5	ดำเนินงานและพบปะ เพื่อแก้ปัญหาที่อาจจะเกิดขึ้นเป็นระยะๆ และรายงานความคืบหน้า	7/6/2004	8/10/2004
6	จัดทำรายงานและพรีเซนเทชัน เพื่อนำเสนอความคืบหน้า	6/9/2004	12/10/2004

ตารางที่ 6-1 การดำเนินงานในภาคเรียนที่ 1

ผลจากการสำรวจพบว่าผลงานที่ใกล้เคียงดังนี้

PoC: Push to talk over cellular (http://www.forum.nokia.com) เป็นบริการใหม่ที่เป็น VoIP บนเครื่อข่ายจีเอสเอ็ม ซึ่งเป็นเพียงการสื่อสารแบบ Half duplex VoIP โดยผ่าน PoC Server



รูปที่ 6-7 รูปแบบการทำงานของ PoC

การดำเนินงานในภาคเรียนที่ 2 มีดังตารางต่อไปนี้

ID	Task Name	Start	Finish	พ.ย. 2004	ธ.ค. 2004	ม.ค. 2005	ก.พ. 2005
1	พัฒนาซอฟต์แวร์ในการทำ Voice Streaming	1/11/2004	10/12/2004				
2	พัฒนาซอฟต์แวร์ในการติดต่อระบบจีพีอาร์เอสด้วย Socket	1/11/2004	10/12/2004				
3	รวมซอฟต์แวร์สองส่วนเข้าด้วยกัน	13/12/2004	11/2/2005				
4	ตรวจสอบและแก้ไขข้อผิดพลาดของซอฟต์แวร์	3/1/2005	18/2/2005				
5	ดำเนินงานและพบปะ เพื่อแก้ปัญหาที่อาจจะเกิดขึ้นเป็นระยะๆ และรายงานความคืบหน้า	1/11/2004	11/3/2005				
6	จัดทำปริญญานิพนธ์และพรีเซนเทชัน เพื่อนำเสนอครั้งสุดท้าย	15/2/2005	11/3/2005				

ตารางที่ 6-2 การดำเนินงานในภาคเรียนที่ 2

บทที่ 7
การทดลองและผลการทดลอง

การทดลองการทำงานของแอพพลิเคชันนั้นเป็นสิ่งที่จำเป็น เพื่อทดสอบผลของการทำงานให้เป็นไปตามที่เราคาดหวังหรือกำหนดไว้ และเพื่อทราบปัญหาต่างๆ ที่เกิดขึ้นเพื่อที่จะได้หาทางแก้ไขปัญหานั้นๆ ได้ อีกทั้งยังสามารถนำข้อมูลที่ได้มาพัฒนาให้แอพพลิเคชันของเรามีความสามารถเพิ่มมากขึ้นได้อีกด้วย

7.1 อุปกรณ์ที่ใช้ในการทดลอง

1. โทรศัพท์เคลื่อนที่ที่ใช้ระบบปฏิบัติการซิมเบียนเวอร์ชัน 7.0s และรองรับโปรโตคอลซิพ เช่น Nokia 6260, Nokia 6600 และ Nokia 7610 จำนวน 2 เครื่อง
2. เครื่องคอมพิวเตอร์ที่ลงโปรแกรม Series 60 SDKs เวอร์ชัน 2.0 และ Series 60 SIP Plug-in เวอร์ชัน 1.0
3. บลูทูธโมดูลเพื่อใช้ในการติดต่อกันระหว่างโทรศัพท์เคลื่อนที่และเครื่องคอมพิวเตอร์

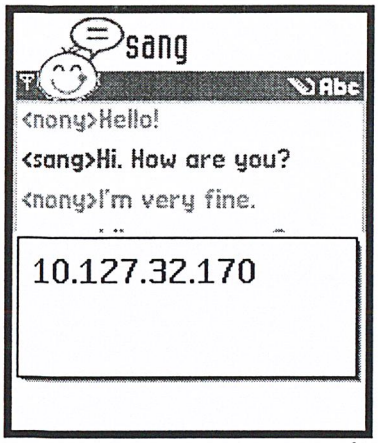
7.2 สิ่งที่ต้องการทดลอง

1. การเปิดซ็อกเก็ตเพื่อสร้างการเชื่อมต่อระหว่างโทรศัพท์เคลื่อนที่ 2 เครื่อง
2. การบันทึกเสียงพูดลงในโทรศัพท์เคลื่อนที่
3. การเข้ารหัส การถอดรหัส การบีบอัดและการขยายข้อมูล
4. การรับส่งแพ็กเก็ตของข้อมูลระหว่างโทรศัพท์เคลื่อนที่ 2 เครื่อง

7.2.1 การเปิดซ็อกเก็ต

การเปิดซ็อกเก็ตเพื่อสร้างการเชื่อมต่อระหว่างโทรศัพท์เคลื่อนที่ทั้งสองนั้น เป็นสิ่งจำเป็นที่สุดของแอพพลิเคชันของเรา เพราะการที่เราจะส่งข้อมูลใดๆ ระหว่างโทรศัพท์เคลื่อนที่สองเครื่องนั้น เราจะต้องมีการเปิดซ็อกเก็ตเพื่อสร้างการเชื่อมต่อเสียก่อน

สำหรับการทดลองการเปิดซ็อกเก็ตนั้น จะเริ่มต้นด้วยการให้โทรศัพท์เคลื่อนที่ทั้งสองทำการรีจิสเตอร์ไปยังซิพเซิร์ฟเวอร์ (SIP Server) ที่กำหนดไว้เสียก่อน จากนั้นให้ฝั่งใดฝั่งหนึ่งทำหน้าที่เป็นไคลเอนท์ร้องขอการเชื่อมต่อไปยังอีกฝั่งหนึ่งให้เป็นเซิร์ฟเวอร์และเปิดพอร์ทรอการเชื่อมต่อจากฝ่ายที่ร้องขอ จากนั้นฝ่ายที่ร้องขอก็จะทำการเชื่อมต่อไปยังหมายเลขไอพีแอดเดรสของฝั่งที่รอการเชื่อมต่อและหมายเลขพอร์ทที่กำหนดไว้



รูปที่ 7-1 หมายเลขไอพีแอดเดรสของฝ่ายเซิร์ฟเวอร์ที่ถูกส่งให้ฝ่ายไคลเอนท์

7.2.2 การบันทึกเสียงพูดลงในโทรศัพท์เคลื่อนที่

สำหรับการบันทึกเสียงพูดลงในโทรศัพท์เคลื่อนที่นั้นเราใช้ API ที่มีมาให้ในการบันทึกเสียง โดยเราทำการทดลองด้วยการบันทึกเสียงพูดเพื่อบันทึกเป็นไฟล์ จากนั้นจึงนำไฟล์ที่ได้กลับมาลองเล่นดู ถ้าเสียงที่ได้ออกมาเหมือนกับเสียงที่ถูกบันทึก แสดงว่าการบันทึกเสียงพูดของเราทำงานได้อย่างถูกต้อง และในทางกลับกันถ้าเสียงที่ได้ออกมาไม่เหมือนกับเสียงที่ถูกบันทึก เราก็ต้องมาตรวจสอบแก้ไขใหม่อีกครั้ง โดยเราต้องแน่ใจด้วยว่าฟังก์ชันของการเล่นเสียงนั้นทำงานได้อย่างถูกต้อง

7.2.3 การเข้ารหัส การถอดรหัส การบีบอัดและการขยายข้อมูล

ในส่วนนี้หน้าที่โดยรวมก็คือ พยายามทำให้เสียงที่ถูกบันทึกมามีขนาดที่เล็กลงด้วยการเข้ารหัสและบีบอัด และต้องใช้การถอดรหัสและการขยายข้อมูล ทำให้เสียงนั้นกลับมาเหมือนเดิม โดยในการทดลองเราจะนำไฟล์เสียงที่ได้จากการบันทึกมาทำการเข้ารหัสและบีบอัดแล้วเก็บลงบัฟเฟอร์ จากนั้นจึงนำข้อมูลในบัฟเฟอร์มาถอดรหัสและขยายออกเป็นสัญญาณเสียงดังเดิม ทดลองนำเอาไฟล์เสียงที่ได้จากการถอดรหัสและขยายแล้วมาเปรียบเทียบกับไฟล์เสียงก่อนที่จะมีการเข้ารหัสและบีบอัดว่าเหมือนกันหรือไม่ ถ้าเหมือนกันก็แสดงว่าการเข้ารหัส การถอดรหัส การบีบอัดและการขยายข้อมูลทำงานได้อย่างถูกต้อง

7.2.4 การรับส่งแพ็กเก็ตของข้อมูล

หลังจากที่เราทำการบันทึกเสียงพูด และผ่านขั้นตอนของการเข้ารหัสและบีบอัดมาเรียบร้อยแล้วนั้น ส่วนต่อไปก็จะเป็นส่วนที่ข้อมูลเกิดการรับส่งระหว่างคู่สนทนาจริงๆ โดยข้อมูลที่รับส่งกันนั้นจะถูกแปลงให้อยู่ในรูปของแพ็กเก็ตก่อน ดังนั้นเราจึงต้องแน่ใจด้วยว่าการรับส่งแพ็กเก็ตของข้อมูลนั้นทำงานได้อย่างถูกต้อง สำหรับการทดลองของการรับส่งแพ็กเก็ตของข้อมูลเราจึงทดลองโดยการนำไฟล์ที่เป็นข้อความล้วนมาใช้ในการทดสอบ ทั้งขั้นตอนการแปลงเป็นแพ็กเก็ต การส่งแพ็กเก็ตและการนำข้อมูลในแพ็กเก็ตไปใช้ เพราะเราสามารถที่จะตรวจสอบการทำงานได้ง่ายโดยการดูจากข้อความภายในไฟล์ จะทำให้เราสามารถรู้ได้ว่ามีส่วนใดบ้างที่สูญหายไป

และจากการทดลองในขั้นนี้ทำให้แอพพลิชันของเรามีความสามารถในการรับส่งไฟล์ถึงกันระหว่างโทรศัพท์เคลื่อนที่ได้อีกด้วย

รูปที่ 7-2 สถานะการทำงานที่เกิดขึ้นระหว่างการรับส่งไฟล์

บทที่ 8
บทวิจารณ์และสรุป

8.1 บทวิจารณ์และสรุป

แอพพลิเคชันที่จัดทำขึ้นสามารถทำได้ตามความต้องการของผู้จัดทำดังนี้ แอพพลิเคชันสามารถที่จะเปิดซ็อกเก็ตเพื่อสร้างการเชื่อมต่อไปยังโทรศัพท์ของคู่สนทนาได้อย่างถูกต้อง สามารถส่งเสียงได้ในลักษณะของการกดปุ่มแล้วพูด และสามารถส่งไฟล์ถึงกันได้ นั่นก็คือแอพพลิเคชันที่พัฒนาขึ้นมาสามารถเปลี่ยนแปลงวิธีการเชื่อมต่อได้ตามที่ผู้จัดทำต้องการจากการเชื่อมต่อแบบ Circuit Switched Data มาเป็นแบบ Packet Switched Data

ส่วนในเรื่องของการที่ต้องการจะลดค่าใช้จ่ายที่เกิดจากการสื่อสารแบบเดิมๆ นั้น เราจะให้ได้จากในปัจจุบันเมื่อเทียบระหว่างค่าใช้บริการในการโทรและค่าใช้บริการจีพีอาร์เอส เราจะเห็นได้ว่าค่าใช้บริการจีพีอาร์เอสมีสัดส่วนที่ถูกกว่ามาก โดยข้อมูลเสียงขนาด 1 วินาที เมื่อเข้ารหัสแบบเอเอ็มอาร์แล้วจะมีขนาดประมาณ 1 กิโลไบต์ ดังนั้นเมื่อการสื่อสารของแอพพลิเคชันเราเลือกที่จะใช้การติดต่อสื่อสารผ่านทางจีพีอาร์เอส ทำให้ภาระค่าใช้จ่ายที่จะเกิดขึ้นลดลงมาก

8.2 ปัญหาที่พบ

1. Series 60 SDK 2.0 สำหรับระบบปฏิบัติการซิมเบียนได้ลบ API บางตัวไป ทำให้ไม่สามารถอัดเสียงและเข้ารหัสแบบเอเอ็มอาร์ได้ทันที จึงต้องทำให้อัดเสียงและเข้ารหัสแบบพีซีเอ็ม (PCM) ก่อน แล้วหลังจากนั้นถึงจะนำเสียงที่เข้ารหัสแบบพีซีเอ็มมาเข้ารหัสแบบเอเอ็มอาร์อีกที และเวลาที่ต้องการฟังเสียงเราต้องนำเสียงที่ถอดรหัสแบบเอเอ็มอาร์แล้วจะได้เป็นพีซีเอ็มจึงจะฟังได้
2. หลังที่เราได้ไฟล์เสียงมาแล้ว เวลาจะทำการเข้ารหัสและบีบอัดข้อมูล หรือเวลาที่เราถอดรหัสและขยายข้อมูล เครื่องโทรศัพท์เคลื่อนที่ทำงานช้า
3. ในการส่งข้อมูลผ่านระบบจีพีอาร์เอสนั้น จะมีความล่าช้าในการส่งไม่เท่ากัน ซึ่งจะขึ้นกับทราฟฟิก (traffic) ของเครือข่าย
4. มีนักพัฒนาโปรแกรมจำนวนน้อยที่เข้ามาทดลองทำงานกับระบบปฏิบัติการซิมเบียน จึงมีตัวอย่างและความรู้ที่จะอ้างอิงไปถึงน้อยมาก

8.3 แนวทางการปรับปรุงและพัฒนาต่อไป

ภายในปริญญานิพนธ์เล่มนี้ได้อธิบายโปรแกรมในส่วนสำคัญๆไว้อย่างครบถ้วน ง่ายต่อการศึกษาและทำความเข้าใจ เช่น ฟังก์ชันการทำงานของซ็อกเก็ต API โดยที่ในอนาคตที่มีเครือข่ายแบบ 3 จี จะทำให้มีอัตราการรับส่งข้อมูลที่สูงขึ้น ดังนั้นความล่าช้าในการส่งข้อมูลผ่านระบบจีพีอาร์เอสจะไม่เกิดขึ้น

ในการพัฒนาต่อสามารถที่จะทำให้การส่งข้อมูลเสียงนั้นได้รวดเร็วมากขึ้น และสามารถทำการพัฒนาให้มีไคลเอนต์ ได้หลายเครื่องรับข้อมูลเสียงได้พร้อม ๆ กันภายในครั้งเดียว

ภาคผนวก ก
ตัวอย่างซอสโค้ดการทำงานหลักของโปรแกรม

1. ซอสโค้ดในส่วนของการสร้างการเชื่อมต่อจากไคลเอนต์ไปยังเซิร์ฟเวอร์

```
void CSocketsEngine::ConnectL(TUint32 aAddr)
  {
  // Open channel to Socket Server
  User::LeaveIfError(iSocketServ.Connect());

 // Initiate attempt to connect to a socket by IP address
  if (iEngineStatus == ENotConnected)
    {
    // Open a TCP socket
    User::LeaveIfError(iSocket.Open(iSocketServ, KAfInet, KSockStream, KProtocolInetTcp));

    // Set up address information
          iAddress.SetPort(iPort);
          iAddress.SetAddress(aAddr);

   // Initiate socket connection
          iSocket.Connect(iAddress, iStatus);
          ChangeStatus(EConnecting);

   // Start a timeout
          iTimer->After(KTimeOut);

         SetActive();
    }
}
```

เป็นส่วนที่จะทำการเชื่อมต่อไปยังเครื่องเซิร์ฟเวอร์ โดยต้องทำการประกาศตัวแปร iSocketServ ชนิดของข้อมูลเป็น RSocketServ เพื่อทำการติดต่อไปยังซ็อกเก็ตเซิร์ฟเวอร์ของระบบปฏิบัติการซิมเบียน

และต้องประกาศตัวแปร iSocket ชนิดของข้อมูลเป็น RSocket เพื่อใช้สำหรับเชื่อมต่อไปยัง Server โดยใช้ฟังก์ชัน Open() ในการกำหนดชนิดของซ็อกเก็ตและโปรโตคอลที่ใช้ โดยที่ตัวแปร iAddress ชนิดของ จะทำการเก็บค่าแอดเดรสและพอร์ทของเครื่องที่ต้องการติดต่อ จากนั้นใช้คำสั่ง Connect สำหรับติดต่อไปยังเครื่องที่ต้องการโดยส่งตัวแปรแอดเดรสเข้าไป ทำการ SetActive เพื่อรอรับการเชื่อมต่อ

2. ซอสโค้ดในส่วนของการรอรับการเชื่อมต่อจากไคลเอนต์ไปยังเซิร์ฟเวอร์

```
void CSocketsEngine::ListenL( )
  {
  // Open channel to Socket Server
 User::LeaveIfError(iSocketServ.Connect());

 // Initiate attempt to connect to a socket by IP address
  if (iEngineStatus == ENotConnected)
     {
     // Open a TCP socket
     User::LeaveIfError(iSocket.Open(iSocketServ, KAfInet, KSockStream, KProtocolInetTcp));

     // Set up port
            iAddress.SetPort(iPort);
    // Bind port
            iSocket.Bind(iAddress);

     // Listen for connection from client
            iSocket.Listen(KSizeofListenQueue);
    // Create blank socket
            iServSocket.Open(iSocketServ);
     // Initiate socket connection
            iSocket.Accept(iServSocket, iStatus);
            ChangeStatus(EAccepting);
            SetActive();
     }
 }
```

เป็นส่วนที่จะทำการเชื่อมต่อไปยังเครื่องเซิร์ฟเวอร์ โดยต้องทำการประกาศตัวแปร iSocketServ ชนิดของข้อมูลเป็น RSocketServ เพื่อทำการติดต่อไปยังซ็อกเก็ตเซิร์ฟเวอร์ของระบบปฏิบัติการซิมเบียน และต้องประกาศตัวแปร iSocket ชนิดของข้อมูลเป็น RSocket เพื่อใช้สำหรับ Bind โดยที่ต้องกำหนดค่าหมายเลขพอร์ทที่ตัวแปร iAddress ก่อน จากนั้นจะต้องใช้คำสั่ง Listen สำหรับรอการเชื่อมต่อจากไคลแอนท์ และฟังก์ชัน Accept ยอมรับการ หลังจากทำการ SetActive เพื่อไปทำอย่างอื่นต่อไป

3. ซอสโค้ดในส่วนของการรอรับข้อมูล

```
void CSocketsReader::IssueRead()
  {
  // Initiate a new read from socket into iBuffer
  __ASSERT_ALWAYS(!IsActive(), User::Panic(KpanicSocketsEngineRead, EsocketsBadState));
  iSocket.RecvOneOrMore(iBuffer, 0, iStatus, iDummyLength);
  SetActive();
  }
```

ฟังก์ชัน RecvOneOrMore ซึ่งเป็นฟังก์ชันในคลาส Rsocket ใช้ในการรับข้อมูลที่เข้ามาใส่ใน iBuffer ที่สร้างเอาไว้ล่วงหน้า และทำการ SetActive เพื่อรอรับข้อมูลถัดไป

4. ซอสโค้ดในส่วนของการส่งข้อมูล

```
void CSocketsWriter::SendNextPacket()
  {
  if (iTransferBuffer.Length() > 0)
    {
    // Move data from transfer buffer to actual write buffer
    iWriteBuffer = iTransferBuffer;
    iTransferBuffer.Zero();

    iSocket.Write(iWriteBuffer, iStatus); // Initiate actual write

    // Request timeout
        iTimer->After(iTimeOut);
        SetActive();
```

```
            iWriteStatus = Esending;
        }
    else
        {
            iWriteStatus = Ewaiting;
        }
    }
```

ฟังก์ชัน Write ซึ่งเป็นฟังก์ชันในคลาส Rsocket ใช้ในการส่งข้อมูลที่อยู่ใน iWriteBuffer ผ่านทาง iSocket และทำการ SetActive เพื่อส่งข้อมูลถัดไป

5. ซอสโค้ดโค้ดในส่วนของการตัดการเชื่อมต่อ

```
void CSocketsEngine::Disconnect()
    {
    __ASSERT_ALWAYS(iEngineStatus == Econnected, User::Panic(KpanicSocketsEngine,
EsocketsBadState));

    // cancel all outstanding operations
    // since we are connected, the only possibilities are read and write
    iSocketsReader->Cancel();
    iSocketsWriter->Cancel();

     iSocket.Close();
     ChangeStatus(EnotConnected);
    }
```

ฟังก์ชัน Close ซึ่งเป็นฟังก์ชันในคลาส Rsocket ใช้ในการตัดการเชื่อมต่อ

6. ซอสโค้ดโค้ดในส่วนของการอัดเสียง

```
// RAW -> PCM
const TUid KmmfUidControllerAudio = {0x101F5022};
const Tuid KUidFormatRAWRead  = {0x101F5C16};
```

```
const Tuid KUidFormatRAWWrite = {0x101F5C17};

// An existing sound sample file
_LIT(KrecorderFile, "C:\\System\\Apps\\Sound\\record.pcm");

void CrecordEngine::RecordL()
  {
   iMdaAudioRecorderUtility->OpenFileL(KrecorderFile,KmmfUidControllerAudio,
                                       KmmfUidControllerAudio,KUidFormatRAWWrite,
                                       KMMFFourCCCodePCM16 );
  // Record from the device microphone
  iMdaAudioRecorderUtility->SetAudioDeviceMode(CmdaAudioRecorderUtility::Elocal);

  // Set maximum gain for recording
  iMdaAudioRecorderUtility->SetGain(iMdaAudioRecorderUtility->MaxGain());

  // Delete current audio sample from beginning of file
  iMdaAudioRecorderUtility->SetPosition(TtimeIntervalMicroSeconds(0));
  iMdaAudioRecorderUtility->CropL();

  iMdaAudioRecorderUtility->RecordL();
  }
```

การอัดเสียงของเราโดยมีฟังก์ชัน RecordL โดยที่ต้องมีการกำหนดไฟล์ที่ทำการอัดเสียงและชนิดของไฟล์ที่เราต้องการ

7. ซอสโค้ดโค้ดในส่วนของการเล่นเสียง

```
void CInputEngine::Play()
        {
        // if either stream is active, return
        if (iInputStatus!=EnotReady || iOutputStatus!=EnotReady)
                return;
        // open output stream
```

```
        // upon completion will receive callback in
        // MmdaAudioInputStreamCallback::MaoscOpenComplete()
        iOutputStream->Open(&iStreamSettings);
        }

void CinputEngine::MaoscOpenComplete(Tint aError)
        {
        if (aError==KerrNone)
                {
                // output stream opened     uccessfully, set status
                iOutputStatus = Eopen;
                // set stream properties, 16bit 8KHz mono
                iOutputStream->SetAudioPropertiesL(iStreamSettings.iSampleRate,
                        iStreamSettings.iChannels);
                // set volume to 1/4th of stream max volume
                iOutputStream->SetVolume(iOutputStream->MaxVolume()/4);
                // set stream priority to normal and time sensitive
                iOutputStream->SetPriority(EpriorityNormal,
                        EmdaPriorityPreferenceTime);

                // issue WriteL() to write the first audio data block,
                // subsequent calls to WriteL() will be issued in
                // MmdaAudioOutputStreamCallback::MaoscBufferCopied()
                // until whole data buffer is written.
                //iStreamIdx=iStreamStart;
                iStreamIdx=0;

                iOutputStream->WriteL(*iStreamBuffer[iStreamIdx]);
                }
        else
                {
                // output stream open failed
                iOutputStatus = EnotReady;
                }
```

```
    }

void CinputEngine::MaoscBufferCopied(Tint aError, const TdesC8& aBuffer)
    {
    if (aError==KerrNone)
        {
        if (&aBuffer==iStreamBuffer[iStreamEnd])
            {
            iOutputStatus = EnotReady;
            }
        else
            {
            iStreamIdx++;
            // rotate stream if necessary
            if (iStreamIdx==iStreamBuffer.Count()) iStreamIdx=0;
            // issue WriteL() for next audio data block
            iOutputStream->WriteL(*iStreamBuffer[iStreamIdx]);
            }
        }
    else if (aError==KerrAbort)
        {
        // playing was aborted, due to call to CmdaAudioOutputStream::Stop()
        }
    else
        {
        // error writing data to output
        iOutputStatus = EnotReady;
        }
    }

void CinputEngine::MaoscPlayComplete(Tint aError)
    {
    iOutputStatus = EnotReady;
    if (aError==KerrNone)
```

```
        {
        // normal stream closure
        }
    else if (aError==KerrUnderflow)
        {
        // end of audio data stream was reached because of stream underflow,
        }
    else {
        // completed with error(s)
        }
}
```

8. ซอสโค้ดโค้ดในส่วนของการเข้ารหัสเสียงพูดและบีบอัดข้อมูล

```
// Uid of PCM16toAMR codec is 0x101FAF68
CMMFCodec* codec = CMMFCodec::NewL(TUid::Uid(0x101FAF68));
CleanupStack::PushL(codec);

CMMFDescriptorBuffer* srcbuf = CMMFDescriptorBuffer::NewL(320);
CleanupStack::PushL(srcbuf);

// Copy your PCM frame data into srcbuf, for example: srcbuf->Data().Copy(pcmbuf);
CMMFDescriptorBuffer* dstbuf = CMMFDescriptorBuffer::NewL(32);
CleanupStack::PushL(dstbuf);

// file opened ok, proceed writing
// read audio data directly into iStreamBuffer
for (Tint idx=0; idx<iPCMBuffer.Count(); idx++)
        {
        srcbuf->Data().Copy(*iPCMBuffer[idx]);
        TcodecProcessResult result = codec->ProcessL(*srcbuf, *dstbuf);
        audiofile.Write(dstbuf->Data());
        writeidx++;
        // write buffers in correct order, rotate if necessary
```

```
        if (writeidx==iPCMBuffer.Count()) writeidx=0;
        }

// now the dstbuf contains an AMR frame data
CleanupStack::PopAndDestroy(dstbuf);
CleanupStack::PopAndDestroy(srcbuf);
CleanupStack::PopAndDestroy(codec);
```

9. ซอสโค้ดโค้ดในส่วนของการถอดรหัสเสียงพูดและขยายข้อมูล

```
// Uid of AMRtoPCM16 codec is 0x101FAF67
CMMFCodec* codec = CMMFCodec::NewL(TUid::Uid(0x101FAF67));

CleanupStack::PushL(codec);

CMMFDescriptorBuffer* srcbuf = CMMFDescriptorBuffer::NewL(32);
CleanupStack::PushL(srcbuf);

// Copy your PCM frame data into srcbuf, for example: srcbuf->Data().Copy(pcmbuf);
CMMFDescriptorBuffer* dstbuf = CMMFDescriptorBuffer::NewL(1280);
CleanupStack::PushL(dstbuf);

// file opened ok, proceed writing
// read audio data directly into iStreamBuffer
for (TInt idx=0; idx<iAMRBuffer.Count(); idx++)
        {
        srcbuf->Data().Copy(*iAMRBuffer[idx]);
        TCodecProcessResult result = codec->ProcessL(*srcbuf, *dstbuf);
        audiofile.Write(dstbuf->Data());
        writeidx++;
        // write buffers in correct order, rotate if necessary
        if (writeidx==iAMRBuffer.Count()) writeidx=0;
        }
```

```
// now the dstbuf contains an AMR frame data
CleanupStack::PopAndDestroy(dstbuf);
CleanupStack::PopAndDestroy(srcbuf);
CleanupStack::PopAndDestroy(codec);
```

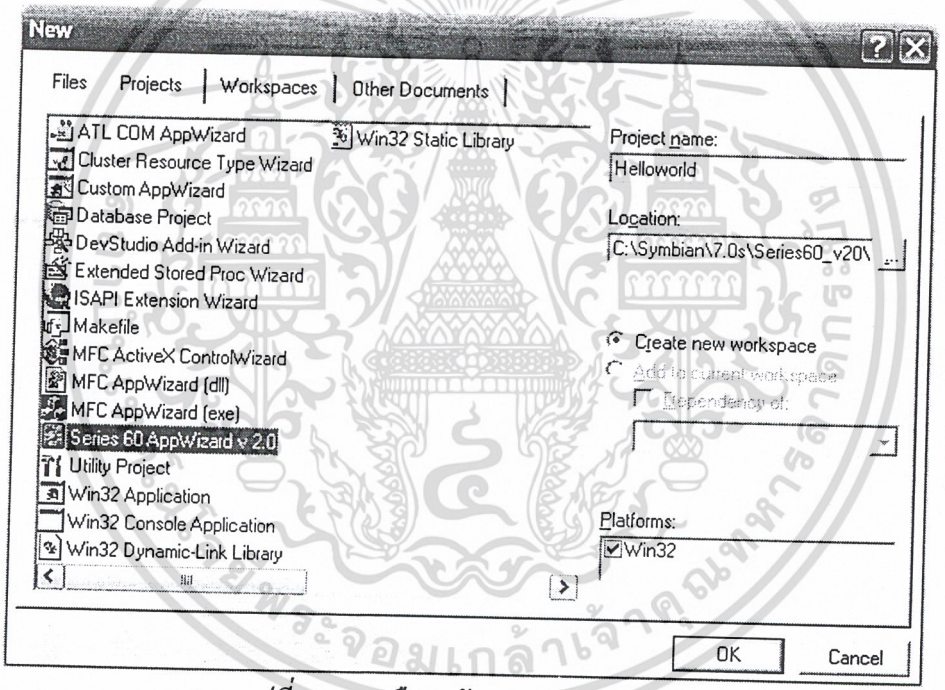
ภาคผนวก ข

Series 60 C++ Software Development Kits (SDKs)

Series 60 C++ Software Development Kits (SDKs) เป็นเครื่องมือที่ช่วยในการพัฒนาแอพพลิเคชันบนอุปกรณ์เคลื่อนที่ พัฒนาด้วยภาษา C++ โดยมีอีมูเลเตอร์มาให้พร้อมเพื่อความสะดวกในการดูผลลัพธ์และการดีบัก

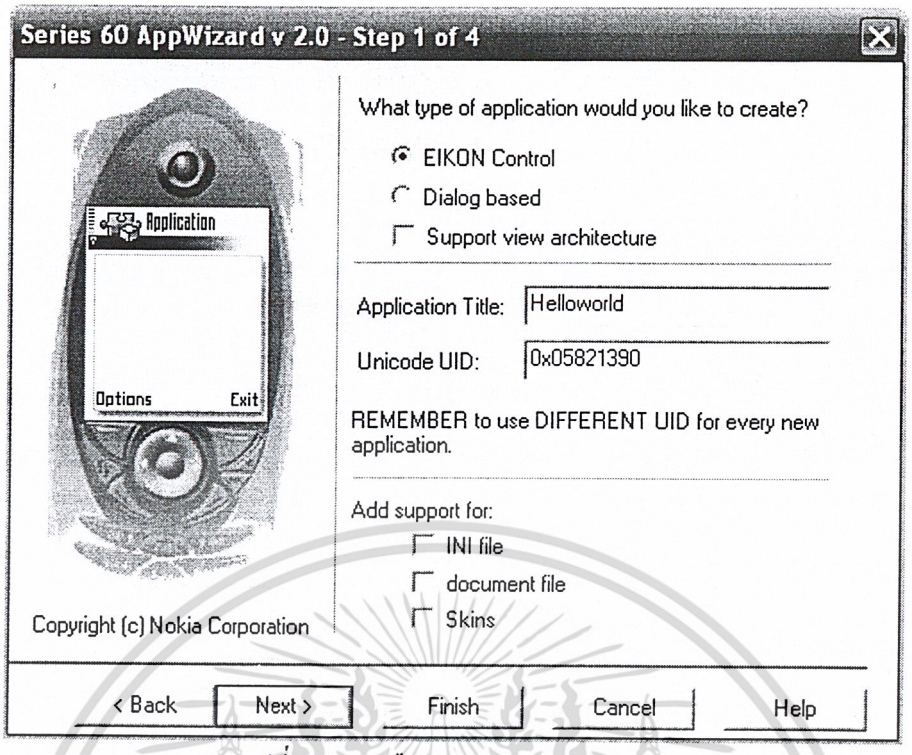
1. การสร้างแอพพลิเคชันใหม่สำหรับอุปกรณ์เคลื่อนที่

1. เปิด Microsoft Visual C++ แล้วเลือก File| New
2. จากนั้นเลือกแท็บโปรเจค แล้วเลือกโปรเจคแบบ Series 60 AppWizard v 2.0



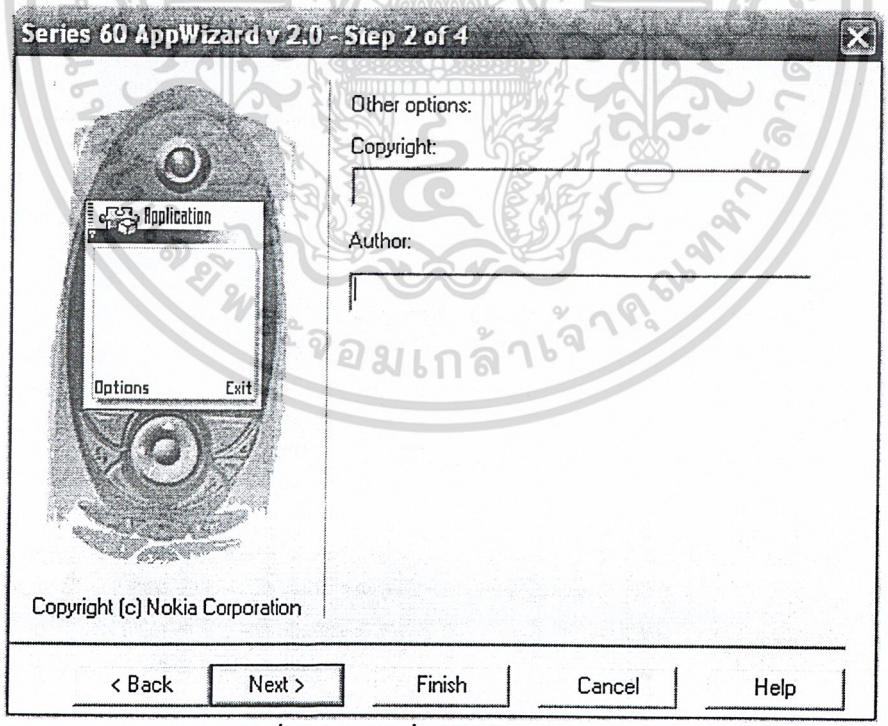
รูปที่ ข-1 การเลือกสร้างแอพพลิเคชันใหม่

3. ใส่ชื่อโปรเจค ยกตัวอย่างเช่น Helloworld แล้วกด OK
4. เลือกประเภทแอพพลิเคชันที่ต้องการจะสร้าง และใส่ชื่อของแอพพลิเคชัน แล้วกด Next



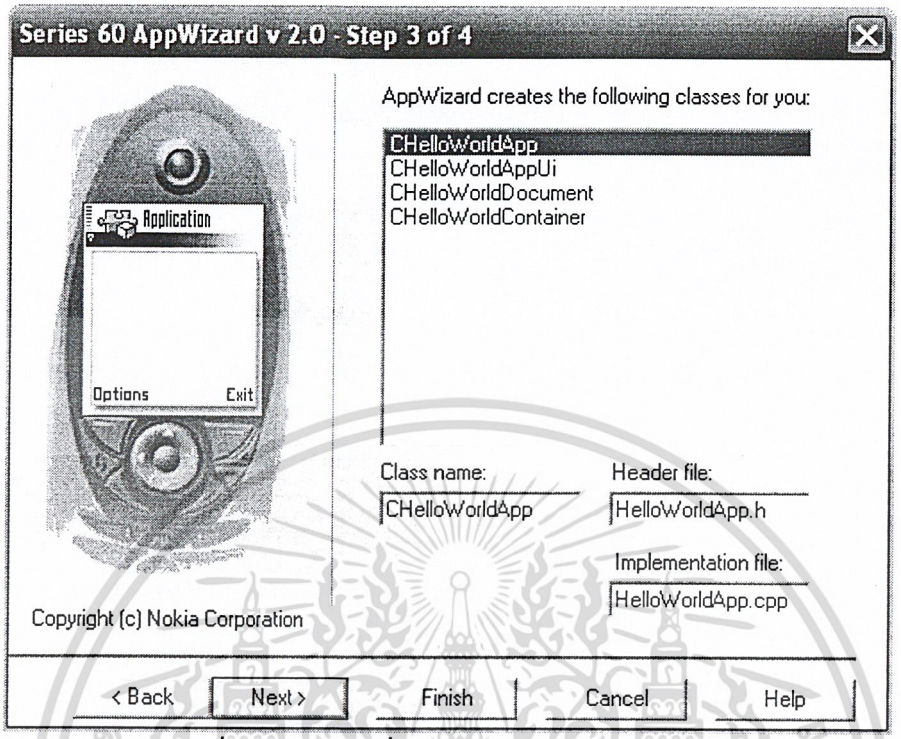
รูปที่ ข-2 การเลือกประเภทแอพพลิเคชัน

5. ใส่ชื่อผู้เขียนแอพพลิเคชัน



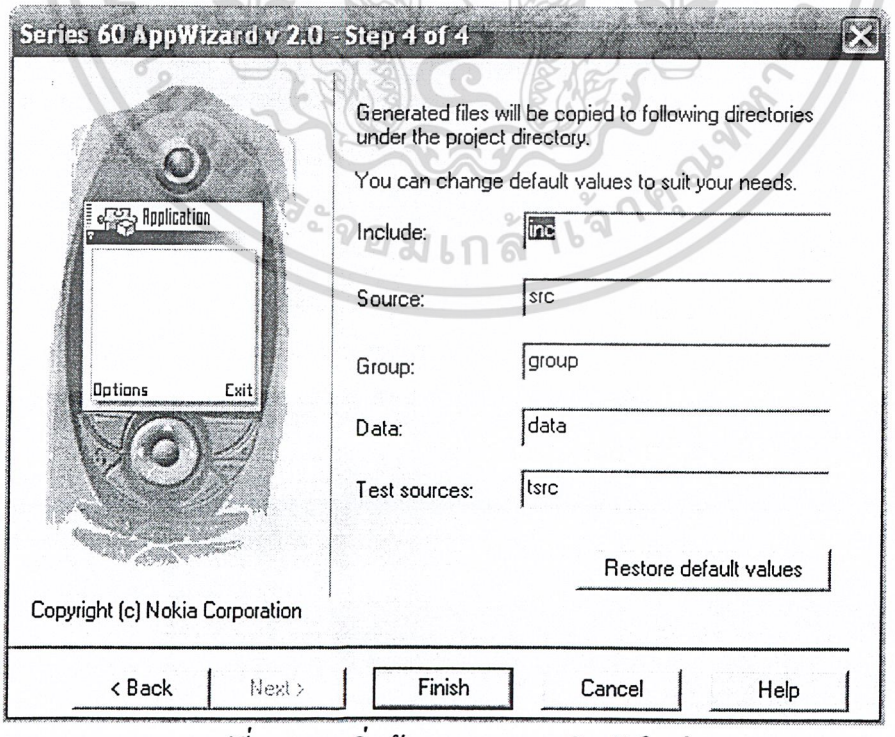
รูปที่ ข-3 การใส่ชื่อผู้เขียนแอพพลิเคชัน

6. Series 60 AppWizard จะสร้างคลาสต่างๆให้ดังรูป



รูปที่ ข-4 คลาสต่างที่ Series 60 AppWizard สร้างให้

7. จะปรากฏไดอะลอกบล็อกที่บอกถึงการอ้างอิง Path ต่างๆ ของโปรแกรม และเมื่อกด Finish แอพพลิเคชันใหม่ก็จะถูกสร้าง



รูปที่ ข-5 การเพิ่มข้อมูลของแอพพลิเคชันใหม่

2. การ Build แอพพลิเคชัน

ในการ Build แอพพลิเคชันนั้น เราจะ Build โดยผ่านทาง Command Line โดยที่เราต้องเปิด Command prompt ก่อน และเปลี่ยน Path ไปที่โปรเจคที่เราต้องการ โดยที่ต้องมีไฟล์ .mmp และไฟล์ bld.inf เราถึงสามารถจะ Build ได้ ยกตัวอย่างเช่น

โปรเจคอยู่ที่ \Symbian \ Series602_0 \HelloWorld \group

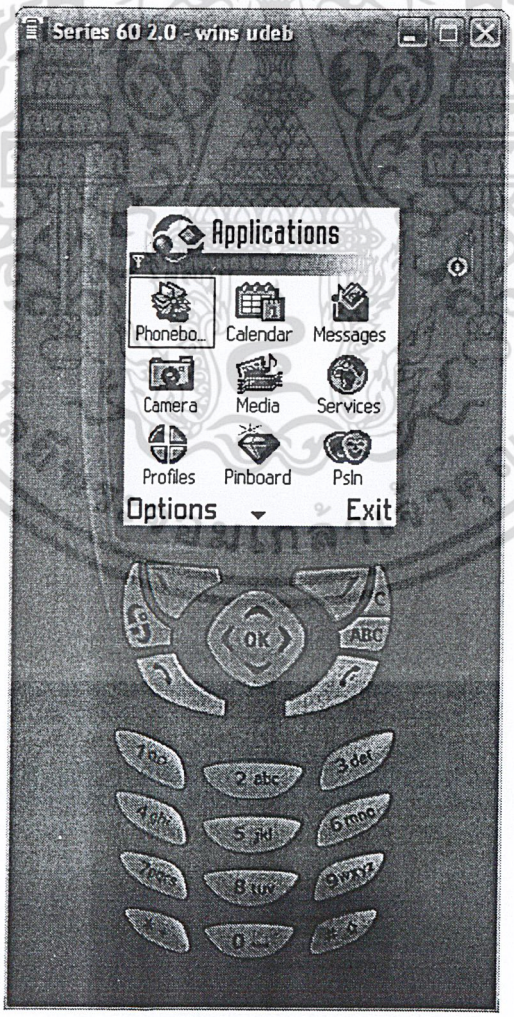
โดยที่เราต้องพิมพ์ blamake bldfiles

รอสักพักจะได้ไฟล์ abld.bat

และต่อมาเราต้องคอมไพล์และลิงค์โปรเจค โดยพิมพ์ abld build wins udeb (สำหรับ Visual C++)

3. การ Run อีมูเลเตอร์

หลังจากที่มีการ Build แอพพลิเคชันเรียบร้อยแล้วสำหรับแพลตฟอร์ม เราสามารถทำการทดสอบการทำงานของแอพพลิเคชันบนอีมูเลเตอร์ได้โดย พิมพ์ epoc.exe จากนั้นจะปรากฏอีมูเลเตอร์ให้สามารถทดสอบการทำงานได้



รูปที่ ข-6 อีมูเลเตอร์

บรรณานุกรม

[1] Leigh Edwards, Richard Barker, and the Staff of EMCC Software Ltd.: "Developing Series 60 Applications A Guide for Symbian OS C++ Developers", Addison Wesley 2004.

[2] Nokia Corporation: "Course #5300 Series 60 C++ Introduction", Nokia Corporation 2004.