



การพัฒนาโพรโทคอลค้นหาเส้นทางสำหรับเครือข่ายเซนเซอร์ไร้สาย
The Routing Protocol Development for Wireless Sensor Networks

ธัญกรณ์ พัฒนไตรวัฒน์

Thanankorn Phatthanatraiwat

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยสงขลานครินทร์

**A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Computer Engineering
Prince of Songkla University**

2554

ลิขสิทธิ์ของมหาวิทยาลัยสงขลานครินทร์

ชื่อวิทยานิพนธ์ การพัฒนาโปรโตคอลค้นหาเส้นทางสำหรับเครือข่ายเซนเซอร์ไร้สาย

ผู้เขียน นายธนัญกรณ์ พัฒนไทรวัฒน์

สาขาวิชา วิศวกรรมคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

คณะกรรมการสอบ

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.วรรณรัช สันติอมรทัต) (ดร.สกุณา เจริญปัญญาศักดิ์)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ศักดิ์ชัย ทิพย์จักษ์รัตน)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.วรรณรัช สันติอมรทัต)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้บัณฑิตวิทยาลัยนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

(ศาสตราจารย์ ดร.อมรรัตน์ พงศ์ดารา)

คณบดีบัณฑิตวิทยาลัย

ชื่อวิทยานิพนธ์ การพัฒนาโพรโทคอลค้นหาเส้นทางสำหรับเครือข่ายเซนเซอร์ไร้สาย
ผู้เขียน นายธนัญกรณ์ พัฒนไทรวัฒน์
สาขาวิชา วิศวกรรมคอมพิวเตอร์
ปีการศึกษา 2553

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้นำเสนอการออกแบบและพัฒนาโพรโทคอลค้นหาเส้นทางสำหรับใช้งานในเครือข่ายเซนเซอร์ไร้สาย เนื่องจากการทำงานในเครือข่ายเซนเซอร์ไร้สายมีลักษณะที่ใกล้เคียงกับการทำงานของเครือข่ายแบบ ad hoc ดังนั้นจึงทำการประยุกต์หลักการของโพรโทคอลค้นหาเส้นทางในเครือข่าย ad hoc มาใช้กับโพรโทคอลบนเครือข่ายเซนเซอร์ไร้สายที่ได้ออกแบบและพัฒนาขึ้นในงานวิจัยนี้ ซึ่งจะทำการทดสอบโพรโทคอลบนโหนดจริงที่ชื่อ Unode เพื่อนำเสนอประสิทธิภาพของโพรโทคอลในด้านอัตราการรับส่งข้อมูลหรือเรียกว่า Packet Delivery Ratio (PDR)

โพรโทคอลค้นหาเส้นทางที่ออกแบบและพัฒนาขึ้นมานั้น มีความสามารถในการค้นหาเส้นทาง รับส่งข้อมูล และสามารถบำรุงรักษาเส้นทางเมื่อเกิดเส้นทางเสียหายในเครือข่าย โดยโพรโทคอลมีการทำงานที่ไม่ซับซ้อน เนื่องจากเครือข่ายเซนเซอร์ไร้สายมีข้อจำกัดในเรื่องของพลังงาน หน่วยความจำ และความสามารถของหน่วยประมวลผล

จากผลการทดสอบพบว่าค่าอัตราการรับส่งข้อมูลของโพรโทคอลที่พัฒนามีค่าสูงถึง 95.2 เปอร์เซ็นต์ แต่เมื่อจำนวนการเชื่อมต่อเพิ่มสูงขึ้น ค่าอัตราการรับส่งข้อมูลจะลดลง ดังนั้นจึงได้ทำการปรับปรุงโพรโทคอลโดยใช้การทำ scheduling ทำให้มีค่าอัตราการรับส่งข้อมูลเพิ่มสูงจากเดิมประมาณ 18 เปอร์เซ็นต์

คำสำคัญ: โพรโทคอล, เครือข่ายเซนเซอร์ไร้สาย, Unode, TinyOS

Thesis Title The Routing Protocol Development for Wireless Sensor Networks
Author Mr. Thanankorn Phatthanatraiwat
Major Program Computer Engineering
Academic Year 2010

ABSTRACT

In this thesis, we present the design and development of protocol for wireless sensor networks. The routing protocol in wireless sensor network is similar routing protocol in Ad hoc networks. We apply concept the routing protocol in Ad hoc networks to our routing protocol. Unode has been used to test our protocol and show the performance in term of Packet Delivery Ratio (PDR).

The our routing protocol have ability to routing, collection data from sensor and maintenance of path when path loss in network. The operation of our routing protocol is simple because wireless sensor networks have limitations in term of energy, memory and ability of processor.

The experimental results show that our protocol can give the highest PDR at 95.2%. However, the PDR has been decreased when the number of connection is increasing. Thus, we improved our protocol using a scheduling algorithm. We found that the PDR improvement is about 18%.

KEYWORDS: WIRELESS SENSOR NETWORK, ROUTING PROTOCOL, UNODE, TINYOS

สารบัญ

	หน้า
บทคัดย่อ.....	(3)
ABSTRACT.....	(4)
สารบัญ.....	(6)
รายการตาราง.....	(9)
รายการภาพประกอบ.....	(10)
บทที่ 1 บทนำ.....	1
1.1 ที่มาและความสำคัญของวิทยานิพนธ์.....	1
1.2 ตรวจสอบเอกสาร.....	2
1.2.1 โพรโทคอล Ad hoc On-demand Distance Vectoring (AODV).....	3
1.2.2 การพัฒนาโพรโทคอล AODV สำหรับเครือข่ายเซนเซอร์ไร้สาย.....	4
1.2.3 โพรโทคอลอื่นๆที่ใช้งานบนระบบปฏิบัติการ TinyOS.....	5
1.2.4 สรุปผลการทบทวนเอกสารวิจัยที่เกี่ยวข้อง.....	6
1.3 วัตถุประสงค์ของวิทยานิพนธ์.....	7
1.4 ขอบเขตของการทำวิทยานิพนธ์.....	7
1.5 ขั้นตอนและวิธีดำเนินงานในวิทยานิพนธ์.....	7
1.6 เครื่องมือที่ใช้ในการทำวิทยานิพนธ์.....	8
1.7 ประโยชน์ที่คาดว่าจะได้รับ.....	8
บทที่ 2 ทฤษฎีและหลักการ.....	9
2.1 เครือข่ายเซนเซอร์ไร้สาย (Wireless Sensor Networks).....	9
2.2 รูปแบบและการทำงานของโพรโทคอล.....	10
2.2.1 โพรโทคอลกลุ่ม Flat.....	10
2.2.2 โพรโทคอลกลุ่ม Hierarchical.....	11
2.2.3 โพรโทคอลกลุ่มเครือข่าย ad hoc.....	11

สารบัญ (ต่อ)

	หน้า
2.2.4 การทำงานของโพรโทคอล AODV	14
2.2.5 การทำงานของโพรโทคอล DSDV	15
2.2.6 การทำงานของโพรโทคอล DSR	15
2.3 โพรโทคอล DYnamic Manet On-demand (DYMO)	15
2.4 คุณสมบัติของ โหนด Unode	16
2.5 ระบบปฏิบัติการ TinyOS 2.0	17
2.5.1 ข้อแตกต่างของสถาปัตยกรรม TinyOS 1.1 และ TinyOS 2.0	17
2.5.2 การทำงานแบบพร้อมกัน (Concurrency)	17
2.6 ภาษา NesC	18
2.6.1 คอมโพเนนต์ (Component)	18
2.6.2 อินเตอร์เฟซ (interface)	19
บทที่ 3 การออกแบบโพรโทคอลค้นหาเส้นทาง	21
3.1 ภาพรวมของโพรโทคอล	21
3.2 รูปแบบของแพ็กเก็ต	21
3.3 การทำงานของโพรโทคอล	23
3.3.1 การกำหนดค่าเริ่มต้น (initial setup)	24
3.3.2 การรับส่งข้อมูล (data collection)	25
3.3.3 การบำรุงรักษาเส้นทาง (maintenance)	26
3.4 การพัฒนาโพรโทคอลบนระบบปฏิบัติการ TinyOS	29
3.4.1 การสร้างโพรโทคอลบน TinyOS2.x	30
3.4.2 การทำงานในระดับชั้น Data Link	30
3.4.3 การทำงานในระดับชั้น Network Layer	31
บทที่ 4 การทดสอบและวิเคราะห์ประสิทธิภาพของโพรโทคอล	37

สารบัญ (ต่อ)

	หน้า
4.1 การทดสอบประสิทธิภาพของ โพรโทคอล	37
4.1.1 Packet Delivery Ratio (PDR).....	37
4.1.2 End-to-End Delay	37
4.1.3 เวลาในการปรับปรุงเส้นทาง	38
4.2 โครงสร้างเครือข่าย (Topology) ที่ใช้ในการทดสอบ.....	38
4.3 สภาพแวดล้อมในการทดลอง	39
4.4 ผลการทดสอบประสิทธิภาพ.....	39
4.4.1 Packet Delivery Ratio (PDR).....	39
4.4.2 การทดสอบประสิทธิภาพของ โพรโทคอลแบบมี Scheduling.....	41
4.4.3 End-to-End Delay	42
4.4.4 การทดสอบความเร็วในการปรับปรุงเส้นทาง	43
4.4.5 ขนาดของ โพรโทคอลที่พัฒนาบนระบบปฏิบัติการ TinyOS.....	44
4.5 การนำไปใช้งานในฟาร์มเลี้ยงกุ้ง.....	44
บทที่ 5 สรุปผลการวิจัยและข้อเสนอแนะ	48
5.1 สรุปผล	48
5.2 ผลที่ได้จากการทำวิทยานิพนธ์ชุดนี้.....	49
5.3 ปัญหาและอุปสรรค.....	49
5.4 ข้อเสนอแนะและแนวทางการพัฒนาในอนาคต	49
บรรณานุกรม.....	51
ภาคผนวก	53
ผลงานตีพิมพ์เผยแพร่จากวิทยานิพนธ์	54
ประวัติผู้เขียน	62

รายการตาราง

	หน้า
ตารางที่ 1-1 เปรียบเทียบโพรโทคอล AODV ที่ได้พัฒนาสำหรับเครือข่ายเซนเซอร์ไร้สาย.....	4
ตารางที่ 3-1 การทำงานของฟิลด์ข้อมูล.....	22
ตารางที่ 4-1 ค่า Packet Delivery Ratio ของรูปแบบการทดสอบที่ 1, 2 และ 3	40
ตารางที่ 4-2 ผลการทดสอบค่า PDR กับจำนวนการเชื่อมต่อ.....	42
ตารางที่ 4-3 ผลการทดสอบ End-to-End Delay.....	43
ตารางที่ 4-4 ผลการทดสอบหาความเร็วในการปรับปรุงเส้นทาง	44
ตารางที่ 4-5 แสดงขนาดของโพรโทคอลที่ได้ทำการพัฒนาขึ้นกับโพรโทคอล DYMO.....	44

รายการภาพประกอบ

	หน้า
ภาพประกอบ 1-1 เส้นทางในเครือข่ายเกิดเสียหาย.....	3
ภาพประกอบ 2-1 เครือข่ายเซนเซอร์ไร้สาย	9
ภาพประกอบ 2-2 ตัวอย่างการทำงานของโปรโตคอลแบบ Proactive	12
ภาพประกอบ 2-3 ตัวอย่างการทำงานของโปรโตคอลแบบ Reactive.....	13
ภาพประกอบ 2-4 Unode	16
ภาพประกอบ 2-5 คอมโพเนนต์แบบ module ชื่อ TimerM.....	18
ภาพประกอบ 2-6 คอมโพเนนต์แบบ configuration ชื่อ BlinkM.....	19
ภาพประกอบ 2-7 อินเตอร์เฟซ Timer	19
ภาพประกอบ 3-1 รูปแบบแพ็กเก็ตในโปรโตคอล	22
ภาพประกอบ 3-2 โครงสร้างเครือข่ายเซนเซอร์ไร้สาย.....	23
ภาพประกอบ 3-3 รูปแบบการเริ่มต้นการค้นหาเส้นทาง	24
ภาพประกอบ 3-4 โหนด gateway ทำการกระจายแพ็กเก็ตให้กับโหนดรอบข้าง.....	25
ภาพประกอบ 3-5 โหนดมีเส้นทางติดต่อกับโหนด gateway	25
ภาพประกอบ 3-6 รูปแบบการรับส่งข้อมูลของโหนด	26
ภาพประกอบ 3-7 รูปแบบการบำรุงรักษาเส้นทางของโปรโตคอล	27
ภาพประกอบ 3-8 เส้นทางในเครือข่ายเกิดการเสียหาย.....	28
ภาพประกอบ 3-9 เส้นทางใหม่ในเครือข่ายในการติดต่อสื่อสาร	28
ภาพประกอบ 3-10 โครงสร้างลำดับชั้น network บนระบบปฏิบัติการ TinyOS2.0.....	29
ภาพประกอบ 3-11 โครงสร้างการทำงานระบบปฏิบัติการ TinyOS.....	30
ภาพประกอบ 3-12 โครงสร้างของคอมโพเนนต์ PNetwork.....	32
ภาพประกอบ 3-13 ฝั่งงานการทำงานของโหนด gateway.....	33
ภาพประกอบ 3-14 ฝั่งงานการทำงานของเซนเซอร์โหนดในการรับแพ็กเก็ต.....	34
ภาพประกอบ 3-15 ฝั่งงานการทำงานของเซนเซอร์โหนดในการส่งข้อมูล	35
ภาพประกอบ 4-1 โครงสร้างเครือข่ายแบบเรียงต่อกันจำนวน 4 hop	38
ภาพประกอบ 4-2 โครงสร้างเครือข่ายแบบมี 2 ทางเลือกแต่ละเส้นทางมีจำนวน 2 hop.....	39
ภาพประกอบ 4-3 โครงสร้างเครือข่ายแบบ 3 hop lollipop.....	39
ภาพประกอบ 4-4 ฟาร์มเลี้ยงลูกกึ่ง	45
ภาพประกอบ 4-5 การติดตั้งโหนดที่นำไปใช้งาน	45

รายการภาพประกอบ (ต่อ)

	หน้า
ภาพประกอบ 4-6 โครงสร้างเครือข่ายที่ทำการติดตั้งเซนเซอร์.....	46
ภาพประกอบ 4-7 ส่วนแสดงผลข้อมูล	46

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของวิทยานิพนธ์

ในปัจจุบันเทคโนโลยีเครือข่ายเซนเซอร์ไร้สายถูกนำมาประยุกต์ใช้งานอย่างแพร่หลาย เช่น การเก็บข้อมูลด้านการเกษตร การเฝ้าระวังผู้บุกรุกด้านการทหาร การเฝ้าระวังปริมาณน้ำฝนเพื่อคาดการณ์อุทกภัยหรือดินถล่ม เป็นต้น เครือข่ายเซนเซอร์ไร้สายประกอบไปด้วยอุปกรณ์ตรวจวัดขนาดเล็กที่เรียกว่าโหนดเป็นจำนวนมาก โหนดประกอบด้วยโมดูลในการสื่อสารแบบไร้สายผ่านคลื่นวิทยุย่านความถี่ 2.4 GHz หน่วยประมวลผล หน่วยความจำ ตัวตรวจวัดและแหล่งจ่ายพลังงาน นอกจากนี้เครือข่ายเซนเซอร์ไร้สายจำเป็นที่จะต้องมีอุปกรณ์เชื่อมต่อกับเครือข่ายอื่นหรือเครื่องแม่ข่าย ซึ่งจะเรียกอุปกรณ์นี้ว่า gateway เป็นโหนดที่มีความสามารถสูงกว่าโหนดทั่วไป รวมทั้งจะต้องมีหน่วยความจำที่มีขนาดใหญ่เพียงพอสำหรับงานประยุกต์นั้นๆเพื่อใช้สำหรับเก็บข้อมูลทั้งหมดที่ได้รับจากโหนดในเครือข่าย

เนื่องจากการประยุกต์ใช้งานเครือข่ายเซนเซอร์ไร้สายมีข้อจำกัดในเรื่องของแหล่งจ่ายพลังงานให้กับโหนดที่มีจำกัด และความสามารถของหน่วยประมวลผล ซึ่งมีประสิทธิภาพไม่สูง เพราะแนวคิดของโหนดในเครือข่ายเซนเซอร์ไร้สายจะต้องมีขนาดเล็ก และราคาถูก ข้อจำกัดเหล่านี้ยังส่งผลโดยตรงต่อการรับส่งข้อมูล เพราะกำลังส่งของโมดูลการสื่อสารจะน้อยทำให้การส่งข้อมูลได้ภายในระยะทางที่สั้น (short range communication) ดังนั้นในการทำงานต้องกระจายโหนดจำนวนมากลงไปในพื้นที่ที่สนใจ ซึ่งการส่งข้อมูลภายในเครือข่ายไม่ได้เป็นไปในลักษณะ single hop เท่านั้น โหนดที่อยู่ห่างกันเกินจากรัศมีที่ใช้ในการส่งข้อมูล จะเกิดการส่งข้อมูลในอีกลักษณะคือ โหนดจะส่งข้อมูลผ่านไปยังโหนดข้างเคียง (neighbor) หรือเรียกว่าการส่งข้อมูลแบบ multi-hop โดยการส่งข้อมูลลักษณะนี้โมดูลการสื่อสารสามารถใช้กำลังส่งน้อยๆ ทำให้ช่วยประหยัดพลังงานได้

นอกจากนี้การพัฒนาโปรโตคอลสำหรับค้นหาเส้นทางบนเครือข่ายเซนเซอร์ไร้สายจะต้องคำนึงทั้งในเรื่องการใช้พลังงานและความรวดเร็วในการสื่อสาร ฉะนั้นโปรโตคอลค้นหาเส้นทางควรจะต้องมีขนาดเล็กและมีการออกแบบให้ทำงานเฉพาะที่จำเป็นสำหรับเครือข่ายเซนเซอร์ไร้สายของแต่ละงานประยุกต์เท่านั้น เพื่อช่วยประหยัดขนาดของหน่วยความจำบนโหนด

เมื่อโพรโทคอลไม่มีความซับซ้อนทำให้สามารถจัดการการสื่อสารได้อย่างรวดเร็ว และตอบสนองกับความต้องการของผู้ใช้ได้ทันทีทันใดแบบเรียลไทม์ ด้วยขนาดแบนด์วิดท์ที่ใหญ่ของเครือข่ายไร้สายมาตรฐาน IEEE 802.11 และไม่มีข้อจำกัดในเรื่องของพลังงาน โพรโทคอลค้นหาเส้นทางที่มีอยู่เดิมในเครือข่ายไร้สายตามมาตรฐาน IEEE 802.11 จึงไม่เหมาะสมสำหรับการนำมาใช้งานบนเครือข่ายเซนเซอร์ไร้สายบนมาตรฐาน IEEE 802.15.4 ที่มีอัตราการส่งข้อมูลเพียง 256 kbps

งานวิจัยนี้จึงเป็นการออกแบบและพัฒนาโพรโทคอลค้นหาเส้นทางที่มีการทำงานที่ไม่ซับซ้อนเหมาะสำหรับเครือข่ายเซนเซอร์ไร้สาย ให้มีความสามารถในการค้นหาเส้นทาง การรับส่งข้อมูล และการบำรุงรักษาเส้นทาง ทำงานตามมาตรฐาน IEEE 802.15.4 ซึ่งประกอบด้วยชั้น physical layer และชั้น MAC layer โดยที่ชั้น physical layer จะทำงานที่ความถี่ 2.4 ถึง 2.4853 GHz อัตราการส่งที่ 250 kbps และมีจำนวน 16 ช่องสัญญาณคือช่องที่ 11 ถึง 26 นอกจากนี้ได้ทำการพัฒนาโพรโทคอลใช้งานจริงบน Unode ที่ทำการพัฒนาด้วยสถาปัตยกรรมเครือข่ายของระบบปฏิบัติการ TinyOS2.0 [1] ซึ่งแบ่งโมดูลรองรับการทำงานของเครือข่ายไว้ 3 ชั้นคือ physical, data link และ network

โพรโทคอลที่ได้ออกแบบจะถูกนำไปประยุกต์ใช้งานจริงกับการเก็บข้อมูลในฟาร์มเพาะเลี้ยงลูกกุ้งที่ใช้โหนดจำนวนประมาณ 30 โหนด ซึ่งโหนดจะทำหน้าที่ทั้งตรวจวัดอุณหภูมิของน้ำและควบคุมการเปิด-ปิดเครื่องปรับอากาศ โดยข้อกำหนดที่สำคัญคือระบบจะถูกควบคุมทั้งแบบอัตโนมัติตามค่าที่ตั้งไว้และควบคุมโดยผู้ใช้แบบเรียลไทม์ด้วยโหนดสถานีฐาน (base station) โดยโหนดสถานีฐานนี้จะเชื่อมต่อเข้ากับคอมพิวเตอร์เพื่อทำหน้าที่ควบคุมการทำงานของเครือข่ายและเก็บข้อมูล โดยมีวิธีการทำงานคือสถานีฐานจะส่งคำสั่งไปควบคุมเครื่องปรับอากาศ ตั้งค่าการอ่านอุณหภูมิจากโหนด ดังนั้นโหนดสถานีฐานจะทำหน้าที่เป็นตัวตรวจหาและสร้างเส้นทางเชื่อมต่อ โดยที่โพรโทคอลที่พัฒนาขึ้นจะต้องมีขนาดเล็ก สามารถใช้งานได้จริงบนโหนดที่มีหน่วยความจำจำกัด และสามารถเพิ่มอัตราการรับส่งข้อมูลเมื่อมีจำนวนการเชื่อมต่อเพิ่มขึ้น

1.2 ตรวจสอบเอกสาร

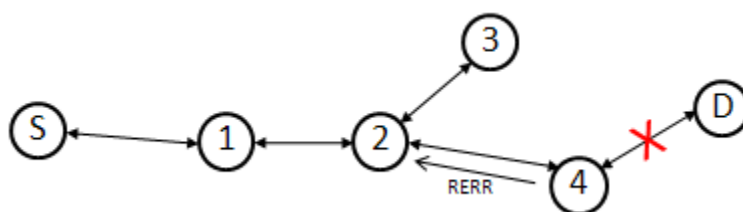
โพรโทคอล Ad hoc On-demand Distance Vectoring (AODV) [2] กำลังได้รับความนิยมในการนำไปใช้งานบนเครือข่ายเซนเซอร์ไร้สาย เนื่องจากการทำงานของเครือข่ายแบบ

ad hoc คล้ายกับแนวคิดของเครือข่ายเซนเซอร์ไร้สาย ดังนั้นงานวิจัยนี้จึงเลือกศึกษาโพรโทคอล AODV นี้เพื่อเป็นแนวทางของการพัฒนาโพรโทคอลที่เหมาะสมสำหรับเครือข่ายเซนเซอร์ไร้สาย

1.2.1 โพรโทคอล Ad hoc On-demand Distance Vectoring (AODV)

โพรโทคอล AODV เป็นโพรโทคอลที่ช่วยในการค้นหาเส้นทางในเครือข่าย ทำให้โหนดสมาชิกสามารถติดต่อสื่อสารกัน โดยเส้นทางในการติดต่อสื่อสารอาจจะมีได้หลายเส้นทาง โพรโทคอล AODV จะเริ่มดำเนินการค้นหาเส้นทางโดยการกระจายแพ็กเก็ต Route Request (RREQ) ไปให้โหนดรอบข้างรวมถึงโหนดปลายทาง เมื่อโหนดปลายทางได้รับแพ็กเก็ต RREQ แล้ว โหนดปลายทางจะทำการส่งแพ็กเก็ต Route Reply (RREP) ไปให้กับโหนดต้นทาง เพื่อให้โหนดต้นทางทำการเลือกเส้นทางในการติดต่อสื่อสารกับโหนดปลายทาง

ในการใช้งานเครือข่าย ad hoc โหนดสามารถเคลื่อนที่ได้ จึงเป็นเหตุผลให้โพรโทคอล AODV มีกระบวนการในการตรวจสอบเส้นทางในเครือข่ายอยู่เป็นระยะ โดยทำการกระจายแพ็กเก็ตที่ชื่อว่า Hello ให้กับโหนดรอบข้าง เพื่อตรวจสอบว่าสามารถติดต่อถึงโหนดรอบข้างได้หรือไม่ดังในภาพประกอบ 1-1 แสดงให้เห็นว่า เมื่อโหนดหมายเลข 4 ทำการตรวจสอบโหนดรอบข้างแล้ว พบว่าเส้นทางเชื่อมต่อกับโหนด D (โหนดปลายทาง) เสียหาย โหนดหมายเลข 4 จะกระจายแพ็กเก็ต Route Error (RERR) ไปให้โหนดรอบข้างรับทราบ เมื่อโหนด S (โหนดต้นทาง) ได้รับแพ็กเก็ต RERR โหนด S จะลบเส้นทางสื่อสารถึงโหนด D ในตารางเส้นทางของโหนด ถ้าโหนด S ต้องการติดต่อถึงโหนด D อีกครั้ง โหนด S จะเริ่มกระบวนการค้นหาเส้นทางใหม่อีกครั้ง



ภาพประกอบ 1-1 เส้นทางในเครือข่ายเกิดเสียหาย

จากการทำงานของโพรโทคอล AODV พบว่ามีการทำงานบางส่วนที่เครือข่ายเซนเซอร์ไร้สายไม่มีความจำเป็นต้องใช้งานเช่น การตรวจสอบเส้นทางโดยใช้แพ็กเก็ต Hello อยู่ตลอดเวลา เนื่องจากงานประยุกต์ในเครือข่ายเซนเซอร์ไร้สาย โหนดส่วนใหญ่ไม่ได้เคลื่อนที่ ทำให้

การตรวจสอบเส้นทางด้วยวิธีการกระจายแพ็กเก็ต Hello ในโพรโทคอล AODV จึงเป็นการสิ้นเปลืองพลังงานของโหนด

1.2.2 การพัฒนาโพรโทคอล AODV สำหรับเครือข่ายเซนเซอร์ไร้สาย

มีงานวิจัยหลายชิ้นที่ได้ทำการพัฒนาโพรโทคอล AODV เพื่อใช้งานในเครือข่ายเซนเซอร์ไร้สาย โดยทำการพัฒนาปรับลดการทำงานบางส่วนที่ไม่จำเป็นออก เนื่องจากเครือข่ายเซนเซอร์ไร้สายมีข้อจำกัดหลายด้านเช่นพลังงานของโหนด การประมวลผล และหน่วยความจำของโหนด เป็นต้น ทำให้ไม่สามารถที่จะนำโพรโทคอล AODV ดั้งเดิมทั้งหมดมาใช้งานในเครือข่ายเซนเซอร์ไร้สายได้ทันที

ตารางที่ 1-1 เปรียบเทียบโพรโทคอล AODV ที่ได้พัฒนาสำหรับเครือข่ายเซนเซอร์ไร้สาย

	Connectivity maintenance mechanism	RERR	Local repair	Only destination generates RREP	Routing metric	Specification and/or implementation status
AODV	Hello message, LLN, etc.	Yes	Yes	No	Hop count	Experimental RFC; Implemented for several platforms
AODVjr.	Connect messages	No	No	Yes	Fastest RREP	Implemented for NS2 simulator
TinyAODV	LLN	Yes	No	Yes	Hop count	Implemented for TinyOS
NST-AODV	LLN	Yes	Yes	No	Hop count	Implemented for TinyOS

ในตารางที่ 1-1 แสดงให้เห็นถึงการเปรียบเทียบคุณลักษณะของโพรโทคอล AODV ที่ได้มีการพัฒนาขึ้นสำหรับเครือข่ายเซนเซอร์ไร้สาย ได้แก่ โพรโทคอล AODVjr. [3] โพรโทคอล tinyAODV [4] และโพรโทคอล Not so tiny (NST-AODV) [5]

โพรโทคอล AODVjr. เป็นโพรโทคอลได้รับการพัฒนามาจากโพรโทคอล AODV โดยทำการตัดส่วนของการตรวจสอบเส้นทางออก ซึ่งได้แก่แพ็กเก็ต Hello ดังนั้นจึงไม่มีการส่งแพ็กเก็ต RERR สำหรับส่วนของการเลือกเส้นทางจะไม่ใช้จำนวน hop มาพิจารณา แต่จะใช้

ความเร็วในการได้รับแพ็กเก็ต RREP เป็นตัวแปรในการพิจารณาเลือกเส้นทาง ในงานวิจัย [3] ได้มีการทดสอบประสิทธิภาพการทำงานของโปรโตคอล AODVjr. บน Network Simulator 2 (NS-2) ด้วยสภาพการทำงานแบบเครือข่ายเซนเซอร์ไร้สายโดยที่ไม่มีโหนดเคลื่อนที่ พบว่าค่า Packet Delivery Ratio (PDR) ของโปรโตคอล AODVjr. มีค่าใกล้เคียงกันกับโปรโตคอล AODV อีกทั้งโปรโตคอล AODVjr. มีจำนวนแพ็กเก็ตในเครือข่ายน้อยกว่าโปรโตคอล AODV

สำหรับโปรโตคอล tinyAODV และโปรโตคอล NST-AODV ถูกพัฒนามาจากโปรโตคอล AODV ที่นำไปสร้างบนระบบปฏิบัติการ TinyOS 1.1 โดยทั้งสองโปรโตคอลได้ตัดความสามารถบางส่วนของโปรโตคอล AODV ออกไป ซึ่งมีรายละเอียดดังต่อไปนี้

โปรโตคอล tinyAODV ตัดความสามารถในการตรวจสอบเส้นทางโดยใช้แพ็กเก็ต Hello และโหนดที่สามารถสร้างแพ็กเก็ต RREP มีเพียงแค่โหนดปลายทางเท่านั้น และไม่มีกระบวนการซ่อมแซมเส้นทางในการติดต่อสื่อสารกันของโหนดในเครือข่าย เมื่อพบว่ามีครบของเส้นทางจากแพ็กเก็ต RERR จะทำการเริ่มกระบวนการหาเส้นทางใหม่ที่ ส่วนโปรโตคอล NST-AODV โหนดทุกตัวสามารถสร้างแพ็กเก็ต RREP ได้ และการตรวจสอบเส้นทาง ไม่ได้ใช้แพ็กเก็ต Hello แต่จะใช้ข้อมูลจากการส่งแพ็กเก็ตให้โหนดรอบข้างไม่สำเร็จ ซึ่งกระบวนการนี้จะทำการตรวจสอบในระดับชั้น link layer

1.2.3 โปรโตคอลอื่นๆที่ใช้งานบนระบบปฏิบัติการ TinyOS

ในบทความ [6] ได้นำเสนอผลการวิเคราะห์ประสิทธิภาพของโปรโตคอล AODV และ Destination-Sequenced Distance Vector (DSDV) โดยทำการสร้างโปรโตคอลบนโปรแกรมจำลองการทำงานของระบบปฏิบัติการ TinyOS ที่เรียกว่า TOSSIM บทความนี้ได้ทำการทดสอบการทำงานบนเครือข่ายขนาด 5 ถึง 30 โหนด ผลการทดสอบพบว่าที่ 5 โหนด พบว่า Packet Delivery Ratio (PDR) ของโปรโตคอล AODV ให้ค่า 95 เปอร์เซ็นต์ และ PDR ของโปรโตคอล DSDV ให้ค่า 80 เปอร์เซ็นต์ ส่วนผลการทดสอบที่ 20 ถึง 30 โหนด ค่า PDR ของโปรโตคอล AODV ให้ค่า 72 เปอร์เซ็นต์ และค่า PDR ของโปรโตคอล DSDV ให้ค่า 66 เปอร์เซ็นต์ จากผลการทดสอบพบว่าโปรโตคอล AODV มีค่า PDR มากกว่า โปรโตคอล DSDV

ในการทดสอบหาค่าหน่วงเวลาแบบ End-to-End Delay ของทั้งสองโปรโตคอลจะเริ่มจากที่ 5 วินาทีสำหรับ 5 โหนด และ 10 วินาทีสำหรับ 10 โหนด เมื่อจำนวนโหนดเพิ่มขึ้นเป็น 15 โหนด โปรโตคอล AODV และ DSDV จะมีค่าหน่วงเวลาที่ 13 และ 9 วินาทีตามลำดับ ค่าหน่วง

เวลาของโพรโทคอล AODV มากครั้งที่ 20 วินาทีเมื่อมีโหนด 20 ถึง 30 โหนด และค่าหน่วยเวลาของโพรโทคอล DSDV มากครั้งที่ 15 วินาทีเมื่อมีโหนด 20 ถึง 30 โหนด จากการทดสอบพบว่าโพรโทคอล DSDV จะน้อยกว่าโพรโทคอล AODV ประมาณ 25 เปอร์เซ็นต์ ดังนั้นโพรโทคอล AODV มีจุดเด่นในเรื่องของค่า PDR ในขณะที่โพรโทคอล DSDV มีจุดเด่นในเรื่องของความเร็ว ดังนั้นการเลือกใช้โพรโทคอลควรเลือกใช้ตามลักษณะของงานประยุกต์ที่ต้องการ

1.2.4 สรุปผลการทบทวนเอกสารวิจัยที่เกี่ยวข้อง

จากการศึกษางานวิจัยอื่นๆที่เกี่ยวข้องพบว่าโพรโทคอล AODV แบบมาตรฐานมีขนาดใหญ่ไม่เหมาะสมสำหรับการใช้งานบนโหนดที่มีข้อจำกัดเรื่องพลังงานซึ่งโหนดทำการประมวลผลด้วยไมโครคอนโทรลเลอร์ที่ความถี่เพียง 8 MHz และมีขนาดของหน่วยความจำจำกัด ตัวอย่างเช่นบนโหนด Unode ที่จะใช้ในวิทยานิพนธ์นี้มีหน่วยความจำชนิด RAM และ Flash อยู่ที่ 10KB และ 48KB ตามลำดับ

สำหรับโพรโทคอล AODV หรือ DSDV ที่ถูกพัฒนาเพื่อให้ใช้ในเครือข่ายเซนเซอร์ไร้สาย ในงานวิจัยก่อนหน้านี้พบว่าเป็นการทดสอบโพรโทคอลเฉพาะในโปรแกรมจำลองการทำงานเท่านั้น ดังนั้นถ้าต้องการนำมาใช้งานจริงจะต้องทำการพัฒนาขึ้นมาใหม่เพื่อให้รองรับกับสถาปัตยกรรมของระบบปฏิบัติการ TinyOS บนโหนดสำหรับโพรโทคอล tinyAODV ที่ได้รับการพัฒนาบนระบบปฏิบัติการ TinyOS1.1 แล้ว แต่ก็ไม่สามารถนำมาใช้งานได้จริง เนื่องจากโครงสร้างการทำงานของระบบปฏิบัติการ TinyOS ในปัจจุบันเป็น TinyOS2.0 ซึ่งมีสถาปัตยกรรมแตกต่างจาก TinyOS1.1 ดังนั้นการนำ tinyAODV มาใช้งานจะต้องมีการพัฒนาขึ้นมาใหม่โดยเฉพาะสำหรับ TinyOS2.0 นอกจากนี้โพรโทคอล tinyAODV ไม่มีส่วนของการซ่อมแซมเส้นทาง (route maintenance) และไม่รองรับการทำงานแบบ timer-based event ที่จำเป็นจะต้องนำมาใช้ในเครือข่ายเซนเซอร์ไร้สาย

ดังนั้นทางผู้วิจัยจึงทำการพัฒนาโพรโทคอลค้นหาเส้นทางขึ้นมาใหม่ เพื่อให้สามารถนำไปประยุกต์ใช้จริงบน Unode ที่ใช้ระบบปฏิบัติการ TinyOS 2.0 สำหรับระบบควบคุมดูแลฟาร์มเลี้ยงกุ้งแบบอัตโนมัติ

1.3 วัตถุประสงค์ของวิทยานิพนธ์

1. เพื่อศึกษา ออกแบบ และพัฒนาโพรโทคอลค้นหาเส้นทางที่เหมาะสมกับการใช้งานในเครือข่ายเซนเซอร์ไร้สาย
2. เพื่อนำเสนอประสิทธิภาพของโพรโทคอลที่พัฒนาขึ้นในประเด็นของ Packet Delivery Ratio (PDR) และ End-to-End Delay

1.4 ขอบเขตของการทำวิทยานิพนธ์

1. ทดสอบการทำงานของโพรโทคอลบน โหนด platform Unode ที่ใช้ระบบปฏิบัติการ TinyOS
2. สามารถวิเคราะห์ประสิทธิภาพการทำงานของโพรโทคอลในห้องปฏิบัติการและทดสอบการใช้งานจริงจำนวนประมาณ 30 โหนดในฟาร์มกึ่ง

1.5 ขั้นตอนและวิธีดำเนินงานในวิทยานิพนธ์

การดำเนินงานในการทำวิทยานิพนธ์ มีขั้นตอนดังต่อไปนี้

1. ศึกษาค้นคว้าข้อมูลเอกสารที่เกี่ยวข้องกับวิทยานิพนธ์ดังนี้
 - โพรโทคอล AODV
 - โพรโทคอล tinyAODV และโพรโทคอล NST-AODV
 - งานวิจัยอื่นๆ ที่เกี่ยวข้อง
2. วิเคราะห์และออกแบบโพรโทคอล
 - วิเคราะห์โพรโทคอล AODV tinyAODV และโพรโทคอล NST-AODV
 - การออกแบบการติดต่อสื่อสารกันของโพรโทคอล
3. ทดสอบและแก้ไขโพรโทคอล
 - ใช้ภาษา NesC ในการออกแบบการทำงานของโพรโทคอล โดยใช้ระบบปฏิบัติการ TinyOS 2.0
 - ทดสอบการทำงานของโพรโทคอลบน โหนด platform Unode
 - วิเคราะห์หาข้อผิดพลาดและแนวทางในการปรับปรุงแก้ไข
 - ปรับปรุงโพรโทคอลเพื่อให้เหมาะสมกับเครือข่ายเซนเซอร์ไร้สาย
 - ทดสอบการทำงานของโพรโทคอลหลังจากการปรับปรุงแก้ไขแล้ว
4. จัดทำเอกสารรายงานผลการทำวิจัยฉบับสมบูรณ์

1.6 เครื่องมือที่ใช้ในการทำวิทยานิพนธ์

1. ด้านฮาร์ดแวร์

- เครื่องคอมพิวเตอร์ ความเร็วหน่วยประมวลผลที่ 2.66 GHz และมีหน่วยความจำ 4 GB ฮาร์ดดิสก์ความจุ 320 กิกะไบต์ จำนวน 1 ชุด
- อุปกรณ์โหนด Unode จำนวนประมาณ 30 ตัว

2. ด้านซอฟต์แวร์

- ระบบปฏิบัติการ Microsoft Window XP
- โปรแกรม cygwin
- ระบบปฏิบัติการ Ubuntu 9.04
- Tool chain TinyOS 2.0

1.7 ประโยชน์ที่คาดว่าจะได้รับ

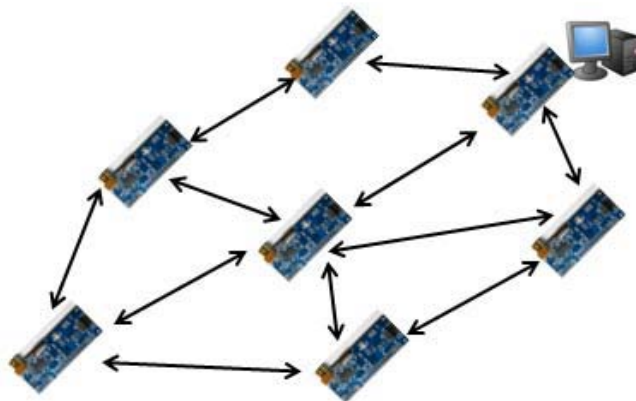
1. ได้โพรโทคอลที่เหมาะสมกับการใช้งานจริงในเครือข่ายเซนเซอร์ไร้สาย
2. สามารถนำโพรโทคอลที่พัฒนาขึ้นไปประยุกต์ใช้งานกับการจัดการฟาร์มเลี้ยงลูกกุ้ง

บทที่ 2

ทฤษฎีและหลักการ

โพรโทคอลในเครือข่ายเซนเซอร์ไร้สายมีการทำงานที่คล้ายกับเครือข่าย ad hoc ดังนั้นนักวิจัยได้นำโพรโทคอลในเครือข่าย ad hoc มาประยุกต์ใช้ในเครือข่ายเซนเซอร์ไร้สาย เพื่อให้สามารถรองรับเครือข่ายที่โหนดเคลื่อนที่ได้ ในบทนี้จึงอธิบายการทำงานของเครือข่ายเซนเซอร์ไร้สาย ข้อแตกต่างระหว่างเครือข่าย ad hoc กับเครือข่ายเซนเซอร์ไร้สาย รูปแบบการทำงานของโพรโทคอลในเครือข่าย ad hoc และเซนเซอร์ไร้สาย คุณลักษณะของโหนด Unode โพรโทคอล DYMO ที่มีใช้งานจริงบนโหนด ระบบปฏิบัติการ TinyOS2.0 และภาษา NesC ที่ใช้พัฒนาโพรโทคอลบนโหนด Unode

2.1 เครือข่ายเซนเซอร์ไร้สาย (Wireless Sensor Networks)



ภาพประกอบ 2-1 เครือข่ายเซนเซอร์ไร้สาย

เครือข่ายเซนเซอร์ไร้สาย [7] เป็นการเชื่อมต่อเครือข่ายด้วยอุปกรณ์ขนาดเล็กที่เรียกว่าโหนด (node) จำนวนมากดังภาพประกอบ 2-1 โดยที่โหนดประกอบด้วยส่วนสำคัญดังนี้

- 1) อุปกรณ์ตรวจวัด (sensor) สภาพแวดล้อมต่างๆ เช่น อุณหภูมิ เสียง การสั่นสะเทือน ความกดอากาศ หรือมลพิษ เป็นต้น
- 2) อุปกรณ์สื่อสารด้วยคลื่นวิทยุ (Radio Frequency, RF) แบบไร้สายบนคลื่นความถี่ 2.4 GHz ที่อัตราการส่งข้อมูล 250 kbps
- 3) ไมโครคอนโทรลเลอร์สำหรับใช้ในการประมวลผลและ
- 4) หน่วยความจำเพื่อเก็บข้อมูลและโปรแกรมบนโหนด โดยการทำงานของเซนเซอร์ไร้สายนั้นเริ่มจากการที่ไมโครคอนโทรลเลอร์ควบคุมอุปกรณ์ตรวจวัดและอุปกรณ์สื่อสาร

ด้วยคลื่นวิทยุ เมื่ออุปกรณ์ตรวจวัดได้ค่าจากการตรวจวัดแล้ว ไมโครคอนโทรลเลอร์จะทำการประมวลผล และค้นหาเส้นทางเพื่อจัดส่งข้อมูลผ่านคลื่นสัญญาณวิทยุกลับไปยังสถานีฐาน

เครือข่าย ad hoc [7] กับเครือข่ายเซนเซอร์ไร้สายมีความคล้ายคลึงกันในลักษณะของการสื่อสารติดต่อกันระหว่างโหนดแบบไร้สาย สามารถทำงานได้โดยไม่จำเป็นต้องมีสถานีฐาน และสามารถจัดการเครือข่ายได้ด้วยตัวเอง (self-configuration) แต่เครือข่ายทั้งสองมีความแตกต่างกันดังนี้

- โครงสร้างการเชื่อมต่อของเครือข่ายเซนเซอร์ไร้สายมีการเปลี่ยนแปลงจากสาเหตุที่สัญญาณวิทยุมีความอ่อนไหวต่อสิ่งแวดล้อม เนื่องจากโมดูลของภาครับส่งสัญญาณจะมีกำลังในการส่งที่ต่ำเพื่อให้โหนดประหยัดพลังงานมากที่สุด ในขณะที่เครือข่าย ad hoc ก็มีการเปลี่ยนแปลงโครงสร้างการเชื่อมต่อบ่อยเนื่องจากโหนดสามารถเคลื่อนที่ไปมาได้อย่างอิสระ
- โหนดในเครือข่ายเซนเซอร์ไร้สายมีข้อจำกัดในด้านของพลังงาน ความสามารถในการประมวลผลและขนาดของหน่วยความจำ

2.2 รูปแบบและการทำงานของโปรโตคอล

ในเครือข่ายเซนเซอร์ไร้สายประกอบไปด้วยโหนดจำนวนมาก ส่วนโปรโตคอลค้นหาเส้นทางจะช่วยในการสร้างเส้นทางเพื่อให้โหนดค้นหาและโหนดปลายทางสามารถส่งข้อมูลถึงกันได้ สำหรับโปรโตคอลค้นหาเส้นทางมีอยู่ด้วยกันหลายรูปแบบ และจากการศึกษาพบว่าโปรโตคอลที่มีการใช้งานในเครือข่ายเซนเซอร์ไร้สาย สามารถนำมาแบ่งตามลักษณะการทำงานออกเป็นกลุ่มดังนี้

2.2.1 โปรโตคอลกลุ่ม Flat

โปรโตคอลในกลุ่มนี้มีการทำงานที่ไม่มีความซับซ้อน เหมาะสำหรับเครือข่ายเซนเซอร์ไร้สายที่มีโหนดจำนวนไม่มาก โหนดในเครือข่ายทุกตัวมีความสามารถเท่าเทียมกัน การค้นหาเส้นทางของโหนดในเครือข่ายสามารถทำได้โดยการกระจาย (broadcast) แพ็กเก็ตเพื่อสร้างเส้นทางเชื่อมต่อของโหนดในเครือข่าย โดยจะเลือกใช้เส้นทางที่มีระยะทางสั้นที่สุด โดยทั่วไปโครงสร้างของเครือข่ายเซนเซอร์ไร้สายที่ใช้โปรโตคอลแบบ flat นี้จะมีลักษณะแบบ data centric โหนดทุกตัวจะส่งข้อมูลไปเก็บยังสถานีฐาน

โพรโทคอลที่อยู่ในกลุ่มการทำงานในลักษณะนี้ได้แก่ Sensor Protocols for Information via Negotiation (SPIN) [8] และ Directed diffusion [9]

2.2.2 โพรโทคอลกลุ่ม Hierarchical

การทำงานของโพรโทคอลในกลุ่มนี้เหมาะสำหรับเครือข่ายที่มีโหนดจำนวนมาก เนื่องจากโดยปกติโหนดจะส่งข้อมูลผ่านแต่ละ hop กลับไปยังสถานีฐาน ซึ่งอาจจะทำให้โหนดที่อยู่ไกลเคียงสถานีฐานจะต้องทำหน้าที่ส่งผ่านข้อมูลแทนโหนดอื่นเป็นจำนวนมาก ทำให้โหนดโดยรอบสถานีฐานจะหมดพลังงานก่อน นอกจากนี้แล้วยังทำให้เกิดความล่าช้าของข้อมูลได้อีกด้วย เพราะข้อมูลจะถูกส่งผ่านโหนดในแต่ละ hop ก่อนจะไปถึงสถานีฐาน ดังนั้นโพรโทคอลกลุ่ม Hierarchical จึงนำเอาหลักการการทำงานของ clustering ด้วยวิธีการจัดโหนดออกเป็นกลุ่มและเลือกหัวหน้าของกลุ่ม (cluster head, CH) ที่สามารถทำหน้าที่รวบรวมข้อมูลจากโหนดสมาชิกแล้วส่งข้อมูลกลับไปยังสถานีฐาน

โพรโทคอลที่อยู่ในกลุ่มการทำงานลักษณะนี้ได้แก่ Low Energy Adaptive Clustering Hierarchy (LEACH) [10] และ โพรโทคอล Power Efficient Gathering in Sensor Information Systems (PEGASIS) [11] เป็นต้น

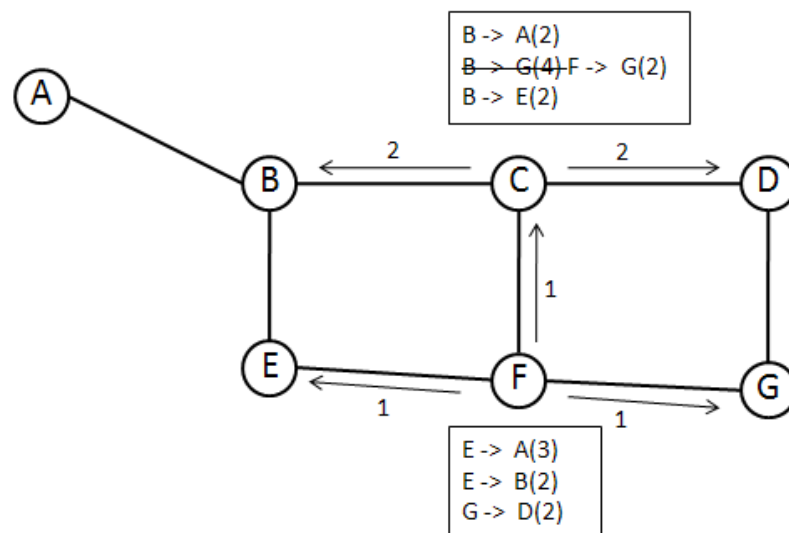
2.2.3 โพรโทคอลกลุ่มเครือข่าย ad hoc

รูปแบบการเชื่อมต่อในเครือข่าย ad hoc มีลักษณะใกล้เคียงกับรูปแบบการเชื่อมต่อของเครือข่ายเซนเซอร์ไร้สาย โดยที่โพรโทคอลในเครือข่าย ad hoc ถูกใช้งานทั่วไปในเครือข่ายไร้สายมาตรฐาน IEEE 802.11 ที่มีการเคลื่อนที่ของโหนดบ่อยครั้ง นอกจากนี้ยังมีอัตราการส่งข้อมูลที่สูงกว่าเครือข่ายเซนเซอร์ไร้สายบนมาตรฐาน IEEE 802.15.4 รวมถึงเครือข่าย ad hoc จะมีกำลังในการส่งที่สูงกว่าเครือข่ายเซนเซอร์ไร้สายเพราะไม่มีข้อจำกัดในเรื่องพลังงาน

งานวิจัยหลายชิ้นจึงได้พยายามปรับแต่งโพรโทคอลในเครือข่าย ad hoc ให้มีขนาดเล็ก และประหยัดพลังงานมากขึ้นเพื่อให้สามารถใช้งานบนเครือข่ายเซนเซอร์ไร้สายได้ ซึ่งตัวอย่างของโพรโทคอลในกลุ่ม ad hoc สามารถอธิบายตามรูปแบบของการเชื่อมต่อได้ดังต่อไปนี้

2.2.3.1 โพรโทคอลแบบ Proactive

การทำงานของโพรโทคอลแบบ Proactive ตัวโหนดที่ทำหน้าที่เป็นสถานีฐานจะเริ่มค้นหาเส้นทางโดยการกระจายแพ็กเก็ตการร้องขอออกไปยังโหนดทุกตัวในเครือข่าย แล้วทำการสร้างเส้นการเชื่อมต่อในลักษณะ many-to-one คือเส้นทางการเชื่อมต่อในทุกโหนดปลายทางมายังสถานีฐาน



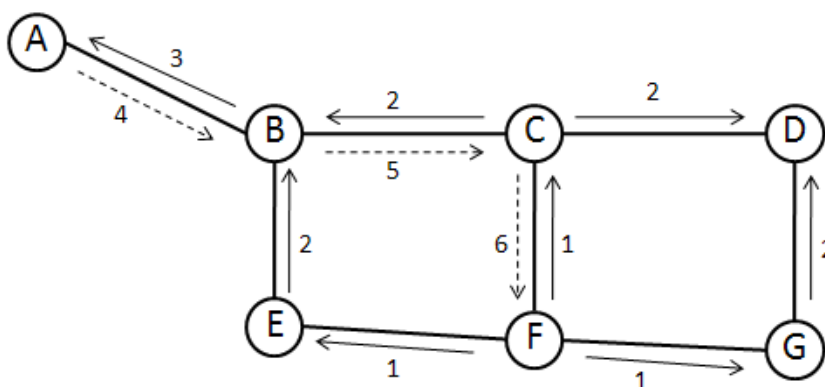
ภาพประกอบ 2-2 ตัวอย่างการทำงานของโพรโทคอลแบบ Proactive

ตัวอย่างการทำงานของโพรโทคอลแบบ Proactive แสดงได้ในภาพประกอบ 2-2 เมื่อเส้นทางการเชื่อมต่อของเครือข่ายที่โหนด F เกิดการเปลี่ยนแปลง 1) โหนด F จะทำการกระจายตารางเส้นทางของตัวเองออกไป 2) เมื่อโหนด C ได้รับตารางเส้นทางจากโหนด F จึงทำการอัปเดตตารางเส้นทางของโหนด C เอง จากเดิมเส้นทางถึงโหนด G ผ่านทางโหนด B ใช้ระยะทาง 4 หน่วย (B->G(4)) แต่เมื่อโหนด F ส่งตารางเส้นทางมาให้โหนด C โหนด C อัปเดตตารางเส้นทางใหม่ เส้นทางถึงโหนด G ผ่านทางโหนด F ใช้ระยะทาง 1 หน่วย (F->G(1))

ตัวอย่างของโพรโทคอลที่มีการทำงานแบบ Proactive ได้แก่ โพรโทคอล Destination Sequenced Distance Vector (DSDV) [12] และโพรโทคอล Optimized Link State Routing (OLSR) [12] เป็นต้น

2.2.3.2 โพรโทคอลแบบ Reactive

การทำงานของโพรโทคอลแบบ Reactive จะเริ่มทำการค้นหาเส้นทาง เมื่อโหนดนั้นต้องการส่งข้อมูลไปหาโหนดปลายทาง โหนดต้นทางจะทำการส่งการร้องขอเส้นทางเข้าสู่เครือข่าย โดยทำการกระจายการร้องขอไปให้โหนดที่อยู่ข้างเคียง



ภาพประกอบ 2-3 ตัวอย่างการทำงานของโพรโทคอลแบบ Reactive

ตัวอย่างการทำงานของโพรโทคอล Reactive แสดงดังภาพประกอบ 2-3 โหนด E ต้องการติดต่อสื่อสารกับโหนด A โหนด F 1) เริ่มทำการกระจายแพ็กเก็ต RREQ ไปให้กับโหนดรอบข้างเพื่อช่วยในการกระจายแพ็กเก็ตไปให้ถึงโหนด A 2) โหนด C และโหนด E ได้รับแพ็กเก็ต RREQ โหนด C และโหนด E ช่วยกระจายแพ็กเก็ต RREQ ให้กับโหนดรอบข้าง 3) โหนด B ได้รับแพ็กเก็ต RREQ โหนด B กระจายแพ็กเก็ต RREQ ถึงโหนดปลายทางคือโหนด A 4) เมื่อโหนด A ได้รับแพ็กเก็ต RREQ โหนด A เลือกเส้นทางในการติดต่อถึงโหนด F โดยส่งแพ็กเก็ตตอบรับ RREP กลับไปให้กลับโหนด F ผ่านทางโหนด B 5) โหนด B ส่งแพ็กเก็ต RREP ให้กับโหนด C 6) โหนด C ส่งแพ็กเก็ต RREP ให้กับโหนด F เมื่อโหนด F ได้รับแพ็กเก็ต RREP ตอบกลับมาแสดงว่าโหนด F มีเส้นทางเชื่อมต่อถึงโหนด A

ตัวอย่างของโพรโทคอลที่มีการทำงานแบบ Reactive ได้แก่ โพรโทคอล Ad hoc On Demand Distance Vector (AODV) และโพรโทคอล Dynamic Source Routing (DSR) [14] เป็นต้น

2.2.3.3 โพรโทคอลแบบ Hybrid

การทำงานของโพรโทคอลแบบ Hybrid จะเป็นการผสมผสานระหว่างโพรโทคอลแบบ Proactive และโพรโทคอลแบบ Reactive เพื่อประยุกต์ใช้กับเครือข่ายขนาดใหญ่ โดยที่เครือข่ายจะถูกแบ่งออกเป็นกลุ่มเล็กๆเรียกว่าคลัสเตอร์ (cluster) และซึ่งการทำงานของโหนดสมาชิกในแต่ละกลุ่มจะเป็นแบบ Proactive ส่วนการติดต่อสื่อสารกันระหว่างโหนดต่างกลุ่มจะมีการทำงานแบบ Reactive

ตัวอย่างของโพรโทคอลที่มีการทำงานแบบ Hybrid จะเหมาะสมกับเครือข่ายที่มีการแบ่งโซนในการจัดวางโหนด เช่น Hazy Sighted Link State (HSLs) [15] เป็นต้น

2.2.4 การทำงานของโพรโทคอล AODV

โพรโทคอล AODV เป็นการทำงานแบบ Source Initiated On-Demand Driven/Reactive คือจะทำการหาเส้นทางก็ต่อเมื่อ โหนดต้นทางต้องการหาเส้นทางเชื่อมต่อถึงโหนดปลายทาง ซึ่งจะเป็นลักษณะของการกระจายแพ็กเก็ตเพื่อหาเส้นทางที่เป็นไปได้จนถึงโหนดปลายทาง จากนั้นจึงจะทำการเลือกเส้นทางเชื่อมต่อระหว่างโหนดต้นทางถึงโหนดปลายทาง ซึ่งการพิจารณาเลือกเส้นทางที่ใช้จำนวน hop น้อยที่สุด

โพรโทคอล AODV มีแพ็กเก็ตที่ช่วยในการค้นหาเส้นทางและซ่อมแซมเส้นทางในการติดต่อสื่อสารด้วยกันคือ แพ็กเก็ต Route Request (RREQ) ช่วยในการร้องขอเส้นทางในการสื่อสาร สำหรับแพ็กเก็ต Route Reply (RREP) ช่วยในการตอบรับการร้องขอเส้นทาง และแพ็กเก็ต Route Error (RERR) ช่วยในการซ่อมแซมเส้นทางในเครือข่าย

เมื่อโหนดต้นทางต้องการติดต่อกับโหนดปลายทางจะเริ่มทำการร้องขอเส้นทางด้วยแพ็กเก็ต RREQ การจ่ายออกไปในเครือข่าย เมื่อโหนดปลายทางได้รับแพ็กเก็ต RREQ โหนดปลายทางจะทำการตอบกลับด้วยการส่งแพ็กเก็ต RREP ให้กับโหนดต้นทาง หลังจากที่โหนดต้นทางได้รับแพ็กเก็ต RREP ก็จะได้เส้นทางในการติดต่อถึงโหนดปลายทาง โพรโทคอล AODV มีกระบวนการตรวจสอบเส้นทางโดยใช้แพ็กเก็ตที่ชื่อว่า Hello ทดสอบการเชื่อมต่อกับโหนดรอบข้างอยู่เป็นระยะๆ เพื่อทำการอัปเดตตารางเส้นทางของโหนดรอบข้างว่าสามารถติดต่อกันได้ แต่เมื่อพบว่าเส้นทางติดต่อสื่อสารได้รับความเสียหาย โหนดจะทำการกระจายแพ็กเก็ต RERR ให้กับโหนดในเครือข่ายเพื่อทำการลบเส้นทางนั้นทิ้ง หากว่าโหนดต้นทางต้องการติดต่อสื่อสารถึงโหนดปลายทาง โหนดต้นทางจะเริ่มค้นหาเส้นทาง

2.2.5 การทำงานของโพรโทคอล DSDV

โพรโทคอล DSDV เป็นโพรโทคอลในกลุ่มของ Proactive การส่งข้อมูลที่เกิดขึ้นในโพรโทคอลนี้จะมีการกำหนดหมายเลขลำดับของแพ็กเก็ตข้อมูลที่มาจากโหนดปลายทาง โหนดแต่ละตัวจะมีตารางเส้นทางในการติดต่อสื่อสารกันและจะเก็บหมายเลขลำดับเข้ามาด้วยเพื่อพิจารณาแพ็กเก็ตข้อมูลที่เข้ามา การซ่อมแซมเส้นทางในเครือข่ายจะเกิดขึ้นเมื่อตารางเส้นทางของโหนดนั้นมีการเปลี่ยนแปลง ในช่วงเวลาที่มีการอัปเดตตารางเส้นทางนั้นจะมีแพ็กเก็ตมากขึ้นในเครือข่าย เนื่องจากแต่ละโหนดจะทำการส่งตารางเส้นทางของตัวเองให้กับโหนดเพื่อนบ้าน ทำให้สิ้นเปลืองพลังงานมากกว่าโพรโทคอลแบบ AODV

2.2.6 การทำงานของโพรโทคอล DSR

โพรโทคอล DSR เป็นโพรโทคอลแบบ On-Demand การทำงานของโพรโทคอล DSR มีดังนี้ เมื่อโหนดต้นทางต้องการที่จะส่งแพ็กเก็ตข้อมูลออกไปยังโหนดปลายทาง โหนดต้นทางทำการตรวจสอบว่ามีเส้นทางในการติดต่อถึงโหนดปลายทางหรือไม่ ถ้ามีโหนดจะส่งแพ็กเก็ตออกไปโดยใช้เส้นทางที่มีอยู่ แต่ถ้าไม่มีเส้นทางในการติดต่อสื่อสารก็ทำการเริ่มค้นหาเส้นทางใหม่ โดยทำการร้องขอเส้นทางเพื่อติดต่อสื่อสารกับโหนดปลายทางนั้น เมื่อได้รับแพ็กเก็ตข้อมูลการตอบรับจากโหนดปลายทาง โหนดต้นทางจะทำการบันทึกจำนวน hop ในการติดต่อสื่อสารที่ได้รับ โดยเส้นทางที่ได้รับมาจะทำการจำไว้ในหน่วยความจำของตัวโหนด

2.3 โพรโทคอล DYnamic Manet On-demand (DYMO)

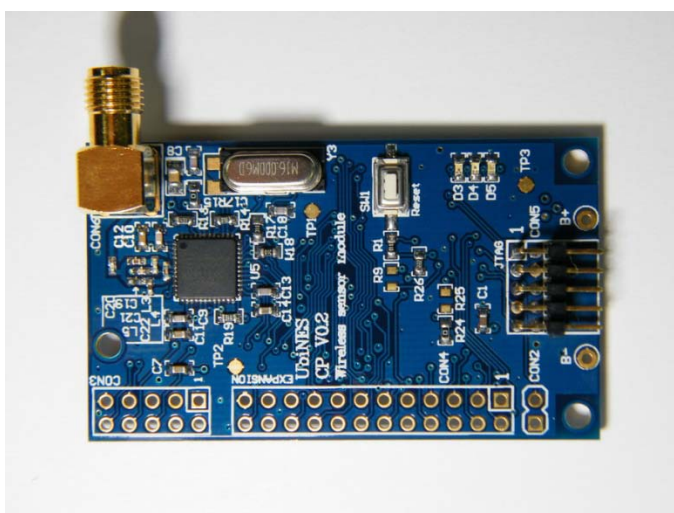
โพรโทคอล DYnamic Manet On-demand (DYMO) [19] มีรูปแบบการทำงานเป็นแบบโพรโทคอล Reactive โหนดจะทำการหาเส้นทางก็ต่อเมื่อต้องการติดต่อกับโหนดปลายทางเท่านั้น เพื่อให้เกิดการส่งข้อมูลในเครือข่ายให้น้อยลง ช่วยในการประหยัดพลังงานและแบนด์วิดท์ของเครือข่าย โพรโทคอล DYMO ได้รับการพัฒนาอยู่บนระบบปฏิบัติการ TinyOS และเป็นโมดูลที่ถูกนำไปรวมเป็นส่วนหนึ่งของ TinyOS2.0 สามารถเรียกใช้งานได้ทันที

การทำงานของโพรโทคอล DYMO เริ่มจากโหนดที่ต้องการเส้นทางในการติดต่อสื่อสารจะส่งแพ็กเก็ต Route Request (RREQ) ให้กับโหนดในเครือข่าย เมื่อแพ็กเก็ต RREQ ถึงโหนดปลายทางแล้วจะทำการส่งแพ็กเก็ต Route Reply (RREP) ตอบรับกลับไปให้โหนดที่ส่งแพ็กเก็ต RREQ มาให้

เมื่อโหนดได้รับแพ็กเก็ต RREQ หรือ RREP จะทำการบันทึกเซนเซอร์โหนดต้นทางกับโหนดที่ส่งแพ็กเก็ตมาให้เพื่อเลือกเป็นเส้นทางในการติดต่อสื่อสารกัน เส้นทางในการติดต่อที่ไม่มีการใช้งานเป็นเวลานาน จะถูกลบทิ้ง เมื่อโหนดต้นทางต้องส่งผ่านเส้นทางนั้น โหนดจะส่งแพ็กเก็ต Route Error (RRER) ให้กับโหนดต้นทางรับทราบ ในกระบวนการซ่อมแซมเส้นทางของโปรโตคอลจะใช้ข้อมูลซึ่งข้อมูลนั้นคือ จำนวน hop หมายเลขของแพ็กเก็ต เพื่อทำการซ่อมแซมเส้นทาง

2.4 คุณสมบัติของโหนด Unode

Unode ดังภาพประกอบ 2-4 เป็นอุปกรณ์สมองกลฝังตัวแบบไร้สายด้วยมาตรฐาน IEEE 802.15.4 ที่ใช้กำลังไฟฟ้าต่ำถูกพัฒนาขึ้นในห้องปฏิบัติการ Ubiquitous Network Embedded System (UbiNES) ภาควิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยสงขลานครินทร์ โดยใช้มาตรฐานของโหนดแบบเดียวกับ Tmote Sky (Telosb) ที่พัฒนาขึ้น ณ มหาวิทยาลัย Berkeley ประเทศสหรัฐอเมริกา ซึ่ง Unode สามารถนำไปใช้ตรวจวัดข้อมูลทางกายภาพได้หลากหลายขึ้นกับตัวตรวจวัดที่นำไปใช้งานควบคู่ไปกับ Unode โดยที่ตัวตรวจวัดสามารถเชื่อมต่อเข้ากับ Unode ผ่านทางพอร์ตมาตรฐานได้แก่ ADC/DAC I2C SPI UART หรือเป็นพอร์ตทั่วไป (GPIO)



ภาพประกอบ 2-4 Unode

ระบบปฏิบัติการที่ถูกนำมาใช้งานบน Unode คือระบบปฏิบัติการแบบ Open Source ที่ชื่อว่า TinyOS ทำงานอยู่บนไมโครคอนโทรลเลอร์กำลังงานต่ำของบริษัท TI รุ่น MSP430F1611 มีความเร็วการทำงานที่ 8 MHz มีหน่วยความจำในไมโครคอนโทรลเลอร์เพื่อใช้ในการเก็บ โปรแกรมและประมวลผลสองชนิดคือ หน่วยความจำแบบ RAM ขนาด 10KB และ

หน่วยความจำแบบ Flash ขนาด 48KB Unode ใช้โมดูลของการรับส่งข้อมูลแบบไร้สายผ่านคลื่นความถี่วิทยุย่าน 2.4 GHz ของบริษัท TI ตระกูล ChipCon รุ่น CC2420 และ Unode สามารถใช้พลังงานจากแบตเตอรี่ขนาด AA จำนวน 2 ก้อน

2.5 ระบบปฏิบัติการ TinyOS 2.0

TinyOS 2.0 [1] เป็นระบบปฏิบัติการแบบเปิด (open-source operating system) ที่ได้รับการออกแบบให้มีสถาปัตยกรรมสำหรับเครือข่ายเซนเซอร์ไร้สายโดยเฉพาะ มีคุณสมบัติดังนี้

- สถาปัตยกรรมของระบบปฏิบัติการเป็นแบบโมดูล ซึ่งช่วยให้การพัฒนาการเขียนโปรแกรมได้รวดเร็วและง่าย ขณะที่โปรแกรมมีขนาดเล็กเหมาะสมในกรณีที่มีข้อจำกัดเรื่องหน่วยความจำของเครือข่ายเซนเซอร์ไร้สาย
- TinyOS มีคลังของโมดูลมากมายเช่น มาตรฐานการสื่อสารของเครือข่าย (network protocol) บริการเช่น ด้านการกระจายตัวของเครือข่าย (distributed services) โปรแกรมขับเครื่องตัวตรวจวัด (sensor driver) เครื่องมือในการเก็บข้อมูล ฯลฯ ซึ่งสามารถนำไปประยุกต์สร้างโปรแกรมสำหรับงานต่างๆ ได้
- สถาปัตยกรรมแบบขับเคลื่อนด้วยเหตุการณ์ (event-driven) ทำให้การทำงานของโหนดจะทำงานเมื่อมีการรับส่งข้อมูลเข้ามาจึงจะทำงานทำให้ระบบประหยัดพลังงานและทำให้การจัดการยืดหยุ่นเหมาะสำหรับการสื่อสารแบบไร้สายในสภาพแวดล้อมจริงที่คาดการณ์ไม่ได้

2.5.1 ข้อแตกต่างของสถาปัตยกรรม TinyOS 1.1 และ TinyOS 2.0

การทำงานของสถาปัตยกรรม TinyOS 2.0 มีความแตกต่างจากการทำงานของสถาปัตยกรรม TinyOS 1.1 ในส่วนของการจัดการคิวงานใน TinyOS 1.1 หาก task ที่ทำงานอยู่เกิดการล้มเหลว ทำให้ระบบทำงานต่อไม่ได้ต้องรอให้ task นั้นเสร็จสิ้นก่อน แต่ใน TinyOS 2.0 ได้พัฒนาแก้ไขในส่วนนี้โดย task แต่ละ task มีช่องการทำงานแยกกัน ทำให้การจัดการคิวงานดีขึ้นในกรณีที่ task ล้มเหลวได้ ส่วนที่ TinyOS 2.0 กับ TinyOS 1.1 แตกต่างอีกคือ ลำดับในการบูตการทำของตัวโหนดเอง

2.5.2 การทำงานแบบพร้อมกัน (Concurrency)

โปรเซสของระบบปฏิบัติการ TinyOS นั้นแบ่งออกได้เป็น 2 ชนิดตามการทำงานของโปรเซสนั้นคือ task และ event โปรเซสชนิด task มีหน้าที่ทำงานต่างๆ ของโปรเซสตั้งแต่คำสั่ง

แรกจนถึงคำสั่งสุดท้ายของโปรเซสนั้น โปรเซสชนิด event ถูกสั่งงานจากโมดูลอื่นๆ หรือเกิดการร้องขอขึ้น

การทำงานของ task จะทำงานของโปรเซสตามลำดับการทำงาน หากเกิด event ขึ้นมาจะทำให้ task ที่ทำงานอยู่ต้องหยุดการทำงานชั่วคราว เพื่อทำงานของ task ที่เกิดจาก event จนกระทั่ง task ที่เกิดจาก event ทำงานเสร็จ task ที่หยุดการทำงานจะกลับมาทำงานต่อจากจุดที่หยุดการทำงานต่อไป

2.6 ภาษา NesC

NesC เป็นภาษาที่ใช้พัฒนาบนระบบปฏิบัติการ TinyOS และเป็นภาษาใหม่ที่มีโครงสร้างแบบคอมโพเนนต์ (structured component-based) ภาษา NesC ถูกพัฒนาขึ้นสำหรับระบบฝังตัวเช่น เครื่องข่ายเซนเซอร์ไร้สาย ภาษา NesC มีลักษณะโครงสร้าง (syntax) แบบภาษา C และสนับสนุนการทำงานแบบพร้อมกัน (concurrency) ของ TinyOS

ในระบบปฏิบัติการ TinyOS กำหนดกรอบของระบบปฏิบัติการอย่างชัดเจนทำให้ภาษา NesC มีการกำหนดคอมโพเนนต์อย่างชัดเจน โดยมีการเชื่อมต่อแบบสองทิศทาง (bidirectional) นอกจากนี้ภาษา NesC ยังกำหนดการทำงานแบบพร้อมกันโดยใช้ task และ event

2.6.1 คอมโพเนนต์ (Component)

คอมโพเนนต์ประกอบไปด้วย อินเตอร์เฟซทำหน้าที่เชื่อมต่อกับคอมโพเนนต์อื่นๆ ที่มีหน้าเฉพาะงานเช่น การสื่อสารและการติดต่อเซนเซอร์ เป็นต้น อินเตอร์เฟซที่ใช้งานในคอมโพเนนต์จะมีการกำหนดคุณสมบัติของแต่ละคอมโพเนนต์ว่าจะใช้อินเตอร์เฟซแบบให้บริการ (provides) หรืออินเตอร์เฟซแบบเรียกใช้ (uses) คอมโพเนนต์ในภาษา NesC มีการทำงานอยู่ 2 รูปแบบ ดังนี้

1. module ประกอบด้วย โค้ดการทำงาน
2. configurations บอกถึงการเชื่อมต่อกันของคอมโพเนนต์ ผ่านอินเตอร์เฟซ ในการเชื่อมต่อแบบนี้เรียกว่า “wiring”

```
module TimerM{
    provides interface Timer;
    uses interface Alarm;
}
```

ภาพประกอบ 2-5 คอมโพเนนต์แบบ module ชื่อ TimerM

ในภาพประกอบ 2-5 แสดงให้เห็นถึงตัวอย่างการประกาศคอมโพเนนต์แบบ module ที่ชื่อว่า TimerM มีการเรียกใช้งานอินเตอร์เฟซทั้ง 2 แบบ ได้แก่อินเตอร์เฟซ Timer ทำงานในแบบให้บริการ และอินเตอร์เฟซ Alarm ทำงานในแบบเรียกใช้

```
configuration BlinkM{
    provides interface SplitControl;
}
implementation
{
    components MainC, BlinkC, LedsC;

    BlinkC->MainC.Boot;
    BlinkC.Leds->LedsC;
}
```

ภาพประกอบ 2-6 คอมโพเนนต์แบบ configuration ชื่อ BlinkM

ในภาพประกอบ 2-6 แสดงให้เห็นถึงตัวอย่างการประกาศคอมโพเนนต์แบบ configuration ที่บอกการเชื่อมต่อของคอมโพเนนต์ ในส่วนของ implementation มีการอธิบายการเชื่อมต่อของคอมโพเนนต์ คอมโพเนนต์แบบ configuration ชื่อว่า BlinkM มีการเชื่อมต่อกันของอินเตอร์เฟซภายในคอมโพเนนต์คือ คอมโพเนนต์ BlinkC เชื่อมต่อกับคอมโพเนนต์ MainC ผ่านอินเตอร์เฟซ Boot และคอมโพเนนต์ BlinkC เชื่อมต่อกับคอมโพเนนต์ LedsC ผ่านอินเตอร์เฟซ Leds

2.6.2 อินเตอร์เฟซ (interface)

อินเตอร์เฟซในภาษา NesC เป็นแบบสองทิศทางคือ เป็นการกำหนดการกระทำระหว่างสองคอมโพเนนต์ ในคอมโพเนนต์จะมีการเรียกใช้อินเตอร์เฟซด้วยกัน 2 แบบ คือ อินเตอร์เฟซแบบให้บริการ (provides) และอินเตอร์เฟซแบบเรียกใช้ (uses) ในแต่ละอินเตอร์เฟซมีการกำหนดฟังก์ชันอยู่ด้วยกัน 2 ชนิดคือ แบบ events และแบบ command

```
interface Timer<tag> {
    command void startOneShot(uint32_t period);
    command void startPeriodic(uint32_t period);
    event void fired();
}
```

ภาพประกอบ 2-7 อินเตอร์เฟซ Timer

ในภาพประกอบ 2-7 แสดงให้เห็นถึงตัวอย่างการประกาศอินเตอร์เฟซ ในส่วนของอินเตอร์เฟซแต่ละอินเตอร์เฟซจะมีฟังก์ชันการทำงาน 2 แบบคือ command จะทำงานเมื่อเรียกใช้งานและ event จะทำงานเมื่อเกิดเหตุการณ์ขึ้นมา ระบบจะทำการเรียกใช้งาน ใน

ภาพประกอบ 2-7 อินเทอร์เฟซ Timer มีฟังก์ชัน startOneShot (period) กับฟังก์ชัน startPeriodic (period) ทำงานแบบ command และมีฟังก์ชัน fired() ทำงานแบบ event จะเกิดการทำงานเมื่อเรียกใช้งาน

บทที่ 3

การออกแบบโพรโทคอลค้นหาเส้นทาง

ในบทนี้จะกล่าวถึงการออกแบบโพรโทคอลค้นหาเส้นทางที่ได้พัฒนาขึ้นสำหรับเครือข่ายเซนเซอร์ไร้สายเพื่อให้สามารถนำไปประยุกต์ใช้งานได้จริงบนโหนดที่พัฒนาขึ้นในห้องปฏิบัติการที่ชื่อ Unode การพัฒนาโพรโทคอลจะทำบนระบบปฏิบัติการ TinyOS โดยที่เนื้อหาในบทนี้จะกล่าวถึงภาพรวมของระบบ รูปแบบของแพ็กเก็ต การทำงานของโพรโทคอลประกอบด้วย 3 ส่วนคือ การกำหนดค่าเริ่มต้น (initial setup) การรับส่งข้อมูลหรือเก็บรวบรวมข้อมูลจากตัวตรวจวัด (data collection) และการบำรุงรักษาเส้นทาง (maintenance) และการพัฒนาโพรโทคอลบนระบบปฏิบัติการ TinyOS

3.1 ภาพรวมของโพรโทคอล

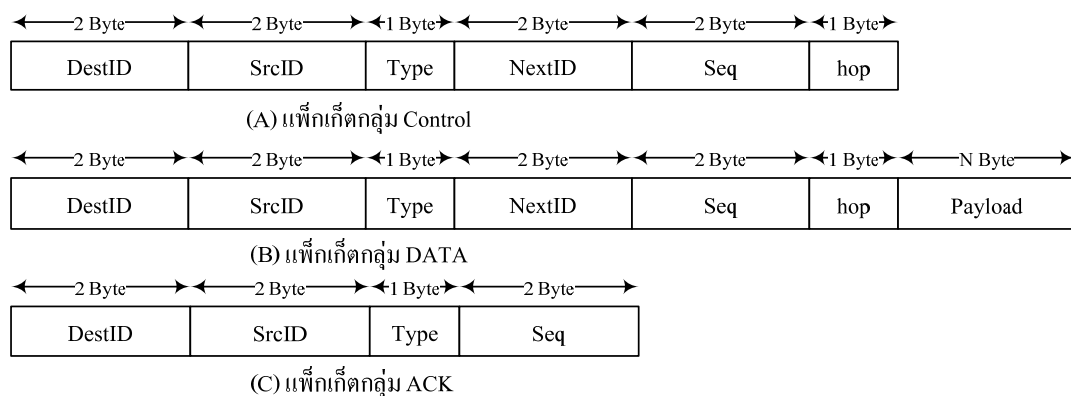
โพรโทคอลที่ได้ออกแบบและพัฒนา มีความสามารถในการค้นหาเส้นทาง การติดต่อสื่อสาร แล้วทำการรับส่งข้อมูล ตรวจสอบเส้นทางและบำรุงรักษาเส้นทางในการติดต่อสื่อสารให้สามารถใช้งานได้ โพรโทคอลที่ออกแบบจะมีรูปแบบของแพ็กเก็ตอยู่ด้วยกัน 4 รูปแบบคือ แพ็กเก็ต Route Request (RREQ) แพ็กเก็ต Route Reply (RREP) แพ็กเก็ต DATA และแพ็กเก็ต ACK ใช้ในการตอบรับการได้รับแพ็กเก็ต DATA

การทำงานของโพรโทคอลเริ่มจากโหนดต้นทางที่สถานีฐาน จะทำการส่งแพ็กเก็ต RREQ ไปให้โหนดในเครือข่าย เพื่อเริ่มกระบวนการค้นหาเส้นทาง เมื่อโหนดใดได้รับแพ็กเก็ต RREQ แล้วจะทำการส่งแพ็กเก็ต RREP เพื่อตอบกลับ หลังจากนั้น โหนดแต่ละโหนดได้รับแพ็กเก็ต RREP แล้วจะจดจำเฉพาะโหนดเพื่อนบ้านไว้ในหน่วยความจำของโหนด โหนดแต่ละโหนดจะทราบถึงเส้นทางติดต่อถึงสถานีฐานผ่านโหนดที่ส่งแพ็กเก็ต RREP ตอบกลับไป ในการเลือกเส้นทางนั้น จะเลือกจากเส้นทางติดต่อสื่อสารถึงสถานีฐานจากความเร็วในการได้รับแพ็กเก็ต RREQ เข้ามา

3.2 รูปแบบของแพ็กเก็ต

ตามที่ได้กล่าวไว้ในข้างต้น แพ็กเก็ตในโพรโทคอลมี 4 รูปแบบ แต่สามารถจัดหมวดหมู่ได้ 3 กลุ่มได้แก่ กลุ่มแพ็กเก็ต Control มีขนาด 10 ไบต์ กลุ่มแพ็กเก็ต DATA มีขนาดขึ้นอยู่กับปริมาณข้อมูลในฟิลด์ข้อมูล Payload และกลุ่มแพ็กเก็ต ACK มีขนาด 7 ไบต์ ดังภาพประกอบ 3-1 ในแต่ละแพ็กเก็ตจะประกอบด้วยฟิลด์ข้อมูล ซึ่งรูปแบบการเรียงตัวของฟิลด์

ข้อมูลจะแตกต่างกันตามกลุ่มที่จำแนกไว้ ความหมายและการทำงานของแต่ละฟิลด์ได้อธิบายไว้ในตารางที่ 3-1



ภาพประกอบ 3-1 รูปแบบแพ็กเก็ตในโปรโตคอล

ตารางที่ 3-1 การทำงานของฟิลด์ข้อมูล

ฟิลด์ข้อมูล	ขนาด (Byte)	การทำงาน
Type	1	ระบุชนิดของแพ็กเก็ต 0x01 : RREQ 0x02 : RREP 0x03 : DATA 0x04 : ACK
DestID	2	หมายเลขโหนดปลายทาง (กำหนด 0xFFFF ในกรณีส่งแบบกระจาย (broadcast))
SrcID	2	หมายเลขโหนดต้นทาง
NextID	2	หมายเลขโหนดถัดไป
seq	2	ลำดับหมายเลขแพ็กเก็ต
hop	1	จำนวนที่แพ็กเก็ตถูกส่งต่อ (ค่าโดยปริยายเท่ากับ 5)
Payload	N	ข้อมูลจากตัวตรวจวัด (สูงสุด 20 ไบต์)

แพ็กเก็ตในแต่ละกลุ่มจะมีฟิลด์ข้อมูลที่เหมือนกันคือ ฟิลด์ข้อมูล “Type” บอกชนิดการทำงานของแพ็กเก็ตซึ่งมีขนาด 1 ไบต์ ฟิลด์ข้อมูล “DestID” บอกถึงหมายเลขโหนดปลายทาง แต่ถ้ามีค่า 0xFFFF หมายถึงมีการส่งแบบกระจาย มีขนาด 2 ไบต์ ฟิลด์ข้อมูล “SrcID”

บอกถึงหมายเลขโหนดต้นทาง มีขนาด 2 ไบต์ และฟิลด์ข้อมูล “seq” บอกถึงลำดับของหมายเลขแพ็กเก็ต แต่แพ็กเก็ตแต่ละกลุ่มมีฟิลด์ข้อมูลในการทำงานที่ต่างกันดังนี้

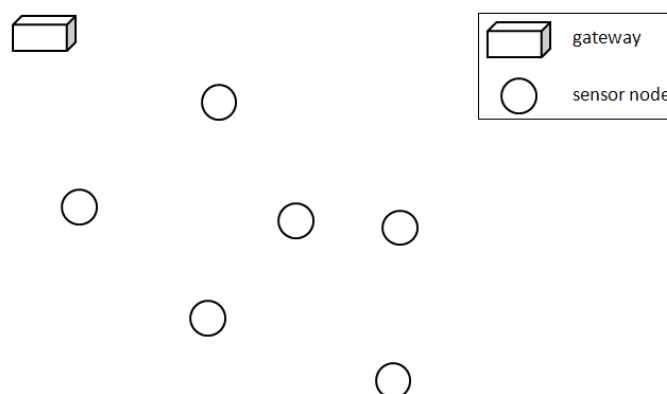
แพ็กเก็ตกลุ่ม CONTROL ในภาพประกอบ 3-1 (A) มีหน้าที่ทำงาน 2 หน้าที่คือ Route Requests (RREQ) ทำหน้าที่ค้นหาเส้นทาง และ Route Reply (RREP) ทำหน้าที่ตอบรับการค้นหาเส้นทาง โดยพิจารณาจากฟิลด์ข้อมูล Type เพื่อบอกถึงการทำงานของแพ็กเก็ต ฟิลด์ข้อมูล Type มีค่าเท่ากับ 0x01 หมายถึงแพ็กเก็ต RREQ แต่ถ้าฟิลด์ข้อมูล Type มีค่าเท่ากับ 0x02 หมายถึงแพ็กเก็ต RREP

แพ็กเก็ตกลุ่ม DATA ในภาพประกอบ 3-1 (B) มีหน้าที่ส่งข้อมูลออกจากโหนดให้กับโหนดรอบข้าง หรือโหนดปลายทาง รูปแบบแพ็กเก็ต DATA มีฟิลด์ข้อมูล Payload เพิ่มขึ้น โดยขนาดของฟิลด์ข้อมูลนี้ขึ้นอยู่กับขนาดของข้อมูล สามารถปรับเปลี่ยนได้ ขนาดข้อมูลมีได้มากที่สุดที่ 20 ไบต์

แพ็กเก็ตกลุ่ม ACK ในภาพประกอบ 3-1 (C) มีหน้าที่ในการตอบรับให้โหนดเพื่อเป็นการยืนยันว่าได้รับแพ็กเก็ต DATA ที่ส่งมาจากโหนดรอบข้าง

3.3 การทำงานของโปรโตคอล

โครงสร้างเครือข่ายเซนเซอร์ไร้สาย โหนดที่อยู่ในเครือข่ายถูกแบ่งตามหน้าที่ในการทำงานออกเป็น 2 ชนิดคือ โหนดที่ทำหน้าที่เป็นสถานีฐาน หรือ gateway และ โหนดที่ทำหน้าที่ตรวจวัด (sensor) ดังในภาพประกอบ 3-2 โดยที่โหนด gateway จะทำหน้าที่ติดต่อสื่อสารกับโหนดในเครือข่ายเพื่อรับข้อมูลที่ได้จากการตรวจวัด

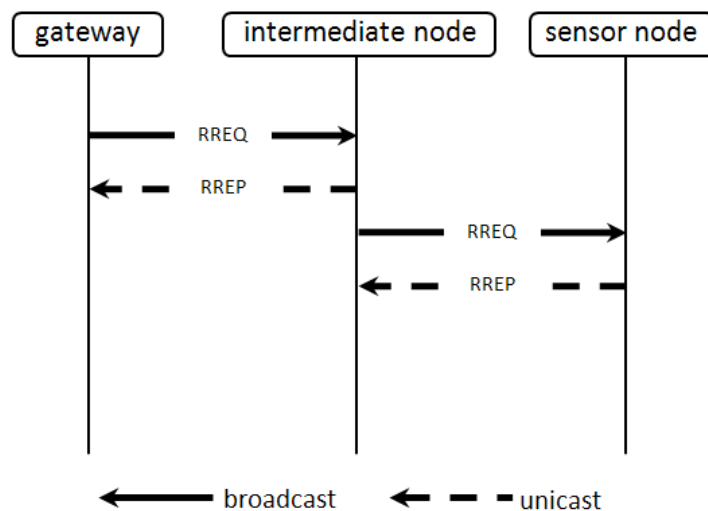


ภาพประกอบ 3-2 โครงสร้างเครือข่ายเซนเซอร์ไร้สาย

สำหรับโพรโทคอลค้นหาเส้นทางที่ได้พัฒนาขึ้นจะแบ่งการทำงานเป็น 3 ส่วน คือ การกำหนดค่าเริ่มต้น (initial setup) การรับส่งข้อมูล (data collection) และการบำรุงรักษาเส้นทาง (maintenance)

3.3.1 การกำหนดค่าเริ่มต้น (initial setup)

การกำหนดค่าเริ่มต้นของโพรโทคอลสามารถอธิบายได้ดังภาพประกอบ 3-3 โหนด gateway ส่งกระจายแพ็กเก็ต RREQ ให้กับโหนดในเครือข่าย เมื่อโหนดได้รับแพ็กเก็ต RREQ โหนดจะทำการบันทึกหมายเลขลำดับแพ็กเก็ตไว้เพื่อนำค่าหมายเลขลำดับแพ็กเก็ตเข้ามาช่วยในการตรวจสอบแพ็กเก็ตข้อมูลซ้ำซ้อนหรือไม่ ถ้าแพ็กเก็ต RREQ นั้นเคยได้รับแล้วจะทำการรีเซ็ตแพ็กเก็ต RREQ ทิ้ง ถ้าไม่เคยได้รับแพ็กเก็ต RREQ โหนดบันทึกข้อมูลของโหนดต้นทางไว้ และทำการส่งแพ็กเก็ต RREP ให้กับโหนดต้นทางที่ส่งแพ็กเก็ต RREQ มาให้ และทำการส่งกระจายแพ็กเก็ต RREQ ให้กับโหนดรอบข้างโดยมีการแก้ไขแพ็กเก็ตในฟิลด์ข้อมูล hop จะทำการลดค่าลง 1 เพื่อช่วยป้องกันแพ็กเก็ตส่งซ้ำอยู่ในเครือข่าย



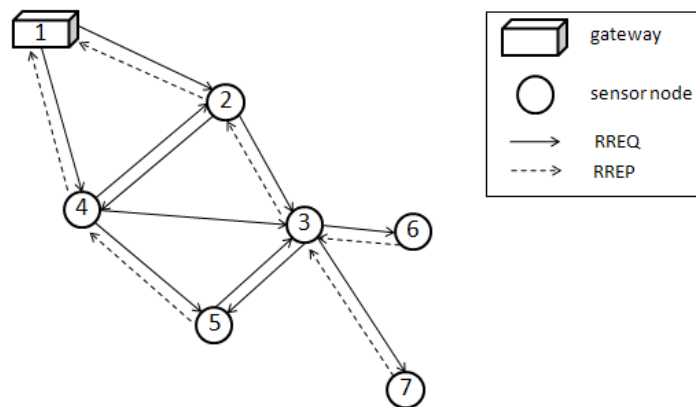
ภาพประกอบ 3-3 รูปแบบการเริ่มต้นการค้นหาเส้นทาง

การที่โหนดได้รับแพ็กเก็ต RREP ตอบรับกลับมาแสดงว่าได้เลือกโหนดนี้เป็นเส้นทางในการติดต่อถึงโหนด gateway ดังนั้นจะมีการบันทึกหมายเลขของโหนดรอบข้างที่ตอบ RREP กลับมาไว้ในหน่วยความจำของโหนดนั้นๆ

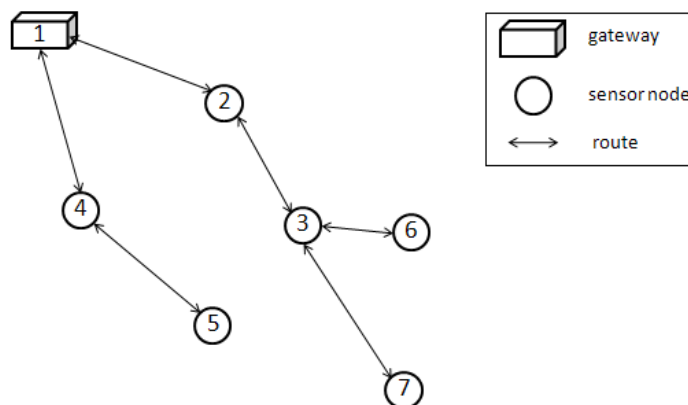
ตัวอย่างในภาพประกอบ 3-4 โหนดหมายเลข 2 ได้รับแพ็กเก็ต RREQ จากโหนดหมายเลข 1 เมื่อโหนด 2 ส่งแพ็กเก็ต RREP ให้กลับโหนดหมายเลข 1 แสดงว่าโหนดหมายเลข 2 เลือกโหนดหมายเลข 1 เป็นเส้นทางในการติดต่อถึงโหนด gateway แล้วจึงทำการกระจายแพ็กเก็ต

RREQ ให้กับโหนดรอบข้างในเครือข่าย เมื่อโหนดหมายเลข 3 ส่งแพ็กเก็ต RREP ให้กลับโหนด 2 แสดงว่าโหนดหมายเลข 3 เลือกโหนดหมายเลข 2 เป็นเส้นทางในการติดต่อถึงโหนด gateway

ในภาพประกอบ 3-5 โหนดในเครือข่ายได้สร้างเส้นทางในการเชื่อมต่อกันถึงโหนด gateway ได้โดยมีเส้นทางในเครือข่ายดังนี้ 1-2-3-6, 1-2-3-7 และ 1-4-5



ภาพประกอบ 3-4 โหนด gateway ทำการกระจายแพ็กเก็ตให้กับโหนดรอบข้าง

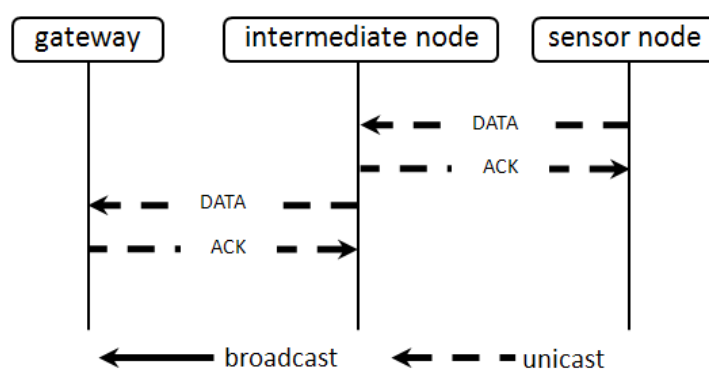


ภาพประกอบ 3-5 โหนดมีเส้นทางติดต่อกับโหนด gateway

3.3.2 การรับส่งข้อมูล (data collection)

การรับส่งข้อมูลจะเริ่มทำงานหลังจากการกำหนดค่าเริ่มต้นสำเร็จ เมื่อโหนด gateway ได้รับทราบเส้นทางการเชื่อมต่อของเครือข่ายทั้งหมดแล้วดังในภาพประกอบ 3-5 ขั้นตอนต่อไปคือการส่งข้อมูลตามเส้นทางที่ได้เลือกกลับมาถึงโหนด gateway โดยจะเริ่มต้นจากโหนดที่เก็บข้อมูลจากตัวตรวจวัด (sensor) โหนดทำการสร้างแพ็กเก็ต DATA เพื่อส่งข้อมูลไปให้กับโหนด

gateway ในแพ็กเก็ต DATA จะทำการระบุหมายเลขโหนดที่ช่วยในการส่งให้กับโหนด gateway จากการกำหนดค่าเริ่มต้นของโพรโทคอลจากภาพประกอบ 3-6 นี้คือ โหนดกลาง (intermediate node) หลังจากที่โหนดกลางได้รับแพ็กเก็ต DATA โหนดกลางจะส่งแพ็กเก็ต ACK ให้กับโหนดที่ส่งแพ็กเก็ตเข้ามา เพื่อเป็นการยืนยันว่าได้รับแพ็กเก็ต DATA แล้ว หลังจากนั้นโหนดกลางจะทำการส่งแพ็กเก็ต DATA ให้กับโหนด gateway เมื่อโหนด gateway ได้รับแพ็กเก็ต DATA โหนด gateway จะส่งแพ็กเก็ต ACK ให้กับโหนดกลางเพื่อเป็นการยืนยันการได้รับแพ็กเก็ต



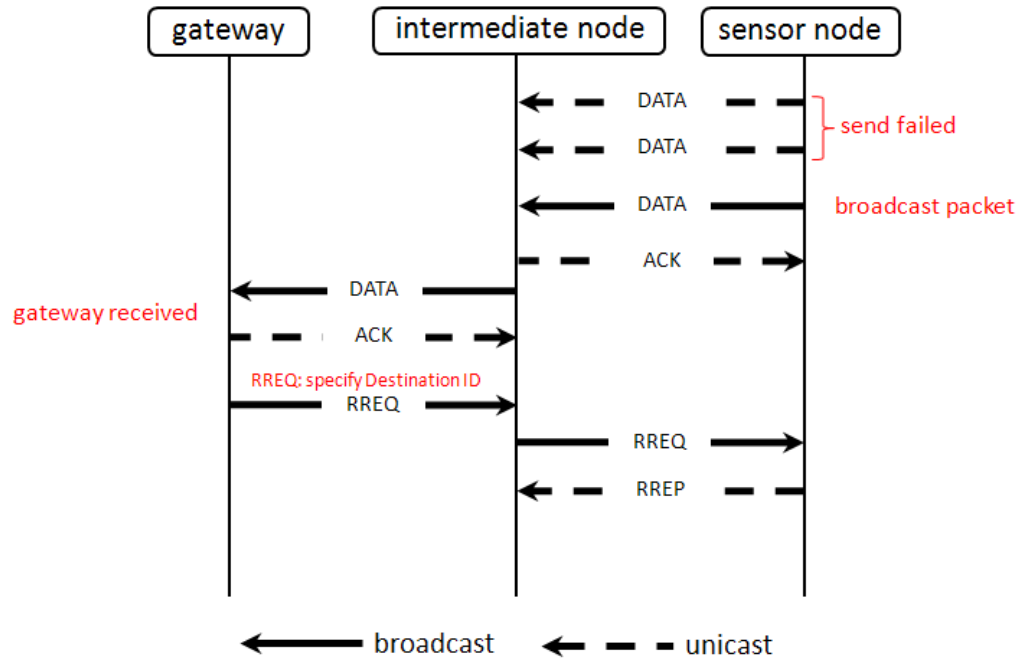
ภาพประกอบ 3-6 รูปแบบการรับส่งข้อมูลของโหนด

3.3.3 การบำรุงรักษาเส้นทาง (maintenance)

ในขั้นตอนนี้จะเป็นการรักษาเส้นทางเชื่อมต่อระหว่างโหนด gateway กับโหนดต่างๆในเครือข่าย โหนด gateway จะทำการตรวจสอบเส้นทางว่าเกิดความเสียหายหรือไม่ โดยสามารถตรวจสอบความเสียหายของเส้นทางได้จากช่วงเวลาที่โหนดทำการส่งแพ็กเก็ตข้อมูลของตัวตรวจวัดกลับมายังโหนด gateway โดยทำการตรวจสอบจากแพ็กเก็ต DATA ที่โหนดส่งเข้ามาดังภาพประกอบ 3-6

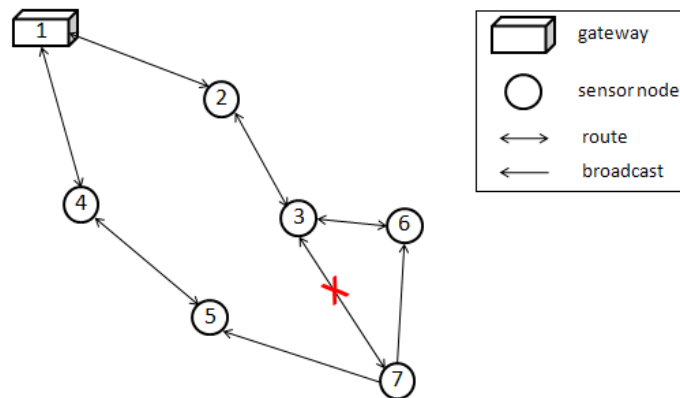
หากข้อมูลที่โหนดส่งถึงโหนด gateway ได้รับแพ็กเก็ต ACK ตอบกลับมาแสดงว่าการส่งข้อมูลสมบูรณ์ แต่ถ้าเกิดการส่งไม่สมบูรณ์ดังภาพประกอบ 3-7 โหนดจะส่งแพ็กเก็ต DATA แล้วไม่ได้รับแพ็กเก็ต ACK กลับมารอบที่ 1 โหนดจะส่งแพ็กเก็ตซ้ำรอบที่ 2 แต่ยังไม่ได้รับแพ็กเก็ต ACK กลับมา โหนดทำการเปลี่ยนหมายเลขโหนดปลายทางเป็น 0xFFFF เพื่อส่งกระจาย (broadcast) ให้โหนดรอบข้าง (intermediate node) ช่วยส่งแพ็กเก็ต DATA ให้กับโหนด gateway เมื่อโหนดรอบข้างได้รับแพ็กเก็ต DATA จะส่งแพ็กเก็ต ACK ให้กับโหนดต้นทางรับทราบ และโหนดรอบข้างจะส่งแพ็กเก็ต DATA ที่ได้รับมาให้กับโหนด gateway เมื่อโหนด gateway พบว่าแพ็กเก็ต DATA ที่เข้ามามีหมายเลขโหนดปลายทาง 0xFFFF ทำให้รู้ว่าเส้นทางเชื่อมต่อของ

โหนดต้นทางมีปัญหา โหนด gateway จะส่งแพ็กเก็ต RREQ โดยมีการระบุหมายเลขโหนดต้นทางเป็นหมายเลขโหนดปลายทาง เมื่อโหนดได้รับแพ็กเก็ต RREQ ที่มีการระบุหมายเลขโหนดปลายทางเป็นหมายเลขของตัวเอง โหนดจะทำการส่งแพ็กเก็ต RREP ให้กับโหนดที่ส่งแพ็กเก็ต RREQ มาให้เพื่อเป็นการเลือกเส้นทางในการเชื่อมต่อกับโหนด gateway ได้



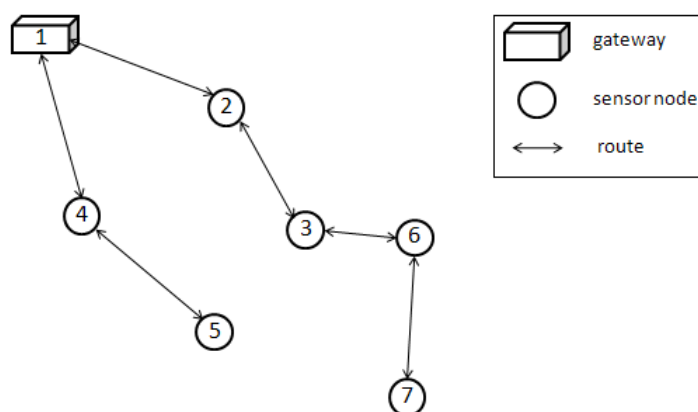
ภาพประกอบ 3-7 รูปแบบการบำรุงรักษาเส้นทางของโปรโตคอล

ตัวอย่างในภาพประกอบ 3-8 เส้นทางเดิมโหนดหมายเลข 7 เชื่อมต่อกับโหนดหมายเลข 3 เมื่อโหนดหมายเลข 7 ทำการส่งแพ็กเก็ต DATA ให้กับโหนดหมายเลข 3 แต่โหนดหมายเลข 7 ไม่ได้รับแพ็กเก็ต ACK กลับมาภายในเวลาที่กำหนดไว้ โหนดหมายเลข 7 จะทำการส่งแพ็กเก็ต DATA ซ้ำ แต่โหนดหมายเลข 7 ยังไม่ได้รับแพ็กเก็ต ACK จากโหนดหมายเลข 3 โหนดหมายเลข 7 ทำการเปลี่ยนหมายเลขโหนดปลายทางเป็น 0xFFFF ซึ่งเป็นรหัสบอกให้โหนด gateway ทราบว่าเกิดการเสียหายของเส้นทางในเครือข่าย



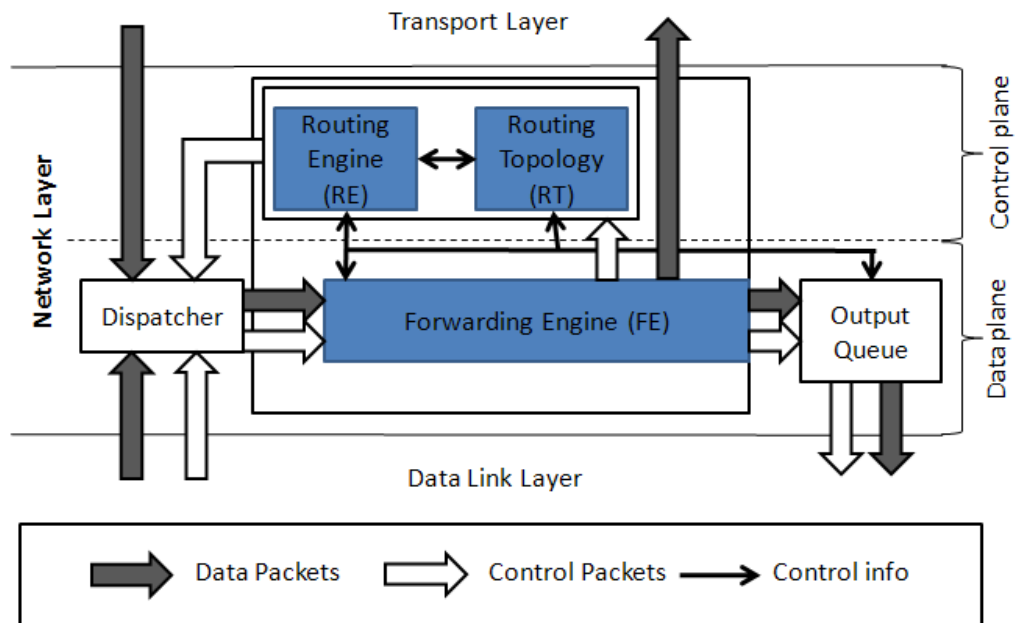
ภาพประกอบ 3-8 เส้นทางในเครือข่ายเกิดการเสียหาย

เมื่อโหนด gateway ได้รับแพ็กเก็ต DATA แล้วตรวจสอบว่าแพ็กเก็ตที่ได้รับมานั้นเป็นการส่งกระจายแพ็กเก็ตเข้ามา และหมายเลขโหนดปลายทางเป็น 0xFFFF ทำให้รู้ว่าเส้นทางในการเชื่อมต่อกันระหว่างโหนดกับโหนด gateway นั้นเสียหาย ดังนั้นโหนด gateway จะทำการแก้ไขเส้นทางที่เสียหายโดยการส่งแพ็กเก็ต RREQ ออกไปให้กับโหนดโดยทำการระบุ หมายเลขโหนดปลายทางเป็นโหนดหมายเลข 7 หลังจากที่โหนด gateway ส่งแพ็กเก็ต RREQ ให้กับโหนดหมายเลข 7 เมื่อโหนดหมายเลข 7 ได้รับจะส่งแพ็กเก็ต RREP ให้กับโหนดที่ส่งแพ็กเก็ต RREQ เพื่อเลือกโหนดนั้นเป็นเส้นทางในการเชื่อมต่อถึงโหนด gateway ทำให้เครือข่ายมีการซ่อมแซมเส้นทางบางส่วนทำให้เกิดการเชื่อมต่อใหม่อีกครั้งดังในภาพประกอบ 3-9 เส้นทางมีการเปลี่ยนแปลงไปจากเส้นทางเดิมคือ 1-2-3-7 เป็น 1-2-3-6-7



ภาพประกอบ 3-9 เส้นทางใหม่ในเครือข่ายในการติดต่อสื่อสาร

3.4 การพัฒนาโปรโตคอลบนระบบปฏิบัติการ TinyOS



ภาพประกอบ 3-10 โครงสร้างลำดับชั้น network บนระบบปฏิบัติการ TinyOS2.0

จากภาพประกอบ 3-10 [18] เป็นโครงสร้างลำดับชั้นของเครือข่ายบนระบบปฏิบัติการ TinyOS2.0 ที่เรียกว่า *modular network layer* ซึ่งออกแบบเน้นให้สะดวกง่ายต่อการสร้างและพัฒนาโปรโตคอลตัวใหม่ โดยที่สามารถให้มีการนำโค้ดกลับมาใช้ใหม่ (code reuse) และใช้โมดูลเดิมที่มีอยู่ในการพัฒนาต่อ ช่วยให้ลดระยะเวลาของการพัฒนาระบบ

สถาปัตยกรรมของเครือข่ายในชั้น Network เป็นส่วนการประมวลผลหลักของโปรโตคอล โดยทำหน้าที่สร้างและประมวลผลแพ็กเก็ตที่เข้ามาในโหนด รวมถึงการติดต่อสื่อสารกับโหนดในเครือข่าย และส่วนการจัดการข้อมูล ชั้น Network นี้แบ่งออกเป็น 2 ชั้นย่อยคือชั้นการควบคุม (control plane) และชั้นของการจัดการข้อมูล (data plane)

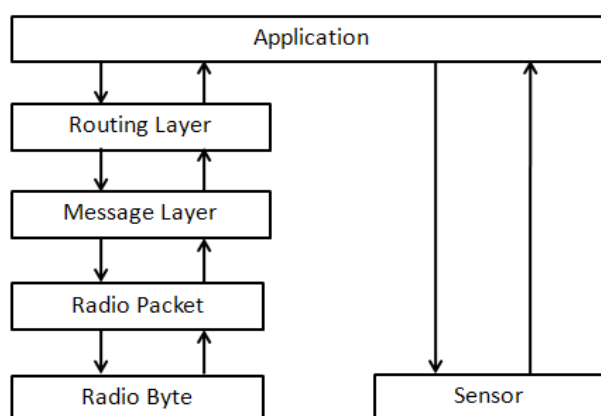
ชั้นย่อยที่ทำหน้าที่ควบคุมประกอบด้วย 2 โมดูลหลักที่สำคัญคือ โมดูล Routing Engine (RE) และ โมดูล Routing Topology (RT) ซึ่งโมดูล RE ทำงานตามอัลกอริทึมที่พัฒนาขึ้นเพื่อค้นหาเส้นทางร่วมกับ โมดูล Routing Topology (RT) ควบคุมการส่งผ่านข้อมูลของโมดูล Forwarding Engine (FE) และควบคุมโมดูล Output Queue

ชั้นย่อยของการจัดการข้อมูลเป็นการไหลของข้อมูลที่ผ่านมายังชั้น Network ประกอบด้วย 1) โมดูล Dispatcher ใช้ทำการตรวจสอบ header ของแพ็กเก็ตข้อมูลและแพ็กเก็ตควบคุมที่มาจาก Network ทั้งชั้น Transport และ Data Link ตามลำดับ เมื่อทำการเพิ่ม header เรียบร้อยแล้วจะจัดส่งเข้าสู่ 2) Forwarding Engine เป็นฟังก์ชันหนึ่งที่ทำให้บริการของโปรโตคอลทำ

หน้าที่ในการเพิ่ม header ก่อนที่จะส่งผ่านข้อมูลออกไป หรือทำหน้าที่ส่งข้อมูลขึ้นไปยังชั้น Transport ในกรณีนี้ แพ็กเก็ตถึงปลายทางเรียบร้อยแล้ว รายละเอียดการเพิ่มส่วน header นั้นได้มาจากส่วนของ RE 3) โมดูล Output Queue จะทำหน้าที่จัดลำดับการส่งแพ็กเก็ตจากโหนด ในที่นี้สามารถนำอัลกอริทึมการทำ scheduling มาประยุกต์ใช้ในโมดูลนี้

3.4.1 การสร้างโปรโตคอลบน TinyOS2.x

เป้าหมายของการออกแบบและพัฒนาโปรโตคอล เพื่อให้สามารถนำไปใช้งานบนโหนดได้จริง และให้สามารถประยุกต์เข้ากับงานควบคุมระบบฟาร์มเลี้ยงลูกกุ้ง การทำงานของโปรโตคอลในหัวข้อ 3.3 ได้ถูกนำมาพัฒนาสร้างด้วยภาษา NesC เพื่อที่จะใช้งานบนระบบปฏิบัติการ TinyOS



ภาพประกอบ 3-11 โครงสร้างการทำงานระบบปฏิบัติการ TinyOS

ในภาพประกอบ 3-11 แสดงโครงสร้างการพัฒนาซอฟต์แวร์โมดูลเพื่อทำงานบนระบบปฏิบัติการ TinyOS แบ่งการพัฒนาโปรแกรมหรือ Application บนระบบปฏิบัติการ TinyOS ออกเป็น 2 ส่วนสำคัญคือ 1) โมดูลที่เกี่ยวข้องกับการติดต่อกับฮาร์ดแวร์หรืออุปกรณ์ต่อพ่วงเช่น ตัวตรวจวัด เป็นต้น 2) โมดูลที่เกี่ยวข้องกับเครือข่ายตั้งแต่ระดับชั้น Physical (จัดการเกี่ยวกับ Radio Byte และ Radio Packet) ชั้น Data Link (จัดการเกี่ยวกับ Message Layer) และ Network (เกี่ยวกับการค้นหาเส้นทาง Routing Layer)

3.4.2 การทำงานในระดับชั้น Data Link

ชั้น Data Link ของสถาปัตยกรรม TinyOS2.0 จะใช้รูปแบบของข้อมูลชื่อ *Active Message (AM)* ใน Active Messages จะประกอบไปด้วยส่วนของ header ที่บ่งบอกถึงการทำงานของแพ็กเก็ตข้อมูลในการรับส่งกันระหว่างโหนด ในระดับชั้นนี้มีการใช้งานอินเตอร์เฟซ

(interface) AMPacket, AMSend, Receive และ SplitControl และมีใช้งานคอมโพเนนต์ (component) ActiveMessageC, AMSenderC และ AMReceiceC

การทำงานของอินเตอร์เฟซที่มีการใช้งานในระดับชั้น data link

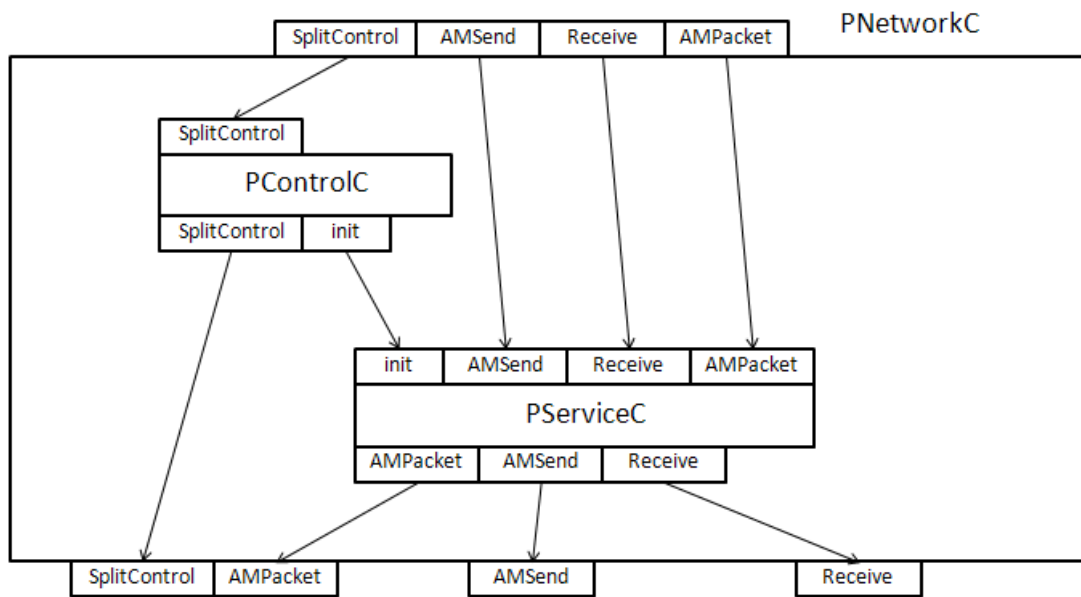
- อินเตอร์เฟซ AMPacket ช่วยในการเข้าถึงและแก้ไขฟิลด์ข้อมูล คือ หมายเลขโหนดปลายทาง หมายเลขโหนดต้นทาง และชนิดของแพ็กเก็ตข้อมูล
- อินเตอร์เฟซ AMSend ช่วยในการส่งแพ็กเก็ตให้กับโหนดรอบข้างหรือโหนด gateway
- อินเตอร์เฟซ Receive ช่วยในการรับแพ็กเก็ตที่ส่งมาจากโหนดรอบข้างหรือโหนด gateway
- อินเตอร์เฟซ SplitControl มีการทำงานอยู่ 2 ส่วนด้วยกันคือการสั่งเริ่มทำงาน (start) และการสั่งหยุดการทำงาน (stop) ของโมดูล

การทำงานของคอมโพเนนต์ที่มีการใช้งานในระดับชั้น data link

- คอมโพเนนต์ ActiveMessageC ทำหน้าที่จัดการแพ็กเก็ตข้อมูล
- คอมโพเนนต์ AMSenderC ทำหน้าที่ในการส่งแพ็กเก็ตข้อมูลให้โหนดรอบข้าง
- คอมโพเนนต์ AMReceiceC ทำหน้าที่ในการรับแพ็กเก็ตข้อมูลจากโหนดรอบข้าง

3.4.3 การทำงานในระดับชั้น Network Layer

ระบบปฏิบัติการ TinyOS ไม่ได้สร้างโมดูลอะไรไว้ในชั้นนี้ ซึ่งเปิดไว้ให้ผู้พัฒนาจะต้องดำเนินการสร้างอัลกอริทึมของการค้นหาเส้นทางและจัดการข้อมูลก่อนที่จะส่งขึ้นไปเครือข่ายชั้นอื่นๆต่อไป ทางผู้วิจัยจึงได้ออกแบบและพัฒนาโปรโทคอลขึ้นมาเพื่อทำงานในส่วนของโมดูล Routing Layer (ดังแสดงไว้ในภาพประกอบ 3-11) โดยได้ทำการออกแบบโมดูลที่จะเข้ามาช่วยในการทำงานควบคุมการรับและส่งข้อมูลของโหนด



ภาพประกอบ 3-12 โครงสร้างของคอมโพเนนต์ PNetwork

โครงสร้างของโพรโทคอลมีคอมโพเนนต์ PNetworkC ที่พัฒนาขึ้นมา โดยการทำงานจะมีการจัดแบ่งส่วนการทำงานอยู่ด้วยกัน 2 คอมโพเนนต์ คือ คอมโพเนนต์ PControlC และคอมโพเนนต์ PServiceC ดังในภาพประกอบ 3-12

การทำงานของคอมโพเนนต์ PControlC ทำหน้าที่สั่งให้คอมโพเนนต์ PServiceC เริ่มทำงาน เพื่อเตรียมพร้อมสำหรับรับและส่งแพ็กเก็ต

การทำงานของคอมโพเนนต์ PServiceC มีการทำงานอยู่ด้วยกันสองส่วนดังนี้

1. ส่วนของ Engine การทำงานในส่วนนี้จะทำหน้าที่ในการสร้างเส้นทางในการติดต่อสื่อสารกับโหนดรอบข้างโดยพิจารณาเส้นทางจากแพ็กเก็ต RREQ ที่ได้รับเข้ามาเพื่อเลือกเส้นทางในการติดต่อสื่อสาร และส่งแพ็กเก็ต RREP ให้กับโหนดที่ส่งแพ็กเก็ต RREQ เพื่อให้โหนดทราบถึงเส้นทางในการติดต่อสื่อสารในเครือข่าย ในส่วนนี้มีการตรวจแพ็กเก็ตข้อมูลที่เข้าจากหมายเลขลำดับของแพ็กเก็ตเพื่อป้องกันแพ็กเก็ตซ้ำ

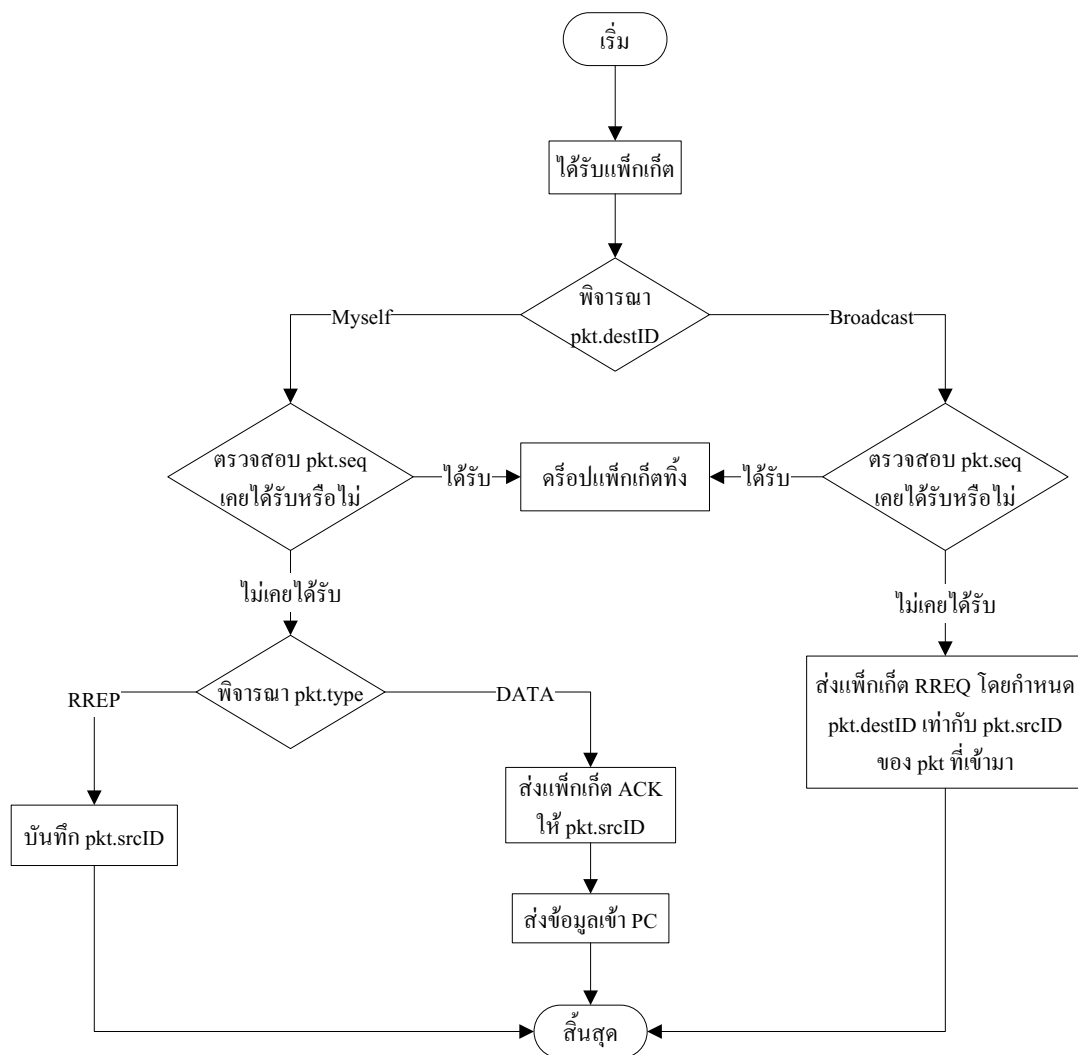
2. ส่วนของ Forwarding การทำงานในส่วนนี้จะทำการตรวจสอบแพ็กเก็ตข้อมูลที่เข้ามาในโหนด และทำการส่งแพ็กเก็ตข้อมูลออกไปโดยนำเส้นทางในการติดต่อสื่อสารมาจากส่วนของ Engine เพื่อทำการส่งข้อมูลไปให้กับโหนดปลายทาง

อัลกอริทึมของโพรโทคอลในตัวโหนด

กำหนดให้	Myself	หมายเลขโหนดตัวเอง
	ListSeq	หมายเลขแพ็กเก็ตที่เคยได้รับ
	NextID	หมายเลขโหนดถัดไป (Default Gateway)

Gateway	หมายเลขโหนด gateway
FSend	จำนวนครั้งที่ส่งไม่สำเร็จ
PC	คอมพิวเตอร์

อัลกอริทึมของโหนด gateway

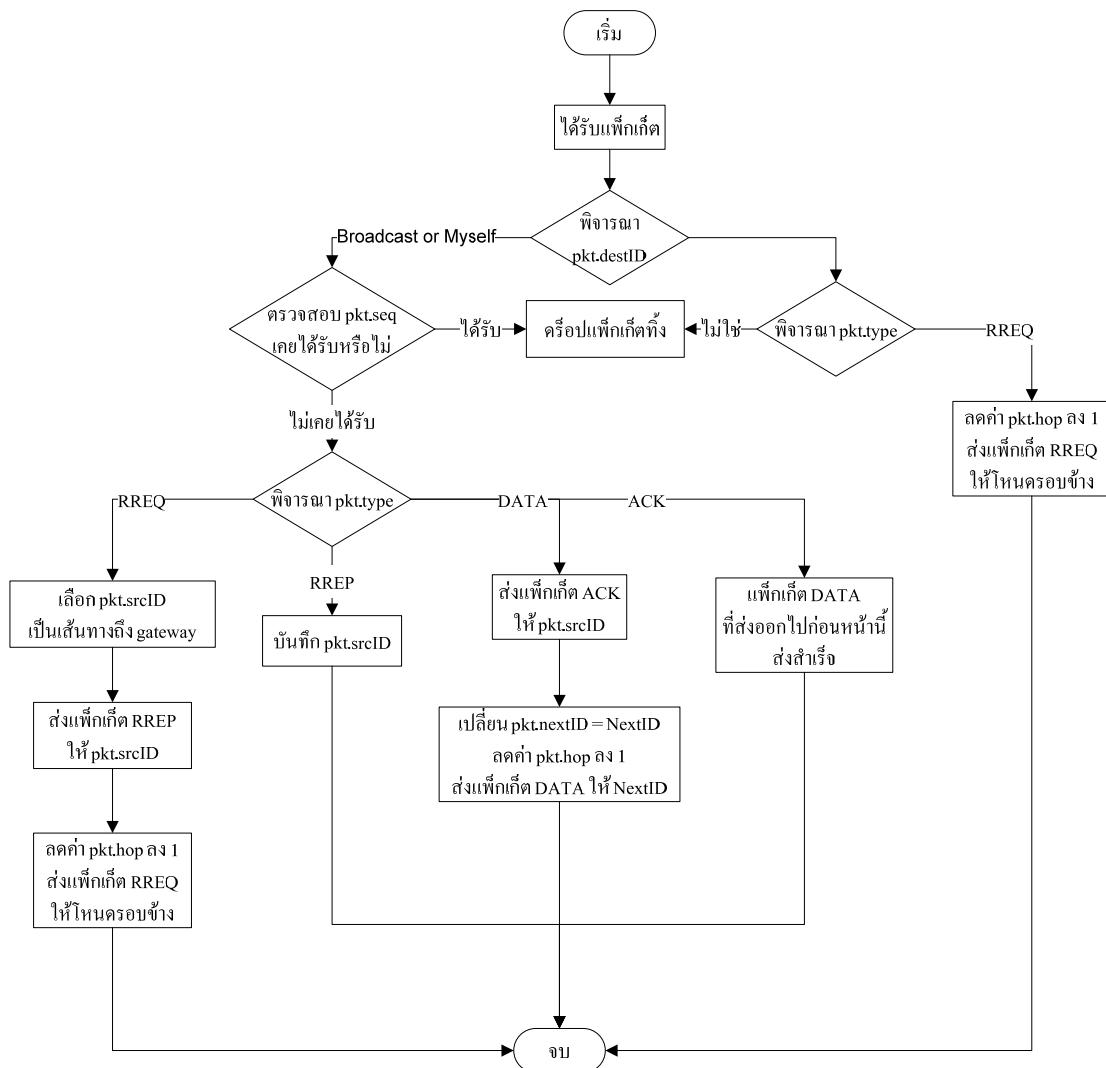


ภาพประกอบ 3-13 ผังงานการทำงานของโหนด gateway

ในภาพประกอบ 3-13 แสดงการทำงานของโหนด gateway เริ่มจากการรับแพ็กเก็ตเข้ามาพิจารณาฟิลด์ข้อมูลหมายเลขโหนดปลายทางคือหมายเลขของโหนด gateway หรือไม่ ตรวจสอบหมายเลขแพ็กเก็ตเคยได้รับหรือไม่ ถ้าเคยได้รับแล้ว จะครีโปกทิ้ง แต่ถ้าไม่เคยได้รับจะแยกชนิดของแพ็กเก็ตเพื่อประมวลผลต่อไป แพ็กเก็ต RREP บอกถึงว่าโหนด gateway เชื่อมต่อกับโหนดใดบ้าง แพ็กเก็ต DATA ที่ได้รับจะทำการส่งเข้าสู่คอมพิวเตอร์เพื่อประมวลต่อไป

ถ้าฟิลด์ข้อมูลหมายเลขโหนดปลายทางเป็น 0xFFFF โหนด gateway รู้ว่าเกิดเส้นทางเสียหายขึ้นของโหนดต้นทางของแพ็กเก็ตนั้น โหนด gateway ส่งแพ็กเก็ต RREQ โดยมีการระบุหมายเลขโหนดปลายทางเป็นหมายเลขโหนดต้นทางที่ส่งแพ็กเก็ตเข้ามา เพื่อทำการซ่อมแซมเส้นทาง

อัลกอริทึมในการรับแพ็กเก็ตของโหนด

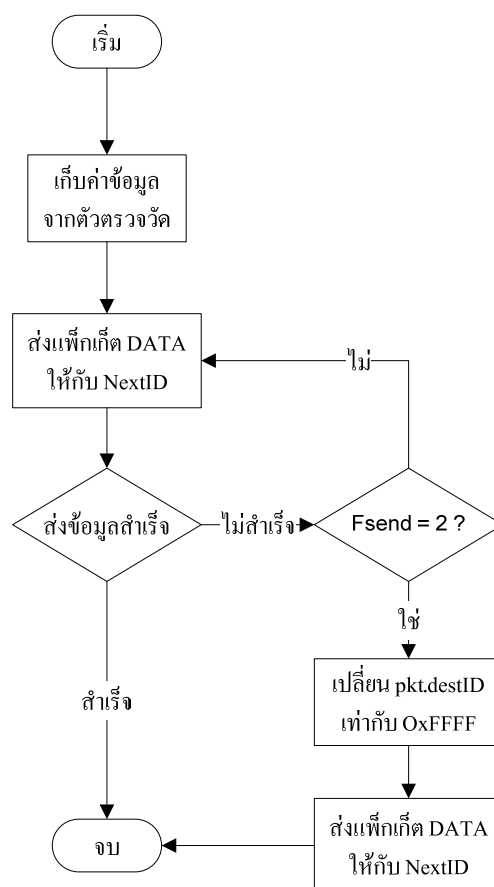


ภาพประกอบ 3-14 ฟังก์ชันการทำงานของเซนเซอร์โหนดในการรับแพ็กเก็ต

ในภาพประกอบ 3-14 แสดงการทำงานของโหนดในการรับแพ็กเก็ตเข้ามา โหนดจะเริ่มพิจารณาจากฟิลด์ข้อมูลหมายเลขโหนดปลายทาง ตรวจสอบหมายเลขแพ็กเก็ตเคยได้รับหรือไม่ ถ้าเคยได้รับแล้ว จะดรอปปั๊กเก็ตทิ้ง แต่ถ้าไม่เคยได้รับจะแยกชนิดของแพ็กเก็ตเพื่อประมวลผลต่อไป

เมื่อโหนดได้รับแพ็กเก็ต RREQ จะเลือกหมายเลขโหนดต้นทางที่ส่งแพ็กเก็ต RREQ เป็นเส้นทางในการติดต่อสื่อสารถึงโหนด gateway ถ้าโหนดได้รับแพ็กเก็ต RREP แสดงว่าโหนดต้นทางที่ส่งแพ็กเก็ต RREP เข้ามา ได้เลือกเซนเซอร์โหนดเป็นเส้นทางในการติดต่อสื่อสารถึงโหนด gateway ถ้าโหนดได้รับแพ็กเก็ต DATA โหนดจะส่งแพ็กเก็ต DATA ต่อไปให้กับโหนด gateway ผ่านทางโหนดที่เป็นเส้นทางในการติดต่อสื่อสารถึงโหนด gateway และถ้าโหนดได้รับแพ็กเก็ต ACK แสดงว่าแพ็กเก็ต DATA ที่โหนดส่งออกไป มีโหนดได้รับแล้ว

อัลกอริทึมในการส่งแพ็กเก็ตของโหนด



ภาพประกอบ 3-15 ผังงานการทำงานของเซนเซอร์โหนดในการส่งข้อมูล

ในภาพประกอบ 3-15 แสดงการทำงานของโหนดในการส่งแพ็กเก็ตข้อมูล โหนดจะนำข้อมูลที่ได้จากตัวตรวจวัด ใส่องในฟิลด์ข้อมูล Payload ของแพ็กเก็ต DATA และระบุหมายเลขโหนดปลายทาง หมายเลขโหนดถัดไป และหมายเลขลำดับแพ็กเก็ต แล้วจึงส่งแพ็กเก็ต DATA ออกไปให้โหนดถัดไป หลังจากนั้นจะรอแพ็กเก็ต ACK ซึ่งถ้าโหนดไม่ได้รับแพ็กเก็ต ACK ก็จะทำการส่งแพ็กเก็ต DATA ไปอีกครั้ง ในครั้งที่สองนี้ถ้าโหนดยังไม่ได้รับแพ็กเก็ต ACK โหนด

ก็จะทำการเปลี่ยนฟิลด์ข้อมูลหมายเลขโหนดปลายทางเป็น 0xFFFF เพื่อส่งให้โหนดรอบข้างช่วยส่งแพ็กเก็ตให้กลับโหนด gateway

บทที่ 4

การทดสอบและวิเคราะห์ประสิทธิภาพของโพรโทคอล

ในบทนี้กล่าวถึงวิธีการทดสอบการทำงานและวิเคราะห์ประสิทธิภาพของโพรโทคอลที่ได้ออกแบบและพัฒนา โดยการทดสอบแบ่งออกเป็น 2 ส่วนคือในระดับห้องปฏิบัติการ และในการนำไปใช้งานจริงในฟาร์มเลี้ยงลูกกึ่งที่อำเภอปะทิว จังหวัดชุมพร

4.1 การทดสอบประสิทธิภาพของโพรโทคอล

การทดสอบประสิทธิภาพของโพรโทคอลจะแสดงถึงความสามารถและคุณสมบัติของโพรโทคอลที่ได้ทำการออกแบบและพัฒนา ในวิทยานิพนธ์ฉบับนี้ จะทำการทดสอบและวิเคราะห์ประสิทธิภาพของโพรโทคอล 3 รูปแบบได้แก่ Packet Delivery Ratio (PDR) End-to-End Delay และเวลาในการปรับปรุงเส้นทาง

4.1.1 Packet Delivery Ratio (PDR)

การทดสอบหาค่า Packet Delivery Ratio (PDR) จะทำการคำนวณจากอัตราส่วนร้อยละระหว่างจำนวนข้อมูลที่ได้รับกับจำนวนข้อมูลที่ส่ง สามารถสรุปเป็นสมการได้ดังนี้คือ

$$\text{Packet Delivery Ratio} = \frac{\#RX}{\#TX} \times 100 \quad (4-1)$$

โดยที่ #RX คือ จำนวนแพ็กเก็ตที่ได้รับและ #TX คือ จำนวนแพ็กเก็ตที่ส่งออก ซึ่ง Packet Delivery Ratio ที่มีค่ามากจะหมายถึงโพรโทคอลที่ออกแบบและพัฒนาสามารถทำงานมีประสิทธิภาพดี

4.1.2 End-to-End Delay

การคำนวณหาค่าหน่วยเวลาหรือที่เรียกว่า End-to-End Delay นั้นสามารถทำได้จากการหาเวลาที่โหนดต้นทางส่งแพ็กเก็ตข้อมูลไปยังโหนดปลายทาง ค่าหน่วยเวลา End-to-End Delay น้อยจะหมายถึงโพรโทคอลสามารถทำงานได้อย่างรวดเร็วมีประสิทธิภาพดี

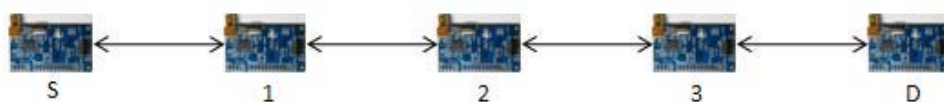
4.1.3 เวลาในการปรับปรุงเส้นทาง

การวัดประสิทธิภาพทางการปรับปรุงเส้นทางของโพรโทคอลนั้น สามารถคำนวณได้จากเวลาที่ใช้ในการปรับปรุงเส้นทางที่เกิดความเสียหาย การวัดเวลาในการปรับปรุงเส้นทาง จะทำโดยการจำลองเหตุการณ์การทำให้เกิดความเสียหายต่อเส้นทางของเครือข่าย (เมื่อโหนดมีเส้นทางในการติดต่อถึงโหนดปลายทางแล้ว จะนำโหนดที่ทำหน้าที่เป็นตัวกลางออก เพื่อให้เส้นทางในการติดต่อเกิดความเสียหาย) หลังจากนั้นจะเริ่มวัดเวลาหลังจากเกิดความเสียหายของเส้นทางไปจนกว่าเครือข่ายสามารถสร้างเส้นทางในการติดต่อสื่อสารได้สำเร็จ ซึ่งค่าเวลาในการทำการปรับปรุงเส้นทางนั้น มีค่าน้อยจะหมายถึงโพรโทคอลสามารถทำการซ่อมแซมเส้นทางได้รวดเร็ว

4.2 โครงสร้างเครือข่าย (Topology) ที่ใช้ในการทดสอบ

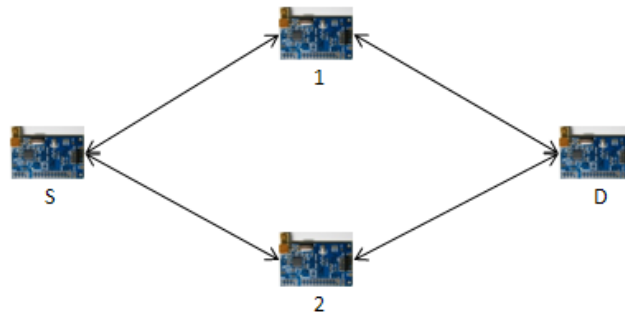
เมื่อได้ทราบวิธีการทดสอบประสิทธิภาพของโพรโทคอลแล้ว ในส่วนนี้จะอธิบายรูปแบบของการจัดวางเครือข่ายเพื่อใช้ในการทดสอบประสิทธิภาพของโพรโทคอลนั้น กำหนดให้ S คือโหนดต้นทาง และ D คือโหนดปลายทาง สามารถกำหนดโครงสร้างเครือข่ายด้วยกัน 3 แบบคือ

1. โครงสร้างเครือข่ายแบบเรียงต่อกันจำนวน 4 hop ดังภาพประกอบ 4-1 ในโครงสร้างเครือข่ายแบบนี้จะใช้เพื่อทดสอบการทำงานของโหนดในกรณีที่มีเส้นทางในการติดต่อสื่อสารเพียงเส้นเดียว คือ $S \leftrightarrow 1 \leftrightarrow 2 \leftrightarrow 3 \leftrightarrow D$



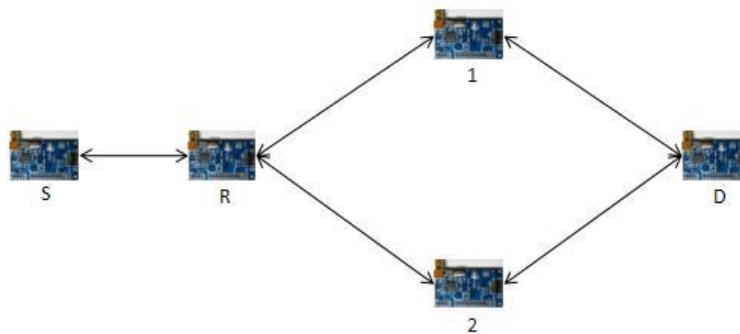
ภาพประกอบ 4-1 โครงสร้างเครือข่ายแบบเรียงต่อกันจำนวน 4 hop

2. โครงสร้างเครือข่ายแบบมี 2 ทางเลือก โดยที่แต่ละเส้นทางมีจำนวน 2 hop ดังภาพประกอบ 4-2 โครงสร้างเครือข่ายแบบนี้ใช้เพื่อทดสอบการเลือกสลับเส้นทางเมื่อเส้นทางใดเส้นทางหนึ่งได้รับความเสียหาย สรุปการเชื่อมต่อได้ดังนี้ $S \leftrightarrow 1 \leftrightarrow D$ และ $S \leftrightarrow 2 \leftrightarrow D$



ภาพประกอบ 4-2 โครงสร้างเครือข่ายแบบมี 2 ทางเลือกแต่ละเส้นทางมีจำนวน 2 hop

3. โครงสร้างเครือข่ายแบบ 3 hop lollipop ดังภาพประกอบ 4-3 โครงสร้างเครือข่ายแบบนี้ถูกวางขึ้นเพื่อต้องการทดสอบการรับส่งข้อมูล 3 hop พร้อมทั้งให้เกิดการสลับเส้นทางเมื่อเกิดความเสียหายของเส้นทาง โดยสามารถสรุปการเชื่อมต่อได้ดังนี้ $S \leftrightarrow R \leftrightarrow 1 \leftrightarrow D$ และ $S \leftrightarrow R \leftrightarrow 2 \leftrightarrow D$



ภาพประกอบ 4-3 โครงสร้างเครือข่ายแบบ 3 hop lollipop

4.3 สภาพแวดล้อมในการทดลอง

- ระยะห่างระหว่างโหนดเป็น 1 เมตร
- กำหนดระดับกำลังงานในการรับส่งข้อมูลเท่ากับ 11 หรือมีค่าเท่ากับ -10 dbm
- กำหนดใช้ช่องสัญญาณที่ 26 [17] เนื่องจากช่องสัญญาณที่ 26 ของ IEEE 802.15.4 ทับซ้อนกับช่องสัญญาณของ IEEE 802.11 น้อยที่สุด

4.4 ผลการทดสอบประสิทธิภาพ

4.4.1 Packet Delivery Ratio (PDR)

รูปแบบการทดสอบหาค่า Packet Delivery Ratio มีด้วยกัน 3 สถานการณ์ดังนี้

1. โหนด S ส่งข้อมูลไปยังโหนด D เพียงตัวเดียว เรียกว่ารูปแบบการทดสอบที่ 1
2. โหนดทุกตัวส่งข้อมูลไปที่โหนด D พร้อมกัน เรียกว่ารูปแบบการทดสอบที่ 2

3. โหนดทุกตัวทำหน้าที่ส่งข้อมูลไปยังโหนด D ไม่พร้อมกัน โดยให้คำนวณการเริ่มส่งข้อมูลของแต่ละโหนดจากสมการที่ (4-2) เรียกว่ารูปแบบการทดสอบที่ 3

$$t_{\text{start}} = 5 \times \# \text{hop} \quad (4-2)$$

โหนด S จะทำการส่งแพ็คเกจข้อมูลให้กับโหนด D 100 ครั้ง โดยที่ขนาดของข้อมูลหรือ Payload ที่ใช้ส่งนั้นมีขนาดเท่ากับ 4 Byte การทดสอบหาค่า PDR เลือกใช้โครงสร้างเครือข่ายทั้ง 3 แบบคือ โครงสร้างเครือข่ายแบบเรียงต่อกันจำนวน 4 hop โครงสร้างเครือข่ายแบบมีสองเส้นทางแต่ละเส้นทางมี 2 hop และโครงสร้างเครือข่ายแบบ 3 hop lollipop ตามรูปแบบการทดสอบที่ 1 ถึง 3 ได้ผลดังตารางที่ 4-1 พบว่ารูปแบบการทดสอบที่ 1 คือโหนด S ส่งข้อมูลไปยังโหนด D เพียงตัวเดียว ในรูปแบบการเชื่อมต่อแบบสองเส้นทางโดยแต่ละเส้นทางมี 2 hop ให้ค่า PDR ที่สุดที่ 99.2 เปอร์เซ็นต์ ตามด้วยโครงสร้างเครือข่ายแบบ 3 hop lollipop ที่ 98.4 เปอร์เซ็นต์ และโครงสร้างเครือข่ายแบบเรียงต่อกันจำนวน 4 hop มีค่าน้อยที่สุดที่ 97.7 เปอร์เซ็นต์ จากผลการทดลองพบโปรโตคอลที่พัฒนาขึ้นสามารถทำงานได้อย่างมีประสิทธิภาพในสถานการณ์ปกติ ไม่มีโหนดหลายตัวส่งข้อมูลไปยังโหนดปลายทางในเวลาเดียวกัน

ตารางที่ 4-1 ค่า Packet Delivery Ratio ของรูปแบบการทดสอบที่ 1, 2 และ 3

โครงสร้างเครือข่าย	Packet Delivery Ratio (%)		
	รูปแบบที่ 1	รูปแบบที่ 2	รูปแบบที่ 3
เรียงต่อกันจำนวน 4 hop	97.7	84.5	94.5
สองเส้นทางแต่ละเส้นทางมี 2 hop	99.2	81.2	95.2
3 hop lollipop	98.4	80.7	94.6

สำหรับการทดสอบในรูปแบบที่ 2 ได้ค่า PDR ของโครงสร้างเครือข่ายแบบเรียงต่อกันจำนวน 4 hop โครงสร้างเครือข่ายแบบมีสองเส้นทางแต่ละเส้นทางมี 2 hop และโครงสร้างเครือข่ายแบบ 3 hop lollipop เป็น 84.5 เปอร์เซ็นต์ 81.2 เปอร์เซ็นต์ และ 80.7 เปอร์เซ็นต์ ตามลำดับ ซึ่งลดลงจากสถานการณ์ในการทดสอบแบบที่ 1 ประมาณ 13 ถึง 18 เปอร์เซ็นต์ เนื่องจากเป็นสถานการณ์ที่โหนดทุกตัวต่างส่งข้อมูลออกมาพร้อมกันในเวลาเดียวกันไปยังโหนดปลายทางหรือโหนด D เหตุการณ์เช่นนี้เกิดมาจากการชนกันของข้อมูล เพราะเครือข่ายมีความคับคั่งของแพ็คเกจข้อมูล

เพื่อเป็นการตรวจสอบและศึกษาสาเหตุของการที่ค่า PDR ของโพรโทคอลที่ ออกแบบและพัฒนาลดลงจากสถานการณ์การทดสอบที่ 2 ที่โหนดต่างส่งข้อมูลออกมาพร้อมกันใน เวลาเดียวกับ ดังนั้นจึงจัดการทดสอบแบบที่ 3 ให้โหนดแต่ละตัวมีการส่งข้อมูล ณ เวลาที่แตกต่างกัน โดยกำหนดให้โหนดแต่ละตัวเริ่มทำงานไม่พร้อมกัน ผลการทดสอบค่า PDR ของโครงสร้าง เครือข่ายแบบเรียงต่อกันจำนวน 4 hop แบบมีสองเส้นทางแต่ละเส้นทางมี 2 hop และแบบ 3 hop lollipop พบว่ามีค่าเป็น 94.5 เปอร์เซ็นต์, 95.2 เปอร์เซ็นต์ และ 94.6 เปอร์เซ็นต์ ตามลำดับ ซึ่งค่า PDR ที่ได้ดีขึ้นจนเกือบจะเท่ากับการทดลองในรูปแบบที่ 1 จึงสามารถสรุปได้ว่าค่า PDR ที่ลดลง ในสถานการณ์การทดสอบที่ 2 นั้น เกิดจากการชนกันของข้อมูลอันเนื่องมาจากที่โหนดทั้งหมดต่าง ส่งข้อมูลออกมาพร้อมกันในเวลาเดียวกัน

จากปัญหาที่พบนี้ผู้พัฒนาจึงได้ทำการพัฒนาอัลกอริทึมการทำงาน Scheduling อย่างง่ายเข้ามาช่วยเพิ่มประสิทธิภาพการทำงานของโพรโทคอลที่พัฒนาขึ้น โดยกำหนดให้แต่ละ โหนดมีค่าหน่วงเวลาก่อนที่จะจัดส่งแพ็กเก็ตข้อมูลออกไม่เท่ากัน ซึ่งสามารถกำหนดค่าหน่วงได้ จากผลคูณของค่าคงที่กับหมายเลขของโหนดเช่น โหนดหมายเลข 1 จะมีค่าหน่วงเวลาที่ 5 หน่วย เวลา โหนดหมายเลข 2 จะมีค่าหน่วงเวลาที่ 10 หน่วยเวลา เป็นต้น ซึ่งผลของการทดสอบจะแสดง ไว้ในหัวข้อถัดไป

4.4.2 การทดสอบประสิทธิภาพของโพรโทคอลแบบมี Scheduling

การทดสอบโพรโทคอลแบบมี Scheduling อย่างง่ายโดยการเพิ่มค่าหน่วงของการ ส่งแพ็กเก็ตข้อมูล เพื่อลดปัญหาของแพ็กเก็ตชนกัน ซึ่งการพัฒนาในส่วนของ Scheduling จะทำใน โมดูล OutputQueue ดังภาพประกอบ 3-10 ซึ่งทำกระบวนการ Scheduling อย่างง่ายโดยการเอา หมายเลขโหนด * 100ms เพื่อเป็นค่าหน่วงของการส่งข้อมูลออกของแต่ละโหนด ทำให้มีการหน่วง เวลาก่อนส่งข้อมูลของแต่ละโหนดไม่ตรงกัน ลักษณะของโครงสร้างเครือข่ายแบบดาว (star topology)

การทดสอบประสิทธิภาพของโพรโทคอลในกรณีที่มีจำนวนการเชื่อมต่อมายัง โหนดเดียวกันเพิ่มขึ้น (many-to-one) จะช่วยชี้วัดความสามารถของโพรโทคอลว่าสามารถจัดการ กับการชนกันของข้อมูลได้อย่างไร โดยได้ทำการเชื่อมต่อเครือข่ายแบบ star เพิ่มจำนวนการ เชื่อมต่อจาก 1 โหนดไปจนถึง 5 โหนด ผลการทดสอบแสดงในตารางที่ 4-2 พบว่าค่า PDR ลดลง อย่างรวดเร็วจนกระทั่งเมื่อมีจำนวนการเชื่อมต่อเป็น 5 โหนดค่า PDR มีค่าต่ำกว่า 50 เปอร์เซ็นต์ ดังนั้นจึงได้ทำการนำโพรโทคอลที่พัฒนาแบบมีการทำ scheduling อย่างง่ายนำมาใช้ พบว่าค่า PDR

สูงขึ้นกว่า การใช้โพรโทคอลแบบที่ไม่มี scheduling อยู่ที่ประมาณ 12 ถึง 18เปอร์เซ็นต์ ในแต่ละจำนวนการเชื่อมต่อที่แตกต่างกัน

ตารางที่ 4-2 ผลการทดสอบค่า PDR กับจำนวนการเชื่อมต่อระหว่างโพรโทคอลที่มี scheduling กับโพรโทคอลที่ไม่มี scheduling

จำนวนการสื่อสาร	ค่า Packet Delivery Ratio (%)		Improvement (%)
	ไม่มี scheduling	มี scheduling	
1	100	100	0
2	81.5	93.86	12.36
3	67.6	85.19	17.59
4	56.25	74.21	17.96
5	49.4	64.15	14.75

4.4.3 End-to-End Delay

ในการทดสอบ End-to-End Delay นั้นจะเลือกใช้โครงสร้างเครือข่ายเรียงต่อกันจำนวน 4 hop ดังภาพประกอบ 4-1 เนื่องจากโครงสร้างเครือข่ายทั้ง 3 โครงสร้าง โหนดไม่มีการเคลื่อนที่ ดังนั้นจึงทำการทดสอบโครงสร้างเครือข่ายแบบเรียงต่อกันจำนวน 4 hop เพียงโครงสร้างเครือข่ายเดียวเท่านั้น ในการทดสอบหาค่า End-to-End Delay จะเริ่มทำการทดสอบจากจำนวน hop 1 hop 2 hop 3 hop และสุดท้ายจำนวน 4 hop ตามลำดับ ในแต่ละครั้งของการทดสอบจะทำการทดลองซ้ำ 10 ครั้งแล้วนำมาหาค่าเฉลี่ย ในการทดสอบนี้จะทำการทดสอบสูงสุดที่ 4 hop เนื่องจากเป็นจำนวน hop ที่สูงที่สุดที่จะเกิดขึ้นในการนำไปใช้งานในฟาร์มเลี้ยงกุ้ง

วิธีการทดสอบหา End-to-End Delay ของโหนดในเครือข่าย

1. โหนดปลายทางหรือโหนด D ทำการกระจายแพ็กเก็ตเพื่อค้นหาเส้นทางในการติดต่อสื่อสาร
2. หลังจากที่โหนดทุกโหนดได้เส้นทางในการติดต่อสื่อสารกับโหนด D โหนดแต่ละโหนดจะส่งแพ็กเก็ตข้อมูลถึงโหนด D ค่า End-to-End Delay วัดจากเวลาที่โหนดแต่ละโหนดส่งแพ็กเก็ตถึงโหนด D

ตารางที่ 4-3 ผลการทดสอบ End-to-End Delay

จำนวน hop	เวลา (ms)
1	28.1
2	40.4
3	55.9
4	76.5

ผลการทดสอบ End-to-End Delay แสดงไว้ในตารางที่ 4-3 พบว่าค่าหน่วงเวลา End-to-End Delay จะเพิ่มขึ้นเมื่อมีจำนวน hop เพิ่มขึ้น จากการทดลองทำให้พบแนวโน้มของความสัมพันธ์ระหว่างค่าหน่วงเวลา End-to-End Delay กับจำนวน hop ว่าไม่เป็นเส้นตรง แต่ทั้งนี้จะต้องมีการทดสอบเพิ่มเติมจึงจะสามารถกล่าวสรุปได้อย่างชัดเจน

4.4.4 การทดสอบความเร็วในการปรับปรุงเส้นทาง

ในการทดสอบความเร็วในการปรับปรุงเส้นทางนั้นจะเลือกทดสอบในโครงสร้างเครือข่ายแบบสองเส้นทางแต่ละเส้นทางมี 2 hop และ 3 lollipop เส้นทางการติดต่อสื่อสารของโหนดเกิดความเสียหาย ไม่สามารถติดต่อสื่อสารได้ ทำให้ต้องมีการปรับปรุงเส้นทางเกิดขึ้น

วิธีการทดสอบหาความเร็วในการปรับปรุงเส้นทาง

1. ให้โหนด D ทำการกระจายแพ็กเก็ตเพื่อค้นหาเส้นทางในการติดต่อสื่อสาร
2. โหนด S ได้เส้นทางติดต่อสื่อสารถึงโหนด D โดยผ่านโหนดหมายเลข 1 ในโครงสร้างเครือข่ายแบบสองเส้นทางแต่ละเส้นทางมี 2 hop มีเส้นทาง $S \leftrightarrow 1 \leftrightarrow D$ และในเครือข่ายแบบ 3 lollipop มีเส้นทาง $S \leftrightarrow R \leftrightarrow 1 \leftrightarrow D$
3. หลังจากที่โหนด D ได้รับข้อมูลแล้ว ทำการย้ายโหนดหมายเลข 1 ออกจากโครงสร้างเครือข่ายทำให้เส้นทางสื่อสารระหว่างโหนด S กับโหนด D เกิดเสียหาย
4. โหนด S ส่งกระจายแพ็กเก็ตให้โหนดรอบข้างช่วยส่งแพ็กเก็ตให้กับโหนด D
5. โหนด D ได้รับแพ็กเก็ตข้อมูลที่มีการส่งแบบกระจายจากโหนดรอบข้าง โหนด D ส่งแพ็กเก็ตเพื่อซ่อมแซมเส้นทางสื่อสารระหว่างโหนด D กับโหนด S
6. โหนด S สามารถติดต่อสื่อสารโหนด D ผ่านทางโหนดหมายเลข 2 ได้ ในโครงสร้างเครือข่ายแบบสองเส้นทางแต่ละเส้นทางมี 2 hop มีเส้นทาง $S \leftrightarrow 2 \leftrightarrow D$ และในเครือข่ายแบบ 3 lollipop มีเส้นทาง $S \leftrightarrow R \leftrightarrow 2 \leftrightarrow D$

ตารางที่ 4-4 ผลการทดสอบหาความเร็วในการปรับปรุงเส้นทาง

โครงสร้างเครือข่าย	เวลา (ms)
สองเส้นทางแต่ละเส้นทางมี 2-hop	95.7
3 hop lollipop	132.5

ผลการทดสอบหาความเร็วในการปรับปรุงเส้นทาง แสดงไว้ในดังตารางที่ 4-4 พบว่าในโครงสร้างเครือข่ายแบบสองเส้นทางแต่ละเส้นทางมี 2 hop ใช้เวลาในการปรับปรุงเส้นทาง 95.7ms และโครงสร้างเครือข่าย 3 hop lollipop ใช้เวลาในการปรับปรุงเส้นทาง 132.5ms เวลาในการปรับปรุงเส้นทาง จะเพิ่มเมื่อจำนวน hops ที่อยู่ในเส้นทางมากขึ้น จะใช้เวลาในการปรับปรุงเส้นทาง

4.4.5 ขนาดของโปรโตคอลที่พัฒนาบนระบบปฏิบัติการ TinyOS

ตารางที่ 4-5 แสดงขนาดของโปรโตคอลที่ได้ทำการพัฒนาขึ้นกับโปรโตคอล DYMO

	ROM (Byte)	RAM (Byte)	รวม
DYMO	19,916	940	20,856
โปรโตคอลที่สร้างขึ้นเอง	15,556	1,418	16,974

เพื่อแสดงให้เห็นว่าโปรโตคอลที่พัฒนาขึ้นมีการหน่วยความจำมาน้อยเพียงใด จึงได้ทำการนำเสนอการใช้พื้นที่ในหน่วยความจำ 2 ชนิดคือ ROM สำหรับเก็บโปรโตคอลและ RAM สำหรับเก็บตัวแปรต่างๆที่เกิดขึ้นระหว่างการทำงานของโปรโตคอลดังแสดงในตารางที่ 4-5 เมื่อทำการเปรียบเทียบขนาดของโปรโตคอลแบบที่มีการทำ scheduling เป็นตัวแทนของโปรโตคอลที่ได้ทำการพัฒนาขึ้นในงานวิจัยนี้ พบว่ามีขนาดการใช้งานพื้นที่ในหน่วยความจำต่ำกว่าโปรโตคอล DYMO ของ TinyOS ที่มีอยู่เดิมถึง 4 Kbytes โดยประมาณ ซึ่งโปรโตคอลที่ได้พัฒนาขึ้นเองมีการใช้งานหน่วยความจำชนิด RAM มากกว่า DYMO เนื่องจากมีส่วนของการจองพื้นที่ใช้เก็บข้อมูล (Queue) ซึ่งโปรโตคอล DYMO ไม่มีการจองพื้นที่ในส่วนนี้ ทั้งนี้สามารถปรับลดขนาดของการจองพื้นที่ที่ใช้เก็บข้อมูลได้เมื่อนำไปใช้งานจริง

4.5 การนำไปใช้งานในฟาร์มเลี้ยงกุ้ง

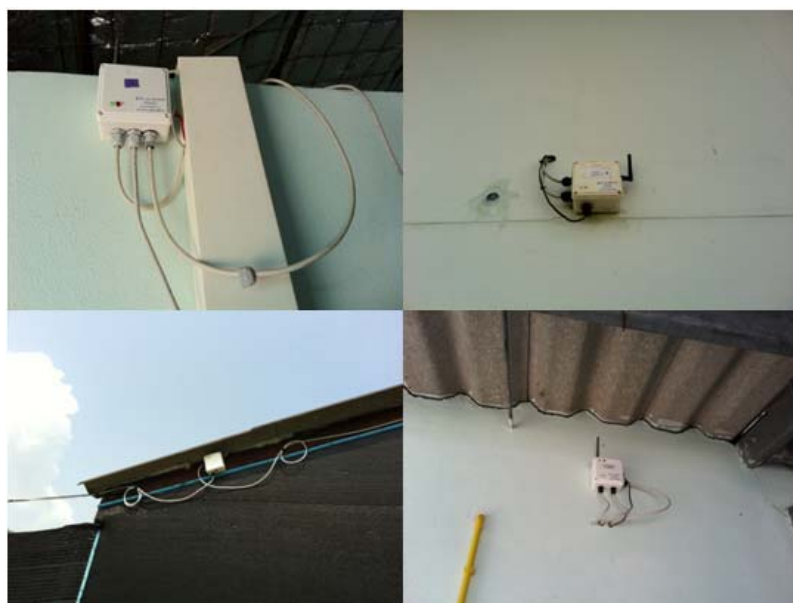
โปรโตคอลที่ได้ทำการพัฒนาขึ้นมา ได้มีการนำไปทดสอบการทำงานในสถานที่จริง ณ ฟาร์มเลี้ยงลูกกุ้งที่อำเภอปะทิว จังหวัดชุมพร ฟาร์มเลี้ยงลูกกุ้งมีลักษณะเป็นโรงเรือนแบบ

ปิด เป็นอาคารคอนกรีตดังในภาพประกอบ 4-4 โหนดที่ได้นำมาใช้งานนั้นจะทำหน้าที่เก็บข้อมูล อุณหภูมิของน้ำและอากาศในบ่อเลี้ยงลูกกุ้งจำนวน 30 จุดกระจายอยู่ในโรงเรือนจำนวน 6 โรงเรือน ภายในฟาร์ม นอกจากนี้ยังมีการควบคุมเครื่องปรับอากาศในแต่ละโรงเรือน โรงเรือนละ 2 จุด โดยเมื่ออุณหภูมิในน้ำและอากาศลดลงถึงค่าที่ตั้งไว้ จะทำการปิดเครื่องปรับอากาศ และจะเปิดอีกครั้งเมื่ออุณหภูมิเพิ่มสูงขึ้นถึงค่าที่ผู้ติดตั้งไว้



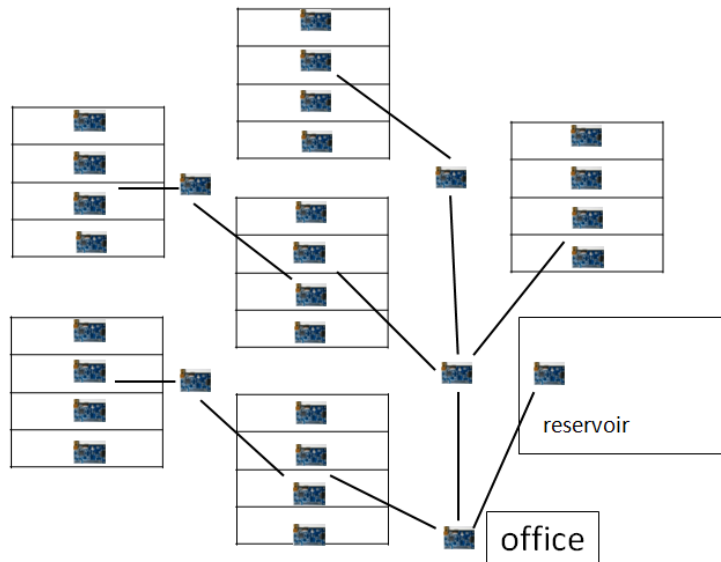
ภาพประกอบ 4-4 ฟาร์มเลี้ยงลูกกุ้ง

การติดตั้งโหนดที่นำไปใช้งานในฟาร์มเลี้ยงลูกกุ้งแต่ละโรงเรือนไม่ได้ติดตั้ง โหนดที่เดิม อาจจะสามารถถูกเคลื่อนย้ายได้ในแต่ละรอบของการตรวจวัดหรือเก็บข้อมูล เนื่องจาก อุปกรณ์การเลี้ยงลูกกุ้งสามารถเปลี่ยนแปลงได้ อีกทั้งในการติดตั้งจะต้องให้โหนดสามารถเคลื่อนย้ายตำแหน่งได้เพื่อหาจุดติดตั้งที่ให้ประสิทธิภาพของการส่งข้อมูลดีที่สุด เพราะโหนดมีความอ่อนไหวต่อน้ำในบ่อเลี้ยงกุ้งและแผ่นเหล็กโครงสร้างของโรงเรือน ดังภาพประกอบ 4-5



ภาพประกอบ 4-5 การติดตั้งโหนดที่นำไปใช้งาน

โครงสร้างเครือข่ายในการติดตั้งโหนดในฟาร์มเลี้ยงลูกกึ่งภาพประกอบ 4-6 มีการติดตั้ง Unode จำนวน 30 ตัวในการใช้งานฟาร์มเลี้ยงลูกกึ่ง โดยในแต่ละเส้นทางมีจำนวน โหนดอยู่ประมาณ 4 โหนด



ภาพประกอบ 4-6 โครงสร้างเครือข่ายที่ทำการติดตั้งเซนเซอร์

ส่วนแสดงข้อมูลที่ได้รับจากโหนดที่ติดตั้งในฟาร์มเลี้ยงลูกกึ่งซึ่งถูกใช้งานเป็นระยะเวลาประมาณ 1 ปี ผลอุณหภูมิในน้ำและอากาศของโหนดทุกตัวแสดงไว้ในภาพประกอบ 4-7 ซึ่งเป็นข้อมูลที่ได้จากโหนดทั้งเครือข่าย รวมถึงสถานะของเครื่องปรับอากาศ



ภาพประกอบ 4-7 ส่วนแสดงผลข้อมูล

การนำโปรโตคอลพัฒนาขึ้นไปประยุกต์ใช้งานจริง การทำงานของโหนดจะมีการเก็บข้อมูลจากเซนเซอร์ และทำการส่งข้อมูลที่ได้ทุกๆ 1 นาที โหนดที่มีการใช้งานประมาณ 30 โหนด ผลการทำงานของโหนดในสถานที่จริงพบว่า ผลการทดสอบค่า Packet Delivery Ratio (PDR) พบว่ามีค่าประมาณ 90 เปอร์เซ็นต์ ดังนั้นเครือข่ายจะต้องมีการปรับปรุงเส้นทางอยู่ที่ประมาณ 10 เปอร์เซ็นต์ ของการใช้งาน สำหรับงานที่เก็บข้อมูลของอุณหภูมิน้ำและอากาศ เนื่องจากการเปลี่ยนแปลงของอุณหภูมิต้องใช้เวลาในการเปลี่ยนแปลง ไม่ได้เปลี่ยนแปลงทันทีทันใด

บทที่ 5

สรุปผลการวิจัยและข้อเสนอแนะ

ในบทนี้จะกล่าวถึงการสรุปผลและข้อเสนอแนะที่ได้จากการดำเนินการทำวิทยานิพนธ์ ตลอดจนปัญหาและอุปสรรคที่เกิดขึ้นขณะทำวิทยานิพนธ์ และท้ายที่สุดจะกล่าวถึงข้อเสนอแนะแก่ผู้สนใจที่จะนำวิทยานิพนธ์ชิ้นนี้เพื่อพัฒนาต่อไป

5.1 สรุปผล

งานวิจัยชิ้นนี้นำเสนอการออกแบบโพรโทคอลเพื่อใช้งานในเครือข่ายเซนเซอร์ไร้สาย โดยในการพัฒนาโพรโทคอลโดยใช้ระบบปฏิบัติการ TinyOS สำหรับอุปกรณ์ในเครือข่ายเซนเซอร์ไร้สายที่ชื่อว่า unode เพื่อให้สามารถนำไปใช้งานได้จริงในเครือข่ายเซนเซอร์ไร้สาย โดยโพรโทคอลมีความสามารถในการค้นหาเส้นทาง รับส่งข้อมูลกันได้ และสามารถทำการซ่อมแซมเส้นทางเมื่อเกิดเส้นทางเสียหายในเครือข่าย

แนวคิดและวิธีการออกแบบและพัฒนาโพรโทคอลโดยนำแนวคิดการทำงานของโพรโทคอล AODV มาประยุกต์ใช้ในการออกแบบการทำงานของโพรโทคอล ซึ่งเครือข่าย ad hoc และเครือข่ายเซนเซอร์ไร้สายมีลักษณะเครือข่ายค่อนข้างคล้ายคลึงกัน เครือข่ายเซนเซอร์ไร้สายมีข้อจำกัดหลายด้านเช่น พลังงานและการประมวลผล เป็นต้น จึงส่งผลกระทบต่อโพรโทคอลที่ได้ออกแบบและพัฒนาขึ้นจะต้องมีการทำงานที่ไม่ซับซ้อน

จากผลการทดสอบประสิทธิภาพของโพรโทคอลที่ออกแบบพบว่า ค่า PDR ของโพรโทคอลอยู่ที่ 84.5 เปอร์เซ็นต์ แต่เมื่อนำการทำ scheduling เข้ามาช่วยในการรับส่งข้อมูลทำให้ค่า PDR อยู่ที่ 95.2 เปอร์เซ็นต์ สูงขึ้นจากโพรโทคอลที่มีการทำ scheduling ประมาณ 10 ถึง 14 เปอร์เซ็นต์ และเมื่อมีจำนวนการสื่อสารมากขึ้นค่า PDR มีค่าลดลงแต่เมื่อนำการทำ scheduling เข้ามาช่วยทำให้ค่า PDR สูงขึ้นประมาณ 12 ถึง 18 เปอร์เซ็นต์ จากผลทดสอบประสิทธิภาพของโพรโทคอลข้างต้นมีเกณฑ์การทำงานที่ดี ผู้ทำวิทยานิพนธ์ได้นำโพรโทคอลที่ได้พัฒนาขึ้นไปประยุกต์ใช้งานจริงในฟาร์มเลี้ยงลูกกึ่งพบว่าค่า PDR ที่ได้จากการนำไปใช้งานมีค่าอยู่ประมาณ 90 เปอร์เซ็นต์

สรุปได้ว่าโพรโทคอลที่ได้ออกแบบนี้มีการทำงานที่ไม่ซับซ้อน สามารถค้นหาเส้นทาง รับส่งข้อมูลได้เหมาะกับงานเครือข่ายเซนเซอร์ไร้สายได้

5.2 ผลที่ได้จากการทำวิทยานิพนธ์ชุดนี้

ผู้ทำวิทยานิพนธ์ได้ศึกษาและออกแบบพัฒนาโพรโทคอลเพื่อนำโพรโทคอลที่พัฒนาขึ้นมาไปประยุกต์ใช้ในงานเครือข่ายเซนเซอร์ไร้สายได้ โดยทำการศึกษาการทำงานของโพรโทคอลในเครือข่าย ad hoc เนื่องจากเครือข่ายเซนเซอร์ไร้สายมีลักษณะการทำงานที่คล้ายคลึงกันกับในเครือข่าย ad hoc โพรโทคอลที่ได้ออกแบบเน้นการทำงานที่ไม่ซับซ้อน รองรับการเปลี่ยนแปลงของโครงสร้างเครือข่ายอันเกิดจากสถานะแวดล้อมให้การรับส่งข้อมูลสามารถทำงานได้ และได้มีการเพิ่มเติมการทำ scheduling เพื่อช่วยในการส่งข้อมูลเพื่อให้โอกาสการชนกันของข้อมูลน้อยลงและค่า PDR สูงขึ้น โพรโทคอลได้พัฒนาบนระบบปฏิบัติการ TinyOS เพื่อให้สามารถใช้กับงานเครือข่ายเซนเซอร์ไร้สายได้ ในวิทยานิพนธ์นี้ได้ผลทดสอบประสิทธิภาพของโพรโทคอลด้วย PDR โพรโทคอลที่ได้พัฒนาขึ้นมามีค่า PDR สูงถึง 95.2 เปอร์เซ็นต์ นอกจากนี้ยังได้เพิ่ม PDR ประมาณ 12 ถึง 18 เปอร์เซ็นต์ เมื่อมีจำนวนการสื่อสารเพิ่มขึ้น ปัจจัยที่มีผลกระทบต่อค่า PDR คือ จำนวนโหนดในเส้นทางสื่อสาร โครงสร้างเครือข่าย และเวลาในการส่งข้อมูล

ผลที่ได้รับจากวิทยานิพนธ์ชุดนี้คือ ได้แนวคิดในการออกแบบและพัฒนาโพรโทคอล ให้สามารถนำไปประยุกต์ใช้งานในเครือข่ายเซนเซอร์ไร้สายได้ และสามารถประยุกต์ใช้ให้เหมาะสมกับงานด้านการควบคุมดูแล (monitor environment) การเลี้ยงลูกกึ่งได้จริงในฟาร์มที่มีจำนวนโหนดประมาณ 30 ตัว

5.3 ปัญหาและอุปสรรค

ปัญหาและอุปสรรคในการดำเนินวิทยานิพนธ์นี้ ในด้านของความรู้ผู้ทำวิทยานิพนธ์ต้องศึกษาการทำงานของโพรโทคอลต่างๆ เพื่อช่วยในการออกแบบพัฒนาโพรโทคอลในด้านการเขียนรายงานเชิงวิชาการได้ไม่ดีเท่าที่ควร และในส่วนของทดสอบประสิทธิภาพของโพรโทคอลพบปัญหาในเรื่องของสถานะแวดล้อมในการทดสอบ เครือข่ายเซนเซอร์ไร้สายมีความอ่อนไหวต่อสถานะแวดล้อมรอบข้างเช่น ความชื้น คลื่นแม่เหล็ก

5.4 ข้อเสนอแนะและแนวทางการพัฒนาในอนาคต

งานวิทยานิพนธ์ชิ้นนี้นำเสนอการพัฒนาและออกแบบโพรโทคอลให้สามารถใช้งานในเครือข่ายเซนเซอร์ไร้สายที่เน้นให้โพรโทคอลมีขนาดเล็ก ไม่ซับซ้อน และสามารถใช้งานได้จริง รวมทั้งการแก้ไขปัญหาการชนของข้อมูลได้อย่างง่าย แต่การทดสอบยังอยู่บนเครือข่ายขนาด

จำนวนไม่เกิน 30 โหนดทำให้นำจะมีการทดสอบกับการใช้งานกับเครือข่ายขนาดใหญ่เช่น จำนวน 100 โหนดเป็นต้นไป

นอกจากนี้สามารถนำโพรโทคอลชนิด AODV หรือ TinyAODV มาสร้างใช้งานจริงบนโหนดแล้วนำมาเปรียบเทียบให้เห็นผลอย่างชัดเจนได้อีกด้วย

บรรณานุกรม

- [1] P. Levis and David Gay, "TinyOS Programming," *Cambridge University Press*, April, 2009.
- [2] C. Perkins *et al.*, "Ad hoc On-Demand Distance Vector (AODV) Routing," [Online]. Available: <http://www.ietf.org/rfc/rfc3561.txt>, July 2003.
- [3] I. Chakeres and L. Klein-Berndt, "AODVjr routing protocol with multiple feedback policy for Zigbee Network," *Proceeding of the IEEE 13th International Symposium on Consumer Electronics (ISCE'09)*, pp. 483-487, 2009.
- [4] TinyAODV Implementation, "TinyOS source code repository," [Online]. Available: [http://TinyOS.cvs.sourceforge.net /viewvc/tinyOS/tinyOS-1.x/contrib/hsn](http://TinyOS.cvs.sourceforge.net/viewvc/tinyOS/tinyOS-1.x/contrib/hsn).
- [5] C. Gomez *et al.*, "Adapting AODV for IEEE 802.15.4 Mesh Sensor Networks: Theoretical Discussion and Performance Evaluation in a Real Environment," *Proceeding of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06)*, pp.1-9, 2006.
- [6] Shailesh A and Notani, "Performance Simulation of Multihop Routing Algorithm for Ad hoc Wireless Sensor Networks Using TOSSIM," *Proceeding of the International Conference on Advanced Communication Technology (ICACT)*, pp.508-513, 2008.
- [7] I.F Akyildiz *et al.*, "A survey on sensor networks," *Communications Magazine, IEEE* , vol.40, no.8, pp. 102- 114, August 2002.
- [8] J. Kulik *et al.*, "Negotiation-based protocols for disseminating information in wireless sensor networks," *Journal Wireless Networks - Selected Papers from Mobicom'99*, vol. 8, pp. 169-185, March-May, 2002.
- [9] C. Intanagonwiwat *et al.*, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," *Proceeding of ACM International Conference on Mobile computing and networking (MobiCom'00)*, pp. 56-67, 2000.
- [10] W. Heinzelman *et al.*, "Energy-Efficient Communication Protocol for Wireless Micro sensor Networks," *Proceeding of the 33rd Hawaii International Conference on System Sciences (HICSS)*, pp.12-20, 2000.

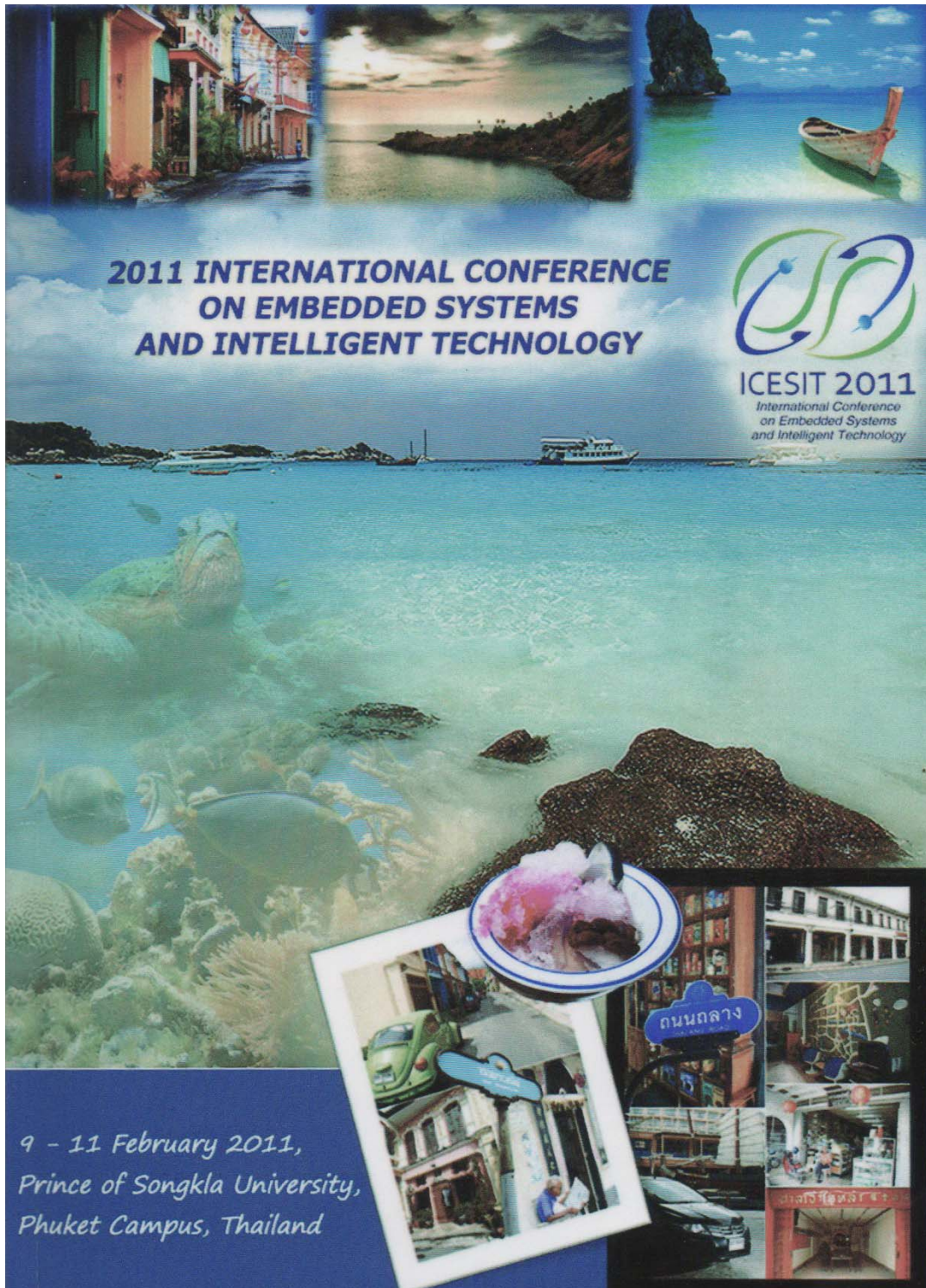
- [11] S. Lindsey and C. Raghavendra, "PEGASIS: Power-Efficient Gathering in Sensor Information Systems," *Proceeding of IEEE Aerospace Conference*, vol. 3, pp. 1125-1130, 2002.
- [12] C. Perkins, P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," *Proceedings of the conference on Communications architectures, protocols and applications (SIGCOMM)*, pp. 234-244, 1994.
- [13] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," [Online]. Available: <http://www.ietf.org/rfc/rfc3626.txt>, October 2003.
- [14] D. Johnson *et al.*, "The dynamic source routing protocol (DSR)," [Online]. Available: <http://www.ietf.org/rfc/rfc4728.txt>, February 2007.
- [15] G. Koltsidas *et al.*, "A performance study of the HSLs routing algorithm for Ad hoc networks," *Proceeding of IEEE 59th Vehicular Technology Conference*, vol.4, pp. 2140 - 2143, 2004.
- [16] N. Jamal *et al.*, "Routing Techniques in Wireless Sensor Networks: A Survey," *IEEE Journal in Wireless Communications*, vol. 11, No. 6, pp. 6-28, 2004.
- [17] Crossbow Company, "Avoiding RF Interference between WiFi and ZigBee." [Online]. Available: <http://www.mobiusconsulting.com/papers/ZigBeeWifiInterference.pdf>
- [18] C. T. Ee *et al.*, "A Modular Network Layer for Sensornets," *Proceeding of Operating Systems Design and Implementation (OSDI)*, pp. 249-262, 2006.
- [19] I. Chakeres and C. Perkins, "Dynamic MANET On-demand (DYMO) routing (work in progress)," [Online]. Available: <http://tools.ietf.org/html/draft-ietf-manet-dymo-21>, July 2010.

ภาคผนวก

ผลงานตีพิมพ์เผยแพร่จากวิทยานิพนธ์



**2011 INTERNATIONAL CONFERENCE ON EMBEDDED SYSTEMS AND
INTELLIGENT TECHNOLOGY (ICESIT2011)
FEB 9, 2011 - FEB 11, 2011, at PHUKET, THAILAND**



**2011 INTERNATIONAL CONFERENCE
ON EMBEDDED SYSTEMS
AND INTELLIGENT TECHNOLOGY**

ICESIT 2011
International Conference
on Embedded Systems
and Intelligent Technology

*9 - 11 February 2011,
Prince of Songkla University,
Phuket Campus, Thailand*

Routing Protocol Development for Real Wireless Sensor Network

S. Phatthanatraiwat, W. Suntiarnrut

*Department of Computer Engineering, Faculty of Engineering,
Prince of Songkla University, Hatyai, Songkhla 90112 Thailand
{sphatthanatraiwat@gmail.com,wannarat@coe.psu.ac.th}*

Abstract: In this paper, wireless sensor network has been employed to collect the data and control the electronic devices in a shrimp hatchery farm. More than 20 our in-house sensor nodes called Unode were used to monitor the water temperature and 24 air-conditions were controlled automatically by 12 nodes. Our routing protocol has been applied in this real system. In this work, we present the packet success ratio and the latency of our routing protocol targeting in the real environment of the shrimp hatchery farm.

Keywords: wireless sensor networks, shrimp hatchery farm, routing protocol

I. INTRODUCTION

Wireless Sensor network (WSN) has been widely used in many applications [1-3] for instances, monitoring environmental or agricultural condition in the fields and enemies detection in the military system. In the past, each sensor is connected by wires to transmit the data back to the server. Therefore, the wired sensor is not flexible and expensive when several sensors are required. In a recent year, wireless sensor network is introduced to be another interested solution. The concept of wireless sensor network composes of a hundred or thousand small devices. Each small device called *mote* or *node* consists of the sensing, data processing and RF communication. Motes can form a network and send data to each other wirelessly.

The IEEE 802.15.4 has been announced as a standard [4] for a low-power Low-Rate Wireless Personal Area Network (LR-WPAN). The network topology in wireless sensor network can be changed during deployment where supported by IEEE 802.15.4.

According to the limited range of radio communication, mote has to forward data over several hops to reach its external base-station (BS). Thus WSN has to be deployed in an ad hoc manner and the routing protocol is also required. Apart of short range communication in WSN, energy is a limitation of mote. The energy source of such small device is a battery. Therefore, energy awareness is needed to be concerned in routing protocol development. Another constrain of WSN is

bandwidth. All these limitations challenge to develop routing protocol in WSN.

The characteristic of wireless sensor network distinguishes from other wireless networks. First, it is not possible to build a global addressing scheme for a large number of sensor nodes. The ID maintenance can be a high overhead. Thus IP-based protocols may not be used in WSN. Second, almost all applications in WSN require the flow of sensed data from multiple sources to a particular base-station (BS). This can make a data collision when every sensor sends the data at the same time. Finally, there is a high probability to have some data redundancy. This redundancy can improve in routing protocol.

This paper proposes a new routing protocol targeting to apply in shrimp hatchery farm. Our routing protocol development needs to concern to data sensing, reporting, fault tolerance, and data aggregation. First, our application requires various data sensing and reporting models. The time-driven delivery model is suitable when the data is monitored periodically. However, the event-driven is also required when motes have to react immediately to sudden changes in the environment. Second, when sensor nodes fail or be blocked due to physical damage or environment interference, it should not affect the sensor network. MAC and routing protocol must find out the new link and forward the data to base station. The last concern is to aggregate the similar packets from different motes. So the number of transmission is reduced.

The remainder of this paper is organized as follows. Section II describes the state-of-the-art in routing protocols of WSN. The requirements of shrimp hatchery farm system are explained shown in section III. The proposed routing protocol is presented in section IV. The environment of the system testing and Unode are explained in section V. In section IV, the evaluation of our routing protocol is discussed. Then we draw the conclusion in section VII.

TABLE I
ROUTING PROTOCOLS IN WSN

Type	Class	Protocol Examples
Network Structure	Flat	SPIN and Direct diffusion
	Hierarchical	LEACH, PEGASIS, TEEN and APTEEN
	Location-based	GPS
	Ad hoc routing	tinyLunar, tinyAODV and LoWPAN-AODV

II. RELATED WORK

The authors in paper [5] survey the state-of-the-art in routing protocols of wireless sensor networks. They classify routing protocols using their network structure and protocol operation as shown in Table I. However, we will only explain the network structure due to it is widely used in WSN.

1) Flat routing protocol

Sometime is called data centric routing. Each node performs the same function, sensing and transmitting the data to base-station. Examples of this routing protocol begin with SPIN[6] and direct diffusion[7]. These two routing protocol were aiming for energy saving and elimination of redundant data. The advantage of SPIN protocol is each node only needs to know its single-hop neighbours. Meanwhile, direct diffusion is suitable to the scenario that BS send queries to the sensor nodes by flooding some tasks and also supports data aggregation and caching very well.

2) Hierarchical routing

This type of routing protocol has some advantages on scalability and efficient communication. The whole idea is to create a cluster head and assign special task to it. This can contribute to overall system scalability, lifetime and energy efficiency. Low Energy Adaptive Clustering Hierarchy (LEACH)[8], Power-Efficient Gathering in Sensor Information Systems (PEGASIS)[9] and Threshold-sensitive Energy Efficient Protocol (TEEN)[10] are well-known examples.

3) Location based routing

This routing protocol focuses on the locations of sensor nodes that can be estimated on the incoming signal strength or using Global Positioning System (GPS). Due to this routing protocol is not suitable for our application domain, we will not discuss in more detail.

4) Ad hoc routing

In wireless sensor network, the data is required to be forwarded through the network and avoiding unnecessary resending. The great number of nodes has been deployed in the field randomly. If some individual node fails temporary or permanently, wireless sensor network has to continue operations. Therefore, the routing protocol is required to have self-organization and self-maintaining. This kind of routing protocol is called ad hoc. Because of the processing, storage capacities and power source limitation, routing protocol in wireless sensor network must be small. Ad hoc On-demand Distance Vector (AODV) is well-known routing protocol for Mobile Ad hoc Network (MANET). This network is widely used in mobile computing devices such as PDA or laptop. There are many differences between MANET and wireless sensor network. First, MANET is aiming to use on high-end device in contrast to WSN. Second, nodes in MANET are allowed to move freely while nodes in WSN are mostly fixed after deployment.

The concept idea of routing protocol in WSN is similar to the routing protocol in ad hoc network because WSNs are adapted from ad hoc network. Ad hoc routing protocols such as LoWPAN-AODV, tinyAODV[11] in TinyOS and tinyLUNAR[12] have been proposed recently to employ in wireless sensor network.

III. REQUIREMENTS

The aim of this work is to implement the ad hoc routing protocols on a real world WSN application. The scenario is the WSN automation systems for shrimp hatchery. There shrimp can be grown under controlled environmental conditions. To control such parameters likes e.g. temperature they have to be measure first. By placing sensor nodes in the hatchery, a higher spatial resolution of the temperature can be measured.

The sensor nodes collect data from temperature sensors corresponding to the programmable frequency. The data will be forwarded hop by hop to gateway node or based station. Then base station could export data to a database and provide access to the data via the internet. It could automatically regulate air conditioning in the hatchery after all data has been computed. Therefore, each sensor reading must reach the user. The WSN can operate in two modes, either sending sensor reading to the base station at the programmable frequency or receiving the command from users to control or reconfigure sensor nodes.

IV. PROPOSED ROUTING PROTOCOL

The proposed routing protocol adapted from concept of ad hoc routing. The proposed routing protocol used flooding to create the route of sensor node. In the proposed routing protocol has three phases as the following:

A. Phase setup

Gateway broadcasts message to the neighbour sensor nodes. Node which is got the message will consider to creating the route. If that node does not used to get the message, node will forward that message to other nodes. The node will be selected to be the path member when it is the quickest respond

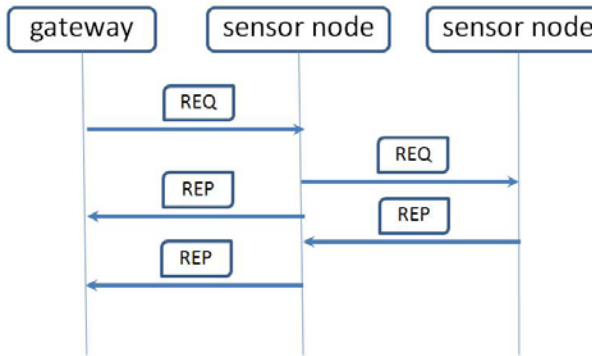


Fig.1 create the route

B. Phase collect data

When the route path is set up, each node will start to get and send the data from its sensors to base station using multi-hop fashion. The data will be buffered on the next node in the sequence list. When the route fails, the data has no need to re-send the data from the original source node.

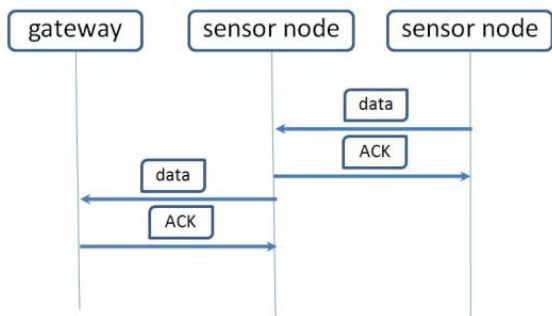


Fig. 2 gateway collected data from sensor node

C. Phase maintenance (connectivity)

When node fails to receive the message in any cases within the interval time, source node will try to send the data twice. If the failure still exists, source node will broadcast message back to gateway.

As soon as gateway gets the message, gateway will be sent message to sensor node, the new route

path will be created after sensor node respond to gateway.

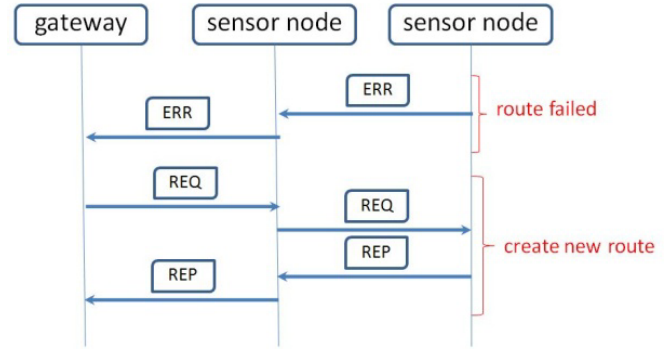


Fig.3 gateway create new route when route failed

V. TEST ENVIRONMENT

The equipment and environment are explained in this

section. Our in-house node named Unode and its operating system (TinyOS) are given in the first part. Then later, the network topologies have been set to evaluate the performance of the proposed routing protocol.

A. Unode and TinyOS

This scenario uses five Unodes running light-weight operating system named TinyOS. Unode is our in-house node which designed to use a 2.4 GHz and support IEEE 802.15.4. Thus CC2420 with 250 kbps data rates is selected to be RF module whereas MSP430f1611 microcontroller is embedded to compute the data. We selected channel 26th to avoid interference with other wireless system[13].

TinyOS has developed by *The University of California, Berkeley* and is programmed by *NesC*. Due to TinyOS was designed for low-power and using a limited resource, it is suitable for WSN. TinyOS supports simple concurrent model with two execution threads.

B. Test topologies

We use three different topologies to evaluate the performance of our routing protocol, using: 4-hop string topology (Fig. 4.a), 2-hop path topology (Fig. 4.b) and 3-hop lollipop topology (Fig. 4.c). All scenarios we test communication between motes S (source) and motes D (Destination) in terms of packet success ratio, end-to-end delay measurements and Route Change (RC) Latency. We set transmission power for transmission range between motes equal to 100 cm.

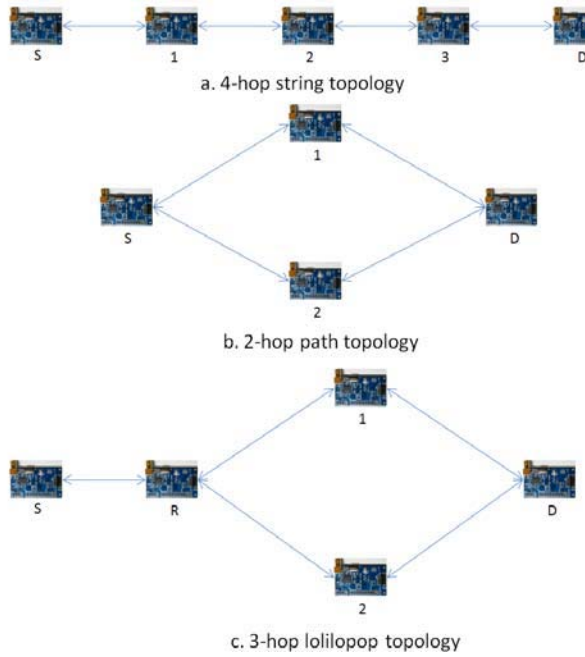


Fig.4 experimental topology

VI. PERFORMANCE EVALUATION

C. Packet success ratio

TABLE 2

DEFINITION OF SCENARIO FOR PACKET SUCCESS RATIO

Scenario	Event
A	The originator mote sends packet data to destination only every 10s
B	Every motes send packet data to destination at the same time (10s)
C	The motes send packet data to destination in order (5, 10, 15)

In experiment we have three scenarios in Table 2. Scenario A source mote (mote S) only sends data packet to destination (mote D), period time to send at 10s. Scenario B every mote in topology sends packet data to mote D, period time to send at 10s. And the last scenario mote S, mote 1, mote 2 and mote 3 send packet data in order period time at (5, 10 and 15).

TABLE 3

PACKET SUCCESS RATIO RESULT

Topology	Scenario		
	A	B	C
4-hop string	97.7%	84.5%	94.5%
2-hop path	99.2%	81.2%	95.2%
3-hop lollipop	98.4%	80.7%	94.6%

Packet success ratio results are shown in table 3. The result of 4-hop string topology, 2-hop path topology and 3-hop lollipop topology are acceptable because packet success ratio of all more than 80%.

D. End-to-end delay measurements

TABLE 4

END-TO-END DELAY RESULT

number of hop	time(ms)
2	40.4
3	55.9
4	76.5

In this experiment we perform end-to-end delay in 4-hop string topology.

Experimental process:

- 1) forcing the motes to be discovered
- 2) motes created route to destination.

After motes created route to destination. The results of end-to-end delay are shown in table 4 and we can conclusion each hop using time estimated 20 ms to end-to-end data delay.

E. RC latency

TABLE 5

DEFINITION OF SCENARIO FOR RC LATENCY

Scenario	Event
A	The originator mote detects a link failure and initiates a new route

In Experiment we perform RC Latency experiment in 2-hop path topology. We perform scenario A in table 5.

Experimental process:

- 1) forcing the motes to be discovered and second step: mote S created route to mote D
- 2) mote S have a route to connection mote D by mote 1
- 3) after that 1 minute we move out mote 1 in topology
- 4) mote S detected link failure and mote S discover route to connection mote D
- 5) mote S connected mote D

We test follow experiment process and RC latency results are shown in Table 6 measurement correspond to average value is equal 95.7 ms for 2-hop topology and 132.5 ms for 3-hop lollipop.

TABLE 6
RC LATENCY RESULT

Topology	RC latency(ms)
2-hop path	95.7
3-hop lollipop	132.5

VII. CONCLUSION AND FUTURE WORK

In this paper, our routing protocol has been proposed. We test routing protocol algorithm in terms of packet success rate, end-to-end delay measurement and RC latency. The performance evaluation of proposed protocol routing in terms of packet success data is acceptable to using in the real system. The performance of end-to-end delay between each hop is estimate at 20 ms.

Additionally, we evaluate RC latency in different topology. And develop proposed routing protocol to better in terms of RC latency and end-to-end delay.

REFERENCE

- [1] F. Akyildiz, W. Su, Y. Sankasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks*, 38:393–422, 2002.
- [2] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In Proc. ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean, 2001.
- [3] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the World with Wireless Sensor Networks. In Proc. Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP 2001), Salt Lake City, Utah, May 2001.
- [4] IEEE 802.15.4, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LRWPANs), IEEE, October 2003.
- [5] Jamal N. Al-Karaki Ahmed E. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey", IEEE in Wireless Communications, Vol. 11, No. 6, pp. 6-28, 2004.
- [6] J. Kulik, W. R. Heinzelman, and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks," *Wireless Networks*, Volume: 8, pp. 169-185, 2002.
- [7] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," *Proceedings of ACM MobiCom '00*, Boston, MA, 2000, pp. 56-67.
- [8] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Micro sensor Networks," *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS '00)*, January 2000.
- [9] S. Lindsey, C. Raghavendra, "PEGASIS: Power-Efficient Gathering in Sensor Information Systems", *IEEE Aerospace Conference Proceedings*, 2002, Vol. 3, 9-16 pp. 1125-1130.
- [10] A. Manjeshwar and D. P. Agarwal, "TEEN: a routing protocol for enhanced efficiency in wireless sensor networks." In 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, April 2001
- [11] C. Gomez, P. Salvatella, O. Alonso, J. Paradells, "Adapting AODV for IEEE 802.15.4 Mesh Sensor Networks:

Theoretical Discussion and Performance Evaluation in a Real Environment," *Proceeding of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06)*, 2006

- [12] Evgeny Osipov, "tinyLUNAR: One-Byte Multihop Communications Through Hybrid Routing in Wireless Sensor Networks," *Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN2007)*, 2007
- [13] "Avoiding RF Interference between WiFi and ZigBee", <http://www.mobiusconsulting.com/papers/ZigBeeWifiInterference.pdf>

ประวัติผู้เขียน

ชื่อ สกุล นายธนัญกรณ์ พัฒนไตรวัฒน์

รหัสประจำตัวนักศึกษา 5110120051

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิศวกรรมศาสตรบัณฑิต (วิศวกรรมคอมพิวเตอร์)	มหาวิทยาลัยสงขลานครินทร์	2550

ทุนการศึกษา (ที่ได้รับในระหว่างการศึกษา)

- ทุนศิษย์ก้นกุฏิ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ประจำปีการศึกษา 2551

การตีพิมพ์เผยแพร่ผลงาน

- T.Phatthanatraiwat and W.Suntiamorntut, "Routing Protocol Development for Real Wireless Sensor Network," *In Proceedings of 4th International Conference on Embedded System and Intelligent Technology 2011 (ICESIT2011)*, Phuket, Thailand, pp. 122-126, 12th – 14th February 2011.