การประสานเวลาแบบกระจายสำหรับเครือข่ายตัวรับรู้แบบไร้สาย

นายกิตติภัทร อภิชาติไตรสรณ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2553

DISTRIBUTED TIME SYNCHRONIZATION FOR WIRELESS SENSOR NETWORKS

Mr. Kittipat Apicharttrisorn

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computer Science

Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University

Academic Year 2010

หัวข้อวิทยานิพนธ์	การประสานเวลาแบบกระจายสำหรับเครือข่ายตัวรับรู้แบบไร้สาย
โดย	นายกิตติภัทร อภิชาติไตรสรณ์
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร.เฉลิมเอก อินทนากรวิวัฒน์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.เกริก ภิรมย์โสภา)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร.เฉลิมเอก อินทนากรวิวัฒน์)

..... กรรมการ
(อาจารย์ ดร.กุลธิดา โรจน์วิบูลย์ชัย)

..... กรรมการภายนอกมหาวิทยาลัย
(รองศาสตราจารย์ ดร.อนันต์ ผลเพิ่ม)

กิตติภัทร อภิชาติไตรสรณ์ : การประสานเวลาแบบกระจายสำหรับเครือข่ายตัวรับรู้แบบไร้สาย. (DISTRIBUTED TIME SYNCHRONIZATION FOR WIRELESS SENSOR NETWORKS) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร. เฉลิมเอก อินทนากรวิวัฒน์, 59 หน้า.

ปัจจุบันสถาบันการศึกษา, ภาคอุตสาหกรรม, และภาคธุกิจ ได้เห็นถึงศักยภาพของการประยุกต์ใช้ ที่มีศักยภาพ ของเครือข่ายตัวรับรู้แบบไร้สาย ข้อมูลของเครือข่ายประเภทนี้มักจะถูกรวมกับเวลาซึ่งจะทำให้ข้อมูลมีประโยชน์และสามารถนำไปใช้ได้ เพื่อที่จะให้โหนดซึ่งกระจายตัวอยู่ในเครือข่ายมีความเข้าใจเวลาที่ตรงกัน โปรโตคอลการประสานเวลาจึงเป็นสิ่งที่จำเป็น โดยการประสานเวลานั้นเป็นหนึ่งในปัญหาที่ต้องแก้ในระบบกระจาย ซึ่ง โปรโตคอลการประสานเวลาที่มีอยู่ในปัจจุบันไม่สามารถทำงานได้ดีในเครือข่ายตัวรับรู้แบบไร้สาย เนื่องจากลักษณะเฉพาะของเครือข่ายตัวรับรู้แบบไร้สาย ทำให้โปรโตคอล ใหม่ๆถูกพัฒนาขึ้นมาเป็นลำดับ

วิทยานิพนธ์ฉบับนี้ได้เสนอการออกแบบและทำให้เกิดผลสำหรับการประสานเวลา ผู้วิจัยได้ออกแบบและเสนอการเฉลี่ยแบบเพิ่มส่วนซึ่งทำให้โหนดสามารถปรับเวลาได้ทันทีที่ได้รับข้อความประสานเวลา และคาบเวลาบิดเบือนซึ่งทำให้โหนดสามารถชดเชยเวลาบิดเบือนได้โดยอัตโนมัติ, ขยายคาบประสานเวลาเพื่อประหยัดพลังงาน, และประมาณค่าชดเชยเวลาบิดเบือนในขณะที่โหนดหลับ ผู้วิจัยได้ ทำให้เกิดผลนั้น ด้วยโปรแกรมภาษา nesC บนอุปกรณ์ตัวรับรู้เพลตฟอร์มโมตรุ่น Telosb ซึ่งใช้ระบบปฏิบัติการ TinyOS และ ผลการทดลองแสดงให้เห็นว่าการประสานเวลาดังกล่าวมีคุณสมบัติ ที่สำคัญสำหรับการนำไปใช้จริงในเครือข่ายตัวรับรู้แบบไร้สาย คือ การกระจาย ใช้พลังงานอย่างมีประสิทธิภาพ มีความแม่นยำ และมีคุณสมบัติเกรเดียน

ภาควิชา.....วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อนิสิต .....

สาขาวิชา....วิทยาศาสตร์คอมพิวเตอร์.... ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก.....

ปีการศึกษา...2553.....

5071404021 : MAJOR COMPUTER SCIENCE

KEYWORDS : TIME SYNCHRONIZATION / WIRELESS SENSOR NETWORKS / DISTRIBUTION / GRADIENT / ENERGY EFFICIENCY

KITTIPAT APICHARTTRISORN: DISTRIBUTED TIME SYNCHRONIZATION FOR WIRELESS SENSOR NETWORKS. THESIS ADVISOR : ASST. PROF. CHALERMEK INTANAGONWIWAT, Ph.D., 59 pp.

In recent years, academic, industrial, and commercial institutions have seen potentially practical applications of wireless sensor networks. Data from such a network is usually embedded with timing information to make them valuable and even usable. To make distributed sensor nodes to have a common notion of time, a time synchronization protocol is needed. Time synchronization has been one of the classical problems in distributed systems. However, existing time synchronization protocols do not work well in wireless sensor networks due to unique characteristics of such networks. Therefore, new synchronization algorithms have been developed.

This thesis proposes the design and implementation of a time synchronization algorithm for wireless sensor networks. The design includes incremental averaging to let nodes synchronize on reception of a time synchronization message. Also, the estimation of a clock skew is measured. The estimated period enables nodes to automatically compensate for a clock skew, to extend synchronization message periods for energy savings, and to calculate clock skew compensation during a sleep period. We implement our software on nesC and evaluate our approach on Telosb motes running TinyOS. Our experimental results indicate that the proposed time synchronization protocol possesses crucial properties for practical uses in wireless sensor networks: distributed computation, energy efficiency, and gradient time.

Department : Computer Engineering ..... Student's Signature .....

Field of Study : Computer Science ..... Advisor's Signature .....

Academic Year : 2010 .....

กิตติกรรมประกาศ

ผู้วิจัยขอขอบพระคุณสำหรับความช่วยเหลือ ข้อชี้แนะ และกำลังใจจากอาจารย์ที่ปรึกษา ผศ.ดร.เฉลิมเอก อินทนากรวิวัฒน์ ซึ่งเป็นอาจารย์ที่มีความสามารถทั้งในด้านการวิจัยและการเรียนการสอน ซึ่งได้ผลักดันให้งานวิจัยมีความก้าวหน้าจนมาเป็นวิทยานิพนธ์ฉบับนี้ในที่สุด

ผู้วิจัยขอขอบพระคุณ ผศ.ดร. เกริก ภิรมย์โสภา อ.ดร. กุลธิดา โรจน์วิบูลย์ชัย และ รศ.ดร. อนันต์ ผลเพิ่ม คณะกรรมการสอบวิทยานิพนธ์ฉบับนี้ ซึ่งได้สละเวลาอันมีค่ามาให้ข้อเสนอแนะและข้อวิจารณ์เชิงวิชาการอันมีค่า อันนำไปสู่การพัฒนางานวิจัยยิ่งขึ้น

ขอขอบคุณจุฬาลงกรณ์มหาวิทยาลัยที่ให้สถานที่ในการทำวิจัย โอกาสในการศึกษาในระดับปริญญาโท คณาจารย์ที่มีความรู้ความสามารถและความใส่ใจในการให้ความรู้และประสบการณ์เป็นอย่างดีเสมอมา และได้พบคนเก่งๆมากมายที่กำลังศึกษาอยู่หรือจบการศึกษาไปแล้วก็ตาม

ขอขอบคุณบัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย ที่เอื้อเฟื้อทุนสำหรับไปเสนอผลงานวิชาการต่างประเทศ ทำให้ผู้วิจัยได้รับประสบการณ์อันมีค่ายิ่ง

ขอขอบคุณนายศุภเสฏฐ์ ชูชัยศรี สำหรับคำแนะนำ, ความช่วยเหลือและกำลังใจ นายธนภูมิ สำหรับคำแนะนำให้รู้จักห้องวิจัยยูบิเนต และ นายศรัณย์ เจนจตุรงค์ นายวันชัย งิ้วลาย นางสาวขวัญจิรา และสมาชิกห้องวิจัย ยูบิเนต สำหรับความคิดเห็นที่เป็นประโยชน์ต่องานวิจัยและกำลังใจที่มีให้เสมอมา

ขอขอบคุณ คุณหญิงภาพร แฝดกลาง คุณบุญรัตน์ อภิชาติไตรสรณ์ และคุณนฤมล แต่เจริญ ที่ให้การสนับสนุนทุนการศึกษา และขอขอบคุณ บิดา มารดา พี่น้อง และญาติทุกๆคนที่ให้กำลังใจแก่ผู้วิจัยตลอดมา

ขอขอบคุณ บริษัท วิทยุการบินแห่งประเทศไทย จำกัด ที่ให้ความสำคัญกับพนักงานในการศึกษาเล่าเรียนในระดับที่สูงขึ้น

สารบัญ

สารบัญตาราง

สารบัญภาพ

คำอธิบายสัญลักษณ์และคำย่อ

NTP	งานวิจัยด้านการประสานเวลา [3]
RBS	งานวิจัยด้านการประสานเวลาสำหรับเครือข่ายตัวรับรู้แบบไร้สาย [8]
TPSN	งานวิจัยด้านการประสานเวลาสำหรับเครือข่ายตัวรับรู้แบบไร้สาย [4]
FTSP	งานวิจัยด้านการประสานเวลาสำหรับเครือข่ายตัวรับรู้แบบไร้สาย [5]
ETA	งานวิจัยด้านการประสานเวลาสำหรับเครือข่ายตัวรับรู้แบบไร้สาย [12]
RITS	งานวิจัยด้านการประสานเวลาสำหรับเครือข่ายตัวรับรู้แบบไร้สาย [12]
RATS	งานวิจัยด้านการประสานเวลาสำหรับเครือข่ายตัวรับรู้แบบไร้สาย [12]
GTSP	งานวิจัยด้านการประสานเวลาสำหรับเครือข่ายตัวรับรู้แบบไร้สาย [6]
EGTSP	งานวิจัยที่เป็นส่วนหนึ่งของวิทยานิพนธ์ฉบับนี้ [20]

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เครือข่ายตัวรับรู้แบบไร้สาย (Wireless Sensor Networks) ประกอบไปด้วยโหนดรับรู้จำนวนมากเชื่อมต่อกันแบบไร้สายขึ้นเป็นเครือข่ายที่สามารถนำไปใช้ได้หลากหลาย โหนดแต่ละโหนดสามารถรับรู้สภาวะแวดล้อม (Sense) ผ่านตัวรับรู้ (Sensor) ได้หลายประเภทเช่น อุณหภูมิ, ความชื้น, การเคลื่อนที่ของยานพาหนะ, ความดัน, เสียง เป็นต้น แต่ละโหนดยังสามารถสื่อสารไร้สายและส่งข้อมูลไปที่โหนดซิ้ง (Sink) โดยสร้างเครือข่ายประเภทแอดฮอก (Ad Hoc Networks) ซึ่งไม่ต้องอาศัยโครงสร้างพื้นฐานการสื่อสารเพื่อส่งข้อมูล ทำให้สามารถนำไปใช้ในพื้นที่ห่างไกลได้ และเนื่องจากโหนดไม่ต้องส่งด้วยกำลังส่งสูง ขนาดของโหนดจึงเล็ก สามารถนำไปวางโดยไม่ไปรบกวนสภาพแวดล้อมเดิมมากนัก ตัวอย่างเช่นในรังนก นอกจากนั้นโหนดส่วนใหญ่จะใช้แบตเตอรี่ที่มีพลังงานจำกัดเป็นแหล่งพลังงาน ทำให้การใช้พลังงานจำเป็นต้องมีประสิทธิภาพให้มากที่สุด

การประยุกต์ใช้งาน (Applications) ของเครือข่ายตัวรับรู้แบบไร้สายสามารถแบ่งตามประเภทได้ดังนี้ [1]

1. ทางการทหาร เช่น การตรวจติดตามข้าศึกภายในสมรภูมิรบ เนื่องจากตัวรับรู้แบบไร้สายไม่จำเป็นจะต้องมีการติดตั้งที่ยุ่งยาก สามารถโปรยลงมาจากเครื่องบิน และทำการสร้างเครือข่ายตรวจติดตามได้ทันที
2. ทางด้านสิ่งแวดล้อม เช่น การตรวจสอบสภาพทางชีววิทยาของพื้นที่ที่สนใจ, ระบบเตือนไฟป่า, ระบบเตือนภัยน้ำท่วม เป็นต้น
3. ทางด้านสาธารณสุข เช่น ระบบเฝ้าสังเกตผู้ป่วยระยะไกล, การตรวจติดตามและเฝ้าสังเกตุคนไข้และแพทย์ภายในโรงพยาบาล, และการบริหารจัดการยาในโรงพยาบาล
4. ภายในที่พักอาศัย เช่น บ้านอัจฉริยะ ซึ่งสามารถตรวจจับบุคคลที่อยู่ในบ้านและทำการเปิดเครื่องปรับอากาศในอุณหภูมิที่เหมาะสม, เปิดเครื่องเสียงสำหรับเพลงโปรด, และชงกาแฟแก้วโปรดได้โดยอัตโนมัติ

5. ทางด้านอื่นๆ เช่น การควบคุมสภาพแวดล้อมภายในอาคารสำนักงาน, พิพิธภัณฑ์เชิงโต้ตอบ, ระบบตรวจจับการโขมยรถ, การจัดการพัสดุคงคลัง, และการตรวจจับยานพาหนะ เป็นต้น

ข้อมูลของเครือข่ายตัวรับรู้แบบไร้สายสามารถรวมกันได้ (Data Fusion) เพื่อที่จะส่งเป็นข้อความรวมแทนที่จะส่งแยกกันทุกข้อความ ทำให้สิ้นเปลืองแบนวิทธ์และพลังงานเป็นอย่างมาก การรวมข้อมูลกันนั้นจำเป็นจะต้องอาศัยเวลาของแต่ละโหนดที่ตรงกัน เพื่อที่จะกำหนดว่าข้อมูลใดที่รวมกันได้หรือรวมกันไม่ได้ ดังนั้นการประสานเวลา (Time Synchronization) จึงมีความจำเป็นต่อเครือข่ายตับรับรู้แบบไร้สาย นอกจากนั้นการประยุกต์ใช้เครือข่ายตัวรับรู้แบบไร้สายเพื่อที่จะนำไปตรวจจับความเร็วและทิศทางของวัตถุที่สนใจ โปรโตคอลการประสานเวลาที่มีความแม่นยำจะทำให้การตรวจจับความเร็วและทิศทางมีความแม่นยำมากขึ้น

ปัญหาเรื่องการประสานเวลานั้นเกิดขึ้นมาในระบบกระจาย โดยเฉพาะเมื่อมีการใช้อินเทอเนตกันอย่างอย่างแพร่หลาย โปรเซสในระบบกระจายซึ่งอาจจะอยู่กันคนละที่และเชื่อมต่อกันด้วยเครือข่ายอินเทอเนต การปรับเวลาของแต่ละโหนดในระบบให้ตรงกันจึงเป็นสิ่งที่สำคัญมาก ไม่เช่นนั้นการประมวลผลอาจจะเกิดความผิดพลาดและส่งผลเสียอย่างร้ายแรง เช่นในระบบธนาคาร [2]

โปรโตคอลถูกนำมาใช้ในอินเทอเนตก็คือ Network Time Protocol (NTP) [3] NTP ใช้เทคนิคการประเมินค่า Round Trip Time ระหว่างโหนดแม่ข่ายเวลาและโหนดลูกข่ายเพื่อประมาณค่าเวลาหน่วงในการส่งข้อความประสานเวลาและทำการชดเชยค่าเวลาตามลำดับ และเพื่อเป็นการชดเชยกับค่าเวลาจิตเตอร์ (Jitter) โหนดจำเป็นต้องส่งข้อความประสานเวลาหลายๆครั้งและทำการคำนวณค่าเวลาที่ต้องทำการชดเชย แต่เนื่องจากความซับซ้อนของโปรโตคอลและความแม่นยำในระดับวินาที ทำให้ NTP ไม่เหมาะกับเครือข่ายตัวรับรู้ไร้สายซึ่งต้องการโปรโตคอลที่ง่าย, ใช้จำนวนข้อความน้อย, และความแม่นยำในระดับมิลลิวินาทีขึ้นไป

โปรโคอล Timing-Sync Protocol for Sensor Networks (TPSN) [4] ได้เสนอส่วนประกอบของเวลาหน่วงที่เกิดขึ้นจากการส่งข้อความแบบไร้สายจากโหนดหนึ่งไปอีกโหนดหนึ่งและเสนอการประทับเวลาในระดับชั้นแมค เพื่อลดผลกระทบของเวลาหน่วงที่เกิดขึ้นในการส่งข้อความประสานเวลา ทำให้การประสานเวลามีความแม่นยำมากขึ้น แต่การทำงานของ TPSN จะสร้างโครงสร้างต้นไม้ขึ้นมาเพื่อให้โหนดลูกประสานเวลากับโหนดแม่จนถึงโหนดราก (Root) ทำให้เกิดการพึ่งพิงโหนดแม่และโหนดรากซึ่งทำให้เกิดลักษณะของการรวมศูนย์ (Centralization) ซึ่งไม่

เหมาะกับเครือข่ายตัวรับรู้แบบไร้สายที่เป็นเครือข่ายแบบพลวัต กล่าวคือโหนดใดๆมีโอกาสตาย, หรือไม่สามารถติดต่อได้, หรือเคลื่อนที่ออกจากกัน ได้ตลอดเวลา

โปรโตคอล Flooding Time Synchronization Protocol (FTSP) [5] ได้เสนอการประสานเวลาโดยใช้การส่งข้อความประสานเวลาจากโหนดรากไปทั่วทั้งเครือข่าย โดยที่ทุกโหนดจะทำการส่งข้อความประสานเวลาต่อไปเรื่อยๆ ทำให้โหนดใดๆมีโอกาสที่จะได้รับข้อความประสานเวลามากขึ้นและโอกาสที่จะประสานเวลาสำเร็จมีสูงขึ้นด้วย และทำให้ FTSP สามารถใช้งานกับเครือข่ายที่มีพลวัตอย่างเครือข่ายตัวรับรู้แบบไร้สายได้ดี อย่างไรก็ตามลักษณะการทำงานของ FTSP ทำให้เกิดความสัมพันธ์เชิงเวลาของโหนดในเครือข่ายเป็นแบบต้นไม้ เนื่องจากโหนดของ FTSP จะไม่นำข้อความที่มีหมายเลขลำดับน้อยกว่าข้อความล่าสุดที่มันนำไปคำนวณมาใช้ ทำให้โหนดมีความสัมพันธ์เชิงเวลาที่แนบแน่นเฉพาะกับโหนดที่ส่งข้อความใหม่ที่สุดให้มันเสมอ ผลที่ตามมาคือ โครงสร้างต้นไม้เชิงเวลาที่เกิดขึ้นกับ FTSP ดังนั้นโหนดที่อยู่ติดกันอาจจะมีความสัมพันธ์เชิงเวลาที่ห่างกันหลายฮอปได้และมีความผิดพลาดในการประสานเวลาสูง

Gradient Time Synchronization Protocol (GTSP) [6] ได้เสนอคุณสมบัติเกรเดียน (Gradient Property) ซึ่งกำหนดว่าค่าใดๆของโหนดที่อยู่ใกล้กันจะมีค่าใกล้กันมากกว่าโหนดที่ไกลกันออกไป ดังนั้นคุณสมบัติของการประสานเวลาแบบเกรเดียนจะต้องทำให้ความผิดพลาดของการประสานเวลาระหว่างโหนดที่อยู่ใกล้กันหรือโหนดเพื่อนบ้านน้อย และความผิดพลาดของการประสานเวลาระหว่างโหนดใดๆในเครือข่ายอยู่ในระดับที่ยอมรับได้ หรือยังคงความเป็นเวลาครอบคลุม (Global Clock) อยู่นั่นเอง ซึ่ง FTSP ไม่มีคุณสมบัติดังกล่าว ดังที่ได้อธิบายไปแล้ว การประสานเวลา GTSP ทำการเฉลี่ยค่าเวลาจากการบรอดคาสข้อความประสานเวลาของเพื่อนบ้าน ทำให้ความสัมพันธ์เชิงเวลาของโหนดเพื่อนบ้านที่อยู่ใกล้กันมีความใกล้ชิดมากกว่าโหนดที่อยู่ไกลออกไป แต่เนื่องจากการเฉลี่ยนั้นเกิดขึ้นทั่วทั้งเครือข่ายทำให้การเกิดประสานเวลาครอบคลุมของเครือข่าย

แต่เนื่องจาก GTSP ใช้คาบเวลาคงที่สำหรับการประมาณค่าและชดเชยเวลาบิดเบือนและเวลาเหลื่อมที่เท่ากัน ทำให้เกิดการใช้พลังงานที่ไม่มีประสิทธิภาพ และทำให้มีความผิดพลาดเกิดขึ้นได้ ผู้วิจัยเสนอว่าการปรับเวลาเหลื่อมสามารถทำได้ทันทีที่ได้รับข้อความประสานเวลา และเวลาบิดเบือนซึ่งจากการทดลองในเบื้องต้นพบว่ามีลักษณะแบบค่อยๆเป็นค่อยๆไป ทำให้ต้องใช้คาบการประมาณค่าเวลาบิดเบือนที่กว้างเพื่อที่จะครอบคลุมเวลาบิดเบือนหนึ่งหน่วย และแสดงออกมาในรูปของคาบเวลาบิดเบือนหนึ่งหน่วย ซึ่งจะทำให้เกิดการชดเชยเวลาบิดเบือนมีคาบการประมาณเวลาบิดเบือนที่ปรับตัวได้ตามความบิดเบือนสัมพัทธ์เทียบกับโหนดเพื่อนบ้าน

และทำให้การประสานเวลาสามารถขยายคาบการประสานเวลาได้ซึ่งทำให้เกิดการประหยัดพลังงานอย่างมาก

จากงานวิจัยทางด้านการประสานเวลาสำหรับเครือข่ายตัวรับรู้แบบไร้สายที่ผ่านมา ผู้วิจัยจึงสนใจที่จะออกแบบโปรโตคอลการประสานเวลาที่มีคุณสมบัติดังนี้

- มีความกระจาย (Distribution) กล่าวคือการประสานเวลานั้นไม่ขึ้นกับโหนดใดโหนดหนึ่ง แต่ทุกโหนดมีส่วนร่วมอย่างเท่าเทียมกันในการประสานเวลา ซึ่งส่งผลให้พลวัตของเครือข่ายส่งผลน้อยมากกับประสิทธิภาพในการประสานเวลา
- มีคุณสมบัติเกรเดียน (Gradient) กล่าวคือ การประสานเวลาจะทำให้ความผิดพลาดระหว่างโหนดเพื่อนบ้านต่ำ และความผิดพลาดระหว่างโหนดใดๆอยู่ในระดับที่รับได้
- ใช้พลังงานอย่างมีประสิทธิภาพ (Energy-Efficiency) เนื่องจากพลังงานแบตเตอรี่ของเครือข่ายตัวรับรู้แบบไร้สายมีอยู่อย่างจำกัด การออกแบบโปรโตคอลซึ่งใช้พลังงานอย่างมีประสิทธิภาพจะช่วยยืดอายุของเครือข่ายได้

1.2 วัตถุประสงค์ของการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อออกแบบโปรโตคอลการประสานเวลาสำหรับเครือข่ายตัวรับรู้แบบไร้สายที่มีคุณสมบัติความกระจาย, คุณสมบัติเกรเดียน และใช้พลังงานอย่างมีประสิทธิภาพ คุณสมบัตินี้ล้วนเป็นคุณสมบัติที่เหมาะสมสำหรับนำไปใช้กับเครือข่ายตัวรับรู้แบบไร้สายได้อย่างมีประสิทธิภาพ

1.3 ขอบเขตของการวิจัย

1. ใช้โปรแกรมภาษา nesC และระบบปฏิบัติการ TinyOS ลงบนอุปกรณ์ตัวรับรู้แบบไร้สาย Telosb
2. โหนดสามารถส่งข้อความแบบบรอดคาสได้
3. โหนดภายในเครือข่ายมีเลขที่ประจำโหนดไม่ซ้ำกัน
4. เวลาบิดเบือนเกิดจากความเบี่ยงเบนทางเวลาซึ่งมีความสัมพันธ์แบบเชิงเส้นในช่วงสั้นๆ

1.4 ข้อจำกัดของการวิจัย

เนื่องจากพฤติกรรมของออสซิลเลเตอร์ที่เป็นต้นทางของสัญญาณนาฬิกานั้นมีลักษณะแบบสุ่มและสร้างแบบจำลองได้ยาก งานวิจัยทางด้านการประสานเวลานั้นจำเป็นต้องใช้อุปกรณ์จริงเพื่อสะท้อนพฤติกรรมจริงเมื่อนำไปใช้งาน จำนวนอุปกรณ์ภายในห้องวิจัยมีอย่างจำกัด ดังนั้นการทดลองจึงมีลักษณะเป็นเครือข่ายขนาดเล็ก ซึ่งได้ควบคุมตัวแปรและเปรียบเทียบให้เห็นประสิทธิภาพของโปรโตคอล

1.5 คำจำกัดความที่ใช้ในการวิจัย

- เวลาท้องถิ่น คือค่าเวลาประจำแต่ละโหนดซึ่งจะเพิ่มขึ้นตามสัญญาณนาฬิกาและไม่สนใจโหนดอื่นๆ
- เวลาครอบคลุม คือเวลาที่โหนดต่างๆพยายามปรับเข้าหากันโดยผ่านทางโปรโตคอลประสานเวลา
- เวลาเหลื่อม คือเวลาที่ไม่เท่ากันของโหนดใดๆ
- เวลาบิดเบือน คือเวลาที่ผิดเพี้ยนไปซึ่งมักจะเกิดจากความเบี่ยงเบนของสัญญาณนาฬิกา
- ความเบี่ยงเบนทางเวลา คือธรรมชาติของแหล่งกำเนิดสัญญาณนาฬิกาอย่างออสซิลเลเตอร์ ซึ่งไม่สามารถสร้างสัญญาณที่ความถี่เท่ากันอย่างสมบูรณ์แบบ ซึ่งความเบี่ยงแบนเพียงเล็กน้อยก็จะทำให้เกิดความบิดเบือนของเวลาได้เมื่อเวลาผ่านไป
- คาบการประสานเวลา คือระยะห่างของการส่งข้อความประสานเวลา เช่น คาบประสานเวลา 30 วินาที คือส่งข้อความประสานเวลาทุกๆ 30 วินาที

1.6 วิธีดำเนินการวิจัย

1. ศึกษางานวิจัยทางด้านเครือข่ายตัวรับรู้แบบไร้สาย
2. ศึกษางานวิจัยทางด้านการประสานเวลาสำหรับตัวรับรู้แบบไร้สาย
3. ออกแบบโปรโตคอลการประสานเวลาสำหรับตัวรับรู้แบบไร้สาย

4. สร้างโปรแกรมตามโปรโตคอลที่ออกแบบไว้
5. ทำการทดลองและบันทึกผลการทดลอง
6. ปรับปรุงแก้ไขโปรแกรม
7. สรุปผลและเรียบเรียงวิทยานิพนธ์

1.7 ผลงานตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของงานวิทยานิพนธ์ได้รับการตีพิมพ์เป็นบทความวิชาการในหัวเรื่อง "Energy Efficient Gradient Time Synchronization for Wireless Sensor Networks" โดย กิตติภัทร อภิชาติไตรสรณ์ ศุภเสฏฐ์ ชูชัยศรี และ เฉลิมเอก อินทนากรวิวัฒน์ ในบันทึกการประชุม "2010 Second International Conference on Computational Intelligence, Communication Systems and Networks" ซึ่งจัดขึ้น ณ โรงแรม Jurys Inn เมืองลิเวอร์พูล (Liverpool) ประเทศอังกฤษ ระหว่างวันที่ 28-30 กรกฎาคม 2553

บทที่ 2

เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 แนวคิดและทฤษฎี

2.1.1 การเฉลี่ยค่า (Averaging)

การเฉลี่ยค่านั้นทำให้ค่าต่างๆที่กระจายตัวอยู่เป็นค่าที่ตรงกันนั่นคือค่าเฉลี่ย การประสานเวลาในวิทยานิพนธ์ฉบับนี้จะทำการเฉลี่ยค่าของออฟเซทเวลาเพื่อที่จะปรับค่าเวลาครอบคลุมให้ตรงกัน สมมติค่าออฟเซทเวลาของโหนด i ที่กำลังคำนวณออฟเซทเวลาเฉลี่ยเมื่อเทียบกับโหนดเพื่อนบ้าน j ใดๆ โดย θ_{ij} คือค่าออฟเซทเวลาของโหนด i เทียบกับโหนด j และ N_i คือจำนวนเพื่อนบ้านของโหนด i

$$\theta_i = \frac{\sum_{j \in N_i} \theta_{ij}}{|N_i| + 1}$$

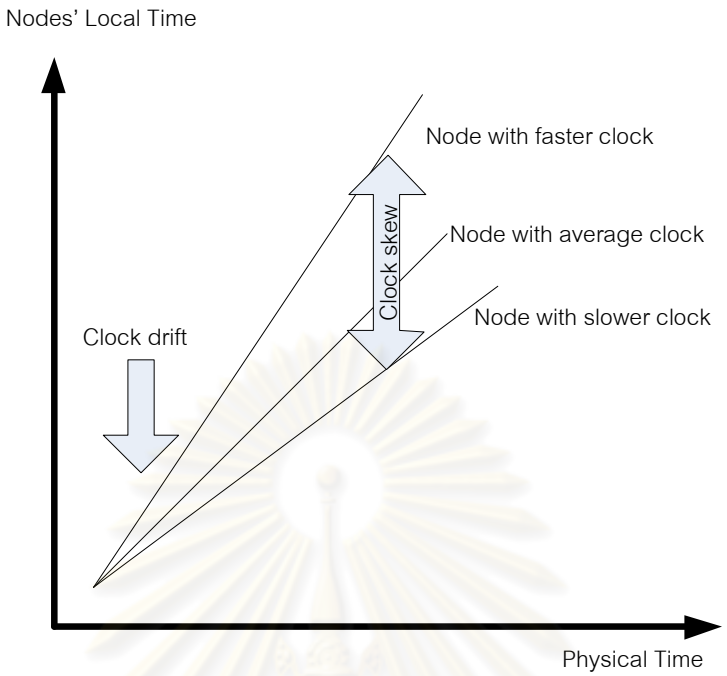
จากสมการข้างต้นจะเห็นได้ว่า โหนดจะต้องรอข้อความประสานเวลาจากโหนดเพื่อนบ้านทุกโหนด จึงจะสามารถคำนวณค่าออฟเซตเวลาที่ต้องชดเชยได้ ในวิทยานิพนธ์ฉบับนี้ผู้วิจัยได้เสนอการเฉลี่ยแบบเพิ่มส่วน (Incremental Averaging) ซึ่งโหนดสามารถทำการเฉลี่ยได้ทันทีที่รับหนึ่งข้อความประสานเวลาจะหนึ่งโหนดเพื่อนบ้านใดๆ ดังสมการ

$$\theta_i(t) = \theta_i(t-1) + \frac{\theta_{ij}(t) + \gamma_i(t-1)}{|N_i + 1|}$$

โดยที่ $\gamma_i(t-1)$ คือเศษเหลือจากการหารรอบที่แล้ว

2.1.2 การเบี่ยงเบนออกของสัญญาณนาฬิกา (Clock Drifting)

สัญญาณนาฬิกาของอุปกรณ์ทางไฟฟ้ามักจะใช้ออสซิลเลเตอร์เป็นแหล่งกำเนิดซึ่งจะกำเนิดสัญญาณทางไฟฟ้าตามความถี่ที่กำหนด เช่น สัญญาณไฟฟ้า 1 KHz จะกำเนิดสัญญาณนาฬิกาที่ความละเอียดหนึ่งมิลลิวินาที และสัญญาณไฟฟ้า 1 MHz จะกำเนิดสัญญาณนาฬิกาที่ความละเอียดหนึ่งไมโครวินาที อย่างไรก็ตามออสซิลเลเตอร์ก็ไม่สามารถทำงานที่ความถี่ตรงกันได้อย่างสมบูรณ์แบบ โดยเฉพาะในตัวรับรู้แบบไร้สายซึ่งมักจะมีขนาดเล็กและราคาถูก ความถี่ที่ไม่เท่ากันนี้จะทำให้เกิดเวลาบิดเบือน (Clock Skew) ซึ่งเป็นปัจจัยที่ทำให้เกิดความผิดพลาดในการประสานเวลาในที่สุดนั่นเอง



ภาพที่ 2.1 แสดงการเบี่ยงเบนของเวลาท้องถิ่นของโหนด

2.1.3 การประทับเวลาในระดับชั้นแมค (MAC-layered Time-stamping)

Ganeriwal et al. [4] ได้จำแนกส่วนประกอบของเวลาหน่วงของการส่งข้อความหนึ่งข้อความแบบไร้สาย ดังนี้

- เวลาส่ง (Send Time) คือเวลาที่เมื่อ Application Layer ต้องการส่งข้อความจนถึงเวลาที่ระดับชั้นแมคด้านล่างได้รับการร้องขอ เวลาหน่วงนี้จะขึ้นกับซอฟแวร์ที่เกี่ยวกับระบบปฏิบัติการ

- เวลาเข้าถึง (Access Time) คือเวลาที่ต้องรอให้ช่องสัญญาณว่าง จึงจะสามารถส่งได้ เวลานี้ขึ้นอยู่กับสภาพของช่องสัญญาณว่าหนาแน่นเพียงไร และโปรโตคอลในการเข้าถึงช่องสัญญาณ (Medium Access Protocol) ว่าจะควบคุมการใช้ช่องสัญญาณไร้สายร่วมกันอย่างไร

- เวลาถ่ายทอด (Transmission Time) คือเวลาที่ใช้ในการส่งข้อความบิตต่อบิต ซึ่งขึ้นอยู่กับขนาดข้อความและความเร็วในการส่งข้อความ

- เวลาแพร่ (Propagation Time) คือเวลาที่สัญญาณเดินทางจากโหนดส่งไปยังโหนดรับ ซึ่งขึ้นอยู่กับความเร็วของคลื่นและระยะทางระหว่างโหนด ซึ่งโดยปกติ ค่าเวลานี้จะน้อยมากเมื่อเทียบกับเวลาอื่นๆ

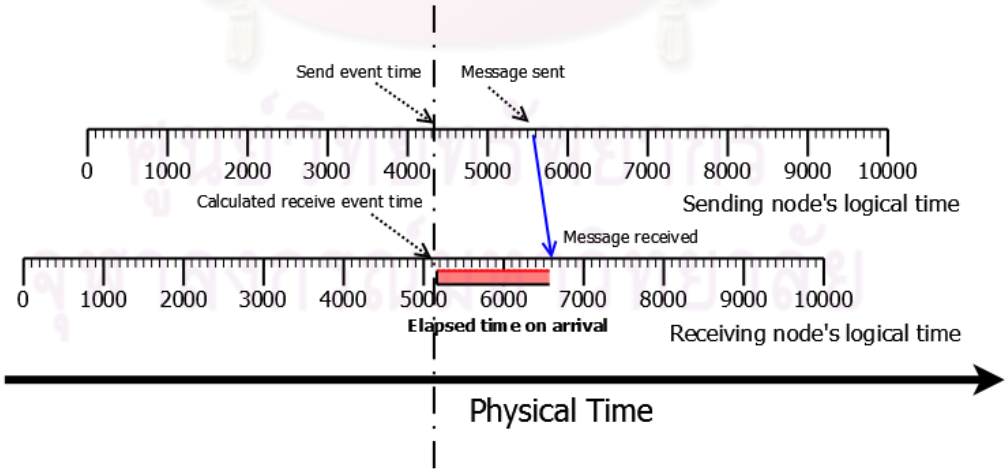
- เวลารับทอด (Reception Time) คือเวลาที่ใช้ในการรับข้อความบิตต่อบิต เป็นแบบเดียวกันกับเวลาถ่ายทอดแต่เกิดที่ด้านรับ

- เวลารับ (Receive Time) คือเวลาที่ระดับชั้นต่างๆสร้างกระแสบิตให้อยู่ในรูปแบบแพคเกจและระดับชั้นแอฟพลิเคชั่นสามารถนำไปประมวลผลต่อไป

หรือสามารถแบ่งเป็นองค์ประกอบหยาบๆได้ คือ เวลาหน่วงด้านส่ง (เวลาส่ง, เวลาเข้าถึง, และเวลาถ่ายทอด), เวลาแพร่ (เกิดขึ้นที่ช่องสัญญาณ), และเวลาหน่วงด้านรับ (เวลารับทอดและเวลารับ

การประทับเวลาในระดับชั้นแมคนั้นคือความสามารถของระดับชั้นแมคที่ประทับเวลาเมื่อบิตแรกของข้อความถูกส่งออกไปที่ด้านส่ง และประทับเวลาเมื่อบิตแรกได้รับที่ด้านรับ ดังนั้นถ้าการประทับเวลาในระดับชั้นแมคทำได้อย่างสมบูรณ์แบบ เวลาหน่วงที่เกิดขึ้นระหว่างการประทับเวลาก็จะมีเพียงเวลาแพร่ซึ่งมีขนาดเล็กมากเมื่อเทียบกับเวลาหน่วงอื่นๆ

นอกจากนี้ Kusy et al. [12] ได้เสนอโมเดลของบริการพื้นฐานสำหรับการประสานเวลา โดยโหนดรับและโหนดส่งสามารถประสานเวลากันได้ด้วยบริการนี้



ภาพที่ 2.2 แสดงการทำงานของ Elapsed time on arrival

จากรูป โหนดส่งจะทำการประทับเวลาของเหตุการณ์ใดๆด้วยเวลาของโหนดส่ง (Send event time) ประมาณ 4300 หน่วยเวลา และเมื่อถึงเวลาที่สามารถส่งข้อความได้ โหนดส่งจะประทับเวลาในระดับชั้นแมค ประมาณ 5600 หน่วยเวลา เพื่อนำมาลบกับเวลาเหตุการณ์ที่บันทึกไว้ก่อนหน้านี้ เรียกว่าค่าเวลาที่เสียไป (Elasped time) เท่ากับ 1300 หน่วยเวลา และทำการแนบค่าเวลานี้ไปกับข้อความด้วย เมื่อโหนดรับประทับเวลาในระดับชั้นแมคด้วยเวลาของโหนดรับแล้ว ประมาณ 6600 หน่วยเวลา โหนดรับจะทำการลบด้วยค่าเวลาที่เสียไปที่โหนดส่งส่งมาให้ 1300 หน่วยเวลา ซึ่งเรียกว่าค่าเวลาที่เสียไปที่ด้านรับ (Elapsed time on arrival) ซึ่งจะได้เวลาเหตุการณ์ของโหนดรับ นั่นคือ 5300 หน่วยเวลา ซึ่งจากรูปจะเห็นว่าเวลาเหตุการณ์ของโหนดรับจริงๆคือ 5100 ความผิดพลาดนี้เกิดจากเวลาแพร่ที่อธิบายไปก่อนหน้านี้นั่นเอง และนอกจากนั้นเวลาที่เสียไปที่ด้านส่งและด้านรับ ในความเป็นจริงนั้นไม่เท่ากัน เนื่องจากสัญญาณนาฬิกามีความเบี่ยงเบน แต่ถ้าเวลานี้มีขนาดเล็กมาก ก็จะเกิดความผิดพลาดเพียงเล็กน้อย

โดยสรุป ถ้าเวลาแพร่และความแตกต่างของเวลาที่เสียไปที่ด้านรับและด้านส่ง มีค่าน้อยมากและสามารถละเลยได้ เราก็จะสามารถคำนวณค่าออฟเซตเวลาระหว่างโหนดรับและโหนดส่งได้ ซึ่งจากตัวอย่างดังรูป จะได้ค่าเป็น 5300 – 4300 = 1000 หน่วยเวลา

ในระบบปฏิบัติการ TinyOS [13] เวอร์ชั่น 2.1 ขึ้นไป ได้นำเอาแนวคิดนี้มาทำให้เกิดผลใน TEP 133 [14] โดยการสร้างส่วนเชื่อมต่อ `TimeSyncAMSend` และ `TimeSyncPacket` ซึ่งมีบริการดังนี้

```
interface TimeSyncAMSend<precision_tag, size_type>
{
        command error_t send(am_addr_t addr, message_t* msg, uint8_t
len, size_type event_time);
        command error_t cancel(message_t* msg);
        event void sendDone(message_t* msg, error_t error);
        command uint8_t maxPayloadLength();
        command void* getPayload(message_t* msg, uint8_t len);
}
```

```
interface TimeSyncPacket<precision_tag, size_type>
{
        command bool isValid(message_t* msg);
        command size_type eventTime(message_t* msg);
}
```

ภาพที่ 2.3 บริการของส่วนเชื่อมต่อ `TimeSyncAMSend` และ `TimeSyncPacket`

การใช้งานบริการดังกล่าวต้องทำการต่อส่วนเชื่อมต่อ `TimeSyncAMSend` และ `TimeSyncPacket` ของ `ActiveMessageC` กับแอฟพลิเคชั่นที่ต้องการใช้งาน และเมื่อต้องการส่งข้อความให้เรียกคำสั่ง `TimeSyncAMSend.send(am_addr_t addr, message_t* msg,`

`uint8_t len, size_type event_time);` โดยให้ใส่ค่าเวลาของเหตุการณ์ไปในอาร์กูเม้นลำดับที่สี่ event_time และที่ด้านรับเมื่อต้องการเรียกค่าเวลาเหตุการณ์ที่ด้านรับก็ให้เรียกคำสั่ง `TimeSyncPacket.eventTime(message_t* msg);` ก็จะคืนค่าออกมาเป็นเวลาเหตุการณ์ที่ด้านรับ

2.1.4 การประสานเวลาสำหรับเครือข่ายตัวรับรู้แบบไร้สาย (Time Synchronization for Wireless Sensor Networks)

2.1.4.1 เทคนิคการประสานเวลา (Time Synchronization Techniques)

วิธีประสานเวลาสำหรับเครือข่ายตัวรับรู้แบบไร้สายสามารถแบ่งคร่าวๆได้เป็นสองแบบคือ แบบด้านรับกับด้านรับ (Receiver-to-receiver synchronization) และแบบด้านส่งกับด้านรับ (Sender-to-receiver synchronization)

1. การประสานเวลาแบบด้านรับกับด้านรับ ได้ถูกเสนอใน RBS โดย Jeremy Elson และทีมงาน [8] เพื่อกำจัดผลกระทบของเวลาหน่วงด้านส่ง โดยการให้โหนดใดๆทำการประทับเวลาข้อความอ้างอิงที่ส่งจากโหนดที่เลือกขึ้นมา และโหนดที่รับก็จะทำการแลกเปลี่ยนเวลาที่รับข้อความอ้างอิง ซึ่งจะสามารถคำนวนค่าออฟเซตเวลาสัมพัทธ์ได้ ด้วยวิธีนี้จะทำให้ผลกระทบของเวลาหน่วงด้านส่งถูกกำจัด เนื่องจากโหนดรับทุกโหนดจะได้รับข้อความด้วยเวลาหน่วงด้านส่งเท่ากันทั้งหมด อย่างไรก็ตามการประสานเวลาด้วยวิธีนี้ก็ยังได้รับผลกระทบจากเวลาแพร่และเวลาหน่วงด้านรับที่ไม่เท่ากันอยู่

2. การประสานเวลาแบบด้านส่งกับด้านรับ เป็นวิธีที่โหนดส่งทำการประทับเวลาและส่งค่าเวลาไปให้โหนดรับซึ่งทำการประทับเวลาด้วย และโหนดรับนำค่าเวลาทั้งสองมาคำนวนเพื่อประสานเวลาได้ ซึ่งวิธีเป็นวิธีที่ใช้กันมานาน เช่นใน NTP [3] แต่เนื่องจากการประสานเวลาด้วยวิธีนี้จะได้รับผลกระทบจากเวลาหน่วงทั้งสามส่วน ทำให้เมื่อนำมาใช้กับเครือข่ายตัวรับรู้แบบไร้สาย RBS มีความผิดพลาดในการประสานเวลาน้อยกว่า NTP เป็นอย่างมาก อย่างไรก็ตาม Ganeriwal et al. [4] ได้เสนอวิธีการประสานเวลาแบบด้านส่งกับด้านรับควบคู่กับการประทับเวลาที่ระดับชั้นแมค และพิสูจน์ทางคณิตศาสตร์ว่า วิธีนี้ให้ผลที่ดีกว่าการประสานเวลาแบบด้านรับกับด้านรับถึงสองเท่า ทำให้งานทางด้านการประสานเวลาในภายหลังมักจะใช้เทคนิคนี้เป็นส่วนใหญ่ รวมถึงในวิทยานิพนธ์ฉบับนี้ด้วย

นอกจากนั้นวิธีการประสานเวลายังสามารถแบ่งเป็นแบบตื่นตัว (Active) กับแบบไม่ตื่นตัว (Passive)

1. การประสานเวลาแบบตื่นตัว คือการประสานเวลาที่ต้องมีการส่งข้อความประสานเวลาอยู่ตลอดเวลา เพื่อทำให้เวลาถูกประสานอยู่ตลอดและแอปพลิเคชั่นสามารถใช้บริการได้ทันที ข้อดีคือ ความผิดพลาดในการประสานเวลาต่ำ แต่ข้อเสียคือมีโอเวอร์เฮดในระบบสูง ตัวอย่างของโปรโตคอลประเภทนี้คือ NTP, RBS, TPSN, FTSP, GTSP และอื่นๆ

2. การประสานเวลาแบบไม่ตื่นตัว คือการประสานเวลาเมื่อมีการส่งข้อมูลหากัน (Data) เท่านั้น ข้อดีคือมีโอเวอร์เฮดต่ำมาก เหมาะสำหรับเครือข่ายที่มีรอบการทำงานสั้น (Low duty cycles) ข้อเสียคือมีความหน่วงของการประสานเวลาและความผิดพลาดในการประสานเวลาสูงกว่า

2.1.4.2 แบบจำลองการประสานเวลา (Time synchronization models)

นอกจากนั้น Ganeriwal et al. [4] ยังได้เสนอรูปแบบของผลลัพธ์ของการประสานเวลาไว้สามประเภทคือ

1. ประเภทลำดับเหตุการณ์ (Event Ordering) ซึ่งจะทำการประสานเวลาเพื่อให้โหนดในเครือข่ายสามารถเรียงลำดับเหตุการณ์ที่เกิดขึ้นได้เท่านั้น ตัวอย่างของโปรโตคอลที่ใช้แบบจำลองนี้คือ Time Synchronization for Ad Hoc Networks [11]

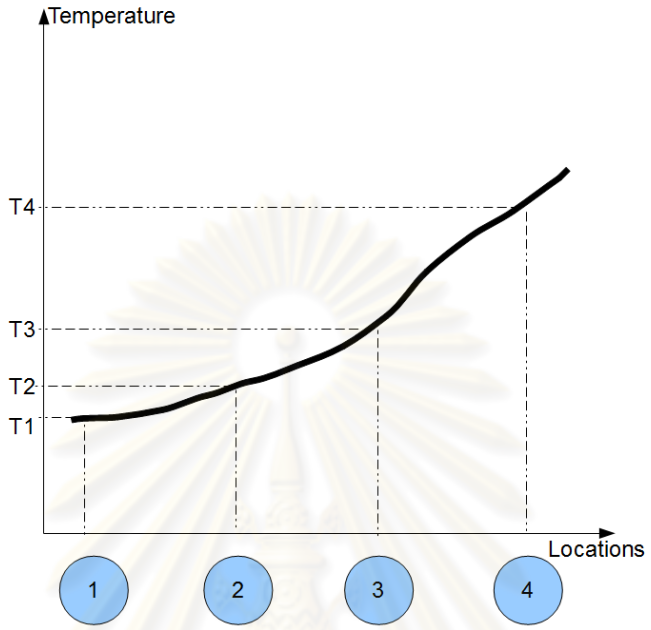
2. ประเภทเวลาสัมพันธ์ (Relative Clocks) โหนดจะทำการเก็บค่าเวลาสัมพัทธ์เทียบกับโหนดเพื่อนบ้านหรือโหนดใดๆ โดยการใช้งานก็ใช้การแปลงเวลาไปมาระหว่างโหนด ตัวอย่างของโปรโตคอลที่ใช้แบบจำลองนี้คือ RBS [8]

3. ประเภทเวลาครอบคลุม (Global Clocks หรือ Always-On) โหนดในเครือข่ายจะทำการรักษาค่าเวลาให้เท่ากันเสมอ เรียกว่าเวลาครอบคุลม การนำไปใช้งานก็เพียงเรียกค่าเวลาครอบคลุมมาใช้ ซึ่งสามารถลำดับเหตุการณ์และรู้เวลาสัมพัทธ์ระหว่างโหนดได้โดยอัตโนมัติ ตัวอย่างของโปรโตคอลที่ใช้แบบจำลองนี้คือ NTP, TPSN, FTSP, GTSP, และโปรโตคอล EGTSP ที่นำเสนอในวิทยานิพนธ์ฉบับนี้เป็นต้น

2.1.5 คุณสมบัติเกรเดียนสำหรับการประสานเวลา

คุณสมบัติเกรเดียนคือคุณสมบัติซึ่งค่าๆหนึ่งค่อยๆ เปลี่ยนแปลงไปตามระนาบที่กำหนด ในที่นี้ก็คือตำแหน่งของโหนดนั่นเอง ตัวอย่างอย่างง่ายก็คืออุณหภูมิ โดยเมื่อโหนดกระจายตัวแบบปกติ ค่าอุณหภูมิจะมีการเปลี่ยนแปลงแบบค่อยเป็นค่อยไป โดยโหนดที่อยู่ติดกัน

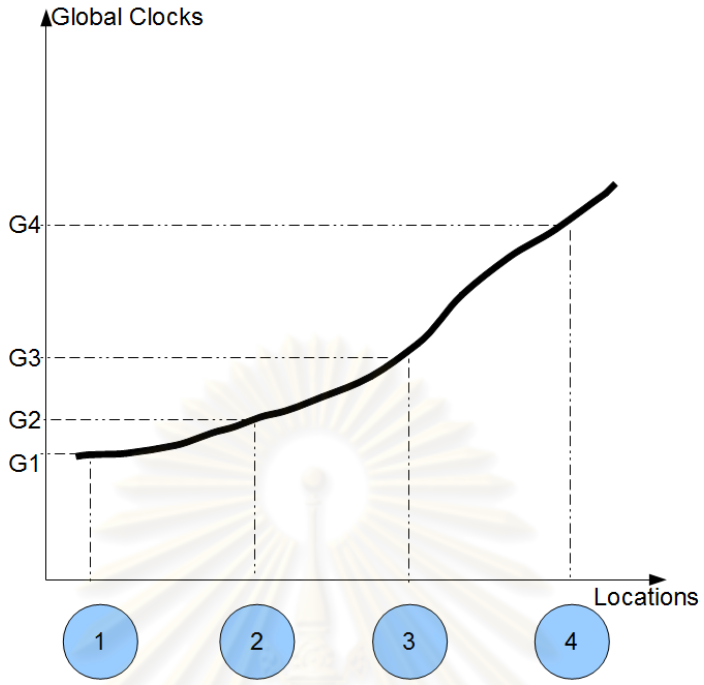
จะอ่านค่าอุณหภูมิได้ใกล้เคียงกัน และโหนดที่อยู่ห่างออกไปจะอ่านค่าอุณหภูมิที่ต่างมากขึ้น ตัวอย่างดังภาพด้านล่าง



ภาพที่ 2.4 แสดงคุณสมบัติเกรเดียนของอุณหภูมิ

จากภาพจะเห็นว่าอุณหภูมิของโหนดที่ 1 คือ T1 มีค่าใกล้เคียงกับอุณหภูมิของโหนดที่ 2 คือ T2 และมีค่าอุณหภูมิแตกต่างจากโหนดที่อยู่ไกลออกไปคือโหนด 3 (T3) และ โหนด 4 (T4) ซึ่งแสดงให้เห็นถึงคุณสมบัติเกรเดียนของอุณหภูมิ

สำหรับการประสานเวลาที่มีคุณสมบัติเกรเดียนนั้น ค่าเวลาครอบคลุมของสองโหนดที่อยู่ติดกันหรือเป็นโหนดเพื่อนบ้านกันต้องมีค่าใกล้เคียงกัน หรือความผิดพลาดในการประสานเวลาต่ำ และค่าเวลาครอบคลุมระหว่างโหนดที่ไม่ได้เป็นเพื่อนบ้านกันสามารถมีค่าแตกต่างกันได้มากขึ้นแต่ยังคงอยู่ในระดับที่ยอมรับได้



ภาพที่ 2.5 แสดงคุณสมบัติเกรเดียนของเวลาครอบคลุม

จากภาพแสดงเวลาครอบคลุมของโหนดต่างๆที่เวลาหนึ่งๆ จะเห็นว่าเมื่อโหนดอยู่ใกล้กันจะมีเวลาครอบคลุมใกล้เคียงกัน หรือความผิดพลาดในการประสานเวลาน้อย และเมื่อโหนดอยู่ไกลกันมากขึ้นจะมีเวลาครอบคลุมต่างกันมากขึ้น หรือความผิดพลาดในการประสานเวลามากขึ้น หรือกล่าวได้ว่าคุณสมบัติเกรเดียนจะพยายามประสานสานเวลาในระดับท้องถิ่นให้ดีที่สุดก่อน และการประสานเวลาของเครือข่ายก็จะขยายวงออกไปโดยอัตโนมัติ

เนื่องจากการประยุกต์ใช้เครือข่ายตัวรับรู้แบบไร้สายนั้น มักจะมีความร่วมมือกันระหว่างโหนดที่อยู่ใกล้กันมากกว่า ดังนั้นคุณสมบัติเกรเดียนจึงเป็นคุณสมบัติที่เหมาะสมกับเครือข่ายประเภทนี้ตัวอย่างเช่น การตรวจจับความทิศทางและความเร็วของวัตถุ

เมื่อโหนดทำการประสานเวลาแบบมีโครงสร้าง ด้วยโปรโตคอลเช่น FTSP หรือ TPSN จะทำให้โหนดใดๆที่อยู่ใกล้กันมีโอกาศอยู่บนคนละกิ่งของโครงสร้างต้นไม้ ทำให้ความสัมพันธ์ของการประสานเวลาห่างกันมากกว่าหนึ่งฮอป หรือสูงที่สุดเป็นสองเท่าของเส้นผ่าศูนย์กลางของเครือข่าย ให้ H_{ij} เป็นจำนวนฮอปความสัมพันธ์ของการประสานเวลา เมื่อโหนด i และ j เป็นเพื่อนบ้านกัน (อยู่ในรัศมีการส่งถึงกัน) ซึ่งความผิดพลาดในการประสานเวลานั้นแปรผันตามค่า H_{ij} และ D คือจำนวนฮอปของเส้นผ่าศูนย์กลางของเครือข่าย เพราะฉะนั้นถ้าการประสานเวลามีลักษณะที่เป็นโครงสร้างต้นไม้ (Structured Synchronization)

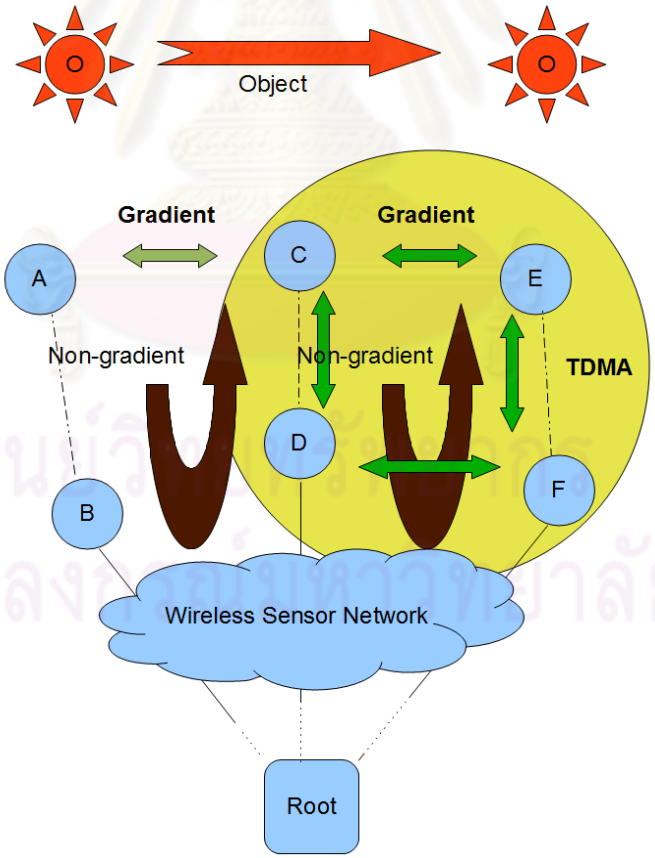
$$1 \le H_{ij} \le 2D$$

เนื่องจากถ้ารากของต้นไม้อยู่ด้านนอกสุดของเครือข่าย และโหนด i และ j อยู่คนละกิ่งหลัก (ที่แยกออกมาจากราก) ก็จะทำให้ H_{ij} สูงที่สุดเท่ากับ 2D ถ้าโหนด i และ j อยู่บนกิ่งเดียวกันและอยู่ติดกัน จะทำให้ H_{ij} น้อยที่สุดเท่ากับ 1 ดังนั้นการประสานเวลาที่วิธีการประสานกับโหนดราก ตำแหน่งของโหนดรากก็จะมีผลกับประสิทธิภาพของการประสานเวลา

แต่ถ้าการประสานเวลานั้นเป็นแบบกระจาย โดยในวิทยานิพนธ์ฉบับนี้ใช้การเฉลี่ยค่าจากทุกโหนดที่เป็นเพื่อนบ้านกัน ดังนั้นการประสานแบบกระจายจึงมีคุณสมบัติเกรเดียน เพราะ

$$H_{ij} = 1$$

ถ้าโหนด i และ j เป็นโหนดเพื่อนบ้านกัน ทำให้การประสานเวลามีความผิดพลาดน้อยและการทำงานร่วมกันของโหนดมีประสิทธิภาพมากยิ่งขึ้น



ภาพที่ 2.6 แสดงตัวอย่างการประยุกต์ใช้คุณสมบัติเกรเดียน

จากภาพเมื่อมีวัตถุ O เคลื่อนที่จากซ้ายไปขวา โหนด A, C, และ E ก็จะทำการประทับเวลาที่ตรวจจับวัตถุได้แล้วส่งไปที่โหนดประมวลผล ดังนั้นถ้าเวลาถูกประสานกันอย่างแม่นยำ การคำนวณทิศทางและความเร็วก็จะมีความแม่นยำ แต่ถ้าเวลาถูกประสานกันอย่างไม่แม่นยำ การคำนวณทิศทางและความเร็วก็จะมีความผิดพลาดได้ โดยจะเห็นว่า เมื่อโหนด A ถึง E ทำการประสานเวลากับโหนดราก ทำให้เกิดโครงสร้างต้นไม้ จะทำให้ความสัมพันธ์ของโหนด A และ C เป็น $1 \le H_{AC} \le 2D$ และความสัมพันธ์ของโหนด A และ E เป็น $2 \le H_{AE} \le 2D$ ดังนั้นทำให้โอกาศที่ความผิดพลาดในการประสานเวลาจะอยู่ในระดับที่สูงนั้นมีมาก และจากภาพ สมมุติว่าโหนด C, D, E และ F ต้องการที่จะส่งข้อมูลแบบ TDMA ถ้าความผิดพลาดในการประสานเวลาระหว่างโหนดมีสูง โอกาศที่เฟรมข้อความจะชนกันก็สูงขึ้น หรือไม่เช่นนั้นก็ต้องเพิ่ม Time Gap ให้กว้างเพื่อป้องกันการชนกัน ทำให้อัตราการใช้ช่องสัญญาณต่ำลง

อย่างไรก็ตามเมื่อระบบเลือกใช้การประสานเวลาแบบกระจายทำให้ $H_{AC} = 1$ และ $H_{AE} = 2$ ทำให้การตรวจจับความเร็วและทิศทางมีความแม่นยำมากขึ้น และ โหนด C, D, E, และ F ซึ่งเป็นโหนดเพื่อนบ้านกัน มีค่า H = 1 ทั้งหมด ทำให้การส่งข้อความแบบสลับเวลามีประสิทธิภาพมากขึ้น

2.2 เอกสารและงานวิจัยที่เกี่ยวข้อง

การประสานเวลาเป็นปัญหาที่ได้รับความสนใจมานาน โดยเฉพาะอย่างยิ่งในระบบกระจายซึ่งแต่ละโหนดมีหน่วยประมวลผลและแหล่งกำเนิดสัญญาณนาฬิกาเป็นของตัวเอง แต่จะต้องทำงานร่วมกันอย่างเป็นระบบ เพื่อให้ผลการทำงานออกมาถูกต้องเสมือนทำงานอยู่บนเครื่องเดียวกัน [7] ตัวอย่างของระบบกระจายเช่น ระบบฐานข้อมูลซึ่งลำดับของการอัพเดทข้อมูลนั้นมีความสำคัญมากเพื่อให้การทำงานนั้นมีความถูกต้อง ดังนั้นความเข้าใจเวลาที่ตรงกัน (A common notion of time) จึงเป็นสิ่งที่สำคัญเช่นกัน

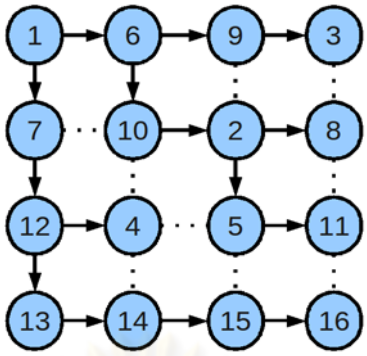
ด้วยความจำเป็นดังกล่าว Network Time Protocol (NTP) [3] จึงถูกนำมาใช้อย่างแพร่หลาย แต่เนื่องจากข้อสมมติฐานที่ว่าการส่งข้อมูลสามารถส่งออกได้ทันทีเนื่องจาก NTP ถูกออกแบบมาสำหรับเครือข่ายแบบมีสายที่สามารถส่งได้อย่างรวดเร็วหรือมีความหน่วงน้อยมาก แต่ NTP มีความแม่นยำในระดับวินาที NTP จึงไม่เหมาะกับการนำมาใช้กับเครือข่ายตัวรับรู้แบบไร้สาย ซึ่งต้องการความแม่นยำในระดับมิลลิวินาที และการส่งข้อมูลแต่ละครั้งมีความหน่วงที่ไม่คงที่ขนาดหลายร้อยมิลลิวินาที ส่งผลให้ความแม่นยำลดลงไปอย่างมาก

Reference Broadcast Synchronization (RBS) [8] แก้ปัญหาดังกล่าวด้วยการให้โหนดใดโหนดหนึ่งทำหน้าที่บรอดคาสข้อความ และให้โหนดที่อยู่ข้างเคียงประทับตราเวลาและทำการแลกเปลี่ยนค่าเวลาดังกล่าวเพื่อประสานเวลากัน การบรอดคาสข้อความดังกล่าว

เปรียบเสมือนการสร้างจุดอ้างอิงให้กับโหนดรอบข้างและเป็นการกำจัดความหน่วงการส่งและความหน่วงการเข้าถึงช่องสัญญาณ ซึ่งเป็นความหน่วงที่มีขนาดใหญ่ที่สุด [9] สำหรับการประสานเวลาแบบทั่วทั้งเครือข่าย (Network-wide Synchronization) ก็จะใช้โหนดที่อยู่ระหว่างหลายๆบรอดคาสโดเมนทำหน้าที่เป็นเกตเวย์ในการแปลงเวลาจากโดเมนหนึ่งไปหาอีกโดเมนหนึ่ง

Timing-sync Protocol for Sensor Networks (TPSN) [4] ได้พิสูจน์ทางคณิตศาสตร์และทำการทดลองให้เห็นว่า Sender-to-Receiver Synchronization จะมีความแม่นยำเป็นสองเท่าของ Receiver-to-Receiver Synchronization ถ้าใช้การตราเวลาในระดับชั้นแมคเข้ามาช่วย และการประสานเวลาแบบทั่วทั้งเครือข่ายทำได้โดยการสร้างโครงสร้างต้นไม้ซึ่งมีการเลือกโหนดรากขึ้นมา และโหนดลูก (Child node) จะทำการประสานเวลากับโหนดแม่ (Parent node) เรื่อยไปจนถึงโหนดราก ก็จะทำให้ประสานเวลาแบบทั่วทั้งเครือข่ายได้ อย่างไรก็ตาม TPSN ถูกมองว่าเป็น อัลกอริทึมแบบศูนย์กลาง (Centralized Algorithm) เนื่องจากโหนดรากเป็นโหนดกลางในการประสานเวลา เมื่อโหนดรากตายลงหรือไม่สามารถติดต่อได้ ประสิทธิภาพของการประสานเวลาก็จะลดลงไปอย่างมาก และโหนดลูกก็จะต้องพึ่งพาโหนดแม่ตลอดเวลา เมื่อโหนดแม่ตายไปหรือติดต่อไม่ได้ก็จะต้องหาโหนดแม่ใหม่และเริ่มกระบวนการประสานเวลาใหม่อีกครั้งหนึ่ง

Flooding Time Synchronization Protocol (FTSP) [5] ได้เสนอวิธีการประสานเวลาโดยใช้การส่งท่วมเครือข่าย (Flooding) เพื่อให้โหนดใดๆมีโอกาสรับข้อความประสานเวลามากขึ้นเพราะเมื่อโหนดใดโหนดหนึ่งตายไป ก็จะมีโหนดอื่นๆทำการส่งข้อความประสานเวลาออกมาเสมอ การทำงานของ FTSP เปรียบเสมือนการส่งข้อมูลซ้ำๆเพื่อให้โหนดประสานเวลาอยู่ได้ตลอดเวลาไม่ว่าจะรับข้อความจากโหนดใด อย่างไรก็ตามถึงแม้ว่า FTSP จะไม่มีการสร้างโครงสร้างต้นไม้ขึ้นมาแต่ความสัมพันธ์เชิงเวลาของโหนดในเครือข่ายนั้นเป็นโครงสร้างต้นไม้อย่างหลีกเลี่ยงไม่ได้ เนื่องจากโหนดใดๆจะคำนวณอัลกอริทึมการประสานเวลาจากข้อความประสานเวลาที่มีหมายเลขมากที่สุดเสมอ ดังนั้นโหนดจะรับข้อความประสานเวลาเข้ามาคำนวณจากโหนดที่ส่งข้อความให้มันก่อนเสมอ และจะโยนข้อความจากโหนดอื่นๆทิ้งเพราะมองว่าซ้ำกัน ดังนั้นความสัมพันธ์เชิงเวลาของโหนดใดๆก็จะแนบแน่นกับโหนดที่ส่งข้อความให้มันเป็นโหนดแรก และไม่มีความสัมพันธ์กับโหนดข้างเคียง ทำให้เกิดเป็นโครงสร้างต้นไม้เชิงเวลาขึ้น



ภาพที่ 2.7 แสดงความสัมพันธ์เชิงเวลาของโหนด FTSP

จากรูปจะเห็นว่าโหนดที่ 10 ถึงแม้ว่าจะได้รับข้อความประสานเวลาจากโหนดที่ 6 และ 7 แต่เนื่องจากมันได้รับข้อความจากโหนด 6 ก่อนเสมอ ทำให้โหนด 10 มีความสัมพันธ์เชิงเวลาเสมือนโหนดลูกของโหนดที่ 6 และไม่มีความสัมพันธ์กับโหนด 7 และเมื่อพิจารณาโหนดที่ 11 และ 16 ถึงแม้ว่าจะเป็นเพื่อนบ้านกันแต่มีความสัมพันธ์เชิงเวลาถึง 11 ฮอป ทำให้ความผิดพลาดในการประสานเวลาของโหนดทั้งสองมีสูง FTSP จึงไม่มีคุณสมบัติเกรเดียน [6]

Gradient Time Synchronization Protocol (GTSP) [6] ได้เสนอวิธีการประสานเวลาโดยใช้ข้อความประสานเวลาจากโหนดเพื่อนบ้าน นำมาเฉลี่ยค่าออฟเซตเวลา (Time Offset) และสกิลเวลา (Time Skew) เมื่อเทียบกับโหนดเพื่อนบ้าน ทำให้สามารถประสานเวลากันได้ และเมื่อทุกโหนดทำการเฉลี่ยค่าไปเรื่อย ก็จะสามารถประสานเวลาทั่วทั้งเครือข่ายได้ และเนื่องจากการประสานเวลาจะใช้ข้อความจากโหนดเพื่อนบ้านทั้งหมด ความสัมพันธ์เชิงเวลาระหว่างโหนดเพื่อนบ้านจะเป็นตอนเดียว (Single Hop) เสมอ และทำให้ความผิดพลาดในการประสานเวลาระหว่างโหนดเพื่อนบ้านต่ำในขณะที่สามารถรักษาความผิดพลาดในการประสานเวลาของทั้งเครือข่ายอยู่ในระดับที่ยอมรับได้ ซึ่งทั้งหมดเป็นคุณสมบัติเกรเดียนที่เหมาะสมสำหรับเครือข่ายตัวรับรู้แบบไร้สาย

งานวิจัยชิ้นอื่นๆที่น่าสนใจประกอบไปด้วย Post-facto Synchronization [10] ซึ่งจะทำการประสานเวลาหลังจากที่มีเหตุการณ์ (Event) ที่สนใจเกิดขึ้น โดยไม่ต้องส่งข้อความประสานเวลาตลอดเวลา และงานของ Kay Romer [11] จะทำการประสานเวลาเพื่อลำดับเหตุการณ์ให้ถูกต้อง งานวิจัย Time Diffusion [22] ทำการสร้างโครงสร้างต้นไม้หลายต้นเพื่อทำการเฉลี่ยค่าเวลาโดยมี Master node และ Diffused leader หลายโหนดเป็นรากของต้นไม้ ซึ่งทำให้ความซับซ้อนในการประมวลผลและขนาดหน่วยความจำสูง งาน TinySync [21] เสนอการประมวลแบบใหม่มาแทนที่ Linear Regression ซึ่งให้ความแม่นยำที่สูงกว่า และสามารถตรวจจับความเบี่ยงเบนออกของเวลาที่ไม่เป็นเชิงเส้นได้ ETA [12] ได้เสนอบริการพื้นฐานสำหรับการประสานเวลาและเสนอโปรโตคอลแบบไม่ตื่นตัว RITS และ แบบตื่นตัว RATS ซึ่งทำงานคล้ายกับ

FTSP งาน ACES [18] ใช้เทคนิคตัวกรองคาลมัน (Kalman Filter) ในการติดตามเวลาของโหนดเพื่อนบ้าน ซึ่งได้เสนอแบบจำลองของสัญญาณนาฬิกาซึ่งมีทั้งส่วนที่เป็นเชิงเส้นและเชิงสุ่ม แต่การคำนวณของตัวกรองคาลมันก็มีความซับซ้อนที่สูงมาก

บทที่ 3

การออกแบบการประสานเวลาแบบกระจาย สำหรับเครือข่ายตัวรับรู้แบบไร้สาย

การออกแบบโปรโตคอลการประสานเวลาแบบกระจายสำหรับเครือข่ายตัวรับรู้แบบไร้สายในวิทยานิพนธ์ฉบับนี้ ได้คำนึงถึงปัจจัยสำคัญด้วยกันสามประการคือ

1) มีคุณสมบัติเกรเดียน (Gradient Orientation) ซึ่งเป็นคุณสมบัติที่ทำให้โหนดที่อยู่ในบริเวณเดียวกันมีความแนบแน่นการประสานเวลาสูง ส่งผลให้เมื่อแอปพลิเคชั่นเรียกใช้งานได้มีประสิทธิภาพมากขึ้น เช่นการเข้าถึงช่องสัญญาณแบบแบ่งเวลา (TDMA) การรวมข้อมูล (Data Fusion) หรือการตรวจจับวัตถุ (Object Tracking)

2) ความแม่นยำในการประสานเวลา (Accuracy) การประสานเวลาที่ดีนั้นจะต้องมีความแม่นยำสูง นั่นคือเวลาครอบคลุมของแต่ละโหนดที่เวลากายภาพเดียวกันนั้นต้องเท่ากันหรือต่างกันเพียงเล็กน้อย

3) การใช้พลังงาน (Energy Consumption) เนื่องจากพลังงานที่มีอยู่อย่างจำกัดของเครือข่ายตัวรับรู้ ทำให้การใช้พลังงานจะต้องเป็นไปอย่างมีประสิทธิภาพ

3.1 ข้อสมมุติฐาน (Assumptions)

1. โหนดสามารถบรอดคาสข้อความไปบนช่องสัญญาณไร้สายได้
2. โหนดมีหมายเลขประจำตัวที่ไม่ซ้ำกันเองในเครือข่าย
3. ไม่มีโหนดภายนอกทำการโจมตีการประสานเวลา
4. โหนดเพื่อนบ้านสามารถตื่นมาทำการประสานเวลาอย่างน้อยทุกคาบการประสานเวลา
5. ระบบมีการพลวัตไม่มากจนเกินไป
6. โหนดมีการกระจายตัวของตำแหน่งแบบปกติ และความหนาแน่นของโหนดไม่มากเสียจนทำให้มีการชนกันของเพ็คเกจและระบบไม่สามารถทำงานได้

3.2 เครื่องมือที่ใช้ในการวิจัย

1. คอมพิวเตอร์ HP, Dell
2. อุปกรณ์ตัวรับรู้ไร้สายเพลตฟอร์ม Mote รุ่น Telosb [17]
3. ระบบปฏิบัติการ Ubuntu [16] และ TinyOS [13]
4. โปรแกรมภาษา nesC [15] , Java, และ Python

3.3 โปรโตคอลการประสานเวลา (Time Synchronization Protocol)

โปรโตคอลการประสานที่ดีและเหมาะสมกับการนำไปใช้กับเครือข่ายตัวรับรู้แบบไร้สายนั้นจะต้องมีคุณสมบัติเกรเดียน ใช้พลังงานอย่างมีประสิทธิภาพ และมีความแม่นยำสูง โหนดตัวรับรู้แต่ละตัวมีค่าเวลาท้องถิ่น (Local Clock) ซึ่งเกิดขึ้นจากสัญญาณของเวลาฮาร์ดแวร์ (Hardware Clock) ซึ่งเพิ่มขึ้นเรื่อยๆตามความถี่ของออสซิลเลเตอร์ และค่าเริ่มต้นใดๆ ดังสมการ

$$L_i(t) = \int_{t_o}^{t} h_i(\tau) d\tau + \Theta_i(t_0) \tag{3.1}$$

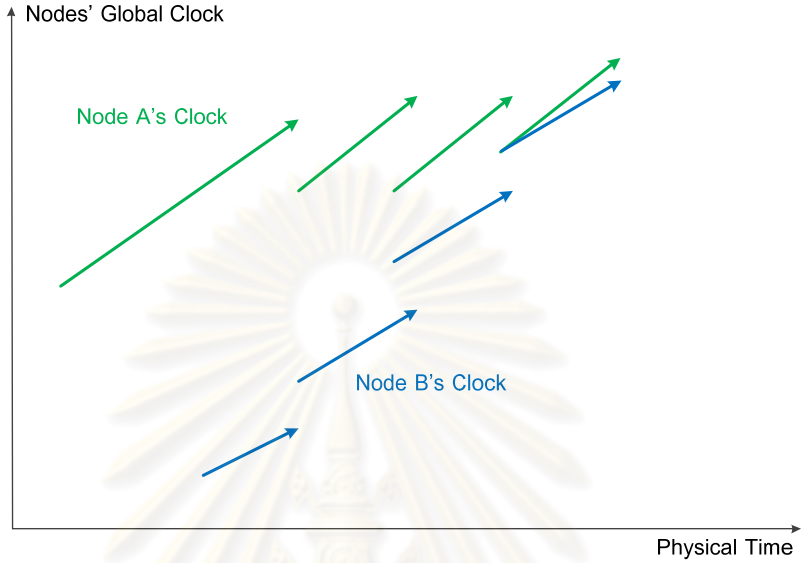
โดยที่ $L_i(t)$ คือเวลาท้องถิ่นของโหนด i ที่เวลา t , $h_i(\tau)$ คือเวลาฮาร์ดแวร์ที่คาบเวลาใดๆ และ $\Theta_i(t_0)$ คือค่าเวลาเริ่มต้นที่ t_0 ใดๆ ซึ่งเวลาดังกล่าวเป็นจำนวนสัญญาณนาฬิกาซึ่งเป็นเลขจำนวนเต็มที่เพิ่มขึ้นทีละหนึ่งหน่วยตามความถี่ของออสซิลเลเตอร์ และเมื่อโหนดมีการส่งข้อความประสานเวลาทุกๆคาบการประสานเวลา (Synchronization Period) เช่น ทุกๆ 30 วินาที โหนดเพื่อนบ้านที่รับข้อความได้จะมีการชดเชยค่าการประสานเวลา เพื่อให้ค่าเวลาครอบคลุม (Global Clock) ตรงกัน ดังสมการ

$$G_i(t) = L_i(t) + \theta_i(t) + \omega_i(t) \tag{3.2}$$

โดยที่ $G_i(t)$ คือเวลาครอบคลุมของโหนด i ที่เวลา t , $\theta_i(t)$ และ $\omega_i(t)$ คือค่าชดเชยเวลาเหลื่อมและเวลาบิดเบือนที่เวลา t ตามลำดับ ดังนั้นการประสานเวลาจะประกอบไปด้วยส่วนชดเชยค่าเวลาเหลื่อมและส่วนชดเชยค่าเวลาบิดเบือน จากนั้นจะกล่าวถึงพลวัตเครือข่ายและการวิเคราะห์อัลกอริทึม

3.3.1 การชดเชยค่าเวลาเหลื่อม

แนวคิดของการชดเชยค่าเวลาเหลื่อมอธิบายดังภาพ



ภาพที่ 3.1 แสดงแนวคิดของการประสานเวลาเหลื่อม

จากภาพเริ่มต้นนั้นโหนดจะมีเวลาครอบคลุมเท่ากับเวลาท้องถิ่น และเมื่อโหนดมีการส่งข้อความประสานเวลากันจะทำการชดเชยเวลาเหลื่อมเพื่อที่จะปรับเวลาครอบคลุมให้เท่ากัน โดยโหนดที่มีค่าเวลาที่มากกว่าก็จะลดเวลาของตัวเองลง และโหนดที่มีค่าเวลาน้อยกว่าก็จะเพิ่มเวลาขึ้น จนกระทั่งเวลาครอบคลุมมีค่าเท่าๆกัน

โดยค่าชดเชยเวลาเหลื่อมคำนวณจากสมการและรหัสเทียมดังนี้

$$\theta_i(t) = \theta_i(t-1) + \frac{G_j(t-1) - G_i(t-1) + \gamma_i(t-1)}{|N_i(t-1)| + 1} \qquad (3.3)$$

$$\gamma_i(t) = \{G_j(t-1) - G_i(t-1)\} \bmod \{|N_i| + 1\}$$

โดยที่ G_j และ G_i คือเวลาครอบคลุมของโหนด j และ i ตามลำดับและ γ_i คือเศษเหลือจากหารจำนวนเต็มด้วยจำนวนเต็มของ i ซึ่งจะป้องกันไม่ให้การหารปัดเศษทิ้งปัดเศษทิ้งตลอดเวลา และ $|N_i(t-1)|$ คือจำนวนของโหนดเพื่อบ้านซึ่งมีการเพิ่มขึ้นและลดลงจากพลวัตของเครือข่าย

1	Create a table item to store reference points, *table[N]* where N is the maximum number of neighbors
2	Node i receives a message from node j
3	Node i timestamp the message and store it to *table.receiveLocalTime*
4	Node i converts the local time to the corresponding global time and stores it to *table.receiveGlobalTime*
5	*table.sendGlobalTime <- msg.sendGlobalTime* *table.sendLocalTime <- msg.sendLocalTime*
6	Node i calculate time offset relative to node j *table.timeOffset <- table.sendGlobalTime - receiveGlobalTime*

7	if table.timeOffset > JUMP_THRESHOLD
	$\theta_i(t) = \theta_i(t-1) + \text{table.timeOffset}$
	else
	$\theta_i(t) = \theta_i(t-1) + \frac{\text{table.timeOffset} + \gamma_i(t-1)}{\lvert N_i(t-1)+1 \rvert}$
	end if

ภาพที่ 3.2 รหัสเทียมแสดงการหาค่าชดเชยเวลาเหลื่อม

จากภาพที่ 3.2 สามารถอธิบายรหัสเทียมได้ดังนี้

- บรรทัดที่ 1 โหนดจะทำการสร้างตารางเก็บจุดอ้างอิงในการประสานเวลาไว้ในรูปแบบของ STRUCT ดังตารางที่ 3.1 โดยจำนวนของแถวขึ้นอยู่กับจำนวนเพื่อนบ้านมากที่สุด สมมติว่าจำนวนเพื่อนบ้านมากที่สุดเท่ากับ 8 ก็สามารถสร้างตารางดังกล่าวจำนวนแปดแถว

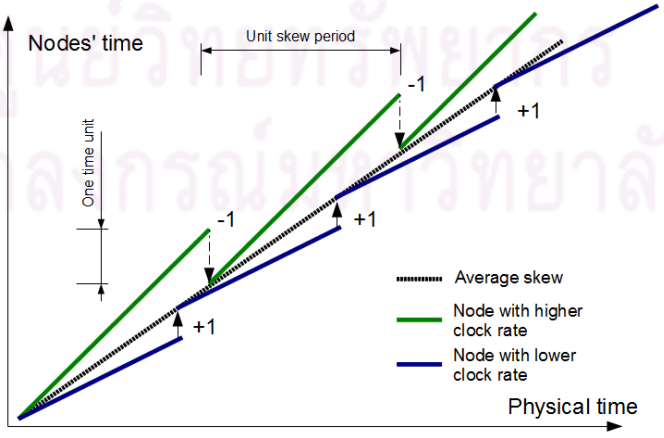
- บรรทัดที่ 2 และ 3 โหนดรับข้อความประสานเวลาด้วยการประทับเวลาในระดับชั้นแมค และเก็บข้อความในตารางใน *table.receiveLocalTime*

- บรรทัดที่ 4 โหนดแปลงเวลารับท้องถิ่นให้เป็นเวลาครอบคลุมตามสมการ (3.2) และเก็บใน*table.receiveGlobalTime*
- บรรทัดที่ 5 ทำการคัดลอกค่าเวลาของโหนดส่งมาจากข้อความประสานเวลา
- บรรทัดที่ 6 คำนวณค่าเวลาเหลื่อมโดยเอาเวลาครอบคลุมของโหนดส่งมาลบกับของโหนดรับ
- บรรทัดที่ 7 ทำการเปรียบเทียบเวลาเหลื่อมกับค่าขีดแบ่งใดๆ เพื่อทำการกระโดดค่าเวลาครอบคลุม ทำให้เวลาครอบคลุมของเครือข่ายลู่เข้าอย่างรวดเร็ว แต่ถ้าเวลาเหลื่อมน้อยกว่าค่าขีดแบ่งก็จะทำการเฉลี่ยแบบเพิ่มส่วนดังที่ได้อธิบายไปในบทที่ 2

อัตราการปรับค่าเวลาเหลื่อมของโปรโตคอลที่ได้ออกแบบ (DTSP) ไว้จะเท่ากับ N/B เมื่อ N คือจำนวนโหนดเพื่อนบ้านและ B คือคาบการประสานเวลา ในขณะที่อัตราการปรับค่าเวลาเหลื่อมของโปรโตคอล GTSP เท่ากับ 1/B เท่านั้น ทำให้ DTSP สามารถปรับเวลาได้รวดเร็วกว่า

3.3.2 การชดเชยค่าเวลาบิดเบือน

เมื่อโหนดทำการชดเชยค่าเวลาเหลื่อมไปช่วงเวลาหนึ่งแล้วค่าเวลาครอบคลุมจะสามารถปรับให้เท่ากันได้ แต่เวลาบิดเบือนซึ่งเกิดจากการเบี่ยงเบนทางเวลาก็จะทำให้ค่าเวลาครอบคลุมเกิดความผิดพลาดขึ้นอีก เพราะฉะนั้นโปรโตคอลการประสานเวลาจึงต้องทำการชดเชยเวลาบิดเบือนดังกล่าวด้วย เพื่อเพิ่มความแม่นยำให้กับการประสานเวลา



ภาพที่ 3.3 แสดงแนวคิดการประสานเวลาบิดเบือน

จากภาพจะเห็นได้ว่าถึงแม้ว่าโหนดจะมีค่าเวลาเริ่มต้นเท่ากัน แต่เนื่องจากการเบี่ยงเบนทางเวลาทำให้โหนดจะมีค่าความผิดเพี้ยนเกิดขึ้นเมื่อเวลาผ่านไป เรียกว่าเวลาบิดเบือน ดังนั้นเพื่อให้การประสานเวลามีประสิทธิภาพ จะต้องมีการชดเชยเวลาบิดเบือนด้วยรหัสเทียมดังนี้

1	Create two table items to store capturing reference points, *startCapture[N]* and *stopCapture[N]*
2	Start capturing by copying *table[N]* to *startCapture[N]*
3	Start the capturePeriod timer, allowing time for the table to be updated.
4	Stop capturing by copying *table[N]* to *stopCapture[N]*
5	Node i calculate relative average skew by $SK_i = \frac{\sum_{j \in N_i} \frac{stopCapture[j].sendLocalTime - startCapture[j].sendLocalTime}{stopCapture[i].receiveLocalTime - startCapture[i].receiveLocalTime} + 1}{|N_i| + 1}$
6	Node i calculates a skew period by $SP_i = \frac{1}{SK_i - 1}$

7	if (SP_i > 0)
	when $t = t + SP_i$, $\omega_i = \omega_i + 1$
	else if (SP_i < 0)
	when $t = t + SP_i$, $\omega_i = \omega_i - 1$
	else
	extend the capture period

8	Use SP as an adaptive estimation period for the next estimation

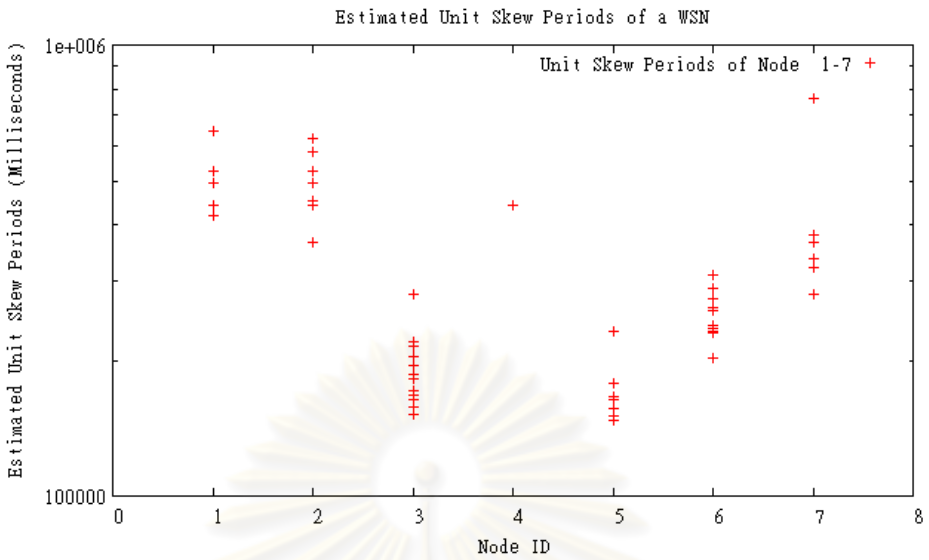
ภาพที่ 3.4 รหัสเทียมแสดงการหาค่าชดเชยเวลาบิดเบือน

สามารถอธิบายรหัสเทียมได้ดังนี้

- บรรทัดที่ 1 โหนดทำการสร้างตารางเพิ่มอีกสองตารางเพื่อเก็บจุดอ้างอิงตอนเริ่มต้น startCapture และตอนท้าย stopCapture ของการประมาณค่าเวลาบิดเบือน

- บรรทัดที่ 2 โหนดเริ่มการประมาณโดยการคัดลอกค่าในตาราง table ไปที่ startCapture
- บรรทัดที่ 3 เริ่มต้นตัวจับเวลาเพื่อให้ table ถูกปรับให้เป็นปัจจุบัน
- บรรทัดที่ 4 หยุดการประมาณค่าโดยการคัดลอกค่าในตาราง table ไปที่ stopCapture
- บรรทัดที่ 5 จากค่าในตาราง startCapture และ stopCapture ที่ได้ โหนดทำการคำนวณค่าเวลาบิดเบือนเฉลี่ยดังสมการ
- บรรทัดที่ 6 แปลงค่าเวลาบิดเบือนเฉลี่ย (Average Skew, SK)เป็นคาบเวลาบิดเบือนหนึ่งหน่วย (Unit Skew Period, SP) ตัวอย่างเช่น คำนวนค่าเวลาบิดเบือนเฉลี่ยได้ 0.9999875 ซึ่งแสดงว่าโหนดมีความเบี่ยงเบนทางเวลาเร็วกว่าโหนดเพื่อนบ้าน เมื่อทำการแปลงเป็นคาบเวลาบิดเบือน – 80000
- บรรทัดที่ 7 เริ่มต้นตัวจับเวลาเพื่อชดเชยค่าเวลาบิดเบือน
- บรรทัดที่ 8 โหนดนำค่า SP มาพิจารณาคาบการประมาณค่าเวลาบิดเบือนในรอบถัดไป โดยโหนดที่มีค่าเวลาบิดเบือนต่างจากค่าเฉลี่ยของโหนดเพื่อนบ้านมาก ก็จะมีคาบการประมาณที่แคบกว่าโหนดที่มีค่าเวลาบิดเบือนต่างจากค่าเฉลี่ยของโหนดเพื่อนบ้านน้อย ทำให้คาบการประมาณค่าเวลาบิดเบือนนั้นสามารถปรับตัวได้ตามพฤติกรรมของสัญญาณนาฬิกาของแต่ละโหนดเมื่อเทียบกับโหนดเพื่อนบ้าน

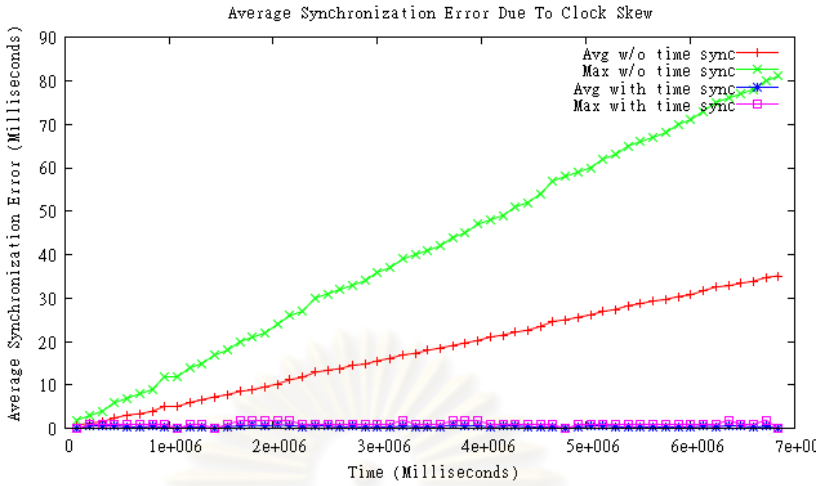
การประมาณค่าเวลาบิดเบือนนั้นจะมีลักษณะที่แตกต่างจากของ GTSP คือมีคาบการประมาณค่าที่กว้างกว่าและปรับตัวได้ ซึ่งจะทำให้การประมาณค่ามีประสิทธิภาพมากกว่า เนื่องจากการเบี่ยงเบนของสัญญาณนาฬิกามีลักษณะค่อยๆเป็นค่อยๆไป ทำให้เวลาบิดเบือนมีค่าน้อยมากในช่วงเวลาหนึ่งๆ และจะมีค่าหนึ่งหน่วยเวลาเมื่อครบคาบเวลาบิดเบือน จากการทดลองโดยใช้โหนด 7 โหนดทำการประสานเวลาและคำนวณคาบเวลาบิดเบือนพบว่ามีค่ามากกว่า 100,000 มิลลิวินาที เกือบถึง 200,000 และบางโหนดมีค่ามากกว่า 400,000 ถึง 1,000,000 มิลลิวินาที ดังรูป



ภาพที่ 3.5 แสดงค่าประมาณคาบเวลาบิดเบือนหนึ่งหน่วยจากการทดลอง

ดังนั้น คาบเวลาในการประมาณค่าเวลาบิดเบือนที่สั้นกว่า เท่ากับคาบการประสานเวลานั้นไม่สามารถประมาณคาบเวลาบิดเบือนได้อย่างมีประสิทธิภาพ เนื่องจากมีคาบดังกล่าวสั้นกว่าคาบที่เกิดขึ้นจริง

นอกจากนั้นเวลาในการทดลองเพื่อแสดงค่าเวลาบิดเบือนที่เกิดขึ้นจริงของเครือข่ายขนาด 7 โหนด ประมาณ 7,000,000 มิลลิวินาที (สองชั่วโมง) เวลาบิดเบือนมากที่สุดประมาณ 81 มิลลิวินาที ดังรูป ซึ่งเท่ากับ 84,000 มิลลิวินาทีต่อหนึ่งหน่วยเวลาบิดเบือน ซึ่งแสดงให้เห็นว่าค่าเวลาบิดเบือนนั้นมีพฤติกรรมแบบค่อยๆเป็นค่อยๆไป คือมีการเปลี่ยนแปลงทีละน้อย ดังนั้นคาบเวลาในการประมาณค่าเวลาบิดเบือนนั้นต้องมีความกว้างที่ครอบคลุมการเกิดเวลาบิดเบือนที่เกิดขึ้นจริง จึงจะมีประสิทธิภาพ



ภาพที่ 3.6 แสดงค่าเวลาบิดเบือนที่เกิดขึ้นจริง

ถึงแม้ว่า GTSP สามารถปรับคาบการประสานเวลาให้มากขึ้นเพื่อทำให้คาบการประมาณค่าเวลาบิดเบือนกว้างขึ้นได้ แต่ก็จะทำให้การชดเชยเวลาเหลื่อมทำได้ช้าด้วยทำให้การลู่เข้าของเวลาครอบคลุมนั้นช้าลงอย่างมากและความผิดพลาดในการประสานเวลาสูงขึ้นตามไปด้วย

โครงสร้างข้อมูลของตาราง table เป็นแบบ STRUCT ดังนี้

Field	Length (Bits)	Description
nodeID	8	หมายเลยประจำตัวของโหนด
seqNum	8	หมายเลขลำดับของข้อความประสานเวลา
sendGlobalTime	32	เวลาครอบคลุมของโหนดส่ง
receiveGlobalTime	32	เวลาครอบคลุมของโหนดรับ
timeOffset	32	เวลาเหลื่อมสัมพัทธ์
sendLocalTime	32	เวลาท้องถิ่นของโหนดส่ง
receiveLocalTime	23	เวลาท้องถิ่นของโหนดรับ

ตารางที่ 3.1 รายละเอียดตารางที่ใช้เก็บข้อมูลการประสานเวลา

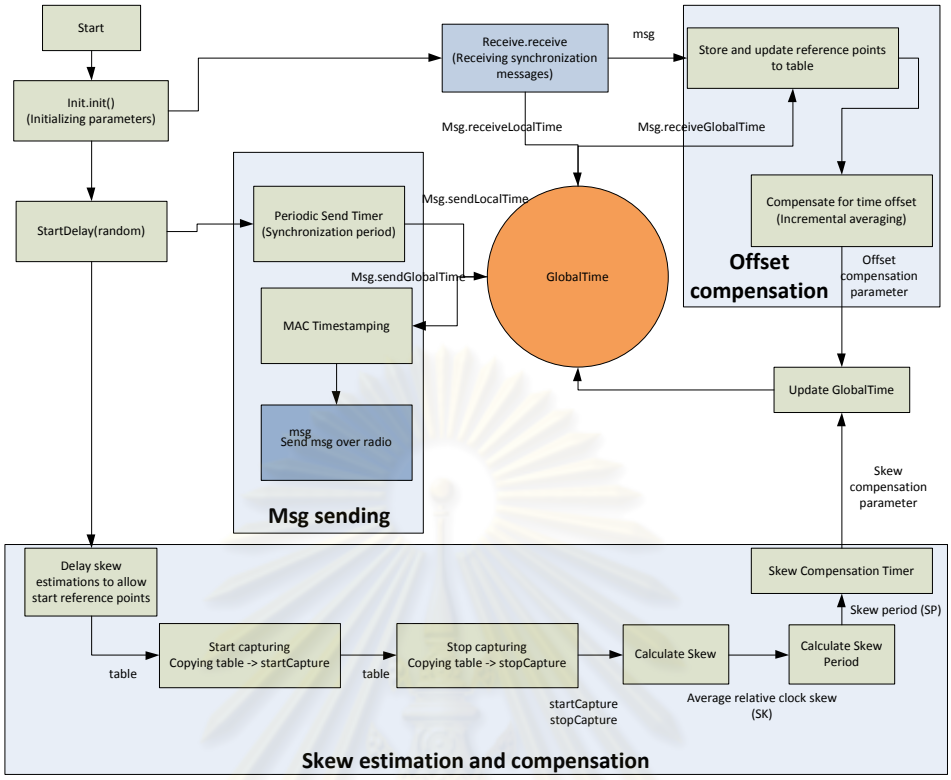
และข้อความประสานเวลามีรูปแบบดังนี้

Field	Length (Bits)	Description
nodeID	8	หมายเลขประจำตัวของโหนด
seqNum	8	หมายเลขลำดับของข้อความประสานเวลา
sendGlobalTime	32	เวลาครอบคลุมของโหนดส่ง
receiveGlobalTime	32	เวลาครอบคลุมของโหนดรับ
sendLocalTime	32	เวลาท้องถิ่นของโหนดส่ง
receiveLocalTime	23	เวลาท้องถิ่นของโหนดรับ

ตารางที่ 3.2 แสดงส่วนต่างๆของข้อความประสานเวลา

ฟิลล์ receiveGlobalTime และ receiveLocalTime ของข้อความประสานเวลา มีไว้เพื่อความสะดวกในการเก็บและส่งค่าเพื่อคำนวนระหว่างฟังก์ชั่นเท่านั้น

สามารถสรุปการทำงานของอัลกอริทึมของโปรโตคอลประสานเวลาดังภาพของแผนภูมิสถานะด้านล่าง



ภาพที่ 3.7 แสดงแผนภูมิสถานะของอัลกอริทึม

เมื่อโปรโตคอลประสานเวลาสามารถคำนวนค่าคาบเวลาบิดเบือน (Skew Period) ได้แล้ว การประยุกต์ใช้งานเครือข่ายตัวรับรู้แบบไร้สายจะได้รับประโยชน์ดังนี้

1. สามารถขยายคาบการประสานเวลาได้ เนื่องจากโหนดสามารถคำนวณคาบเวลาที่จะเกิดความบิดเบือนทางเวลาได้ เพราะฉะนั้นโหนดก็ไม่จำเป็นต้องส่งข้อความประสานเวลาบ่อยๆในช่วงเวลาดังกล่าว
2. ในการจัดตารางเวลาทำงาน (Sleep-wakeup Scheduling) เมื่อโหนดหลับและไม่มีการรับส่งข้อความใดๆ โหนดก็ยังสามารถคำนวณค่าชดเชยเวลาบิดเบือนได้จากคาบเวลาที่หลับเทียบกับคาบเวลาบิดเบือนได้

อย่างไรก็ตาม การขยายคาบการประสานเวลาจะต้องคำนึงถึงปัจจัยอื่นๆ เช่น

- การตอบสนองของระบบต่อพลวัตเครือข่าย โหนดยิ่งขยายคาบการประสานเวลานาน ยิ่งตอบสนองต่อพลวัตได้ช้า

- การประหยัดพลังงาน โหนดยิ่งขยายคาบการประสานเวลานาน ยิ่งประหยัดพลังงานมาก
- ความผิดพลาดการประเวลา โหนดยิ่งขยายคาบการประสานเวลานาน ยิ่งมีความผิดพลาดการประสานเวลาสูงขึ้น

3.3.3 พลวัตเครือข่าย (Network Dynamics)

เครือข่ายตัวรับรู้แบบไร้สายนั้นสามารถเกิดการพลวัตของเครือข่ายได้ เช่น โหนดเคลื่อนที่ โหนดตาย หรือโหนดเกิดใหม่ เนื่องจากโปรโตคอลประสานเวลาในวิทยานิพนธ์ฉบับนี้ใช้การส่งข้อความและประมวลผลเพื่อประสานเวลาภายในบริเวณของโหนดเพื่อนบ้าน เพราะฉะนั้นการที่โหนดตายไป หรือเกิดใหม่นั้น โปรโตคอลสามารถตรวจสอบได้โดยใช้ข้อความประสานเวลาหรือข้อความฮัลโหลแยกออกมา เพื่อเป็นการตรวจสอบว่าโหนดเพื่อนบ้านยังอยู่หรือ หรือมีโหนดใหม่เข้ามาเป็นสมาชิกของเพื่อนบ้านหรือไม่ ขึ้นอยู่กับดีกรีความพลวัตของเครือข่ายและการตอบสนองต่อความพลวัตของระบบที่ต้องการ โดยสามารถแบ่งประเภทของพลวัตได้ดังนี้

1. *โหนดตาย* โหนดในระบบสามารถตั้งค่า Hello_timeout เพื่อลบโหนดเพื่อนบ้านที่ไม่มีการติดต่อมาเป็นระยะเวลาหนึ่ง และโหนดจะลดจำนวนของเพื่อนบ้าน $|N_i(t)|$ เพื่อคำนวณค่าเวลาเหลื่อมและเริ่มต้นการคำนวณเวลาบิดเบือนเฉลี่ยใหม่
2. *โหนดใหม่* เมื่อมีโหนดใหม่เข้ามาบริเวณเพื่อนบ้าน โหนดจะทำการเพิ่มโหนดดังกล่าวเข้าไปในสมาชิกของโหนดเพื่อนบ้าน และเพิ่มค่า $|N_i(t)|$ และเริ่มต้นการคำนวณเวลาบิดเบือนเฉลี่ยใหม่
3. *โหนดเคลื่อนที่* ระบบก็จะทำงานเหมือนกับการตายไปของโหนดจากเพื่อนบ้านกลุ่มหนึ่งและเข้าไปเป็นโหนดใหม่ของอีกกลุ่มเพื่อนบ้าน
4. *โหนดหลับและตื่น* การหลับและตื่นของโหนดเป็นรูปแบบพิเศษของเครือข่ายตัวรับรู้แบบไร้สาย เพื่อประหยัดพลังงานเมื่อไม่ต้องรับส่งข้อความแบบไร้สาย โดยโหนดจะต้องตื่นขึ้นมาภายในคาบการประสานเวลาจึงจะสามารถประสานเวลาได้ และโหนดจะต้องไปหลับไปนานกว่า Hello_timeout ที่ทำให้โหนดอื่นๆเข้าใจว่าเป็นโหนดตาย

ในช่วงเวลาที่โหนดยังไม่สามารถตรวจจับโหนดที่ตายหรือโหนดเคลื่อนที่ออกไปได้นั้น ค่าน้ำหนักของแต่ละโหนดที่นำมาเฉลี่ยค่านั้นจะรวมกันแล้วไม่เท่ากับหนึ่ง ทำให้การเฉลี่ย

ค่านั้นลู่เข้าช้าลง ขึ้นอยู่กับสัดส่วนของโหนดที่ตายหรือโหนดเคลื่อนที่ออกไป และเมื่อหมดเวลา Hello_timeout แล้วโหนดก็สามารถเฉลี่ยค่าได้ถูกต้องต่อไป

3.3.4 วิเคราะห์อัลกอริทึม (Analysis of Algorithms)

อัลกอริทึมการประสานเวลาสามารถวิเคราะห์ได้ดังนี้

1. ความซับซ้อนเชิงการคำนวณ (Computational Complexity) เนื่องจากการคำนวณเพื่อประสานเวลานั้น ใช้การเฉลี่ยค่าทั้งเวลาเหลื่อมและเวลาบิดเบือน ดังนั้นความซับซ้อนเชิงการคำนวณคือ O(C) โดยที่ C คือค่าคงที่ของจำนวนโหนดเพื่อนบ้านสูงสุดที่ระบบสามารถรับได้ กล่าวคือการส่งข้อความไม่มีการชนกันมากจนทำให้ระบบทำงานไม่ได้

2. ความซับซ้อนเชิงหน่วยความจำ (Memory Complexity) เนื่องจากโปรโตคอลจะเก็บจุดอ้างอิงไว้ในตารางตามจำนวนโหนดเพื่อนบ้าน ทั้งในการชดเชยเวลาเหลื่อมและเวลาบิดเบือน ดังนั้น ความซับซ้อนเชิงหน่วยความจำ คือ O(C)

3. ปริมาณข้อความ (Message Overhead) เนื่องจากโหนดเพื่อนบ้านจะทำการส่งข้อความประสานเวลาหนึ่งข้อความทุกคาบการประสานเวลา ดังนั้น ปริมาณข้อความคือ O(C/B) โดยที่ B คือคาบการประสานเวลา

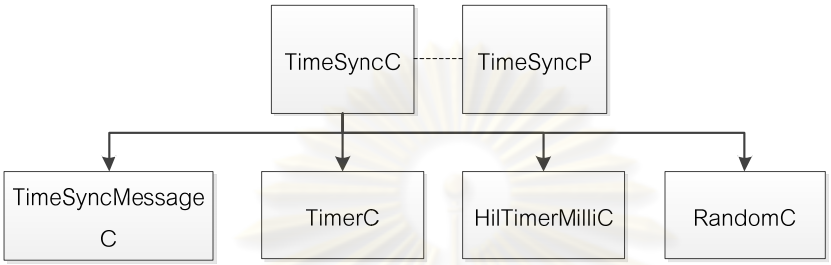
4. ความขยายตัวได้ (Scalability) เนื่องจากโปรโตคอลการประสานเวลาแบบกระจายนี้ใช้ข้อมูลท้องถิ่น (จากโหนดเพื่อนบ้าน) แต่สามารถประสานเวลาครอบคลุมของเครือข่ายได้ นอกจากนั้น ความซับซ้อนเชิงการคำนวณ ความซับซ้อนเชิงหน่วยความจำ ปริมาณข้อความ มีอัตราการเติบโตของเป็นฟังก์ชั่นของจำนวนโหนดเพื่อนบ้าน ดังนั้นโปรโตคอลการประสานเวลาแบบกระจายมีความขยายตัวได้ ภายใต้เงื่อนไขดังนี้

    a. เครือข่ายมีโหนดกระจายตัวแบบปกติหรือโหนดไม่อยู่อย่างหนาแน่นเกินไปจนทำให้ข้อความในระบบชนกันจนไม่สามารถทำงานได้ หรือ จำนวนโหนดเพื่อนบ้านสูงสุดเป็นค่าคงที่ค่าหนึ่ง เมื่อเครือข่ายเชื่อมต่อกันแบบหลายฮอปแล้ว ขนาดของเครือข่ายจะมากกว่าจำนวนโหนดเพื่อนบ้านที่คงที่ไปเรื่อย ทำให้อัลกอริทึมมีความสามารถในการขยายตัวได้

    b. เมื่อพิจารณาความผิดพลาดในการประสานเวลาสูงสุดซึ่งเพิ่มขึ้นตามจำนวนฮอปของเส้นผ่าศูนย์กลางเครือข่าย (D) ดังนั้นความขยายตัวได้ของโปรโตคอลจะต้องพิจารณาความผิดพลาดในการประสานเวลาสูงสุดที่ยอมรับได้ของระบบด้วย

3.4 การทำให้เกิดผล (Implementation)

โปรโตคอลการประสานเวลาที่ได้ออกแบบไว้ถูกทำให้เกิดผลโดยใช้โปรแกรมภาษา nesC และระบบปฏิบัติการ TinyOS ซึ่งผู้วิจัยศึกษาจาก [18] และ [19] โดยมีส่วนประกอบที่ใช้งานดังรูป

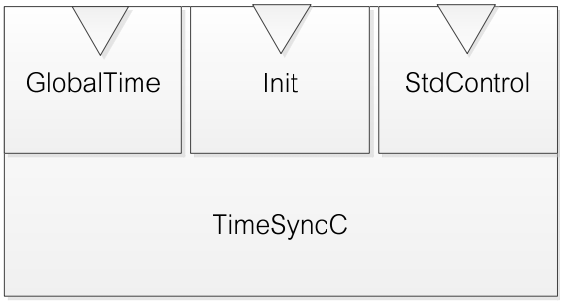


ภาพที่ 3.8 แสดงส่วนประกอบพื้นฐานที่ใช้งาน

โดยที่แต่ละส่วนประกอบทำหน้าที่ดังนี้

- HilMilliTimerC เป็นส่วนประกอบที่แสดงค่าสัญญาณนาฬิกาท้องถิ่น ที่ความถี่ 1 kHz
- TimerC เป็นส่วนประกอบที่ใช้ควบคุมการทำงานของโปรโตคอล เช่น กำหนด Synchronization period, skew capture period, และ skew compensation period
- ActiveMessageC เป็นส่วนประกอบที่ควบคุมการรับส่งข้อความของตัวรับรู้
- TimeSyncC เป็นส่วนประกอบของโปรโตคอลซึ่งทำการเรียกส่วนประกอบพื้นฐานขึ้นมาใช้, ทำการต่อสายส่วนประกอบเข้าด้วยกัน, และแสดงส่วนเชื่อมต่อที่ให้บริการ
- TimeSyncP เป็นส่วนประกอบหลักของโปรโตคอล แสดงอัลกอริทึมการทำงานของโปรโตคอล และโดยปกติจะไม่อนุญาตให้ส่วนประกอบภายนอกมาทำการเชื่อมต่อ

และเนื่องจากส่วนประกอบการประสานเวลาเป็นส่วนประกอบพื้นฐานที่ต้องการให้ส่วนประกอบด้านบนมาเรียกใช้งาน ดังนั้นจึงจำเป็นต้องมีการประกาศส่วนเชื่อมต่อสำหรับให้บริการ (Providing Interfaces) ดังนี้



ภาพที่ 3.9 แสดงส่วนเชื่อมต่อสำหรับให้บริการ

โดยที่ส่วนเชื่อมต่อต่างๆมีหน้าที่ดังนี้

- GlobalTime เป็นส่วนเชื่อมต่อหลักให้เรียกใช้เพื่อที่จะขอค่าเวลาครอบคลุมไปใช้
- Init เป็นส่วนเชื่อมต่อที่ใช้ในการเริ่มการทำงานของส่วนประกอบ
- StdControl ใช้ในการสั่งให้ทำงานหรือหยุดทำงานเมื่อส่วนต่างๆพร้อมทำงานหรือหยุดการทำงาน

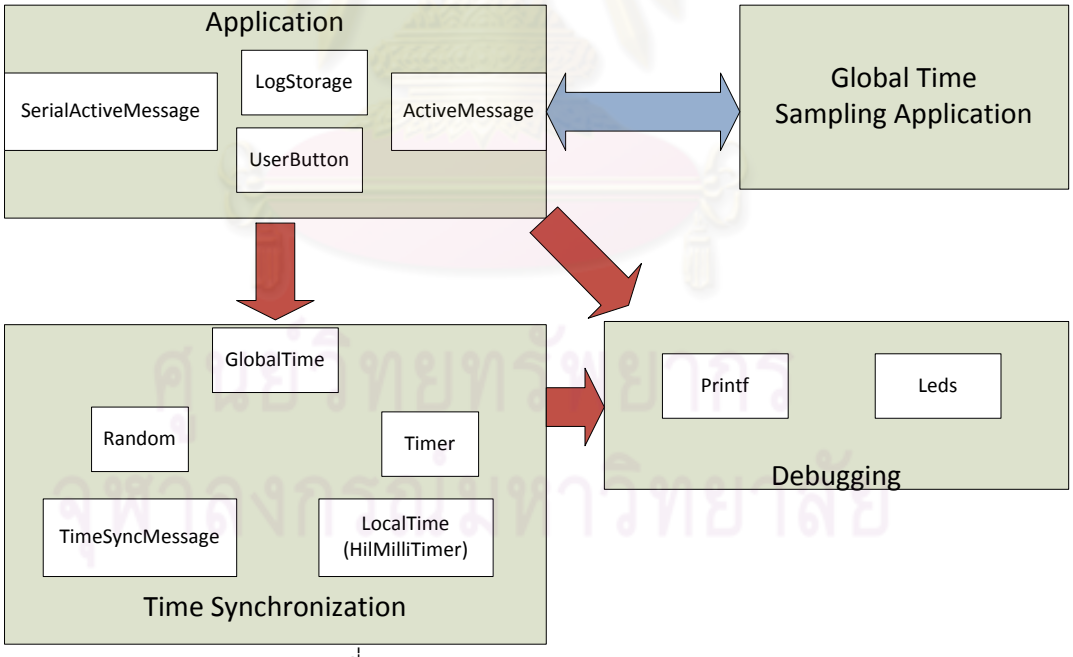
และภายในส่วนประกอบ TimeSyncP ประกอบไปด้วยฟังก์ชัน, คำสั่ง, หรือตัวจัดการเหตุการณ์ (Event handler) ต่างๆ ดังนี้

ชื่อและส่วนหัว	หน้าที่การทำงาน
command error_t Init.init()	ใช้ในการกำหนดค่าเริ่มต้นต่างๆให้กับตัวแปรที่ใช้
event message_t* Receive.receive(message_t* msg, void* payload, uint8_t len)	ใช้ในการกำหนดการทำงานเมื่อได้รับข้อความและการเรียกค่าเวลาเหตุการณ์ด้านรับสามารถทำได้ที่นี่
void task processMsg()	ใช้ในการประมวลผลต่อจากการรับข้อความ
void updateOffset (int32_t timeOffset)	คำนวณค่าเฉลี่ยแบบเพิ่มส่วนเพื่อชดเชยค่าเวลาเหลื่อม
task void sendMsg()	ทำการส่งข้อความ โดยจะต้องเรียกค่าเวลาท้องถิ่นขึ้นมา

	เพื่อกำหนดเป็นเวลาเหตุการณ์ด้านส่งและใส่เข้าไปในข้อความประสานเวลา
event void AMSend.sendDone(message_t* ptr, error_t error)	จะทำงานเมื่อส่งข้อความสำเร็จ ใช้ในการเพิ่มค่าลำดับข้อความ seqNum
void task calculateDrift()	ใช้ในการคำนวณคาบเวลาบิดเบือน

ตารางที่ 3.3 แสดงฟังก์ชั่นการทำงานที่สำคัญของโปรโตคอล

การทำให้เกิดผลของโปรโตคอลประสานเวลานั้นไม่ได้มีเพียงแต่ส่วนของโปรโตคอลเพียงอย่างเดียว ยังต้องอาศัยการสร้างแอปพลิเคชั่นขึ้นมาเรียกใช้โปรโตคอลและทำการรายงานค่าเวลาครอบคลุม, แอปพลิเคชั่นที่ทำการส่งข้อความเก็บตัวอย่างเวลาครอบคลุม, และส่วนหาความผิดพลาด (Debugging) ดังรูป



ภาพที่ 3.10 แสดงสถาปัตยกรรมของระบบ

ดังรายละเอียดดังนี้

1. ส่วน Application คือส่วนที่อยู่ในชั้นสูงสุดที่ทำการเรียกโปรโตคอลประสานเวลาขึ้นมาใช้และทำการติดต่อกับแอปพลิเคชั่นเก็บตัวอย่างเพื่อทำหน้าที่เสมือนเหตุการณ์ที่สนใจให้ทำการประทับเวลาแล้วนำมาเปรียบเทียบกันในการทดลอง โดยที่ในการนำไปใช้งานจริงอาจจะใช้เหตุการณ์ที่สนใจจริงๆแทน เช่น อุณหภูมิเกินค่าที่กำหนด หรือเสียงดังกว่าที่กำหนด ส่วน Application ในระบบที่ทำการทดลองจะต้องใช้งานส่วนย่อยดังนี้

    a. ActiveMessage เพื่อทำการรับข้อความเก็บตัวอย่างจากแอปพลิเคชั่นเก็บตัวอย่างทุกๆคาบเวลาเก็บตัวอย่าง

    b. SerialActiveMessage ทำการนำค่าตัวอย่างส่งไปยังพอร์ตอนุกรม ซึ่งสำหรับอุปกรณ์รุ่น Telosb จะเป็นพอร์ตยูเอสบี เพื่อรายงานตัวอย่างเวลาครอบคลุม

    c. LogStorage หรือถ้าไม่ได้เชื่อมต่อพอร์ตอนุกรมไว้แต่ใช้พลังงานแบตเตอรี่แทน ก็สามารถเก็บค่าตัวอย่างเวลาครอบคลุมไว้ได้โดยการใช้ LogStorage ซึ่งจะเก็บค่าต่างๆไว้ในหน่วยความจำเฟรซ และไม่หายไปเมื่อไม่มีไฟเลี้ยง

    d. UserButton คือส่วนเชื่อมต่อกับผู้ใช้งานเพื่อส่งสัญญาณการเริ่มดึงค่าที่เก็บไว้ในหน่วยความจำเฟรซแล้วส่งไปที่พอร์ตอนุกรม

2. ส่วน Time Synchronization คือส่วนของโปรโตคอลประสานเวลาซึ่งใช้ส่วนย่อยดังนี้

    a. GlobalTime คือส่วนโปรโตคอลประสานเวลาให้บริการ โดยหลักๆก็จะถูกเรียกค่าเวลาครอบคลุมผ่านส่วนย่อยนี้

    b. Random คือส่วนที่สร้างตัวเลขสุ่มขึ้นมาเพื่อให้โปรโตคอลเริ่มส่งข้อความไม่ตรงกัน ลดอัตราการชนกันของข้อความได้ เนื่องจากข้อความจะถูกส่งที่คาบประสานเวลาเท่ากัน ดังนั้นถ้าข้อความเริ่มชนกันก็จะชนกันไปตลอด

    c. Timer คือตัวจับเวลาซึ่งมีประโยชน์มากในการควบคุมการทำงานของโปรโตคอล (ดูแผนภูมิสถานะ) โดยมีทั้งแบบจับเวลาครั้งเดียวและแบบทุกคาบเวลา

    d. LocalTime เป็นส่วนที่สร้างค่านาฬิกาท้องถิ่น ค่านี้จะเพิ่มขึ้นทีละหนึ่งตามความถี่ของสัญญาณนาฬิกากายภาพ

e. TimeSyncMessage คือส่วนที่ใช้รับส่งข้อความประสานเวลา โดยอาศัยหลักการการชดเชยเวลาที่ล่วงเลยไปที่ด้านรับดังที่ได้อธิบายในหัวข้อ 2.1.3

3. ส่วน Debugging ใช้การหาข้อบกพร่องของการทำงานของโปรโตคอลได้ เช่นอาจจะสั่งให้ไฟ LED กระพิบตามเงื่อนไขที่กำหนด หรือสามารถพิมพ์ค่าใดๆที่สนใจออกมาดูได้โดยใช้คลังโปรแกรม Printf

4. ส่วน Sampling Application ส่วนนี้จะทำงานเสมือนว่าเกิดเหตุการณ์ใดๆที่สนใจขึ้นแล้วให้โหนดต่างๆทำการประทับเวลาครอบคลุม เพื่อวัดคุณสมบัติความผิดพลาดในการประสานเวลา สำหรับระบบที่ทำการทดลองได้ใช้ RadioCountToLeds ซึ่งเป็นแอปพลิเคชั่นที่มีอยู่แล้วใน TinyOS ซึ่งจะทำการบรอดคาสข้อความตามคาบเวลาที่กำหนดได้และเพิ่มหมายเลขลำดับของข้อความไปเรื่อยๆ

บทที่ 4

ผลการทดลองและวิเคราะห์ผลการทดลอง

การทดลองใช้โปรแกรมภาษา nesC และใช้ระบบปฏิบัติการ TinyOS ทำการคอมไพล์ติดตั้งลงบนอุปกรณ์เพลตฟอร์มโมตรุ่น Telosb ดังรูป

ภาพที่ 4.1 อุปกรณ์ตัวรับรู้แบบไร้สายรุ่น Telosb

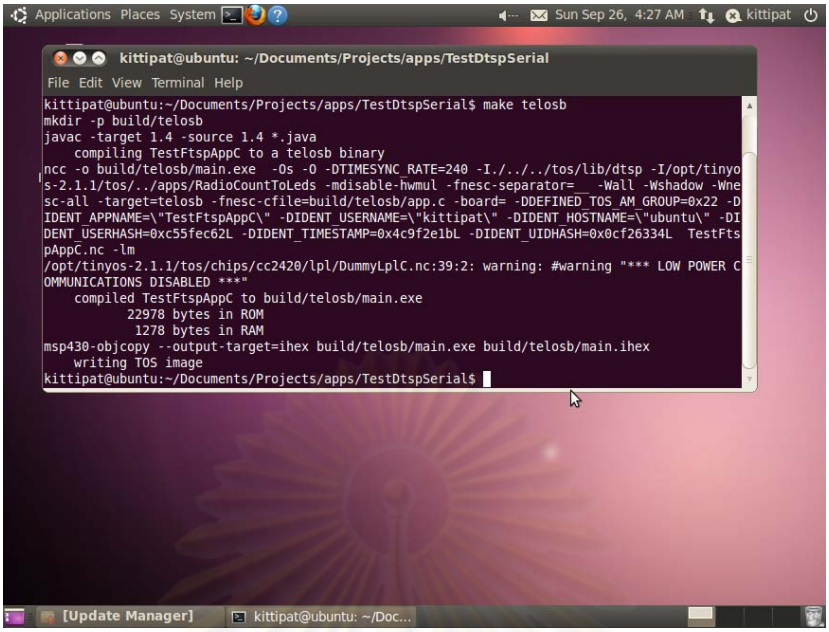
โดยอุปกรณ์สามารถใช้ถ่านขนาด AA สองก้อน หรือใช้พลังงานจากพอร์ตยูเอสบีดังรูป

ภาพที่ 4.2 อุปกรณ์ตัวรับรู้แบบไร้สายรุ่น Telosb บนพอร์ตยูเอสบี

และเมื่อต้องการต่ออุปกรณ์หลายๆตัวโดยใช้พอร์ตยูเอสบี สามารถใช้ยูเอสบีฮับดังรูป

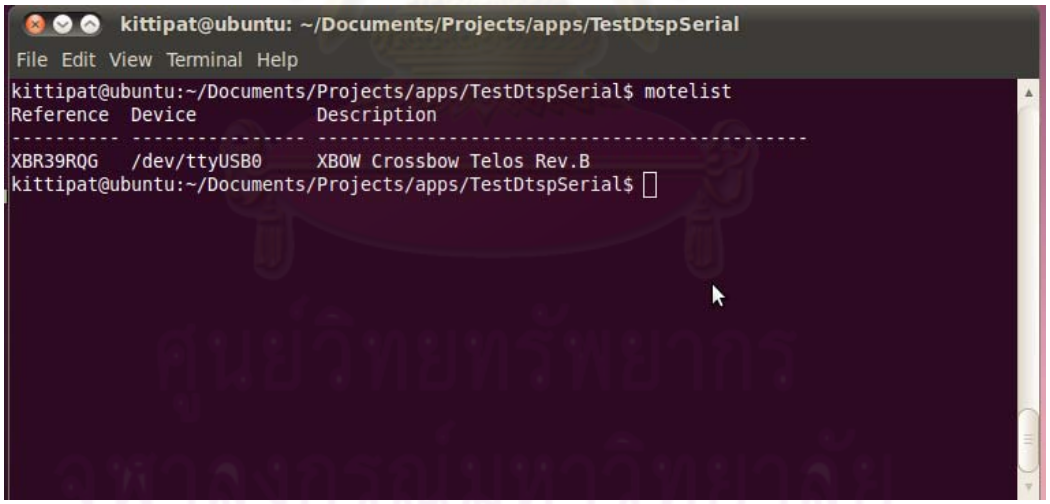
ภาพที่ 4.3 อุปกรณ์ตัวรับรู้แบบไร้สายรุ่น Telosb หลายตัวบนยูเอสบีฮับ

โดยการทดลองนั้นใช้คอมพิวเตอร์ซึ่งทำการลงระบบปฏิบัติการ Ubuntu และชุดโปรแกรมพัฒนาระบบปฏิบัติการ TinyOS เรียบร้อย และเข้าไปยังโฟลเดอร์ ที่เก็บไฟล์โปรแกรมไว้แล้วทำการคอมไพล์โดยใช้คำสั่ง make telosb เพื่อระบุเพลตฟอร์มที่เราต้องการติดตั้งในที่นี้คือ Telosb ดังรูป



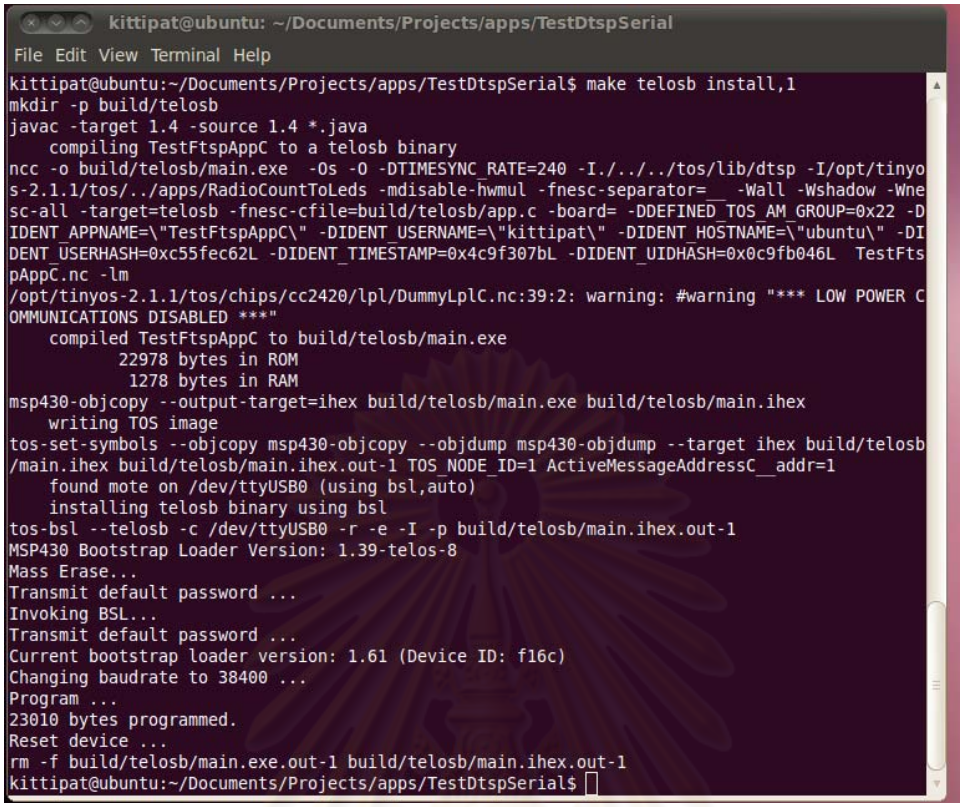
ภาพที่ 4.4 แสดงการคอมไพล์โปรแกรม

และเมื่อต้องการติดตั้งโปรแกรมลงไปในอุปกรณ์จริง ให้เสียบอุปกรณ์ที่พอร์ตยูเอสบี และเรียกคำสั่ง motelist เพื่อแสดงอุปกรณ์ที่ระบบมองเห็นรวมถึงพอร์ตที่เชื่อมต่ออยู่ดังรูป



ภาพที่ 4.5 แสดงการเรียกดูโหนดที่เชื่อมต่ออยู่

และเมื่อต้องการติดตั้งโปรแกรมลงไปในอุปกรณ์ดังกล่าวก็ให้ใช้คำสั่ง make telosb install,1 โดยเลข 1 คือหมายเลขของโหนด ดังรูป



ภาพที่ 4.6 แสดงการติดตั้งโปรแกรม

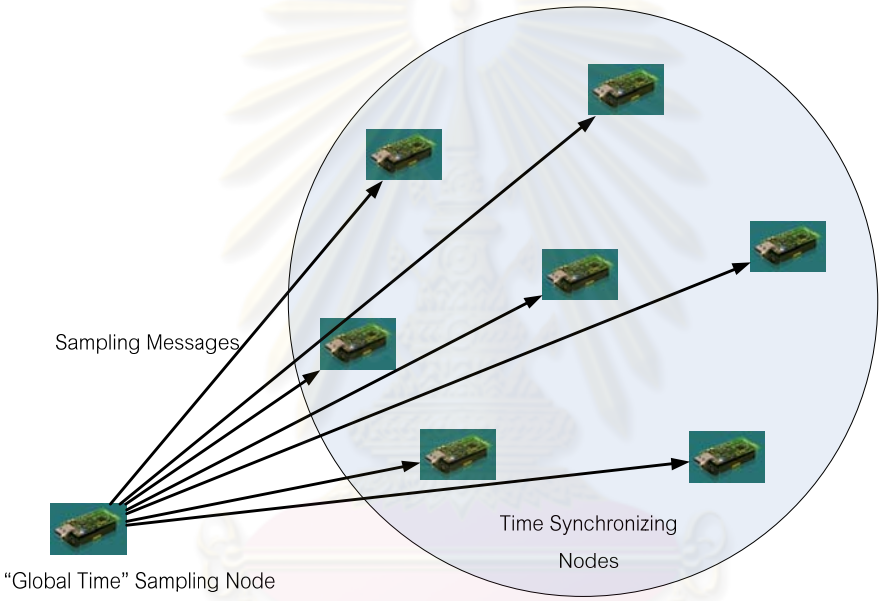
โดยเราสามารถระบุค่าต่างๆใน Makefile ได้ดังนี้

- COMPONENT=TestFtspAppC ระบุโปรแกรมที่ต้องการคอมไพล์และติดตั้ง
- PFLAGS += -DTIMESYNC_RATE=30 กำหนดค่าคาบประสานเวลาเริ่มต้น
- PFLAGS += -DTIMESYNC_DEBUG กำหนดให้การทำงานเป็นแบบฮอปเดียวหรือหลายฮอป
- PFLAGS += -I./../../tos/lib/dtsp เพื่อระบบโปรแกรมที่ต้องการเรียกใช้ ในที่นี้คือโปรโตคอลประสานเวลาที่ต้องการทดสอบ
- CFLAGS += -DCC2420_DEF_RFPOWER=5 ใช้ในการระบุพลังงานที่ใช้ในการส่งข้อความ
- CFLAGS += -DCC2420_DEF_CHANNEL=22 ใช้ในการระบุช่องสัญญาณไร้สายที่ต้องการใช้ ค่าปกติคือ 23

- CFLAGS += -I$(TOSDIR)/lib/printf ระบุเมื่อต้องการใช้คลังโปรแกรม Printf เมื่อต้องการให้โหนดแสดงข้อความหรือค่าต่างๆที่คอมพิวเตอร์

- CFLAGS += -DPRINTF_BUFFER_SIZE=2024 ระบุขนาดหน่วยความจำที่ต้องการใช้ในการส่งข้อความ Printf

โดยที่ในการทดลองได้ใช้โหนดทั้งหมด 8 โหนด โดยมี 7 โหนดทำการประสานเวลากัน และอีกหนึ่งโหนดทำหน้าที่บรอดคาสข้อความเก็บตัวอย่างเวลาครอบคลุมเพื่อเปรียบเทียบกันและคำนวณความผิดพลาดในการประสานเวลา ดังรูป



ภาพที่ 4.7 แสดงการเก็บตัวอย่างเวลาครอบคลุม

จากรูป โหนดเก็บตัวอย่างจะบรอดคาสข้อความเก็บตัวอย่างทุกๆคาบการเก็บตัวอย่าง เพื่อให้โหนดที่กำลังประสานเวลากันอยู่ทำการประทับเวลาครอบคลุม ซึ่งข้อความเก็บตัวอย่างนี้จะทำหน้าที่เป็นตัวแทนของเวลากายภาพ (Physical Time) เดียวกัน และเก็บค่าเวลาในหน่วยความจำเฟรช (Flash Memory) ซึ่งผู้วิจัยสามารถเก็บค่าเหล่านี้จากทุกโหนดมาทำการวิเคราะห์ข้อมูล

4.1 ผลการทดลองและการเปรียบเทียบ

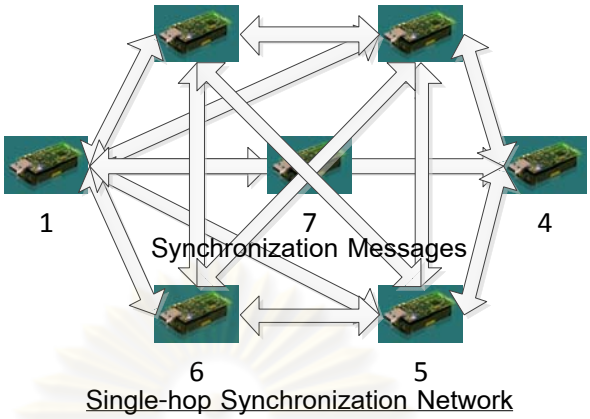
ผู้วิจัยได้ทำการทดลองและเปรียบเทียบประสิทธิภาพการทำงานของโปรโตคอลสองตัว คือ GTSP [6] และ DTSP และใช้ตัวชี้วัด (Metrics) ดังนี้

1. การมีคุณสมบัติเกรเดียน (Gradient Orientation)
2. ความผิดพลาดในการประสานเวลา (Time Synchronization Errors)
3. การใช้พลังงาน (Energy Consumption)

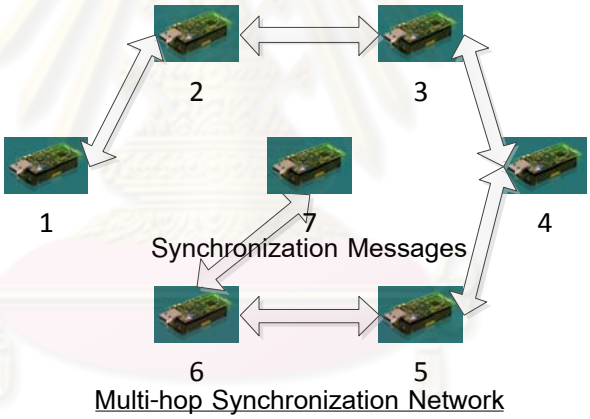
การทดลองได้แบ่งออกเป็นสองประเภทคือการประสานเวลาหนึ่งฮอป (Single-hop synchronization) และการประสานเวลาหลายฮอป (Multi-hop synchronization) ในการประสานเวลาหนึ่งฮอปนั้นโหนดทุกโหนดสามารถรับส่งข้อความประสานเวลากันได้ทั้งหมด ทำให้ความสัมพันธ์เชิงเวลานั้นเป็นแบบฮอปเดียว การทดลองการประสานเวลาแบบหนึ่งฮอปมีวัตถุเพื่อวัดการทำงานของการประสานเวลาในเบื้องต้น ว่าสามารถทำงานได้ตามที่ออกแบบไว้ และสามารถคงค่าความผิดพลาดในการประสานเวลาอยู่ในระดับที่ต่ำไว้ได้

ในขณะที่ในการประสานเวลาหลายฮอป จะใช้ซอฟแวร์บังคับให้โหนดที่มีหมายเลขติดกันสามารถรับส่งข้อความประสานเวลาได้และโหนดที่อยู่ถัดออกไปไม่สามารถคุยกันได้ (ข้อความถูกโยนทิ้ง) ทำให้ความสัมพันธ์เชิงเวลาของเครือข่ายเป็นแบบหลายฮอป การประสานเวลาหลายฮอปมีวัตุประสงค์เพื่อวัดประสิทธิภาพในแง่ของความผิดพลาดในการประสานเวลาที่จะสะสมเพิ่มขึ้นเมื่อประสานเวลาต่อกันหลายๆฮอป

รูปด้านล่างแสดงการรับส่งข้อความประสานเวลาสำหรับการทดลองทั้งสองแบบ



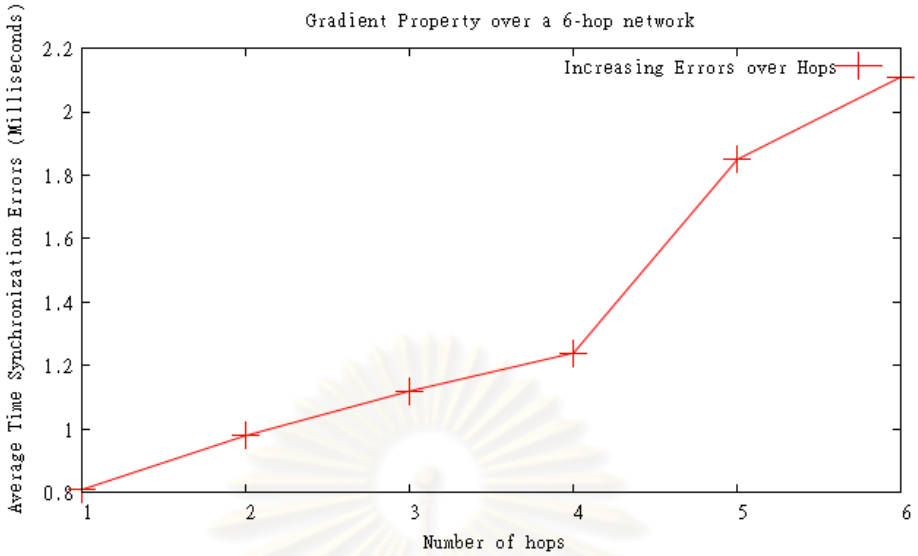
ภาพที่ 4.8 แสดงการทดลองเครือข่ายการประสานเวลาแบบฮอปเดียว



ภาพที่ 4.9 แสดงการทดลองเครือข่ายการประสานเวลาแบบหลายฮอป

4.1.1 การมีคุณสมบัติเกรเดียน (Gradient Orientation)

จากการทดลองเครือข่ายแบบหลายฮอปดังภาพที่ 4.9 แล้วนำค่าตัวอย่างมาทำการหาค่าความผิดพลาดในการประสานเวลาโดยนำค่าเวลาครอบคลุมของโหนดที่อยู่ห่างกัน 1 ถึง 6 ฮอป มาเปรียบเทียบกัน



ภาพที่ 4.10 แสดงความสัมพันธ์ระหว่างจำนวนฮอปกับความผิดพลาดในการประสานเวลา

จากภาพจะเห็นว่าเมื่อจำนวนฮอปเพิ่มมากขึ้นความผิดพลาดในการประสานเวลาก็เพิ่มมากขึ้นตามไปด้วย ซึ่งเป็นความสัมพันธ์ปกติของการประสานเวลา แต่เมื่อพิจารณาเครือข่ายที่มีเส้นผ่าศูนย์กลางเท่ากับ D ฮอป สำหรับโปรโตคอลแบบมีโครงสร้าง ความผิดพลาดในการประสานเวลาจะมีค่าสูงสุด เท่ากับความผิดพลาดในการประสานเวลา 2*D ฮอป แต่สำหรับโปรโตคอลแบบกระจาย ความผิดพลาดในการประสานเวลาจะมีค่าสูงสุด เท่ากับความผิดพลาดในการประสานเวลา D ฮอป

สมมุติว่าเครือข่ายมีขนาดสองฮอปจะมีความผิดพลาดในการประสานเวลาสำหรับโปรโตคอลแบบโครงสร้างต้นไม้สูงสุดประมาณ 1.2 มิลลิวินาที ในขณะที่ความผิดพลาดในการประสานเวลาสำหรับโปรโตคอลแบบกระจายสูงสุดประมาณ 1 มิลลิวินาที และสำหรับเครือข่ายมีขนาดสามฮอปจะมีความผิดพลาดในการประสานเวลาสำหรับโครงสร้างต้นไม้สูงสุดประมาณ 2.1 มิลลิวินาที ในขณะที่ความผิดพลาดในการประสานเวลาสำหรับโปรโตคอลแบบกระจายสูงสุดประมาณ 1.2 มิลลิวินาที ซึ่งจะเห็นได้ว่าค่าความแตกต่างระหว่างค่าความผิดพลาดสูงสุดนั้นจะเพิ่มขึ้นเมื่อขนาดเครือข่ายใหญ่ขึ้น ซึ่งทำให้เห็นถึงข้อดีของคุณสมบัติเกรเดียนของการประสานเวลา

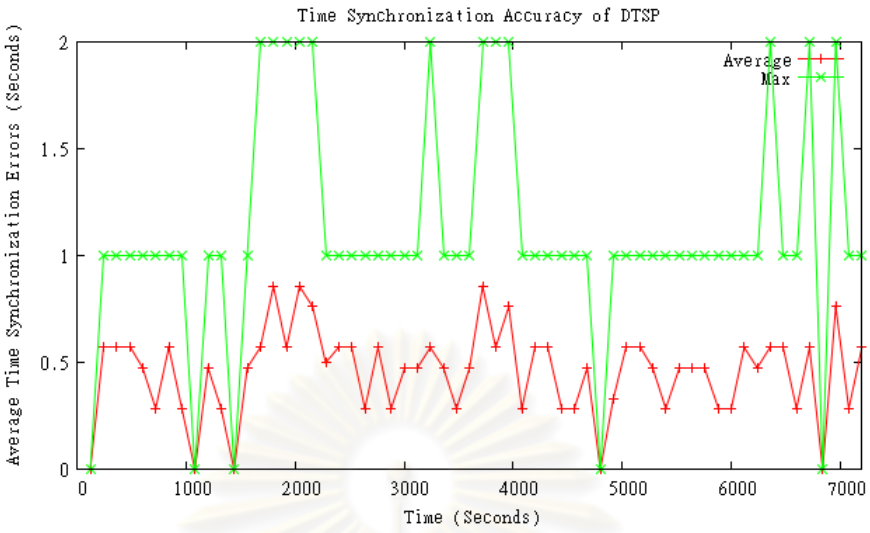
4.1.2 ความผิดพลาดในการประสานเวลา (Time Synchronization Errors)

ความผิดพลาดในการประสานเวลาเป็นตัวชี้วัดความแม่นยำในการประสานเวลาของโปรโตคอล เพราะเมื่อเกิดเหตุการณ์ใดๆที่เวลาหนึ่งๆขึ้น ระดับชั้นแอปพลิเคชั่นจะทำการเรียกค่าเวลาครอบคลุมจากโปรโตคอล ซึ่งเหตุการณ์เดียวกันก็ควรจะได้เวลาครอบคลุมเท่ากัน ดังนั้นการวัดประสิทธิภาพด้วยความผิดพลาดในการประสานเวลานั้นจะต้องสร้างเหตุการณ์สมมุติขึ้นมาเพื่อให้โหนดต่างๆทำการประทับเวลาครอบคลุม ในการทดลองได้ใช้การบรอดคาสข้อความเป็นเหตุการณ์สมมุติเนื่องจากข้อความแบบไร้สายนั้นมีความเร็วสูงมาก ทำให้โหนดต่างๆได้รับข้อความพร้อมๆกัน เมื่อโหนดทำการประทับเวลาแล้วก็จะเก็บค่าลงใน Flash Memory เพื่อถูกดึงค่าออกมาเมื่อเสร็จสิ้นการทดลอง ความผิดพลาดในการประสานเวลาของโปรโตคอลประสานเวลาที่ดีนั้นต้องมีค่าน้อยๆ โดยถ้าค่าความผิดพลาดเป็นศูนย์จะเรียกว่า การประสานเวลาสมบูรณ์แบบ (Perfect Time Synchronization)

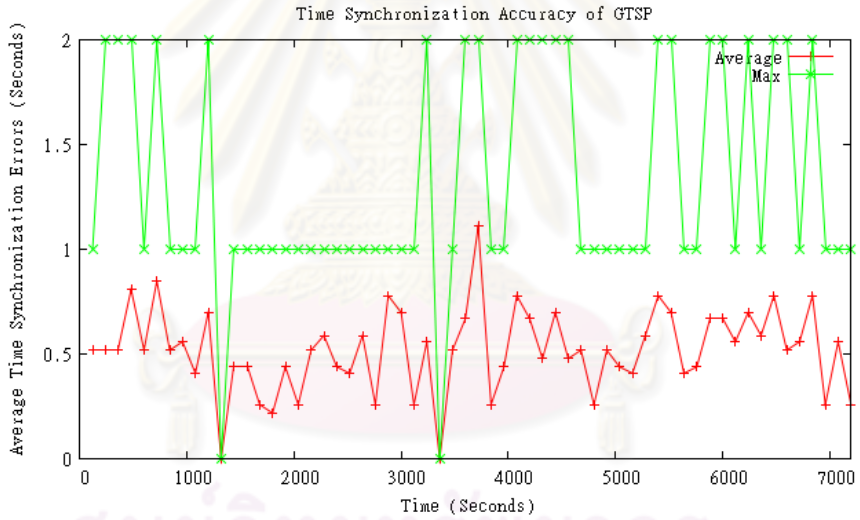
จากการทดลองโดยวัดความผิดพลาดในการประสานเวลาของโปรโตคอล DTSP เทียบกับ GTSP บนเครือข่ายหนึ่งฮอปดังภาพที่ 4.8 โดยตั้งค่าการทดลองดังนี้

ตัวแปรสำหรับการทดลอง	จำนวน / ขนาด
จำนวนโหนด	7
คาบการประสานเวลา	30 วินาที
คาบเวลาการเก็บตัวอย่างเวลาครอบคลุม	120 วินาที
คาบการประมาณค่าเวลาบิดเบือน	30 วินาที (GTSP) ปรับตัวได้ (DTSP)
เวลาของการทดลอง (Experimental Period)	7200 วินาที

ตารางที่ 4.1 แสดงการตั้งค่าการทดลองความแม่นยำในการประสานเวลา



ภาพที่ 4.11 แสดงความแม่นยำในการประสานเวลาของโปรโตคอล DTSP



ภาพที่ 4.12 แสดงความแม่นยำในการประสานเวลาของโปรโตคอล GTSP

โปรโตคอล	ความผิดพลาดในการประสานเวลา		
	ค่าเฉลี่ย	ค่าเบี่ยงเบนมาตราฐาน	ค่าสูงสุดเฉลี่ย
DTSP	0.45	0.20	1.12
GTSP	0.52	0.20	1.31

ตารางที่ 4.2 แสดงการเปรียบเทียบความแม่นยำในการประสานเวลา

จากตารางที่ 4.2 โปรโตคอล DTSP มีค่าความผิดพลาดในการประสานเวลาเฉลี่ยและสูงสุดเฉลี่ยน้อยกว่า GTSP ประมาณ 15% เนื่องจาก DTSP มีอัตราการปรับค่าเวลาเหลื่อมที่เร็วกว่าและมีการประมาณค่าเวลาบิดเบือนที่แม่นยำกว่า

คาบการประสานเวลา (วินาที)	ความผิดพลาดในการประสานเวลา		
	ค่าเฉลี่ย	ค่าเบี่ยงเบนมาตราฐาน	ค่าสูงสุดเฉลี่ย
30	0.45	0.20	1.12
120	0.55	0.22	1.31
240	0.60	0.29	1.40

ตารางที่ 4.3 แสดงการเปรียบเทียบความผิดพลาดในการประสานเวลาของ DTSP ที่คาบการประสานเวลาต่างๆ

จากตาราง ค่าคาบการประสานเวลาที่ขยายจาก 30 วินาที เป็น 120 วินาทีและ 240 วินาที หรือ สี่เท่าและแปดเท่าตามลำดับ ซึ่งทำให้ความผิดพลาดในการประสานเวลาเพิ่มขึ้น 22% และ 33% ตามลำดับนั้น แสดงให้เห็นถึง Trade-off ของระบบซึ่งเมื่อขยายคาบการประสานเวลาซึ่งสามารถทำให้ลดจำนวนข้อความในระบบได้ ก็ทำให้ความแม่นยำในการประสานเวลาเพิ่มขึ้นเพียงเล็กน้อย

4.1.3 การใช้พลังงาน (Energy Consumption)

เนื่องจากพลังงานนั้นมีอยู่อย่างจำกัดในเครือข่ายตัวรับรู้แบบไร้สาย โปรโตคอลต่างๆที่สนับสนุนการทำงานของเครือข่ายนั้นจำเป็นต้องใช้พลังงานอย่างมีประสิทธิภาพ การใช้

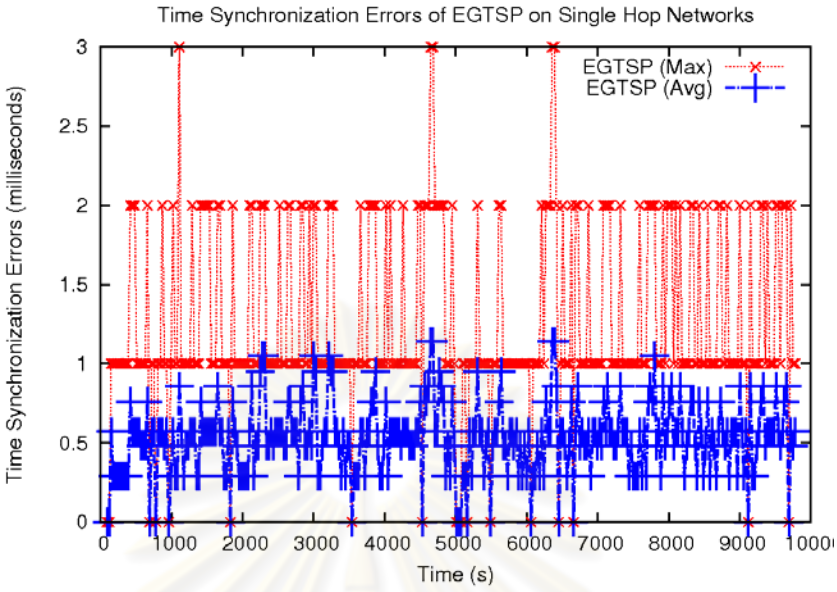
พลังงานของโปรโตคอลประสานเวลาในการทดลองนี้ ใช้การวัดจำนวนข้อความประสานเวลาที่ถูกส่งออกมาจากแต่ละโหนดรวมกัน ซึ่งโปรโตคอลที่ทำการเปรียบเทียบคือ EGTSP ซึ่งคือโปรโตคอล DTSP ที่วัดผลด้านการประหยัดพลังงาน กับ GTSP ใช้ข้อความประสานเวลาขนาดเท่ากันคือ 140 บิต

และผลการทดลองในส่วนของพลังงานนั้นมาจากงาน EGTSP [20] ที่ได้รับการตีพิมพ์ในรายงานการประชุมระดับนานาชาติและเป็นส่วนหนึ่งของวิทยานิพนธ์ ซึ่งใช้ค่าในการทดลองดังนี้

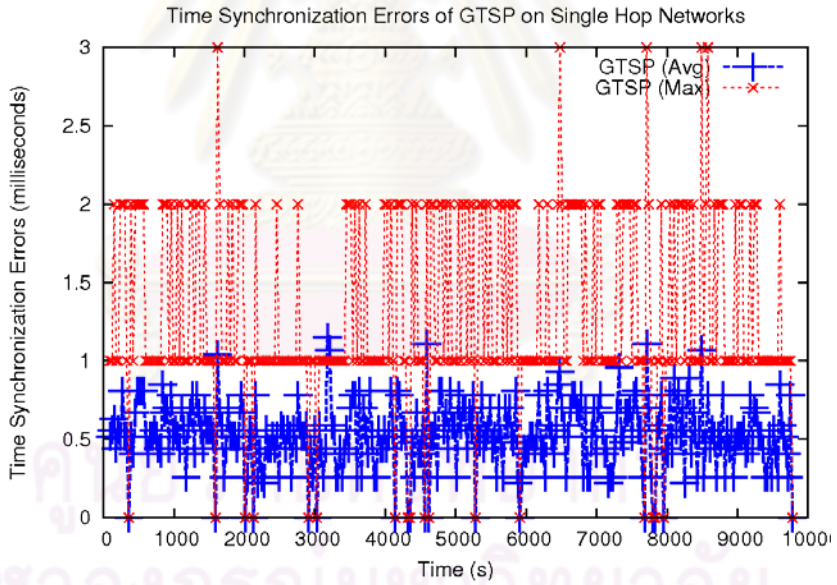
ตัวแปรสำหรับการทดลอง	จำนวน / ขนาด
จำนวนโหนด	7
คาบการประสานเวลาเริ่มต้น (Initial Synchronization Period)	30 วินาที
คาบการประสานเวลาที่ถูกขยาย (Extended Synchronization Period)	200 วินาที (EGTSP)
คาบเวลาการเก็บตัวอย่างเวลาครอบคลุม (Global Time Sampling Period)	30 วินาที
คาบเวลาการประมาณค่าเวลาบิดเบือน (Skew Estimation Period)	30 วินาที (GTSP) 300 วินาที (EGTSP)
เวลาของการทดลอง (Experimental Period)	10000 วินาที

ตารางที่ 4.4 แสดงการตั้งค่าการทดลองเพื่อวัดการใช้พลังงาน

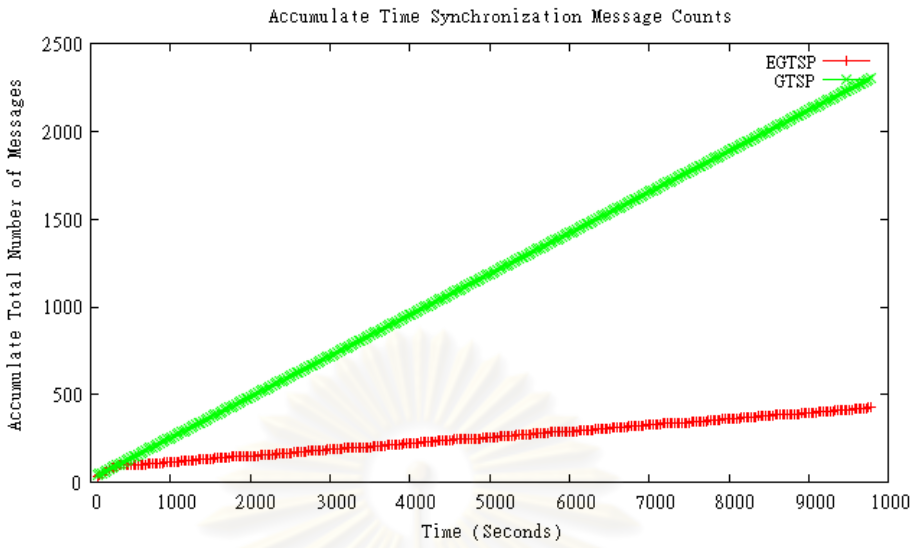
จากการทดลองบนเครือข่ายแบบหนึ่งฮอปดังภาพที่ 4.8 สามารถแสดงกราฟความผิดพลาดในการประสานเวลาเฉลี่ยเทียบกับเวลาของการทดลอง ของ EGTSP และ GTSP และกราฟแสดงจำนวนข้อความประสานเวลาทั้งหมดสะสม ของทั้ง EGTSP และ GTSP ดังนี้



ภาพที่ 4.13 กราฟแสดงความผิดพลาดเฉลี่ยในการประสานเวลาของ EGTSP



ภาพที่ 4.14 กราฟแสดงความผิดพลาดเฉลี่ยในการประสานเวลาของ GTSP



ภาพที่ 4.15 แสดงจำนวนข้อความทั้งหมดสะสมเทียบกับเวลา

จากภาพที่ 4.13 และ 4.14 ความแม่นยำในการประสานเวลาของโปรโตคอลทั้งสองทำงานได้พอๆกัน มีความผิดพลาดในการประสานเวลาสูงสุดที่ 3 มิลลิวินาที และระดับของความผิดพลาดในการประสานเวลาเฉลี่ยอยู่ในระดับเดียวกันตลอดการทดลอง และเมื่อเฉลี่ยทุกตัวอย่างแล้วจะได้ค่าดังตารางที่ 4.5

และเมื่อพิจารณาภาพที่ 4.15 ซึ่งจำนวนข้อความทั้งหมดสะสมเทียบกับเวลาของโปรโตคอล EGTSP และ GTSP จะเห็นว่า เมื่อเริ่มต้น จำนวนข้อความการประสานเวลามีจำนวนเท่าๆกัน แต่เมื่อเวลาผ่านไป โปรโตคอล EGTSP สามารถขยายคาบการประสานเวลาได้ทำให้จำนวนข้อความลดลงอย่างมาก โดยที่ไม่ทำให้ความแม่นยำในการประสานเวลาลดลง เนื่องจากโปรโตคอล EGTSP มีการชดเชยเวลาเหลื่อมที่ถี่กว่าเป็นจำนวนเท่าของโหนดเพื่อนบ้าน ทำให้มีใช้พลังงานในการคำนวนบ่อยกว่า แต่ว่าพลังงานที่ใช้ไปในการส่งข้อความแบบไร้สายหนึ่งบิตนั้นสามารถนำไปใช้คำนวนหนึ่งได้ถึง 1,000 บรรทัด [1] และสมมุติว่า GTSP ต้องการขยายคาบการประสานเวลาเพื่อประหยัดพลังงานในการส่งข้อความประสานเวลาเช่นเดียวกัน ก็จะทำให้การชดเชยค่าเวลาเหลื่อมและการลู่เข้าของเวลาครอบคลุมทำได้ช้าลงตามลำดับ ทำให้ความแม่นยำในการประสานเวลาลดลงตามไปด้วย

โปรโตคอล	ความผิดพลาดในการประสานเวลา (มิลลิวินาที)		จำนวนข้อความประสานเวลาทั้งหมด
	ค่าเฉลี่ย	ค่าเบี่ยงเบนมาตารฐาน	
GTSP	0.52	0.22	2303
EGTSP	0.52	0.23	427

ตารางที่ 4.5 เปรียบเทียบประสิทธิภาพของ GTSP และ EGTSP แบบฮอปเดียว

เมื่อพิจารณาตารางที่ 4.5 และสมมุติให้พลังงานที่ใช้ในการส่งข้อความหนึ่งข้อความ (ที่มีขนาดเท่ากัน) ใช้พลังงานเท่ากัน และพลังงานที่ใช้ในการประมวลผลมีค่าน้อยมากเมื่อเทียบกับพลังงานที่ใช้ในการส่งข้อความแบบไร้สาย EGTSP สามารถประหยัดพลังงานได้มากกว่าห้าเท่า โดยไม่สูญเสียความแม่นยำในการประสานเวลา

4.2 **สรุปผลการทดลอง**

จากผลการทดลองสามารถสรุปผลการทดลองได้ดังนี้

1. โปรโตคอลการประสานเวลาในวิทยานิพนธ์ฉบับนี้มีคุณสมบัติเกรเดียนบนระนาบของเครือข่ายโดยมีความผิดพลาดในการประสานเวลาเฉลี่ยของหนึ่งฮอปอยู่ที่ 0.8 มิลลิวินาที และหกฮอปอยู่ที่ 2.11 มิลลิวินาที ซึ่งเมื่อหารต่อฮอปแล้วเท่ากับ 0.35 มิลลิวินาที ซึ่งเป็นค่าเฉลี่ยที่น้อยกว่าหน่วยย่อยที่สุดของเวลาคือ 1 มิลลิวินาที

2. โปรโตคอลการประสานเวลาในวิทยานิพนธ์ฉบับนี้เมื่อเปรียบเทียบกับงาน GTSP แล้วมีความแม่นยำมากกว่าประมาณ 15% เนื่องจากมีอัตราการชดเชยเวลาเหลื่อมที่มากกว่าและมีประสิทธิภาพในการประมาณค่าเวลาเหลื่อมที่ดีกว่า และเมื่อขยายคาบการประสานเวลาเป็น 4 เท่า และ 8 เท่า ความแม่นยำในการประสานเวลาลดลง 22% และ 33% ตามลำดับ

3. โปรโตคอลการประสานเวลาในวิทยานิพนธ์ฉบับนี้เมื่อเทียบกับ GTSP ใช้พลังงานในการส่งข้อความประสานเวลาต่ำกว่า 5 เท่า เมื่อสมมุติให้พลังงานที่ใช้ในการส่งข้อความหนึ่งข้อความ (ที่มีขนาดเท่ากัน) ใช้พลังงานเท่ากัน และพลังงานที่ใช้ในการประมวลผลมีค่าน้อยมากเมื่อเทียบกับพลังงานที่ใช้ในการส่งข้อความแบบไร้สาย [1] เนื่องจากโปรโตคอล

สามารถขยายคาบการประสานเวลาแต่มีอัตราการคำนวนที่ถี่กว่า GTSP จึงทำให้โปรโตคอลไม่ทำให้สูญเสียความแม่นยำ

บทที่ 5
สรุปผลการวิจัย ข้อจำกัด และข้อเสนอแนะ

5.1 สรุปผลการวิจัย

งานวิจัยฉบับนี้ได้เสนอ การประสานเวลาแบบกระจายสำหรับเครือข่ายตัวรับรู้แบบไร้สาย โดยเริ่มต้นจากการศึกษางานวิจัยทางด้านการประสานเวลาสำหรับเครือข่ายตัวรับรู้แบบไร้สาย ซึ่งเป็นปัญหามีผลงานทางวิชาการตีพิมพ์ในวารสาร และรายงานการประชุมที่มีชื่อเสียงทั่วโลกอย่างต่อเนื่อง และผู้วิจัยได้ออกแบบโปรโตคอล โดยคำนึงถึงการนำไปใช้งานจริงในเครือข่ายตัวรับรู้แบบไร้สาย และได้นำเอาการออกแบบดังกล่าวไปทำให้เกิดผล รวมทั้งทดลองบนอุปกรณ์ตัวรับรู้เพลตฟอร์มโมตรุ่น Telosb และ ระบบปฏิบัติการ TinyOS

ผู้วิจัยได้นำเสนอการประสานเวลาแบบกระจายโดยโหนดจะประสานเวลาไปที่ค่าเฉลี่ยของโหนดเพื่อนบ้าน และเสนอการเฉลี่ยแบบเพิ่มส่วน ซึ่งสามารถทำให้โหนดปรับการประสานเวลาได้อย่างรวดเร็ว ทันทีที่ได้รับข้อความประสานเวลา และได้นำเสนอการประมาณความบิดเบือนทางเวลา ซึ่งเกิดจากความเบี่ยงเบนทางเวลาของสัญญาณนาฬิกา โดยใช้คาบการประมากว้างและปรับตัวได้ และนำเสนออยู่ในรูปแบบของคาบเวลาบิดเบือน ซึ่งทำให้โหนดสามารถชดเชยเวลาบิดเบือนได้โดยอัตโนมัติ สามารถขยายคาบประสานเวลาออกไปเพื่อประหยัดพลังงาน นอกจากนั้นโปรโตคอลยังสามารถปรับตัวกับความพลวัตของเครือข่ายและขยายตัวได้ภายใต้เงื่อนไขความหนาแน่นของโหนดไม่สูงจนเกินไปและความผิดพลาดในการประสานเวลาที่เกิดจากจำนวนฮอปสูงสุดของเครือข่ายอยู่ในระดับที่ยอมรับได้

โปรโตคอลดังกล่าว ผู้วิจัยได้แสดงให้เห็นและเปรียบเทียบผลการทดลองกับโปรโตคอล GTSP ว่า สามารถใช้พลังงานที่มีอยู่อย่างจำกัดได้มีประสิทธิภาพกว่าโดยไม่สูญเสียความแม่นยำในการประสานเวลา และมีคุณสมบัติเกรเดียน เนื่องจากความผิดพลาดในการประสานเวลาระหว่างโหนดเพื่อนบ้านอยู่ในระดับที่ต่ำและสูงขึ้นตามจำนวนฮอปที่อยู่ห่างออกไปและมีความแม่นยำกว่า

5.2 ข้อจำกัด

การทำงานของโปรโตคอลประสานเวลานั้นจะทำงานได้ในเครือข่ายที่มีพลวัตน้อย มีความหนาแน่นของเครือข่ายปานกลาง และมีคาบเวลาในการหลับไม่สูงกว่าคาบการประสานเวลาและสามารถตื่นขึ้นมาพร้อมกันเพื่อรับส่งข้อความ นอกจากนั้นความเบี่ยงเบนทางเวลา

สัมพัทธ์จะเป็นเชิงเส้นในช่วงเวลาสั้นๆเท่านั้น เมื่อพิจารณาเวลาที่กว้างมากๆความเบี่ยงเบนทางเวลาสัมพัทธ์อาจจะไม่เป็นเชิงเส้น ทำให้เกิดความผิดพลาดในการประมาณค่าเวลาบิดเบือนได้

5.3 ข้อเสนอแนะ

การประสานเวลาซึ่งใช้การเฉลี่ยค่านั้น สามารถใช้เทคนิคการเฉลี่ยแบบมีน้ำหนักได้ เช่น ถ้ารู้ตำแหน่งของโหนด เราสามารถให้น้ำหนักของการเฉลี่ยกับโหนดที่อยู่ใกล้มากกว่า ทำให้คุณสมบัติเกรเดียนของการประสานเวลามีความละเอียดมากขึ้น

การประมาณและชดเชยเวลาบิดเบือนนั้นถ้าสามารถทำให้มีความแม่นยำมากขึ้นก็จะสามารถทำให้ประสิทธิภาพของการประสานเวลาเพิ่มขึ้นด้วย เช่นการประมาณค่าแบบเพิ่มส่วน (Incremental estimation) และการตรวจจับความไม่เชิงเส้นของความเบี่ยงเบนทางเวลา

เนื่องจากการประสานเวลาเป็นส่วนที่สำคัญสำหรับการทำงานที่ถูกต้องของเครือข่ายตัวรับรู้แบบไร้สาย การโจมตีการประสานเวลาของเครือข่ายจึงเป็นที่สนใจของวงการวิจัยในขณะนี้ โดยอาจจะอยู่ในรูปของการสุ่มค่าเวลาปลอมและส่งเข้าไปในเครือข่าย ทำให้การประสานเวลาไม่สำเร็จหรือไม่ลู่เข้า (Non-convergence) นั่นเอง โดยอาจจะอาศัยการตรวจจับโหนดประเภทนี้และทำการตัดออกจากเครือข่ายประสานเวลา (Synchronizing Networks)

การสร้างแบบจำลองของสัญญาณนาฬิกาที่มีความใกล้เคียงกับพฤติกรรมของสัญญาณนาฬิกาบนอุปกรณ์จริง ก็จะทำให้การทดสอบประสิทธิภาพของการประสานเวลาทำได้บนโปรแกรมจำลอง ทำให้การวิจัยและพัฒนาโปรโตคอลมีความสะดวกมากขึ้น นอกจากนั้น ระบบทดสอบ (Testbed) ทางด้านเครือข่ายตัวรับรู้แบบไร้สายที่เปิดให้บุคคลภายนอกเข้าไปใช้งานได้ ก็จะทำให้การทำให้เกิดผลบนเครือข่ายมีความสมจริงมากขึ้น

รายการอ้างอิง

[1] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y. and Cayirci, E. A survey on sensor networks. IEEE Communications Magazine 40 (2002): 102-114.

[2] Lamport, L. Time, clocks, and the ordering of events in a distributed system. Communication of the ACM 21 (July 1978): 558-565.

[3] Mills, D. L. Internet Time Synchronization: the Network Time Protocol. IEEE Transactions on Communications 39 (October 1991): 1482-1493.

[4] Ganeriwal, S., Kumar, R., and Srivastava, M. B. Timing-sync protocol for sensor networks. In Proceedings of the 1st international Conference on Embedded Networked Sensor Systems (SenSys) (November 2003): 138-149.

[5] Maróti, M., Kusy, B., Simon, G., and Lédeczi, Á. The flooding time synchronization protocol. In Proceedings of the 2nd international Conference on Embedded Networked Sensor Systems (SenSys) (November 2004): 39-49.

[6] Sommer, P. and Wattenhofer, R. Gradient clock synchronization in wireless sensor networks. In Proceedings of the 2009 international Conference on Information Processing in Sensor Networks (April 2009): 37-48.

[7] Coulouris, G. Dollimore, J. and Kindberg, T. Time and Global States. , Distributed Systems: Concepts and Design, pp. 433-464. Addison Wisley/Pearson Education, 2005

[8] Elson, J., Girod, L., and Estrin, D. Fine-grained network time synchronization using reference broadcasts. In Proceedings of the 5th Symposium on Operating Systems Design and Implementation. (December 2002): 147-163.

[9] Sundararaman, B. Buy, U. and others. Clock synchronization for wireless sensor networks: a survey. Ad Hoc Networks Vol. 3 No. 3 (2005): 281-323.

[10] Elson, J. and Römer, K. Wireless sensor networks: a new regime for time synchronization. SIGCOMM Computer Communication Review 33 1 (January 2003): 149-154.

[11] Römer, K. 2001. Time synchronization in ad hoc networks. In Proceedings of the 2nd ACM international Symposium on Mobile Ad Hoc Computing 2 (October 2001): 173-182.

[12] Kusy, B., Dutta, P., Levis, P., Maroti, M., Ledeczi, A., and Culler, D. Elapsed time on arrival: a simple and versatile primitive for canonical time synchronisation services. International Journal on Ad Hoc Ubiquitous Computing 1 4 (July 2006): 239-251.

[13] Levis, P. TinyOS. [Online]. Available from: http://code.google.com/p/tinyos-main/ [2010, September 26]

[14] Maroti M. Packet-level time synchronization. [Online]. Available from: http://www.tinyos.net/tinyos-2.1.0/doc/html/tep133.html [2010, September 26]

[15] Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., and Culler, D. The *nesC* language: A holistic approach to networked embedded systems. In Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (June 2003): 1-11.

[16] Canonical Ltd. Ubuntu. [Online]. Available from: http://www.ubuntu.com/ [2010, September 26]

[17] Polastre, J., Szewczyk, R., and Culler, D. Telos: enabling ultra-low power wireless research. In Proceedings of the 4th international Symposium on information Processing in Sensor Networks (April 2005): 364-369.

[18] Hamilton, B. R., Ma, X., Zhao, Q., and Xu, J. ACES: adaptive clock estimation and synchronization using Kalman filtering. In Proceedings of the 14th ACM international Conference on Mobile Computing and Networking 14 (September 2008): 152-162.

[19] Phillips, L. TinyOS Programming. Cambridge University Press, 2009.

[20] Apicharttrisorn, K., Choochaisri, S., and Intanagonwiwat, C. Energy-Efficient Gradient Time Synchronization for Wireless Sensor Networks. 2nd Int Conference on Computational Intelligence, Communication Systems and Networks (CICSyN2010). (July 2010): 124-129.

[21] Yoon, S., Veerarittiphan, C., and Sichitiu, M. L. Tiny-sync: Tight time synchronization for wireless sensor networks. ACM Transactions on Sensor Networks 3 2 (June 2007).

[22] Su, W. and Akyildiz, I. F. Time-diffusion synchronization protocol for wireless sensor networks. IEEE/ACM Transactions on Network 13 (April 2005): 384-397.

[23] Levis, P., Lee, N., Welsh, M., and Culler, D. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In Proceedings of the 1st international Conference on Embedded Networked Sensor Systems (November 2003): 126-137.

ประวัติผู้เขียนวิทยานิพนธ์

นายกิตติภัทร อภิชาติไตรสรณ์ เกิดเมื่อวันที่ 2 กันยายน พ.ศ. 2525 ที่จังหวัดสมุทรปราการ สำเร็จการศึกษาระดับประถมศึกษาจากโรงเรียนอัสสัมชัญ สำโรง จังหวัดสมุทรปราการ สำเร็จการศึกษาระดับมัธยมศึกษา จากโรงเรียนหอวัง จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ ในปีการศึกษา 2547 และเข้าศึกษาในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2550

ผู้เขียนวิทยานิพนธ์ได้ตีพิมพ์ผลงานทางวิชาการในรายงานการประชุม “2010 Second International Conference on Computational Intelligence, Communication Systems and Networks” ชื่อว่า “Energy Efficient Gradient Time Synchronization for Wireless Sensor Networks” ณ เมืองลิเวอร์พูล ประเทศอังกฤษ เมื่อเดือนกรกฎาคม พ.ศ. 2553