โปรโตคอลค้นหาเส้นทางเอบีอาร์ด้วยตำแหน่งของโฮมาย์โฮสท์สำหรับเครือข่ายไร้สายแอดฮอก

LOCATION-BASED ASSOCIATIVITY-BASED ROUTING PROTOCOL FOR AD-HOC WIRELESS NETWORKS

มนูญ คัมมกสิกิจ

MANOON KUMMAKASIKIT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2549

ISBN 974-15-2335-1

โปรโตคอลค้นหาเส้นทางเอบีอาร์ด้วยตำแหน่งของโมบายล์โฮสท์ สำหรับเครือข่ายไร้สายแอดฮอค

LOCATION-BASED ASSOCIATIVITY-BASED ROUTING PROTOCOL FOR AD-HOC WIRELESS NETWORKS

มนูญ คัมมกสิกิจ

MANOON KUMMAKASIKIT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2549

ISBN 974-15-2335-1

LOCATION-BASED ASSOCIATIVITY-BASED ROUTING PROTOCOL FOR AD-HOC WIRELESS NETWORKS

MANOON KUMMAKASIKIT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF ENGINEERING IN COMPUTER ENGINEERING
SCHOOL OF GRADUATE STUDIES
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2006

ISBN 974-15-2335-1

หัวข้อวิทยานิพนธ์	โปรโตคอลค้นหาเส้นทางเอบีอาร์ด้วยตำแหน่งของโมบายล์โฮสท์สำหรับเครือข่ายสายแอดฮอค
นักศึกษา	นายมนูญ คัมมกสิกิจ
รหัสนักศึกษา	45061054
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
พ.ศ.	2549
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ.ดร.ศักดิ์ชัย ทิพย์จักษุรัตน์

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้เราได้นำเสนอวิธีการปรับปรุงโปรโตคอลค้นหาเส้นทางเอบีอาร์ในเครือข่ายไร้สายแอดฮอค เนื่องจากกระบวนการค้นหาเส้นทางในโปรโตคอลเอบีอาร์ โมไบล์โฮสต์ต้นทางเมื่อต้องการเส้นทางไปยังโมไบล์โฮสต์ปลายทาง จะบรอดคาสต์เมสเสจร้องขอเส้นทางไปสู่โมไบล์โฮสต์ต่างๆ ทั้งหมดในเครือข่าย การบรอดคาสต์เช่นนี้ทำให้โปรโตคอลเอบีอาร์สามารถค้นหาเส้นทางที่มีเสถียรภาพสูงสุดในเวลานั้นได้ แต่จากคุณสมบัติของโมไบล์โฮสต์ในเครือข่ายไร้สายแอดฮอคซึ่งแต่ละโมไบล์โฮสต์สามารถเคลื่อนที่ได้อย่างอิสระ ความเร็วการเคลื่อนที่ของโมไบล์โฮสต์ที่สูงขึ้นจะส่งผลโดยตรงต่อปริมาณโอเวอร์เฮดที่ใช้ในการค้นหาเส้นทาง นอกจากนี้การบรอดคาสต์เมสเสจร้องขอเส้นทางไปทั้งเครือข่ายยังก่อให้เกิดปริมาณโอเวอร์เฮดเป็นจำนวนมาก

ดังนั้นเราจึงได้นำเสนอการปรับปรุงโปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์ โดยตำแหน่งของโมไบล์โฮสต์ ข้อมูลตำแหน่งของโมไบล์โฮสต์นี้จะถูกใช้เพื่อคาดเดาตำแหน่งหรือพื้นที่คาดหวังของโมไบล์โฮสต์ต่างๆที่ต้องการค้นหาเส้นทาง เมสเสจร้องขอเส้นทางจะถูกจำกัดให้อยู่ในพื้นที่เพียงส่วนหนึ่งของเครือข่ายที่คาดหวังว่า จะพบเส้นทางอย่างน้อย 1 เส้นทางซึ่งเรียกว่า พื้นที่การส่งต่อเมสเสจร้องขอเส้นทางที่รวมเอาพื้นที่คาดหวังของโมไบล์โฮสต์ปลายทางเอาไว้ จากกระบวนการจำกัดพื้นที่การบรอดคาสต์เฉพาะกลุ่มนี้ ทำให้สามารถลดปริมาณโอเวอร์เฮดของเมสเสจค้นหาเส้นทางได้ สำหรับโมไบล์โฮสต์ที่มีตำแหน่งอยู่นอกพื้นที่การส่งต่อเมสเสจจะถูกรบกวนจากกระบวนการค้นหาเส้นทางน้อยลง

เราทำการวัดประสิทธิภาพของโปรโตคอลที่เรานำเสนอด้วยการจำลองการทำงานพบว่าโปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์สามารถปรับปรุงประสิทธิภาพของโปรโตคอลค้นหาเส้นทางเอบีอาร์ ในเทอมของโอเวอร์เฮด ดีเลย์ และภาระที่เกิดขึ้นต่อโมไบล์โฮสต์ให้ลดลง นอกจากนี้ยังเพิ่มอัตราการรับส่งข้อมูล และประสิทธิภาพการรับส่งข้อมูลให้สูงขึ้นเมื่อเทียบกับโปรโตคอลค้นหาเส้นทางเอบีอาร์

Thesis Title	Location-Based Associativity-Based Routing Protocol for Ad-Hoc Wireless Networks
Student	Mr. Manoon Kummakasikit
Student ID.	45061054
Degree	Master of Engineering
Programme	Computer Engineering
Year	2006
Thesis Advisor	Asst. Prof. Dr. Sakchai Thipchaksurat

ABSTRACT

In this thesis, we proposed the improvement of Associativity-Based Routing protocol (ABR) in Ad-Hoc wireless networks. In route discovery process of ABR protocol, when mobile host needs a route to destination. It initially broadcasts the route request message to entire of the network. Result in the routing protocol ABR can gets the best stability route in searching duration. But one of the wireless ad-hoc network characteristics, each mobile host can freely move. Higher movement speed result in higher route discovery overhead. Additionally, broadcasting to entire of the network consumes more resource.

We proposed the Modification of Associativity-Based Routing protocol (MABR), by considering mobile host's position. This position information is used to predict the position or expected area of arbitrary mobile host in the network. Route request messages are only broadcasted to limited area that expects 1 or more route will be founded. Called, request zone which included an expected area of the desired host. Also this limitation can reduces more route request messages. Also mobile hosts which are outside the request zone can avoid the route request interruption.

We evaluate performance of our proposed protocol by means of simulation. Simulation results showed that MABR can provide the better performance such as lower overhead, delay, mobile load, higher packet delivery ratio and data throughput comparing with those of ABR protocol.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้อย่างดีด้วยคำแนะนำ และคำปรึกษาจาก ผศ.ดร.ศักดิ์ชัย ทิพย์-จักษุรัตน์ ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ และรศ.ดร.รัตติกร วรากูลศิริพันธุ์ หัวหน้าห้องปฏิบัติการวิจัยเครือข่ายสื่อสาร สำนักวิจัยการสื่อสารและเทคโนโลยีสารสนเทศ (ReCCIT) ข้าพเจ้ารู้สึกทราบซึ้งในความอนุเคราะห์จากท่านอาจารย์ทั้งสองท่าน และขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกคนที่ให้คำแนะนำต่างๆ และคอยให้กำลังใจเสมอมา

ขอขอบคุณบัณฑิตศึกษาและบัณฑิตวิทยาลัย คณะวิศวกรรมศาสตร์ที่ให้ความช่วยเหลือในเรื่องต่างๆ

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจและให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

มนูญ คัมมกสิกิจ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การสื่อสารแบบไร้สายในปัจจุบันได้เข้ามามีบทบาทมากขึ้นในชีวิตประจำวัน เนื่องจากมีความคล่องตัวในการสื่อสารข้อมูล สามารถนำไปประยุกต์ใช้งานได้อย่างหลากหลาย สำหรับระบบเครือข่ายไร้สาย Wireless Local Area Networks หรือ WLAN คือระบบไร้สายชนิดหนึ่ง ที่แบ่งการทำงานออกเป็นสองโหมดคือ Infrastructure และ Ad-hoc ในโหมด Infrastructure แต่ละโมไบล์โฮสต์ซึ่งเป็นลูกข่ายจะมีการติดต่อสื่อสารโดยผ่านตัวกลางที่ทำหน้าที่กระจายสัญญาณเรียกว่าสถานีฐาน (Base Station หรือ Access Point) บริเวณพื้นที่ของเครือข่ายจะถูกจำกัดโดยรัศมีการรับ-ส่งของสถานีฐาน สำหรับการทำงานในโหมดแอดฮอค โมไบล์โฮสต์แต่ละตัวจะเป็นอิสระจากกัน การสื่อสารไม่จำเป็นต้องผ่านตัวกลางที่เป็นสถานีฐาน (Base Station) แต่จะมีการส่งข้อมูลต่อกันไปในลักษณะ Store and Forward ซึ่งแต่ละโมไบล์โฮสต์จะทำตัวเสมือนเราท์เตอร์ที่รับและส่งต่อข้อมูลไปยังโมไบล์โฮสต์ที่อยู่ใกล้เคียงภายในรัศมีการรับ-ส่งจากโมไบล์โฮสต์ต้นทางไปจนถึงโมไบล์โฮสต์ปลายทาง

เนื่องจากการเคลื่อนที่อย่างเป็นอิสระและไม่มีตัวกลางในการควบคุมการสื่อสารข้อมูลทำให้ต้องมีกระบวนการในการจัดการ เพื่อสร้างและดูแลเส้นทางในการสื่อสารข้อมูล เราเรียกกระบวนการเหล่านี้ว่าโปรโตคอลค้นหาเส้นทาง (Routing Protocols) ในวิทยานิพนธ์ฉบับนี้จะแสดงให้เห็นถึงปัญหาของการค้นหาเส้นทางในกลุ่มของโปรโตคอลค้นหาเส้นทางที่เป็นแบบออนดีมานด์ (On-Demand Routing Protocols) โดยโมไบล์โฮสต์ต้นทางจะเป็นผู้ค้นหาเส้นทางไปยังโมไบล์โฮสต์ปลายทาง ปัญหาที่เกิดขึ้นเกิดจากการสูญเสียแบนด์วิธในการส่งเมสเสสร้องขอเส้นทางไปยังเครือข่าย ซึ่งเมสเสสร้องขอเส้นทางนี้จะถูกบรอดคาสต์ออกไปทั้งเครือข่ายเพื่อหาเส้นทางที่เป็นไปได้ การบรอดคาสต์เพื่อหาเส้นทางนี้ ก่อให้เกิดการสูญเสียแบนด์วิธมากเกินความจำเป็น ซึ่งในกระบวนการสุดท้าย เส้นทางจะถูกเลือกนำมาใช้ในการสื่อสารเพียงเส้นทางเดียวเท่านั้น

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

วิทยานิพนธ์ฉบับนี้มีวัตถุประสงค์เพื่อศึกษาและปรับปรุงโปรโตคอลการค้นหาเส้นทางในกลุ่มของโปรโตคอลค้นหาเส้นทางแบบออนดีมานด์ (On-Demand Routing Protocols) โดยในงานวิจัยนี้เราพิจารณาโปรโตคอลค้นหาเส้นทางเอบีอาร์ (Associativity-Based Routing protocol:

ABR) โดยการปรับปรุงการค้นหาเส้นทางด้วยการนำแนวคิดการจำกัดพื้นที่การบรอดคาสต์เมสเสสร้องขอเส้นทางจากโปรโตคอลแอลเออาร์ (Location Aided Routing protocol : LAR) ที่มีการใช้ข้อมูลตำแหน่งของโมไบล์โฮสต์มาประมาณหรือคาดการณ์ตำแหน่งของโมไบล์โฮสต์ปลายทาง เพื่อลดโอเวอร์เฮดที่เกิดจากการบรอดคาสต์เมสเสสร้องขอเส้นทางในเครือข่ายให้อยู่ในเฉพาะพื้นที่การส่งต่อ (Request Zone) ซึ่งทำให้การค้นหาเส้นทางสามารถลดโอเวอร์เฮดและเพิ่มประสิทธิภาพการสื่อสารข้อมูลของเครือข่ายให้สูงขึ้น

1.3 สมมติฐานของการศึกษา

สำหรับโปรโตคอลค้นหาเส้นทางเอบีอาร์ (Associativity Based Routing Protocol : ABR) เป็นโปรโตคอลหนึ่งในกลุ่มของ On-Demand Routing Protocol ซึ่งมีลักษณะเด่นคือโอเวอร์เฮดที่ใช้ในการค้นหาเส้นทางต่ำ เนื่องจากไม่จำเป็นต้องมีขบวนการค้นหาหรือเก็บรักษาเส้นทาง การค้นหาเส้นทางจะเริ่มต้นขึ้นเมื่อมีความต้องการใช้งานเส้นทาง สำหรับโปรโตคอลต้นแบบเอบีอาร์เป็นโปรโตคอลที่มีแนวคิดเพื่อพยายามลดโอเวอร์เฮดที่ใช้ในการค้นหาเส้นทาง (Route Discovery Process) โดยใช้ค่าเสถียรภาพ (Stability) นำมาบ่งบอกความสามารถในการเชื่อมต่อของคู่โมไบล์โฮสต์ต่างๆในเครือข่าย ซึ่งการใช้ค่าเสถียรภาพทำให้มีการเลือกเส้นทางที่คาดว่าจะสามารถใช้งานได้นานกว่า จึงทำให้สามารถลดปริมาณโอเวอร์เฮดที่เกิดขึ้นการการค้นหาเส้นทางทดแทนเนื่องจากความเสียหายที่เกิดขึ้นกับเส้นทางที่ใช้งานอยู่ ซึ่งมักเกิดขึ้นบ่อยครั้งเนื่องจากเครือข่ายไร้สายมักมีการเคลื่อนที่อยู่เสมอ และคาดเดาลักษณะเครือข่ายที่เปลี่ยนแปลงได้ยาก

แต่อย่างไรก็ตามจากการศึกษายังพบข้อด้อยที่เกิดขึ้นจากปริมาณโอเวอร์เฮดที่เกิดจากกระบวนการส่งเมสเสจร้องขอเส้นทาง (Route Request Message) ไปยังเครือข่ายซึ่งเมสเสจร้องขอเส้นทางจะถูกบรอดคาสต์ไปทั่วทั้งเครือข่าย เพื่อหาเส้นทางที่เหมาะสมที่สุด การบรอดคาสต์ดังกล่าวส่งผลให้ทั้งเครือข่ายต้องตอบสนองต่อรีเควสที่เข้ามาทันที ซึ่งส่งผลต่อระบบและความสามารถในการรับส่งข้อมูลของเครือข่าย

เราได้นำแนวคิดการใช้ข้อมูลตำแหน่งมาเพื่อทำนายอาณาบริเวณของโมไบล์โฮสต์ต่างๆ ในเครือข่าย เพื่อให้สามารถจำกัดบริเวณการค้นหาเส้นทางให้อยู่ในพื้นที่ (Request Zone) ซึ่งคาดหวังว่าจะมีเส้นทางไปยังโมไบล์โฮสต์ที่ต้องการค้นหา การจำกัดบริเวณของการบรอดคาสต์เมสเสจร้องขอเส้นทางดังกล่าวทำให้สามารถโอเวอร์เฮดได้ทั้งในส่วนของการค้นหาเส้นทางใหม่ (ไม่มีเส้นทางอยู่เดิมหรือเป็นเส้นทางที่ไม่ได้ใช้งานแล้วช่วงเวลาหนึ่งและถูกลบออกจากตารางเส้นทาง) รวมถึงกระบวนการค้นหาเส้นทางทดแทน

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

วิทยานิพนธ์นี้ได้นำแนวคิดของการใช้ข้อมูลตำแหน่งของโมไบล์โฮสต์เพื่อจำกัดพื้นที่การ บรอดคาสต์เมสเสจร้องขอเส้นทางจากโปรโตคอลแอลเออาร์ (Location-Aided Routing Protocol: LAR) ซึ่งสมมติว่าแต่ละโมไบล์โฮสต์รู้ตำแหน่งพิกัดของตนเองบนพื้นโลกและสามารถเรียกใช้ข้อมูลตำแหน่งนี้ได้ตลอดเวลา โดยในงานวิจัยเราได้สมมติว่าข้อมูลตำแหน่งที่โมไบล์โฮสต์นำมาใช้ในการคำนวณนั้นถูกต้องและไม่มีความผิดพลาด โดยได้นำแนวความคิดของโปรโตคอลแอลเออาร์ไปปรับปรุงกับโปรโตคอลต้นแบบเพื่อให้มีประสิทธิภาพที่สูงขึ้น

1.5 การเปรียบเทียบระหว่างวิธีการที่นำเสนอกับวิธีการแบบพื้นฐาน

โปรโตคอลต้นแบบเอบีอาร์ (Associativity Based-Routing Protocol: ABR) มีข้อเด่นในการลดปริมาณโอเวอร์เฮดของการค้นหาเส้นทางทดแทนโดยเลือกเส้นทางที่คาดว่าจะสามารถใช้สื่อสารข้อมูลได้นาน สำหรับโปรโตคอลที่นำเสนอนอกจากจะรวมเอาความสามารถในการลดโอเวอร์เฮดของการค้นหาเส้นทางทดแทนแล้ว ยังสามารถลดปริมาณโอเวอร์เฮดที่เกิดจากการค้นหาเส้นทางใหม่ซึ่งอาจจะเคยหรือไม่เคยมีข้อมูลเส้นทางในตารางเส้นทาง (Routing Table) มาก่อนได้ โดยงานวิจัยที่นำเสนอจะยังคงความสามารถเดิมของโปรโตคอลต้นแบบไว้ทั้งหมด แต่เพิ่มส่วนของการคำนวณพื้นที่การส่งต่อเมสเสจร้องขอเส้นทางและวิธีการจัดการข้อมูลเส้นทางซึ่งได้สร้างตารางตำแหน่งที่เรียกว่าพีไอที (Position Information Table: PIT) เพื่อเก็บข้อมูลตำแหน่งของโมไบล์โฮสต์ในเครือข่ายและเวลาที่ได้ทำการบันทึกข้อมูลตำแหน่งของโมไบล์โฮสต์นั้น สำหรับตารางข้อมูลตำแหน่งจะมีการปรับปรุงไปตามเมสเสจต่างๆที่ได้รับมาจากเครือข่าย

1.6 ขอบเขตการวิจัย

วิทยานิพนธ์นี้เราปรับปรุงโปรโตคอลค้นหาเส้นทางให้มีประสิทธิภาพที่ดีขึ้นและเปรียบเทียบประสิทธิภาพที่ได้กับโปรโตคอลต้นแบบเอบีอาร์ รวมทั้งพยายามหาข้อจำกัดโดยหาสภาวะแวดล้อมที่เหมาะสมและไม่เหมาะสมกับงานวิจัย เราได้วัดประสิทธิภาพของโปรโตคอลที่นำเสนอด้วยการจำลองการทำงานของระบบ ด้วยโปรแกรมจำลองการทำงานเครือข่ายไร้สายเอ็นเอสทู (Network Simulator version 2: NS-2) โดยทำการเปรียบเทียบกับโปรโตคอลค้นหาเส้นทางเอบีอาร์ ประสิทธิภาพที่เราสนใจคือ โอเวอร์เฮด (Control Overhead), อัตราส่วนความสามารถในการรับส่งข้อมูล (Packet Delivery Fraction), ดีเลย์ (Delay), ค่าประสิทธิภาพในการรับส่งข้อมูล (Data Throughput) และภาระโหลดที่เกิดขึ้นต่อโมไบล์โฮสต์ (Mobile Host Relay Load)

1.7 ขั้นตอนของการศึกษา

วิทยานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 7 บทคือ

บทที่ 1 กล่าวถึงความเป็นมาของงานวิจัย ความมุ่งหมายและวัตถุประสงค์ สมมติฐาน ทฤษฎีที่ใช้ ขอบเขตของการวิจัย และขั้นตอนการศึกษา

บทที่ 2 กล่าวถึงกลุ่มโปรโตคอลค้นหาเส้นทางแบบต่างๆ ที่สำคัญและนิยมอยู่ในปัจจุบัน ซึ่งประกอบด้วยโปรโตคอลค้นหาเส้นทางแบบ Table-Driven Routing Protocol และ Source-initiated On-demand Routing Protocol

บทที่ 3 กล่าวถึงวิธีการใช้ข้อมูลตำแหน่งในโปรโตคอลค้นหาเส้นทาง

บทที่ 4 กล่าวถึงโปรโตคอลที่เกี่ยวข้องกับงานวิจัย ซึ่งประกอบด้วยโปรโตคอลต้นแบบที่นำมาปรับปรุงการค้นหาเส้นทางเอบีอาร์ (Associativity Based-Routing protocol: ABR) และโปรโตคอลที่นำแนวคิดการใช้ข้อมูลตำแหน่งเพื่อลดปริมาณการบรอดคาสต์เมสเสจร้องขอเส้นทางแอลเออาร์ (Location-Aided Routing Protocol: LAR)

บทที่ 5 ขั้นตอนการอิมพลีเมนต์โปรโตคอลลเอบีอาร์และการอิมพลีเมนต์โปรโตคอลที่นำเสนอรวมถึงผลและปัญหาที่เกิดขึ้นจากการจำลองการทำงาน

บทที่ 6 ผลการทดลองหรือวิเคราะห์ซึ่งได้รับโดยนำมาเปรียบเทียบประสิทธิภาพกับโปรโตคอลค้นหาเส้นทางเอบีอาร์

บทที่ 7 สรุปผลการวิจัยและข้อเสนอแนะ

บทที่ 2
โปรโตคอลการค้นหาเส้นทาง

หัวข้อนี้จะกล่าวถึงเครือข่ายไร้สายแอดฮอคและกระบวนการค้นหาเส้นทางต่างๆ รวมถึงแนะนำโปรโตคอลต่างๆที่ใช้ในการค้นหาเส้นทาง ซึ่งแบ่งออกเป็นสองกลุ่มคือ Table Driven Routing Protocol และ On-demand Routing Protocol

เครือข่ายไร้สายแอดฮอคประกอบด้วยโมไบล์โฮสต์ที่เคลื่อนที่อย่างอิสระภายในพื้นที่ของเครือข่าย ซึ่งเชื่อมต่อกันด้วยรัศมีทำการของแต่ละโมไบล์โฮสต์ เนื่องจากแต่ละโมไบล์โฮสต์มีการเคลื่อนที่อย่างอิสระทำให้โครงสร้างของเครือข่ายมีการเปลี่ยนแปลงอย่างไม่มีรูปแบบที่แน่นอนตลอดเวลา ปัญหาสำคัญคือการค้นหาเส้นทางระหว่างโมไบล์โฮสต์ต้นทางไปยังปลายทางซึ่งซับซ้อนกว่าเมื่อเทียบกับระบบการส่งข้อมูลที่ใช้เพียงฮอพเดียว (Single Hop) เช่นในระบบเครือข่ายโทรศัพท์เคลื่อนที่เซลลูลาร์ ซึ่งเครือข่ายประเภทนี้แต่ละโมไบล์โฮสต์จะต้องติดต่อผ่านไปยังตัวกลางที่เป็นสถานีฐาน (Base Station) เท่านั้น ก่อนที่จะส่งต่อข้อมูลต่อไปยังโมไบล์โฮสต์ปลายทาง สำหรับโปรโตคอลต่างๆที่ใช้ในการค้นหาเส้นทางสำหรับเครือข่ายไร้สายแอดฮอคนั้นมีหลายรูปแบบแบ่งได้เป็นสองกลุ่มคือ

2.1 Table-Driven Routing Protocols

โปรโตคอลกลุ่มนี้จะมีการเก็บข้อมูลเส้นทางเอาไว้ภายในโมไบล์โฮสต์เรียกว่าตารางเส้นทาง (Routing Table)โดยข้อมูลเส้นทางในตารางเส้นทางต้องมีการปรับปรุงให้ใหม่อยู่เสมอ โดยการปรับปรุงข้อมูลในตารางเส้นทางเกิดขึ้นเมื่อเครือข่ายมีการเปลี่ยนแปลงรูปแบบโครงสร้าง สำหรับการปรับปรุงข้อมูลในตารางเส้นทางอาจจะปรับปรุงทั้งเครือข่ายหรือปรับปรุงเฉพาะบางส่วนที่มีโครงสร้างเครือข่ายเปลี่ยนแปลงไปก็ได้ ทั้งนี้ขึ้นอยู่กับรายละเอียดของแต่ละโปรโตคอล สำหรับกลุ่มของโปรโตคอลแบบนี้จะมีการค้นหาเส้นทางไว้ก่อน ซึ่งไม่ทราบล่วงหน้าว่าเส้นทางใดจะถูกใช้เมื่อใด จึงมีการเก็บเส้นทางเอาไว้จำนวนมาก และจะต้องมีการปรับปรุงข้อมูลในตารางเส้นทางของตนให้ใหม่อยู่เสมอเพื่อให้พร้อมหากมีความต้องการใช้เส้นทางในอนาคต ลักษณะการค้นหาเส้นทางและปรับปรุงตารางเส้นทางนั้น แต่ละโมไบล์โฮสต์จะส่งข้อมูลเส้นทาง (Route Update Message) ไปยังโมไบล์โฮสต์ข้างเคียงทุกโมไบล์โฮสต์ เพื่อให้โมไบล์โฮสต์ที่อยู่ข้างเคียงรับทราบสถานะการเชื่อมต่อและเก็บข้อมูลจากโมไบล์โฮสต์ต่างๆ ภายในเครือข่ายเพื่อนำมาสร้างหรือปรับปรุงตารางเส้นทางจากข้อมูลเหล่านี้ ตัวอย่างของโปรโตคอลที่เป็นที่รู้จักในกลุ่มของ Table-Driven Routing มีดังนี้

2.1.1 DBF (Distributed Bellman-Ford) [1]

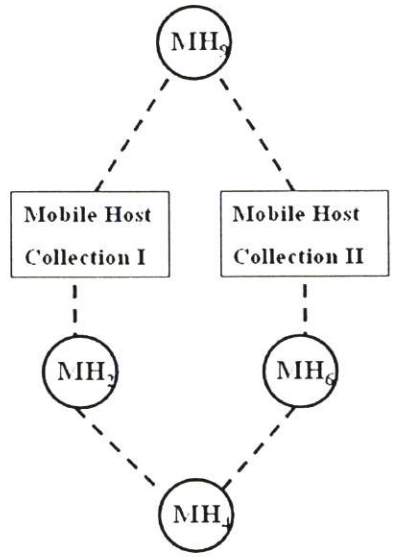
อัลกอริทึมดีบีเอฟเป็นอัลกอริทึมที่ถูกพัฒนาขึ้นเพื่อนำไปใช้งานในการค้นหาเส้นทางในระบบ ARPANET ซึ่งเป็นที่รู้จักคือ (Routing Internet Protocol: RIP) และยังคงมีการใช้งานอยู่จนถึงปัจจุบันในอินเทอร์เน็ตโดเมนบางแห่ง แนวคิดของโปรโตคอลจะพยายามให้ข้อมูลในตารางเส้นทางใหม่อยู่เสมอและพร้อมนำมาใช้งานได้ตลอดเวลา ซึ่งตารางเส้นทางจะเก็บข้อมูลของโมไบล์โฮสต์ปลายทางไว้มากที่สุดเท่าที่สามารถทำได้ ข้อมูลในตารางข้อมูลจะเก็บเส้นทางไปยังโมไบล์โฮสต์ปลายทางเอาไว้ตลอดจนเก็บข้อมูลของเส้นทางนั้น เช่น ดีเลย์ (Delay), จำนวนฮอพ (Hop Count) เป็นต้น แต่ละโมไบล์โฮสต์จะเริ่มต้นโดยการส่งเมสเสจซึ่งมีข้อมูลของเส้นทางไปยังโมไบล์โฮสต์ข้างเคียงเพื่อบอกถึงเส้นทางและข้อมูลระยะทางที่ตนมี จากข้อมูลตารางเส้นทาง โมไบล์โฮสต์ใดที่ได้รับเมสเสจจะทำการตรวจสอบและปรับปรุงตารางเส้นทางของตนให้สอดคล้องกับเมสเสจที่ได้รับมา สำหรับช่วงเวลาหนึ่ง จะมีเส้นทางเพียงเส้นทางเดียวที่ถูกเลือกให้เป็นเส้นทางที่ดีที่สุดจากเส้นทางทั้งหมดที่ไปยังปลายทางได้ โดยเส้นทางที่ดีที่สุดนี้พิจารณาได้จากสมการ

$$D(i, j) = \min[d(i,k) + D(k, j)] \tag{2.1}$$

โดยที่ D(i,j) คือค่าระยะเส้นทางที่สั้นที่สุดจากโมไบล์โฮสต์ i ไปยังโมไบล์โฮสต์ j และ d(i, k) เป็นเวลาที่ใช้ในการส่งข้อมูลจากโมไบล์โฮสต์ i ไปยังโมไบล์โฮสต์ k โดยที่โมไบล์โฮสต์ k เป็นโมไบล์โฮสต์ข้างเคียงของโมไบล์โฮสต์ i หลังจากนั้นจะคำนวณค่าระยะทางแล้วส่งข้อมูลเส้นทางและระยะทางแลกเปลี่ยนไปให้กับโมไบล์โฮสต์ข้างเคียง

2.1.2 DSDV (Destination-Segmented Distance-Vector Routing Protocol) [2]

DSDV เป็นโปรโตคอลเพื่อค้นหาเส้นทางที่ใช้หลักการของ Bellman-Ford โปรโตคอลนี้จะพิจารณาเส้นทางที่ไม่ก่อให้เกิดลูปขึ้น ตารางเส้นทาง (Routing Table) จะเก็บเส้นทางไปยังโมไบล์โฮสต์อื่นๆ ในเครือข่ายให้ได้มากที่สุดเท่าที่จะเป็นไปได้ นอกจากนี้จะเก็บข้อมูลจำนวนฮอพ (Hop) ซึ่งเป็นระยะทางไปยังปลายทางไว้ในตารางเส้นทางนี้ด้วย สำหรับข้อมูลเส้นทางจะถูกกำกับไว้ด้วยลำดับหมายเลข (Sequence Number) ซึ่งกำหนดโดยโมไบล์โฮสต์ปลายทาง ด้วยวิธีการที่ใช้ลำดับหมายเลขนี้จะทำให้สามารถแยกความแตกต่างระหว่างเส้นทางเก่าและเส้นทางใหม่ที่ถูกสร้างขึ้น เพื่อป้องกันปัญหาของการเกิดลูปในเส้นทาง สำหรับขั้นตอนการปรับปรุงตารางเส้นทาง จะแบ่งออกเป็นสองวิธี คือการปรับปรุงตารางเส้นทางโดยแลกเปลี่ยนข้อมูลเส้นทางเพื่อปรับปรุงตารางเส้นทางทั้งเครือข่าย และการปรับปรุงเฉพาะส่วนที่เกิดความเปลี่ยนแปลงของเครือข่าย เพื่อให้ข้อมูลในตารางเส้นทางของทุกๆ โมไบล์โฮสต์สามารถพร้อมที่จะนำมาใช้งานได้ตลอดเวลา



รูปที่ 2.1 การบรอดคาสต์เมสเสจอัพเดทเส้นทาง

จากรูปที่ 2.1 เมื่อโมไบล์โฮสต์ MH_9 บรอดคาสต์เมสเสจอัพเดทไปยังกลุ่มของโมไบล์โฮสต์ I และ II โมไบล์โฮสต์ MH_2 จะส่งต่อข้อมูลเมสเสจอัพเดทเส้นทางต่อไปยังโมไบล์โฮสต์ MH_4 เมื่อโมไบล์โฮสต์ MH_4 ได้รับเมสเสจอัพเดทเส้นทางจะตรวจสอบว่ามีลำดับหมายเลขในเมสเสจเป็นลำดับหมายเลขใหม่ซึ่งเป็นเส้นทางไปยังโมไบล์โฮสต์ MH_9 หรือไม่ ถ้าเป็นชุดข้อมูลลำดับหมายเลขใหม่ จะทำการอัพเดทตารางเส้นทางของตน และบรอดคาสต์เมสเสจอัพเดทเส้นทางต่อไป จากรูปที่ 2.1 เมื่อโมไบล์โฮสต์ MH_6 ได้บรอดคาสต์เมสเสจอัพเดทเส้นทางมายังโมไบล์โฮสต์ MH_4 ด้วยค่าลำดับหมายเลขของเมสเสจค่าเดียวกัน แต่มีเส้นทางที่ดีกว่า หรือสั้นกว่าเส้นทางเดิมที่ MH_4 เคยมีและบันทึกไว้ในตารางเส้นทางแล้ว โมไบล์โฮสต์ MH_4 จะทำการปรับปรุงตารางเส้นทางใหม่ตามข้อมูลเมสเสจอัพเดทเส้นทางที่ได้รับจากโมไบล์โฮสต์ MH_6 แล้วจึงบรอดคาสต์เมสเสจอัพเดทต่อไป

2.1.3 DREAM (Distance Routing Effect Algorithm for Mobility) [3]

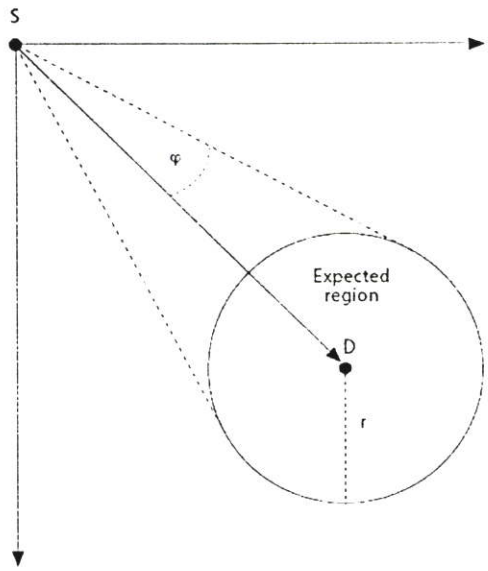
DREAM เป็นโปรโตคอลที่อาศัยการแลกเปลี่ยนข้อมูลตำแหน่งของโมไบล์โฮสต์ โดยแต่ละโมไบล์โฮสต์จะมีตารางตำแหน่ง (Location Table) ที่ใช้สำหรับเก็บข้อมูลตำแหน่งของโมไบล์โฮสต์ต่างๆที่อยู่ในเครือข่าย ซึ่งข้อมูลนี้จะมีการแลกเปลี่ยนกันอยู่ตลอดเวลา โดยการแลกเปลี่ยนข้อมูลตำแหน่งจะเกิดขึ้นเป็นคาบเวลาซึ่งค่าของคาบเวลาอาจจะไม่เท่ากันขึ้นกับระยะห่างของโมไบล์โฮสต์ หากโมไบล์โฮสต์อยู่ห่างกันมาก การปรับปรุงข้อมูลตำแหน่งจะกระทำช้ากว่าการปรับปรุงข้อมูลตำแหน่งที่โมไบล์โฮสต์อยู่ใกล้กว่า ทำให้โปรโตคอล DREAM สามารถจำกัดโอเวอร์เฮดที่เกิดขึ้นจากการปรับปรุงข้อมูลตำแหน่ง

ข้อมูลตำแหน่งที่มีการแลกเปลี่ยนกันระหว่างโมไบล์โฮสต์ต่างๆ จะประกอบด้วยค่าตำแหน่ง ความเร็วและเวลาที่ได้ทำการส่งเมสเสจแลกเปลี่ยนข้อมูลนั้น ในการทำงาน สมมติให้

โมไบล์โฮสต์ S ต้องการติดต่อสื่อสารข้อมูลกับโมไบล์โฮสต์ปลายทาง D โมไบล์โฮสต์ S จะเริ่มต้นโดยการคำนวณค่าพื้นที่วงกลมซึ่งล้อมรอบโมไบล์โฮสต์ปลายทาง D จากข้อมูลตำแหน่งที่มีอยู่ โดยใช้ความเร็วของโมไบล์โฮสต์ D ที่ โมไบล์โฮสต์ S มีอยู่ซึ่งจะได้พื้นที่วงกลมล้อมรอบโมไบล์โฮสต์ D ด้วยรัศมี R จากสมการ

$$R = V_{\max} \times (t_1 - t_0) \tag{2.2}$$

โดยมีจุดศูนย์กลางวงกลมที่ตำแหน่ง (X_d, Y_d) หลังจากที่การคำนวณวงกลมเสร็จสิ้น โมไบล์โฮสต์ S จะถูกสมมติให้เป็นตำแหน่งก้นของกรวยที่มีปากกรวยเป็นวงกลมรัศมี R สำหรับพื้นที่ภายใน



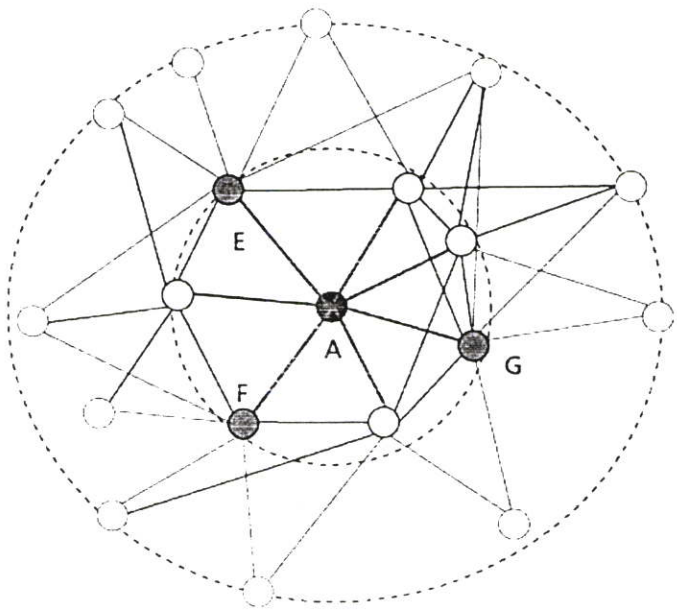
รูปที่ 2.2 การส่งต่อเมสเสจร้องขอเส้นทางของโปรโตคอล DREAM

กรวย จะถูกใช้เป็นพื้นที่ในการส่งต่อข้อมูล สำหรับข้อมูลที่ DREAM ส่งออกไปนั้นจะไม่ใช่เมสเสจร้องขอเส้นทางแต่จะเป็นข้อมูลที่ต้องการสื่อสาร เมื่อโมไบล์โฮสต์ D ได้รับแพคเกจข้อมูลแล้วจะตอบกลับแพคเกจข้อมูลนั้นด้วยแพคเกจตอบรับ (ACK) โดยทั่วไปแล้วแพคเกจ ACK อาจจะส่งกลับไปไม่ถึงโมไบล์โฮสต์ต้นทาง S ได้เนื่องจากไม่มีเส้นทางระหว่างโมไบล์โฮสต์ปลายทาง D และโมไบล์โฮสต์ต้นทาง S หรือเกิดความผิดพลาดระหว่างการส่งข้อมูล

2.1.4 OLSR (Optimized Link State Routing) [4]

โอแอลเอสอาร์เป็นโปรโตคอลที่พิจารณาสถานะของลิงค์การเชื่อมต่อระหว่างโมไบล์โฮสต์ โดยจะมีการแลกเปลี่ยนข้อมูลสถานะการเชื่อมต่อกับโมไบล์โฮสต์ต่างๆ โปรโตคอลจะคำนวณหา Multi-Point Relays (MPRs) ซึ่งเป็นกลุ่มของโมไบล์โฮสต์ที่จะมีการแลกเปลี่ยนข้อมูลสถานะการเชื่อมต่อ สำหรับโมไบล์โฮสต์รอบข้างนอกเหนือจากเอ็มพีอาร์เซตนี้จะไม่มีการ

แลกเปลี่ยนข้อมูลสถานะการเชื่อมต่อ ทั้งนี้เพื่อลดโอเวอร์เฮดในการแลกเปลี่ยนข้อมูลเส้นทางระหว่างโมไบล์โฮสต์ลง ทำให้การบรอดคาสต์เมสเสจร้องขอเส้นทางมีประสิทธิภาพมากขึ้น สมมติโมไบล์โฮสต์ A มีการบรอดคาสต์เมสเสจ Hello ไปยังทุกๆ โมไบล์โฮสต์ข้างเคียงเพื่อแลกเปลี่ยนสถานะการเชื่อมต่อ แต่ละโมไบล์โฮสต์จะคำนวณเอ็มพีอาร์เซต (MPR) ดังในรูปที่ 2.3 เป็นตัวอย่างของเอ็มพีอาร์เซตของโมไบล์โฮสต์ A ที่มีเอ็มพีอาร์เซตเป็นโมไบล์โฮสต์ E, F และ G



รูปที่ 2.3 แสดงเซตเอ็มพีอาร์ของโมไบล์โฮสต์ A

2.2 On-demand Routing Protocols (Source Initiated)

โปรโตคอลในกลุ่มนี้จะแตกต่างจากโปรโตคอลในกลุ่ม Table-driven ที่จะมีการค้นหาเส้นทางและสร้างเส้นทางก็ต่อเมื่อโมไบล์โฮสต์ต้นทางมีความต้องการเส้นทางเท่านั้น โดยโมไบล์โฮสต์ต้นทางจะเริ่มต้นด้วยกระบวนการค้นหาเส้นทางที่เรียกว่า Route Discovery Process ซึ่งโมไบล์โฮสต์ต้นทางจะบรอดคาสต์เมสเสจร้องขอเส้นทางไปยังเครือข่าย โมไบล์โฮสต์ต่างๆ ภายในเครือข่ายจะตอบสนองต่อเมสเสจร้องขอเส้นทาง โดยพยายามส่งต่อเมสเสจไปยังโมไบล์โฮสต์รอบข้างกระจายออกไปทั่วทั้งเครือข่าย เมื่อเมสเสจร้องข้อเส้นทาง เดินทางไปถึงโมไบล์โฮสต์ปลายทาง โมไบล์โฮสต์ปลายทางจะตอบกลับด้วยเมสเสจตอบกลับ (Reply Message) ด้วยเส้นทางเดิมที่เมสเสจร้องขอเส้นทางถูกส่งมาถึง ซึ่งข้อมูลนี้จะถูกนำไปใช้เป็นเส้นทางที่โมไบล์โฮสต์ต้นทางจะนำไปปรับปรุงตารางเส้นทางและใช้เส้นทางนี้ในการสื่อสารข้อมูลต่อไป

หลังจากเส้นทางการสื่อสารข้อมูลถูกสร้างขึ้นแล้ว กระบวนการดูแลรักษาเส้นทาง (Route Maintenance) จะมีหน้าที่ดูแลเส้นทางที่ใช้ในการเชื่อมต่อ หากเส้นทางเกิดความเสียหายก็จะ

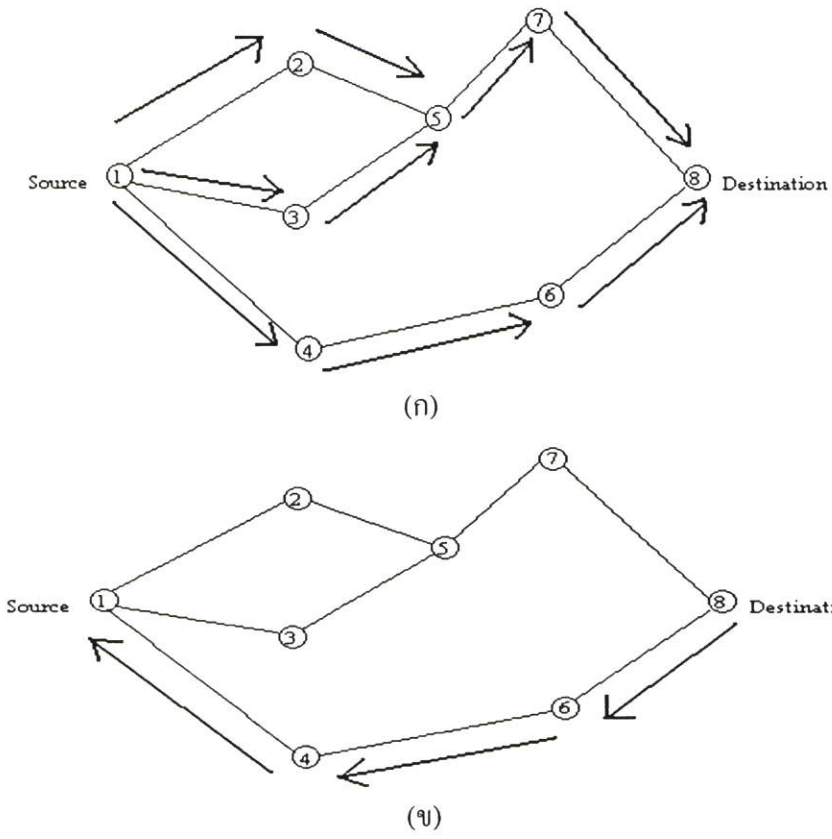
พยายามค้นหาเส้นทางอื่นๆที่ทำให้การสื่อสารข้อมูลดำเนินต่อไปได้ จนกว่าจะมีการยกเลิกการใช้งานเส้นทาง ตัวอย่างของ On-Demand Routing Protocols มีดังต่อไปนี้

2.2.1 AODV (Ad-hoc On-demand Distance Vector Routing Protocol)[5]

AODV เป็นโปรโตคอลค้นหาเส้นทางซึ่งปรับปรุงมาจากโปรโตคอลค้นหาเส้นทาง (Destination-Segmented Distance-Vector Routing Protocol: DSDV) โดยทำการปรับปรุงกระบวนการบรอดคาสต์เพื่อค้นหาเส้นทางและปรับปรุงข้อมูลเส้นทางให้น้อยลง การค้นหาและสร้างเส้นทางจะเกิดขึ้นเมื่อมีความต้องการใช้เส้นทาง

เมื่อโมไบล์โฮสต์ต้นทางต้องการติดต่อไปยังโมไบล์โฮสต์ปลายทางและพบว่าไม่มีข้อมูลเส้นทางไปยังโมไบล์โฮสต์ปลายทางในตารางเส้นทาง โมไบล์โฮสต์ต้นทางจะบรอดคาสต์เมสเสจร้องขอเส้นทาง RREQ (Route Request Message) ไปยังโมไบล์โฮสต์รอบข้าง ซึ่งเมสเสจจะถูกส่งต่อไปจนกระทั่งพบโมไบล์โฮสต์ปลายทางหรือจนกระทั่งพบโมไบล์โฮสต์ระหว่างทาง (Intermediate Mobile Host: IMH) ที่มีเส้นทางไปยังโมไบล์โฮสต์ปลายทางและใหม่ที่สามารถนำข้อมูลนั้นมาใช้เป็นเส้นทางได้ ในรูปที่ 2.4 จะแสดงให้เห็นถึงเมสเสจร้องขอเส้นทางที่ถูกบรอดคาสต์ไปในเครือข่าย

AODV จะใช้หมายเลขลำดับหมายเลขจากเมสเสจ RREQ เพื่อป้องกันปัญหาที่เกิดจากลูปของเส้นทาง แต่ละโมไบล์โฮสต์จะมีการใช้หมายเลขลำดับหมายเลขคล้ายกับไอดีของการบรอดคาสต์ (Broadcast ID) ซึ่งค่าลำดับหมายเลขจะเพิ่มขึ้นทุกๆครั้งที่ RREQ ถูกสร้างขึ้น โดยโมไบล์โฮสต์ต้นทางซึ่งจะแนบข้อมูลลำดับหมายเลขนี้ไปกับข้อมูลตำแหน่งไอพี (IP Address) ของโมไบล์โฮสต์เพื่อให้มั่นใจได้ว่าจะไม่มี RREQ ที่ซ้ำกันถูกส่งอยู่ในเครือข่าย ในช่วงการบรอดคาสต์ RREQ โมไบล์โฮสต์ระหว่างทางจะบันทึกค่าแอดเดรสของโมไบล์โฮสต์ข้างเคียงเอาไว้โดยจะเป็นเมสเสจแรกที่ได้รับมาเท่านั้น ซึ่งถ้ามี RREQ อันเดียวกันแต่มาถึงในภายหลังข้อมูลนี้จะแสดงว่าเป็นข้อมูล RREQ ตัวเดียวกัน และ RREQ ตัวหลังจะถูกดรอบ เมื่อ RREQ ได้ถูกส่งไปถึงโมไบล์โฮสต์ปลายทางหรือโมไบล์โฮสต์ระหว่างทางซึ่งมีข้อมูลเส้นทางที่สามารถติดต่อไปยังโมไบล์โฮสต์ปลายทางได้ จะทำการสร้างเมสเสจตอบกลับที่เรียกว่า RREP (Route Reply Message) ย้อนกลับในทิศทางที่เมสเสจร้องขอเส้นทางได้ถูกส่งมาถึง สำหรับกระบวนการที่เกิดขึ้นเมื่อเส้นทางถูกสร้างขึ้นแล้ว หากโมไบล์โฮสต์ต้นทางได้มีการเคลื่อนที่ออกจากเส้นทางและทำให้เส้นทางการเชื่อมต่อเสียหาย โมไบล์โฮสต์ต้นทางจะเริ่มต้นกระบวนการบรอดคาสต์เมสเสจร้องขอเส้นทางไปยังเครือข่ายใหม่ หากในกรณีที่โมไบล์โฮสต์ระหว่างเส้นทางได้มีการเคลื่อนที่ออกไปทำให้เส้นทางเสียหาย โมไบล์โฮสต์จุดที่เส้นทางเสียหายด้านที่ยังเชื่อมต่อกับโมไบล์โฮสต์ต้นทาง จะส่ง link failure notification Message ย้อนกลับไปยังโมไบล์โฮสต์ต้นทางเพื่อให้โมไบล์โฮสต์ระหว่างเส้นทางนั้นทำการลบเส้นทางนี้ออกไปเนื่องจากเป็นเส้นทางที่ไม่สามารถใช้งานได้อีกต่อไป เมื่อโมไบล์โฮสต์ต้นทางได้รับเมสเสจนี้ก็จะเริ่มกระบวนการค้นหาเส้นทางใหม่อีกครั้ง



รูปที่ 2.4 แสดงการบรอดคาสต์เพื่อหาเส้นทางของโปรโตคอล AODV (route request/reply)

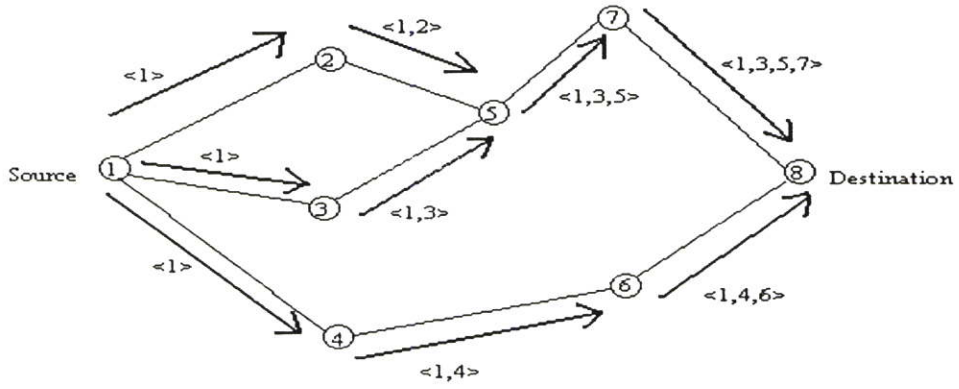
2.2.2 DSR (Dynamic Source Routing) [6]

เป็นโปรโตคอลค้นหาเส้นทางซึ่งโมไบล์โฮสต์ต่างๆจะมีการเก็บข้อมูลเส้นทางลงในความจำชั่วคราว (Cache) ซึ่งข้อมูลในหน่วยความจำนี้จะมีการปรับปรุงอยู่เสมอตามแต่ปริมาณข้อมูลเส้นทางที่ได้รับจากเครือข่าย โปรโตคอลดีเอสอาร์จะประกอบด้วยสองส่วนหลักคือกระบวนการค้นหาเส้นทางและดูแลรักษาเส้นทาง โดยเริ่มต้นจากโมไบล์โฮสต์ต้นทางต้องการสื่อสารกับโมไบล์โฮสต์ปลายทาง จะทำการค้นหาเส้นทางในหน่วยความจำชั่วคราวของตนก่อนว่ามีข้อมูลเส้นทางของโมไบล์โฮสต์ปลายทางอยู่หรือไม่ และถ้ามีจะต้องเป็นข้อมูลเส้นทางที่ยังสามารถนำมาใช้งานได้ แต่ถ้าข้อมูลเส้นทางในหน่วยความจำชั่วคราวไม่มีอยู่หรือมีอยู่แต่ไม่สามารถใช้งานได้แล้ว โมไบล์โฮสต์ต้นทางจะเริ่มต้นค้นหาเส้นทางโดยการบรอดคาสต์เมสเสจร้องขอเส้นทางออกไปยังเครือข่าย สำหรับข้อมูลในเมสเสจร้องขอเส้นทางนี้จะประกอบด้วยแอดเดรสของโมไบล์โฮสต์ปลายทางและต้นทาง รวมถึงไอดีของเมสเสจร้องขอเส้นทาง เมื่อโมไบล์โฮสต์ใดได้รับเมสเสจร้องขอเส้นทางนี้จะทำการตรวจสอบว่าโมไบล์โฮสต์ตนมีข้อมูลเส้นทางไปยังโมไบล์โฮสต์ปลายทางหรือไม่ ถ้าพบว่าไม่มีข้อมูลโมไบล์โฮสต์จะเพียงแค่เพิ่มค่าโมไบล์โฮสต์ไอดีของตนลงในเมสเสจร้องขอเส้นทาง และส่งต่อเมสเสจไปยังโมไบล์โฮสต์รอบข้างต่อไป

เมื่อเมสเสจร้องขอเส้นทางได้ถูกส่งไปถึงโมไบล์โฮสต์ปลายทางหรือถูกส่งไปถึงโมไบล์โฮสต์ระหว่างเส้นทางซึ่งมีข้อมูลเส้นทางสามารถเชื่อมต่อกับโมไบล์โฮสต์ปลายทางได้ โมไบล์โฮสต์

ปลายทางหรือโมไบล์โฮสต์ระหว่างทางจะทำการตอบกลับเมสเสจร้องขอเส้นทางไปยังปลายทางด้วยข้อมูลเส้นทางที่เมสเสจร้องขอเส้นทางเดินทางมาถึง หรืออาจตอบกลับด้วยข้อมูลเส้นทางที่มีอยู่ในหน่วยความจำชั่วคราวในกรณีที่เป็นการตอบกลับจากโมไบล์โฮสต์ระหว่างทาง

สำหรับกระบวนการดูแลรักษาเส้นทางนี้จะใช้เมสเสจเออเรอร์ (Route Error Message) เพื่อแสดงปัญหาเมื่อเลเยอร์ดาต้าลิงค์พบปัญหาของการสื่อสารข้อมูล เมื่อโมไบล์โฮสต์ใดได้รับเมสเสจนี้จะลบเส้นทางที่มีปัญหาออกจากหน่วยความจำเส้นทางชั่วคราว นอกจากนี้ยังใช้เมสเสจตอบรับ (Acknowledgment Message) เพื่อตรวจสอบการทำงานของเส้นทาง

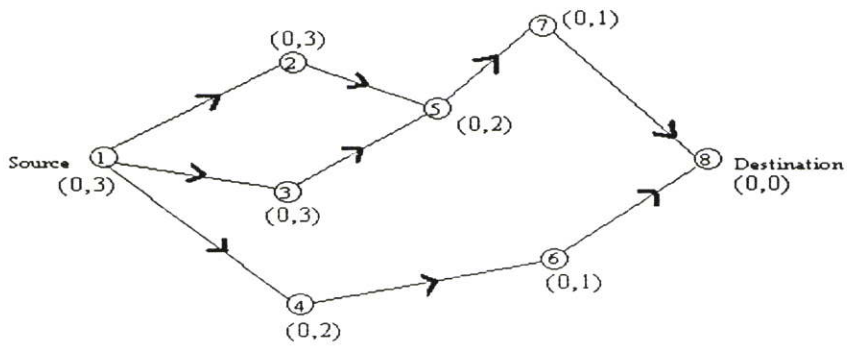


รูปที่ 2.5 การเก็บไอดีและการใช้ไอดีของโมไบล์โฮสต์เป็นเส้นทาง

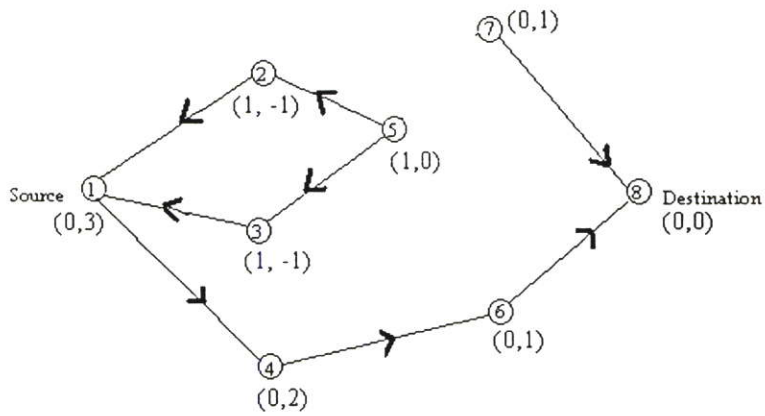
2.2.3 TORA (Temporally Ordered Routing Algorithm) [7]

โปรโตคอลทอราได้ถูกออกแบบมาให้เป็นลักษณะที่ไม่มีลูปด้วยแนวคิดลิงค์รีเวิร์ส โดยได้นำเสนอวิธีการที่มีการส่งข้อมูลเพื่อปรับปรุงตารางเส้นทางให้เฉพาะกับโมไบล์โฮสต์ที่ใกล้เคียงพื้นที่มีการเปลี่ยนแปลงโครงสร้างเครือข่ายเท่านั้น ซึ่งได้แบ่งออกเป็น 3 กระบวนการ คือ การสร้างเส้นทาง การดูแลเส้นทาง และการลบเส้นทาง

ในกระบวนการสร้างเส้นทางและกระบวนการดูแลเส้นทาง โดยจะสร้างกราฟซึ่งมีรากอยู่ที่โมไบล์โฮสต์ปลายทางเรียกว่าดีเอจีกราฟ (Directed Acyclic Graph: DAG) หลังจากนั้นลิงค์ต่างๆจะถูกกำหนดทิศทางให้เป็นทิศทางเข้าหรือออกซึ่งขึ้นกับค่าความสูงกราฟ (Height Metric) ดังแสดงในรูปที่ 2.6 เมื่อเส้นทางมีความเสียหายเกิดขึ้น กระบวนการดูแลเส้นทางจะสร้างกราฟดีเอจีใหม่ ในกรณีที่ลิงค์การเชื่อมต่อของโมไบล์โฮสต์คู่ใดไม่สามารถรับส่งข้อมูลได้ ก็จะมีการเปลี่ยนแปลงค่าไฮต์ให้ลิงค์มีทิศทางตรงข้าม ซึ่งอาจส่งผลให้มีการปรับทิศทางตรงข้ามกับอื่นด้วย หากไม่มีลิงค์ที่ส่งข้อมูลลงสู่รากของกราฟได้ ยังทำให้ค่าความสูงอ้างอิงของกราฟอาจจะมีการเปลี่ยนแปลงไปเมื่อโมไบล์โฮสต์ไม่มีลิงค์ที่สามารถส่งข้อมูลลงสู่รากของกราฟได้เนื่องจากความเสียหายของลิงค์ กระบวนการลบเส้นทาง จะมีการบรอดคาสต์ (Clear Packet: CLR) เพื่อลบเส้นทางที่ไม่สามารถใช้งานได้



(ก)



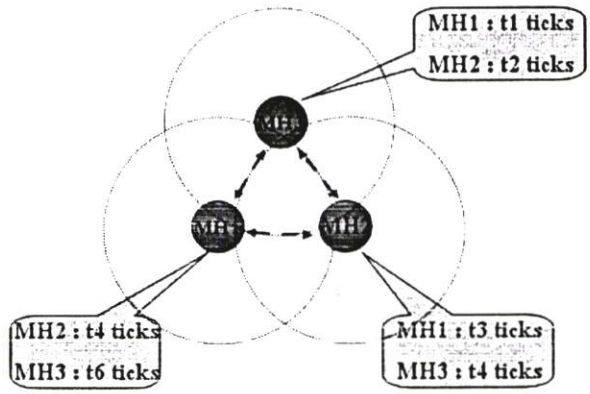
(ข)

รูปที่ 2.6 การสร้างและการทำรีเวิร์สกราฟดีเอจีเมื่อลิงค์เกิดความเสียหาย

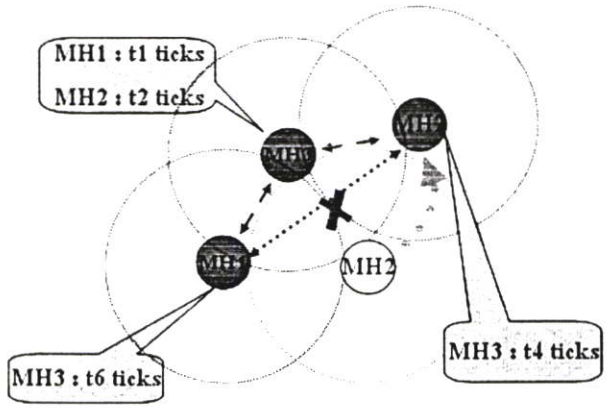
2.2.4 ABR (Associativity-Based Routing Protocol) [8]

โปรโตคอลเอบีอาร์ วิธีการในการค้นหาและเลือกเส้นทางวิธีหนึ่งซึ่งอาศัยหลักการทำงานโดยใช้ associativity tick ซึ่งเป็นตัวตัดสินใจหลักในการเลือกเส้นทาง โดย associativity tick นี้จะมีตารางข้อมูลสำหรับเก็บสถานะของโมไบล์โฮสต์รอบข้าง ข้อมูลนี้เสมือนบอกถึงสถานะของ

ลิงค์ ระหว่างโมไบล์โฮสต์ต่างๆที่อยู่รอบข้าง ทุกๆคาบเวลาหนึ่ง แต่ละโมไบล์โฮสต์จะส่งแพคเกจบีคอน (Beacon) ออกมาเพื่อแจ้งสถานะของตนแก่โมไบล์โฮสต์รอบข้าง เมื่อโมไบล์โฮสต์รอบข้างได้รับบีคอนก็จะบันทึกไว้ว่าได้รับมาจากโมไบล์โฮสต์ไหน จำนวนกี่ครั้ง ดังรูปที่ 2.7 ตัวอย่างเมื่อโมไบล์โฮสต์ MH2 มีการเคลื่อนที่ออกไป ทำให้ลิงค์ระหว่าง MH1 และ MH2 ขาดออกจากกัน เมื่อถึงคาบเวลาการส่งบีคอน โมไบล์โฮสต์ MH1 และ MH2 ก็จะไม่ได้รับบีคอนของกันและกัน ซึ่งหมายถึงสองโมไบล์โฮสต์นี้ ไม่สามารถเชื่อมต่อกันได้แล้ว และจะรีเซต associativity tick



(ก)



(ข)

รูปที่ 2.7 การวัดค่าเสถียรภาพของลิงค์ระหว่างคู่โมไบล์โฮสต์

ของโมไบล์โฮสต์ที่สัมพันธ์กันออกจากตาราง associativity ของตน ด้วยวิธีนี้ เราจะเห็นได้ว่าตลอดช่วงเวลา MH2 และ MH3 จะติดต่อกันอยู่ได้ตลอดเวลา แสดงให้เห็นว่า MH2 และ MH3 จะมีเสถียรภาพของลิงค์ที่ดีกว่า ดังนั้นวิธีการนับบีคอนจึงเป็นวิธีการหนึ่งที่สามารถการคาดเดาลักษณะของลิงค์ระหว่างโมไบล์โฮสต์ว่าเป็นเช่นไร ซึ่งเมสเสจบีคอนนี้เป็นแพคเกจที่มีขนาดเล็กมาก จึงส่งผลต่อเครือข่ายไม่มากเมื่อเทียบกับโอเวอร์เฮดที่จะเกิดขึ้นของกระบวนการค้นหา

เส้นทาง สำหรับเอบีอาร์สามารถแบ่งกระบวนการค้นหาเส้นทางออกเป็นสองส่วนคือ กระบวนการค้นหาเส้นทางและกระบวนการบำรุงเส้นทาง

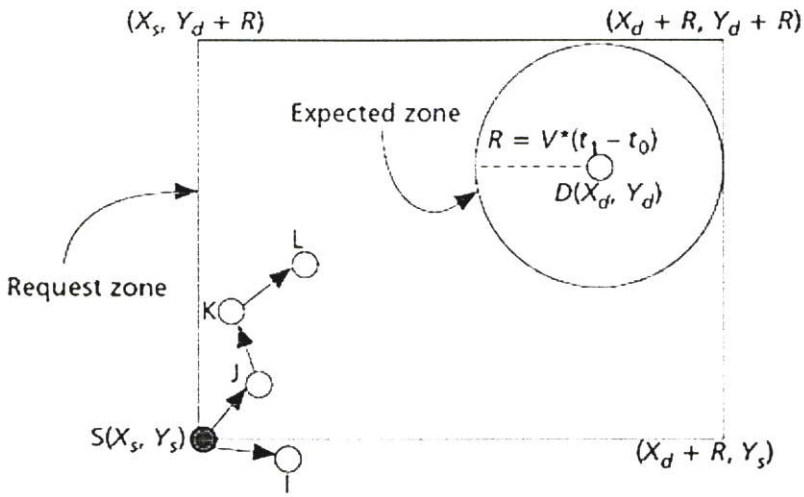
สำหรับกระบวนการค้นหาเส้นเริ่มต้นโดยโมไบล์โฮสต์ S ต้องการค้นหาเส้นทางไปยังโมไบล์โฮสต์ D โมไบล์โฮสต์ S ก็จะส่งเมสเสจบรอดคาสต์คิวรี (BQ) ออกไปยังเครือข่ายโดยส่งไอดีของโมไบล์โฮสต์ D ซึ่งต้องการค้นหา ลงในเมสเสจบีคิว เมื่อโมไบล์โฮสต์ใดๆภายในเครือข่ายได้รับเมสเสจนี้และพบว่าตนเองไม่ใช้โมไบล์โฮสต์ D (Intermediate Mobile host: IM) ก็จะส่งต่อเมสเสจไปจนกว่าจะหาโมไบล์โฮสต์ D ได้พบเรียกว่าเป็นวิธีการฟลัดดิ้ง (Flooding)ในขณะที่แต่ละโมไบล์โฮสต์ช่วยกันบรอดคาสต์เพื่อค้นหาโมไบล์โฮสต์ปลายทางนั้น โมไบล์โฮสต์ที่ได้รับหน้าที่ส่งต่อเมสเสจ (IM) ก็จะใส่ โมไบล์โฮสต์ไอดีของตนเองและค่า associativity tick ของตนกับโมไบล์โฮสต์รอบข้างลงไปในเมสเสจบีคิว เพื่อใช้เป็นเส้นทางติดต่อย้อนกลับในภายหลังและใช้ในการตัดสินใจเลือกเส้นทาง เมื่อเมสเสจร้องขอเส้นทางนี้เดินทางไปถึงโมไบล์โฮสต์ D โมไบล์โฮสต์ D จะรออยู่ช่วงเวลาหนึ่งเพื่อรอรับเมสเสจนี้อีกจากเส้นทางอื่นๆ หลังจากนั้นก็จะเลือกเส้นทางจาก associativity tick ในเมสเสจที่ได้รับมาทั้งหมดและเลือกเส้นทางที่เหมาะสมที่สุด หลังจากนั้นโมไบล์โฮสต์ D ก็จะตอบกลับด้วยเมสเสจตอบกลับ (Reply Message) ที่มีโมไบล์โฮสต์ไอดีต่างๆจากเมสเสจบีคิว ย้อนกลับไปหาโมไบล์โฮสต์ S หลังจากโมไบล์โฮสต์ S ได้รับเมสเสจ ก็จะทราบว่าตนเองจะใช้เส้นทางใดสื่อสารกับโมไบล์โฮสต์ D ได้ โมไบล์โฮสต์ S ก็จะเก็บเส้นทางที่โมไบล์โฮสต์ D ตอบกลับมานั้นลงในตารางเส้นทางของตนเป็นอันเสร็จสิ้นกระบวนการค้นหาเส้นทางซึ่งเรียกว่าเป็นช่วงการทำ BQ-Reply นอกจากนี้แล้วช่วงระหว่างการส่งเมสเสจบีคิวออกไปในเครือข่ายเพื่อหาโมไบล์โฮสต์ D นั้น IM (โมไบล์โฮสต์ที่เป็นตัวกลางระหว่างโมไบล์โฮสต์ต้นทางและปลายทาง) อาจจะใส่ค่าพารามิเตอร์ ตัวอื่นๆลงมาในเมสเสจบีคิวได้อีกหากว่าต้องการ เช่นรีเลย์โหลด ซึ่งบอกถึงภาระการใช้งานของโมไบล์โฮสต์ในเส้นทางขณะนั้นมาด้วย เพื่อหลีกเลี่ยงไม่ให้เกิดการแออัดของข้อมูล ซึ่งส่วนนี้บทความของเราได้พิจารณาถึงแบนด์วิธที่ถูกใช้ไปสำหรับช่วง BQ-Reply นี้

ส่วนของกระบวนการบำรุงรักษาเส้นทางจะมีหน้าที่แก้ปัญหาเมื่อเส้นทางที่ใช้สื่อสารข้อมูลนั้นเกิดปัญหา ซึ่งจะกล่าวถึงคร่าวๆ แบ่งออกเป็น 3 ส่วนคือ การค้นหาเส้นทางเพียงบางส่วน (Partial Route Discovery) ใช้ในการหาเส้นทางใหม่โดยยังคงเส้นทางเก่าในส่วนที่ยังพอใช้ได้อยู่ โดยเมื่อเส้นทางขาด โมไบล์โฮสต์ที่อยู่ในเส้นทาง IM จะพยายามหาเส้นทางไปยังโมไบล์โฮสต์ปลายทางเองก่อน ซึ่งถ้าหาไม่พบก็จะย้อนกลับมายังโมไบล์โฮสต์ต้นทางข้อดีคือเป็นการลดดีเลย์และโอเวอร์เฮดที่จะเกิดขึ้นในการหาเส้นทางใหม่ทั้งหมด แต่หากหาไม่พบก็จะทำให้ดีเลย์และโอเวอร์เฮด มากขึ้นไปอีก ซึ่งวิธีการนี้เรียกว่าโลคอลคิวรี (Local Query) กระบวนการลบเส้นทาง (Invalid Route Erasure) เป็นการส่งเมสเสจเพื่อแจ้งโมไบล์โฮสต์ในเส้นทางถึงสภาพเส้นทางที่มีปัญหา และให้ลบเส้นทางนั้นออกจากแคชเนื่องจากเป็นเส้นทางที่ไม่ได้ใช้แล้ว สำหรับกระบวนการปรับสภาพเส้นทาง (Valid Route Update) เกิดขึ้นเมื่อการทำ โล

คอลคิวรีสามารถค้นหาเส้นทางได้สำเร็จ ก็จะเก็บค่าลงในแคชของโมไบล์โฮสต์ที่อยู่ในเส้นทาง เพื่อบอกว่าโมไบล์โฮสต์ใดบ้างเป็นเส้นทางการเชื่อมต่อ

2.2.5 LAR (Location-Aided Routing Protocol) [9]

ได้นำเสนอวิธีการจำกัดการบรอดคาสต์เมสเสจร้องขอเส้นทางให้ลดน้อยลง โดยสมมติว่าโมไบล์โฮสต์มีข้อมูลตำแหน่งของตนเองอยู่แล้วและเรียกใช้ได้ตลอดเวลาซึ่งเป็นข้อมูลที่ใหม่อยู่เสมอ โดยอาจจะได้มาจากอุปกรณ์ที่เรียกว่าจีพีเอส (Global Positioning System: GPS) ซึ่งมีอยู่ในปัจจุบัน โดยแอลเออาร์ ได้นำเสนอสองส่วนที่เรียกว่าพื้นที่ส่งต่อ (Request Zone) และพื้นที่คาดหวัง (Expected Zone) สำหรับพื้นที่คาดหวังนั้นเมื่อโมไบล์โฮสต์ S ต้องการติดต่อไปยังโมไบล์โฮสต์ D โดยสมมติว่าโมไบล์โฮสต์ S มีข้อมูลตำแหน่งของ D เมื่อเวลา t_0 ที่ผ่านมา สมมติเวลาปัจจุบันเป็น t_1 ดังนั้นเราจะสามารถคำนวณหาพื้นที่คาดหวังของโมไบล์โฮสต์ D ณ เวลา t_1 ในมุมมองของโมไบล์โฮสต์ S ได้ โดยให้ v เป็นความเร็วของโมไบล์โฮสต์ D ดังนั้นโมไบล์โฮสต์ D จะมีพื้นที่คาดหวังอยู่ในรัศมี R ในสมการที่ 2.2 โดยมีศูนย์กลางอยู่ที่จุด Dx_0,Dy_0 เมื่อเวลา t_0 ซึ่งพื้นที่คาดหวังนี้เป็นส่วนที่เราคาดว่าโมไบล์โฮสต์ D จะอยู่ภายในพื้นที่นี้



รูปที่ 2.8 พื้นที่คาดหวัง (Expected Zone)

การหาพื้นที่การส่งต่อเมสเสจร้องขอเส้นทางจะเป็นพื้นที่ทั้งหมดที่จะถูกส่งต่อเมสเสจร้องขอเส้นทางซึ่งพื้นที่นี้ควรจะล้อมพื้นที่ของพื้นที่คาดหวังเอาไว้ ส่วนพื้นที่นอกเหนือจากพื้นที่คาดหวังหาก IM ใดได้รับเมสเสจนี้และพบว่าตนเองอยู่ภายนอกพื้นที่ก็จะดรอบเมสเสจที่ได้รับ ซึ่งเป็นส่วนของการจำกัดไม่ให้มีการบรอดคาสต์ไปทั้งเครือข่าย ในการเลือกพื้นที่การส่งต่อ มองได้ในสองลักษณะคือความเหมาะสมระหว่างพื้นที่การส่งต่อและการผกผันที่เกิดขึ้นของโอเวอร์เฮด เนื่องจากการบรอดคาสต์ไปยังพื้นที่การส่งต่อนั้น หากพื้นที่มีเล็กมากเกินไป อาจทำให้ไม่พบเส้นทางไปยังโมไบล์โฮสต์ปลายทางได้ ซึ่งต้องทำการขยายพื้นที่การส่งต่อให้ใหญ่มากขึ้น การส่งต่อเมสเสจร้องขอเส้นทางในรูปแบบของแอลเออาร์จะเป็นลักษณะพื้นที่สี่เหลี่ยมดังในรูป ที่ 2.8

สมมติว่าโมไบล์โฮสต์ S รู้ตำแหน่งของโมไบล์โฮสต์ D ว่าเป็น X_d,Y_d เมื่อเวลา t_0 ที่ผ่านมา และเมื่อเวลา t_1 โมไบล์โฮสต์ S ได้ทำการค้นหาเส้นทางไปยังโมไบล์โฮสต์ D โดยสมมติว่า S รู้ความเร็วของโมไบล์โฮสต์ D ว่าเป็นความเร็ว v ดังนั้นเมื่อเวลา t_1 โมไบล์โฮสต์ S จะสามารถหาพื้นที่คาดหวังของโมไบล์โฮสต์ D ได้จากสมการที่ 2.2 โดยมีศูนย์กลางที่ (X_d,Y_d) สำหรับพื้นที่การส่งต่อจะเป็นลักษณะของสี่เหลี่ยมผืนผ้าโดยมีโมไบล์โฮสต์ S อยู่รวมภายในพื้นที่การส่งต่อด้วย เมื่อโมไบล์โฮสต์ S บรอดคาสต์เมสเสจร้องขอเส้นทางออกไปยังเครือข่าย โมไบล์โฮสต์ใดๆ ที่ได้รับก็จะตรวจสอบว่าตำแหน่งของตนเองภายในพื้นที่การส่งต่อหรือไม่ ถ้าไม่อยู่ภายในพื้นที่การส่งต่อ โมไบล์โฮสต์จะดรอบเมสเสจที่ได้รับมา แต่ถ้าโมไบล์โฮสต์พบว่าตนเองมีตำแหน่งอยู่ในพื้นที่การส่งต่อแล้วโมไบล์โฮสต์จึงจะส่งต่อเมสเสจไปยังทิศทางสู่พื้นที่คาดหวังต่อไป ดังตัวอย่างโมไบล์โฮสต์ I ซึ่งมีตำแหน่ง เป็น Xi,Yi ได้รับเมสเสจร้องขอเส้นทางและพบว่าตนเองอยู่นอกพื้นที่การส่งต่อจะดรอบเมสเสจ แต่สำหรับโมไบล์โฮสต์ J ซึ่งมีตำแหน่ง เป็น Xj,Yj พบว่าตนเองอยู่ภายในพื้นที่การส่งต่อจึงสามารถส่งต่อ เมสเสจร้องขอเส้นทางได้

บทที่ 3
การใช้ข้อมูลตำแหน่งในโปรโตคอลค้นหาเส้นทาง

โปรโตคอลการค้นหาเส้นทางในบทที่ 2 แสดงลักษณะการค้นหาเส้นทางที่พิจารณาการเปลี่ยนแปลงของเครือข่ายเป็นสำคัญ สำหรับในบทนี้จะแสดงรายละเอียดของโมดูลจีพีเอส (Global Positioning Systems: GPS) ซึ่งมีส่วนเกี่ยวของกับวิทยานิพนธ์ เป็นอุปกรณ์ที่ถูกฝังลงในอุปกรณ์ที่สามารถพกพาหรือเคลื่อนที่ได้เช่นโทรศัพท์เคลื่อนที่ ปาล์มคอมพิวเตอร์ คอมพิวเตอร์พกพา พอคเก็ตพีซี เครื่องนำทางเข็มทิศสำรวจ หรือ อุปกรณ์บลูทูธ เป็นต้น ซึ่งในปัจจุบัน (ปี พ.ศ. 2549) ปัจจัยด้านราคาอุปกรณ์ที่ถูกฝังรวมระบบนำทางจีพีเอสมีราคาถูกคือหลักพันบาทขึ้นไป โดยที่ค่าความผิดพลาดจะอยู่ในช่วง 3-20m (ในระบบสองแกน X, Y) ซึ่งสามารถนำมาใช้งานกับการค้นหาเส้นทางการสื่อสารข้อมูลของเครือข่ายแอดฮอคได้ โดยทั่วไปเครือข่ายแอดฮอคมีพื้นที่การใช้งานราวสิบตารางเมตรจนถึงหลายตารางกิโลเมตร และนอกจากนี้จะกล่าวถึงวิธีการกระจายข้อมูลตำแหน่งให้กับโมไบล์โฮสต์ต่างๆ ในวิธีการค้นหาเส้นทางแบบที่เรียกว่าการค้นหาเส้นทางด้วยข้อมูลตำแหน่ง (Position-Based Routing) ซึ่งแต่ละโปรโตคอลมีวิธีการกระจายข้อมูลตำแหน่งเพื่อเก็บลงโมไบล์โฮสต์ในเครือข่ายแอดฮอค ตัวอย่างเช่นโปรโตคอลค้นหาเส้นทาง DREAM และ LAR ในบทที่ 2 รวมถึงรูปแบบการใช้ข้อมูลตำแหน่งเพื่อส่งแพคเกจข้อมูลไปยังปลายทางโดยไม่จำเป็นต้องเริ่มกระบวนการค้นหาเส้นทางก่อน

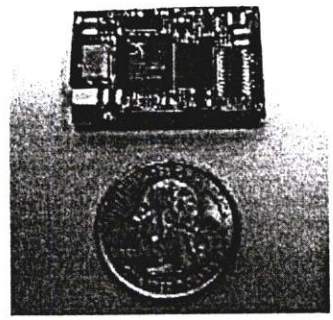
3.1 การหาข้อมูลตำแหน่ง

เนื่องจากโมดูลจีพีเอสมีขนาดเล็ก จึงสามารถนำไปประกอบรวมอยู่กับอุปกรณ์เคลื่อนที่ได้หลากหลาย ระบบจีพีเอสเป็นการใช้ข้อมูลจากดาวเทียมเพื่อแจ้งตำแหน่งของลูกข่ายหรือเครื่องรับจีพีเอส ซึ่งดาวเทียมจะส่งสัญญาณวิทยุมาที่เครื่องรับจีพีเอสซึ่งสัญญาณจะถูกแปลงเป็นข้อมูลตำแหน่ง ความเร็ว และเวลา การคำนวณค่าตำแหน่งจากดาวเทียมจะวัดเวลาในการเดินทางของสัญญาณจากดาวเทียม ซึ่งจะคำนวณได้ข้อมูลตำแหน่งแบบสามแกน X, Y, Z เวลา UTC และความเร็วการเคลื่อนที่ของตัวรับจีพีเอส

3.2 การค้นหาเส้นทางโดยใช้ข้อมูลตำแหน่ง

อัลกอริทึมที่ใช้ในการค้นหาเส้นทางโดยอาศัยข้อมูลตำแหน่ง โดยทั่วไปจะแบ่งออกเป็นสองส่วน คือผู้บริการข้อมูลตำแหน่ง (Location Service) ซึ่งจะเก็บข้อมูลตำแหน่งของโมไบล์

โฮสต์ ต่างๆที่อยู่ในเครือข่าย และอัลกอริทึมที่ใช้ในการค้นหาเส้นทางหรือส่งแพคเกจข้อมูล (Forwarding Strategy)



รูปที่ 3.1 ตัวอย่างจีพีเอสโมดูล Leadtek GPS 9543 (SiRFstar II)

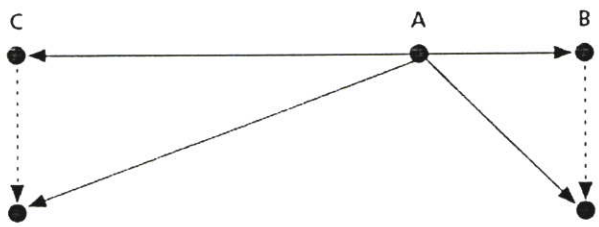
3.2.1 การบริการข้อมูลตำแหน่ง (Location Service)

ส่วนนี้มีหน้าที่ในการเก็บข้อมูลตำแหน่งและกระจายข้อมูลตำแหน่งให้กับโมไบล์โฮสต์ที่ทำการร้องขอ ซึ่งขั้นตอนการเก็บข้อมูลจะเริ่มขึ้นเมื่อโมไบล์โฮสต์ใดเข้ามาสู่เครือข่าย จะมีการส่งข้อมูลตำแหน่งของตนเองไปแจ้งกับโมไบล์โฮสต์ที่เป็นผู้บริการข้อมูลตำแหน่ง เมื่อโมไบล์โฮสต์ใดมีความต้องการข้อมูลตำแหน่งของโมไบล์โฮสต์อื่นในเครือข่ายและตนเองไม่มีข้อมูลนั้น ก็จะร้องขอข้อมูลตำแหน่งไปยังผู้บริการข้อมูลตำแหน่ง สำหรับระบบที่เป็นเซลลูล่าร์ทั่วไป ผู้บริการนี้จะเป็นสถานีฐานตั้งอยู่กับที่ การเก็บข้อมูลตำแหน่งของโมไบล์โฮสต์แบบนี้จะเรียกว่าเป็น Some-for-All ซึ่ง Some หมายถึงลูกข่ายบางตัวในที่นี้คือสถานีฐาน และ all มีความหมายเป็นการเก็บข้อมูลของโมไบล์โฮสต์ในเครือข่ายไว้ทั้งหมด สามารถแบ่งออกได้เป็น 4 รูปแบบของการบริการคือการเก็บข้อมูลแบบ Some-for-Some, Some-for-All, All-for-Some และ All-for-All สำหรับการบริการข้อมูลตำแหน่งที่เป็นแบบ Some-for-All จะมีปัญหาในการทำงานกับเครือแอดฮอค เนื่องจากมีโมไบล์โฮสต์บางตัวเก็บข้อมูลตำแหน่งของโมไบล์โฮสต์ทั้งหมดไว้ หากโมไบล์โฮสต์ที่เก็บข้อมูลตำแหน่งมีการเคลื่อนที่ จะเกิดปัญหาที่เมื่อโมไบล์โฮสต์อื่นมีความต้องการข้อมูลตำแหน่งแต่ไม่สามารถร้องขอไปยังโมไบล์โฮสต์ผู้ให้บริการได้ เนื่องจากไม่ทราบตำแหน่งของโมไบล์โฮสต์ผู้ให้บริการ และเมื่อติดต่อไปยังผู้ให้บริการไม่ได้ก็ไม่สามารถรับข้อมูลตำแหน่งได้อีกเช่นกัน นอกจากนี้ยังเป็นการยากที่รับประกันได้ว่าจะมีโมไบล์โฮสต์บริการข้อมูลตำแหน่งอย่างน้อย 1 ตัวอยู่ในเครือข่ายเสมอ ตัวอย่างการพิจารณาลักษณะของการบริการข้อมูลตำแหน่ง

3.2.1.1 DREAM (Distance Routing Effect Algorithm for Mobility)

แต่ละโมไบล์โฮสต์จะเก็บข้อมูลตำแหน่งซึ่งอาจจะเป็นส่วนหนึ่งหรือทั้งหมดของโมไบล์โฮสต์ในเครือข่าย ซึ่งสามารถจัดได้เป็นกลุ่มวิธีบริการข้อมูลตำแหน่งแบบ All-for-All ในข้อมูลแต่ละชุดจะประกอบด้วยโมไบล์โฮสต์ไอดี ทิศทาง ระยะทางของโมไบล์โฮสต์ และเวลาที่ข้อมูล

ถูกสร้างขึ้น สำหรับความแม่นยำของข้อมูลตำแหน่งจะมีอายุกำกับเอาไว้ แต่ละโมไบล์โฮสต์จะกระจายข้อมูลตำแหน่งไปยังโมไบล์โฮสต์ต่างๆในเครือข่ายเพื่อปรับปรุงข้อมูลตำแหน่งให้มีความถูกต้อง โมไบล์โฮสต์จะควบคุมความถูกต้องของข้อมูลอยู่สองลักษณะ คือความถี่ที่ใช้ในการกระจายข้อมูลปรับปรุงตำแหน่ง (Temporal Resolution) และกำหนดระยะของการกระจายข้อมูลปรับปรุงตำแหน่ง (Spatial Resolution) สำหรับการกำหนดความถี่ของการกระจายข้อมูลตำแหน่งอาจเลือกความถี่ขึ้นกับความเร็วในการเคลื่อนที่ของโมไบล์โฮสต์ เช่นหากโมไบล์โฮสต์มีการเคลื่อนที่ด้วยความเร็วสูงขึ้น ความถี่ในการกระจายข้อมูลปรับปรุงตำแหน่งก็บ่อยมากขึ้นเป็นต้น และสำหรับการกำหนดระยะการกระจายข้อมูลปรับปรุงตำแหน่งถูกใช้ในการแบ่งแยกระหว่างโมไบล์โฮสต์ที่อยู่ใกล้กันและโมไบล์โฮสต์ที่อยู่ห่างออกไป ทำให้สามารถลดปริมาณข้อมูลการปรับปรุงตำแหน่งกรณีโมไบล์โฮสต์อยู่ห่างออกไป ผลที่เกิดจากระยะทางที่ต่างกัน (Distance Effect) แสดงในรูปที่ 3.2 สมมติโมไบล์โฮสต์ A ไม่เคลื่อนที่ และโมไบล์โฮสต์ B, C มีการเคลื่อนที่ในทิศทางเดียวกันและความเร็วเท่ากัน ในมุมมองของโมไบล์โฮสต์ A จะพบว่าการเปลี่ยนแปลงของทิศทางโมไบล์โฮสต์ B มีมากกว่าการเปลี่ยนทิศทางที่เกิดขึ้นจากโมไบล์โฮสต์ C

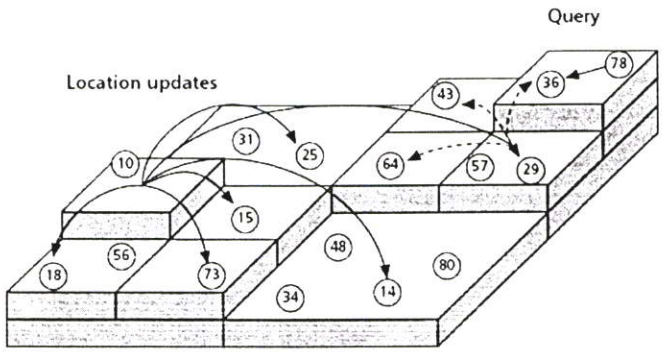


รูปที่ 3.2 ผลของระยะทางของโมไบล์โฮสต์ B และ C ในมุมมองของ A

3.2.1.2 GLS (Grid Location Service) [10]

มีการแบ่งพื้นที่ของเครือข่ายออกเป็นส่วนย่อยสี่เหลี่ยมซ้อนกันเป็นชั้นดังรูปที่ 3.3 ในลำดับชั้นที่ n ใดๆจะประกอบด้วย ส่วนย่อยสี่เหลี่ยมสี่ส่วนในชั้นที่ (n-1) ทำให้เป็นลักษณะที่เรียกว่า Quad Tree แต่ละโมไบล์โฮสต์จะเก็บข้อมูลตำแหน่งของโมไบล์โฮสต์ต่างๆในเครือข่ายที่อยู่ภายในส่วนย่อยสี่เหลี่ยมในชั้นที่ 1 เพื่อเป็นการจำกัดปริมาณข้อมูลที่กระจายออกมาเพื่อปรับปรุงข้อมูลตำแหน่ง ในรูปที่ 3.3 ในการเลือกว่าจะเก็บข้อมูลตำแหน่งไว้ที่ใด โปรโตคอลจะนิยามคำว่าโมไบล์โฮสต์ไอดีใกล้เคียงว่าเป็นไอดีของโมไบล์โฮสต์ที่น้อยที่สุดแต่มากกว่าไอดีโมไบล์โฮสต์ตน จากรูปที่ 3.3 โมไบล์โฮสต์ 10 ต้องการกระจายข้อมูลตำแหน่ง มันจะส่งข้อมูลปรับปรุงตำแหน่งไปยังโมไบล์โฮสต์ไอดีใกล้เคียงให้กับสามส่วนที่อยู่รอบๆในชั้นที่ 1 ดังนั้นข้อมูลตำแหน่งจะถูกเก็บไว้ที่โมไบล์โฮสต์ 15 ,18 และ 73 และทุกๆโมไบล์โฮสต์ที่อยู่ในชั้นที่ 1 เช่นเดียวกับโมไบล์โฮสต์ 10 สามส่วนที่อยู่รอบๆในชั้นที่ 2 ก็เช่นเดียวกัน โมไบล์โฮสต์ที่มีไอดีใกล้เคียงจะถูกเลือกให้มีหน้าเก็บข้อมูลตำแหน่ง ในตัวอย่างนี้คือ 14, 24 และ 29 ซึ่งกระบวนการนี้

จะทำซ้ำไปจนกระทั่งครอบคลุมได้ทั้งเครือข่าย สมมติว่าโมไบล์โฮสต์ 78 ต้องการข้อมูลตำแหน่งของโมไบล์โฮสต์ 10 มันจะค้นหาตำแหน่งของโมไบล์โฮสต์ใกล้เคียงและมีข้อมูลของโมไบล์โฮสต์ 10 ในตัวอย่างนี้คือ 29 แต่โมไบล์โฮสต์ 79 ไม่ทราบว่าโมไบล์โฮสต์ 29 มีข้อมูลดังกล่าวที่ต้องการ ซึ่งตำแหน่งของมันเองจะถูกเก็บไว้ในสามส่วนที่อยู่รอบในชั้นที่ 1 นั่นคือ 36, 43 และ 64



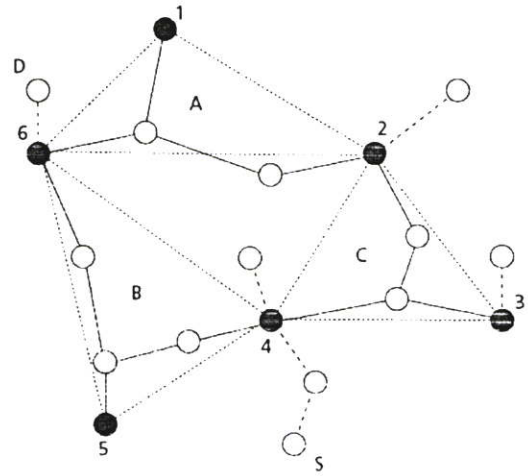
รูปที่ 3.3 จีแอลเอส

สังเกตว่าแต่ละโมไบล์โฮสต์เหล่านี้รวมถึงโมไบล์โฮสต์ 29 จะเป็นโมไบล์โฮสต์ที่อยู่รอบข้างในชั้นที่ 1 ที่มีโมไบล์โฮสต์ไอดีใกล้เคียงกับโมไบล์โฮสต์ 10 ดังนั้นจะมีเส้นทางจากโมไบล์โฮสต์ไอดีมากไปหาโมไบล์โฮสต์น้อยในแต่ละส่วนของทุกๆชั้นเพื่อจะไปถึงโมไบล์โฮสต์ที่เก็บข้อมูลตำแหน่งได้ เมสเสจการร้องขอข้อมูลตำแหน่งจะถูกส่งต่อไปในทิศทางที่มีโมไบล์โฮสต์ไอดีใกล้เคียงกับโมไบล์โฮสต์ที่ร้องขอข้อมูลตำแหน่งทราบ ในตัวอย่างนี้คือโมไบล์โฮสต์ 36 และขั้นตอนจะซ้ำไปจำกระทั่งพบโมไบล์โฮสต์ที่บริการข้อมูลตำแหน่ง ซึ่งโปรโตคอลนี้ทุกโมไบล์โฮสต์มีการเก็บข้อมูลตำแหน่งของโมไบล์โฮสต์อื่นๆบางตัวในเครือข่าย สามารถจัดในกลุ่ม All-for-Some

3.2.1.3 Qourum-Based location service [11]

แนวคิดของระบบควอรัมเป็นที่รู้จักดีจากระบบ Replication ในดาต้าเบสและ Distributed สำหรับข้อมูลปรับปรุงตำแหน่ง (Write Operation) จะถูกส่งไปยังกลุ่มหนึ่ง (ควอรัม) และสำหรับการร้องขอข้อมูลตำแหน่ง (Read Operation) จะอ้างอิงจากค่าของกลุ่ม กลุ่มจะถูกออกแบบให้เมื่อแบ่งกลุ่มแล้วในกลุ่มจะต้องไม่เป็นเซตว่าง เพื่อให้แน่ใจว่าสามารถพบข้อมูลที่มีการปรับปรุงตำแหน่งแล้ว อยู่เสมอ โมไบล์โฮสต์จะส่งเมสเสจปรับปรุงข้อมูลตำแหน่งไปยังโมไบล์โฮสต์หลักที่ใกล้ที่สุด ซึ่งหลังจากนั้นจะทำการเลือกควอรัมของโมไบล์โฮสต์หลักเพื่อให้บริการข้อมูลตำแหน่ง จากรูปที่ 3.4 โมไบล์โฮสต์ D จะส่งข้อมูลปรับปรุงไปยังโมไบล์โฮสต์ 6 ซึ่งจะทำการเลือกควอรัม A พร้อมกับโมไบล์โฮสต์ 1, 2 และ 6 เพื่อให้เก็บข้อมูลตำแหน่ง เมื่อโมไบล์โฮสต์ S ร้องขอข้อมูลตำแหน่ง จะส่งคำร้องขอไปยังโมไบล์โฮสต์หลักที่ใกล้ที่สุดซึ่งอาจเป็นโมไบล์โฮสต์

4 ดังตัวอย่างเลือกควอรัม B ที่ประกอบด้วยโมไบล์โฮสต์ 4, 5 และ 6 ในการร้องขอ หากพิจารณาแนวคิดที่กำหนดว่าการแบ่งกลุ่มแต่ละควอรัมต้องไม่มีควอรัมใดเป็นเซตว่าง เพื่อประกันว่าจะได้รับข้อมูลตำแหน่งที่ต้องการกลับมาอย่างน้อย 1 ตัว ในกรณีที่ข้อมูลตอบกลับการร้องขอตำแหน่งอาจมีได้มากกว่า 1 ตัว จะสังเกตเลือกข้อมูลตำแหน่งที่ใหม่ที่สุดจากที่ได้รับซึ่งแต่ละข้อมูลจะมี Timestamp เอาไว้ เพื่อบอกถึงความใหม่ของข้อมูลตำแหน่ง สำหรับปัญหาของการทำควอรัม คือหากเลือกควอรัมเซตให้มีขนาดใหญ่มากขึ้นจะส่งผลให้เพิ่มประมาณข้อมูลปรับปรุงสูงขึ้นแต่ก็จะทำให้มีเป็นการลดปัญหาที่เกิดจากการการไม่ตอบสนอง



รูปที่ 3.4 ตัวอย่างการแบ่งควอรัม A, B, และ C

เนื่องจากไม่สามารถค้นหาโมไบล์โฮสต์หลักพบ สำหรับแนวคิดของควอรัมนี่จะเป็นสามรูปแบบ คือ Some-for-Some, All-for-All และ All-for-Some ขึ้นกับจำนวนของโมไบล์โฮสต์หลักและการเลือกเซตควอรัม

3.2.1.4 Homezone [12]

วิธีการบริการข้อมูลตำแหน่งแบบโฮมโซนได้มีการนำเสนอแนวคิดจัดการโฮมโซน ตำแหน่งโฮมโซน C โมไบล์โฮสต์สามารถหาตำแหน่งได้จากการใช้แฮซฟังก์ชัน (Hash Function) กับโมไบล์โฮสต์ไอดี ทุกโมไบล์โฮสต์ที่อยู่ภายในพื้นที่วงกลมรัศมี R และมีศูนย์กลางที่จุด C ซึ่งจะบริการข้อมูลให้ เช่นเดียวกับจีแอลเอส ที่ข้อมูลตำแหน่งสามารถหาได้จากเฮซฟังก์ชั่นของผู้ส่งและผู้รับโดยไม่ต้องมีการแลกเปลี่ยนข้อมูลตำแหน่งกัน วิธีการของโฮมโซนจะเป็นลักษณะการบริการแบบ All-for-Some หากโฮมโซนที่กำหนดมีโมไบล์โฮสต์จำนวนน้อย รัศมีของวงกลม R อาจจะขยายออกไปอีก ส่งผลให้เพิ่มการปรับปรุงข้อมูลมากขึ้นเช่นเดียวกับการร้องขอข้อมูล วิธีในการบริการข้อมูลตำแหน่งสามารถแสดงประสิทธิภาพได้ดังตารางที่ 3.1

3.2.2 อัลกอริทึมในการค้นหาเส้นทาง (Forwarding strategy)

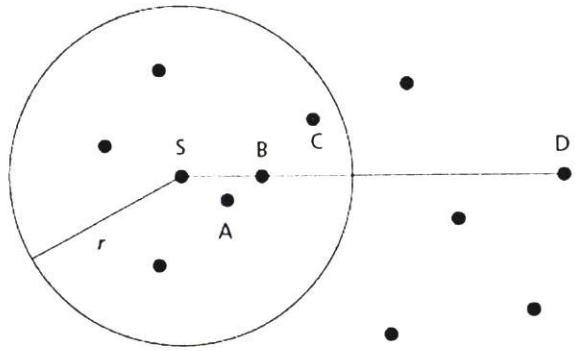
อัลกอริทึมในการค้นหาเส้นทางจะเป็นการใช้ข้อมูลตำแหน่งที่ค้นหาได้รับมาจากขั้นตอนที่ 3.2.1 การบริการข้อมูลตำแหน่ง

ตารางที่ 3.1 แสดงค่าประสิทธิภาพต่างๆของวิธีการบริการข้อมูลตำแหน่ง

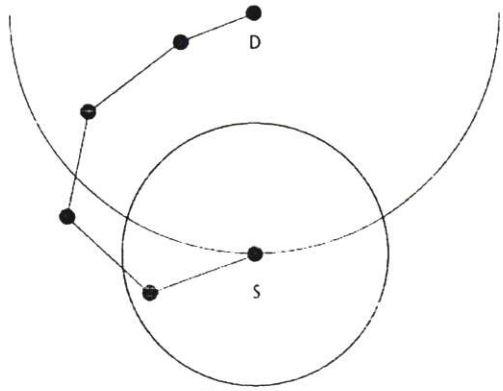
Criterion	DREAM	Quorum system	GLS	Homezone
Type	All-for-all	Some-for-some	All-for-some	All-for-some
Communication complexity (update)	$O(n)$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$
Communication complexity (lookup)	$O(c)$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$
Time complexity (update)	$O(\div n)$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$
Time complexity (lookup)	$O(c)$	$O(\sqrt{n})$	$O(\sqrt{n})$	$O(\sqrt{n})$
State volume	$O(n)$	$O(c)$	$O(log(n))$	$O(c)$
Localized information	Yes	No	Yes	No
Robustness	High	Medium	Medium	Medium
Implementation complexity	Low	High	Medium	Low
Abbreviations: n = number of nodes, c = constant				

3.2.2.1 Greedy Packet Forwarding [13]

ในรูปที่ 3.5 ได้แสดงวิธีการใช้ข้อมูลตำแหน่งค้นหาเส้นทางแบบกรีดี โมไบล์โฮสต์ S และ D จะแทนด้วยโมไบล์โฮสต์ต้นทางและปลายทางตามลำดับ วงกลมที่ล้อมรอบด้วยรัศมี r คือรัศมีทำการของโมไบล์โฮสต์ S ซึ่งการส่งแพคเกจข้อมูลจะส่งไปยังโมไบล์โฮสต์ที่อยู่ใกล้กับเป้าหมายมากที่สุด โดยในแพคเกจข้อมูลจะแนบข้อมูลตำแหน่งที่ได้รับมา ซึ่งในตัวอย่างนี้โมไบล์โฮสต์ที่ใกล้โมไบล์โฮสต์ปลายทางมากที่สุดคือโมไบล์โฮสต์ C วิธีการส่งต่อแบบนี้เรียกว่าเป็น (Most Forward Within R: MFR) ซึ่งเป็นความพยายามเพื่อนลดจำนวนของฮอพในการส่งแพคเกจข้อมูลไปยังโมไบล์โฮสต์ปลายทาง เอ็มเอฟอาร์เป็นวิธีการที่ดีนำมาใช้ได้เพราะโมไบล์โฮสต์ต้นทางไม่สามารถปรับระดับสัญญาณระหว่างผู้รับและผู้ส่งได้ แต่อย่างไรก็ตาม ได้มีผู้นำเสนอวิธีการใหม่ที่ดีกว่าเอ็มเอฟอาร์ซึ่งให้ผู้ส่งสามารถปรับระดับความแรงของสัญญาณเมื่อทำการส่ง



รูปที่ 3.5 การค้นหาเส้นทางแบบกรีดี (Greedy Routing)

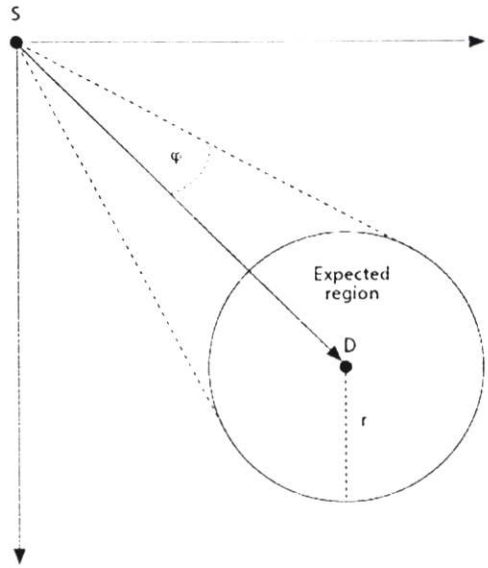


รูปที่ 3.6 ปัญหาที่เกิดขึ้นในการค้นหาเส้นทางแบบกรีดี

แพคเกจข้อมูลได้ เรียกว่า (Nearest with Forward Progress: NFP) ซึ่งแพคเกจข้อมูลจะถูกส่งไปยังโมไบล์โฮสต์ใกล้เคียงกับโมไบล์โฮสต์ต้นทางมากที่สุดแต่อยู่ใกล้กับโมไบล์โฮสต์ปลายทางด้วย ในรูปที่ 3.5 จะสังเกตว่าในกรณีนี้จะเป็นโมไบล์โฮสต์ A ซึ่งวิธีการนี้สามารถลดปริมาณการชนกันของแพคเกจข้อมูล ดังนั้นค่าเฉลี่ยของวิธีการนี้ จะคำนวณได้จาก p * f(a, b) โดยที่ p เป็นความน่าจะเป็นของการส่งแพคเกจข้อมูลได้สำเร็จโดยไม่มีการชนเกิดขึ้น และ f(a. b) คือความคืบหน้าของแพคเกจเมื่อส่งแพคเกจได้สำเร็จจาก a ไปยัง b ซึ่งเอ็นพีเอฟ จะมีค่าสูงกว่า MFR นอกจากนี้ Compass Routing เป็นอีกวิธีการที่นำเสนอโดยเลือกโมไบล์โฮสต์ที่อยู่รอบข้างใกล้ที่สุดกับเส้นตรงที่ลากจากโมไบล์โฮสต์ต้นทางไปยังปลายทาง ในตัวอย่างนี้คือโมไบล์โฮสต์ B ซึ่งวิธีการนี้เป็นไปเพื่อลดจำนวนครั้งในการส่งต่อแพคเกจข้อมูล แต่วิธีการที่เสนอในการส่งต่อข้อมูลแบบกรีดีมีข้อเสียที่ไม่สามารถหาเส้นทางได้ในรูปแบบหนึ่ง ดังแสดงไว้ในรูปที่ 3.6 จะเห็นได้ว่าครึ่งวงกลมรอบ D ซึ่งมีรัศมีที่เป็นระยะทางระหว่าง D และ S และวงกลมรอบโมไบล์โฮสต์ S คือรัศมีทำการของโมไบล์โฮสต์ S จะสังเกตว่าปัญหาเกิดจากกรีดีจะส่งแพคเกจไปในทิศทางใกล้กับโมไบล์โฮสต์ปลายทางเท่านั้นจึงไม่สามารถค้นหาเส้นทางได้พบ แม้จะมีเส้นทางอยู่ก็ตาม

3.2.2.2 DREAM (Distance Routing Effect Algorithm for Mobility)

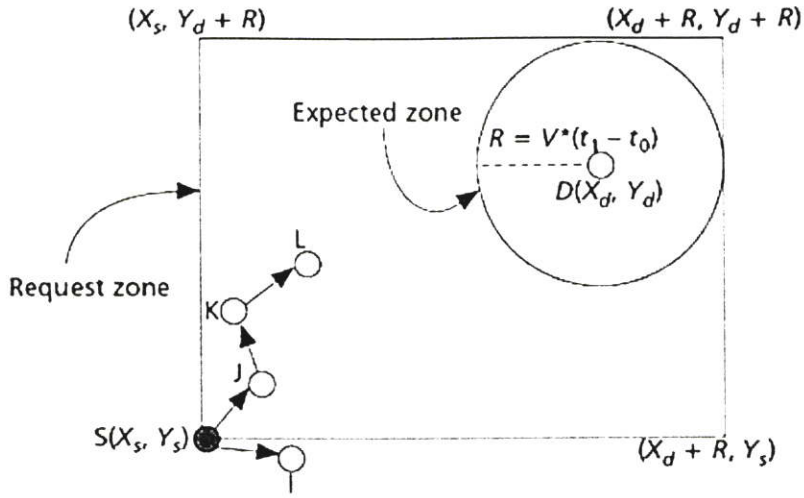
โมไบล์โฮสต์ต้นทาง S จะส่งต่อเมสเสจไปยังฮอพที่ติดกัน ไปในทิศทางของโมไบล์โฮสต์ปลายทาง D ในการค้นหาทิศทางโมไบล์โฮสต์จะคำนวณพื้นที่ซึ่งจะเป็นพื้นที่คาดหวังของโมไบล์โฮสต์ D เรียกว่าเป็นพื้นที่คาดหวัง (Expected Region) ดังรูปที่ 3.7 พื้นที่คาดหวังถูกคำนวณออกมาเป็นพื้นที่วงกลมรัศมี R ซึ่งคำนวณรัศมีจาก $(t_1-t_0)*v_{max}$ โดยที่ t_1 เป็นเวลาปัจจุบัน t_0 เป็นเวลาที่โมไบล์โฮสต์ S ได้เก็บข้อมูลตำแหน่งของโมไบล์โฮสต์ปลายทาง D เอาไว้ และ v_{max} แทนด้วยความเร็วการเคลื่อนที่สูงสุดของโมไบล์โฮสต์ในเครือข่าย ดังในรูปที่ 3.7 โมไบล์โฮสต์ระหว่างทางจะใช้ข้อมูลตำแหน่งโมไบล์โฮสต์ D นี้ในการส่งต่อเมสเสจให้อยู่ในเส้นตรงมีมุมไม่เกิน φ จากเส้นตรงที่ลากระหว่างโมไบล์โฮสต์ต้นทาง S และโมไบล์โฮสต์ปลายทาง D



รูปที่ 3.7 การส่งต่อเมสเสจร้องขอเส้นทางของโปรโตคอล DREAM

3.2.2.3 LAR (Location Aided Routing)

การใช้ข้อมูลของโปรโตคอลนี้เพื่อลดเมสเสจการค้นหาเส้นทางที่เกิดขึ้นในกระบวนการค้นหาเส้นทาง (Route Discovery Process) ให้อยู่ในบริเวณจำกัดโดยจะคำนวณพื้นที่คาดหวังไว้เช่นเดียวกับโปรโตคอล DREAM แต่จะเพิ่มการคำนวณพื้นที่ซึ่งถูกเรียกว่าพื้นที่การส่งต่อ (Request Zone) เมสเสจร้องขอเส้นทาง ซึ่งพื้นที่การส่งต่อจะมีเงื่อนไขในการกำหนดว่าควรมีโมไบล์โฮสต์ต้นทาง และพื้นที่คาดหวังอยู่ภายในด้วย



รูปที่ 3.8 การส่งต่อแมสเสจร้องขอเส้นทางของโปรโตคอลแอลเออาร์

3.2.2.4 Terminodes Routing [14]

เป็นวิธีการที่ผสมกันระหว่างการแบ่งชั้น (Hierarchical) รวมกับการใช้ข้อมูลตำแหน่งเพื่อการค้นหาเส้นทาง เทอมิโหนด จะถูกแบ่งออกเป็นสองชั้น แพคเกจข้อมูลจะถูกส่งต่อไปตามทิศทางโมไบล์โฮสต์ปลายทางแบบ Distance Vector Routing ในกรณีที่ระยะห่างมากขึ้นจะเปลี่ยนโหมดการทำงานเป็นกรีดีและเมื่อแพคเกจเข้าใกล้พื้นที่ของผู้รับ จะเปลี่ยนวิธีไปส่งต่อแพคเกจไปเป็นวิธีแบบการค้นหาภายใน (Local Routing) ซึ่งผู้นำเสนอแสดงให้เห็นว่าสามารถปรับปรุงประสิทธิภาพของอัตราส่วนการรับส่งและโอเวอร์เฮดให้ดีขึ้นได้ ในการป้องกันปัญหาของกรีดีที่จะเกิดขึ้นจากการไม่สามารถค้นหาเส้นทางได้ ผู้ส่งจะแนบลิสรายชื่อข้อมูลตำแหน่งลงในแพคเกจเฮดเดอร์ ซึ่งโมไบล์โฮสต์ S จำเป็นต้องทราบว่าตำแหน่งใดบ้างที่เหมาะสมและสามารถไปถึงโมไบล์โฮสต์ปลายทางได้ ผู้ส่งจะร้องขอข้อมูลเหล่านี้ได้จากโมไบล์โฮสต์ต่างๆ โดยใช้วิธีการแบบค้นหาภายใน เมื่อผู้ส่งได้รับข้อมูลเหล่านี้แล้วจะตรวจสอบข้อมูลทุกช่วงเวลาว่าข้อมูลยังใช้ได้อยู่หรือไม่

3.2.2.5 Grid Routing [15]

วิธีการนี้จะคล้ายกับเทอมิโหนด โดยระยะทางที่อยู่ภายในบริเวณที่กำหนดจะใช้โปรโตคอลค้นหาเส้นทางแบบ Distance Vector และใช้วิธี Position Routing เมื่อมีการส่งแพคเกจออกไปในระยะไกล วิธีการของกริด นอกจากจะเป็นลักษณะการแบ่งเป็นชั้นเพื่อให้สามารถสเกลขนาดให้ใหญ่หรือเล็กลงได้ยังทำให้โมไบล์โฮสต์ซึ่งไม่ทราบตำแหน่งของตนเองสามารถใช้งานในเครือข่ายได้ แนวคิดหลักคือการมีโมไบล์โฮสต์อย่างน้อย 1 ตัว ที่เก็บข้อมูลตำแหน่งเมื่อทำงานในโหมด Distance Vector ซึ่งโมไบล์โฮสต์เหล่านี้จะเปรียบเสมือนพร็อกซี่สำหรับโมไบล์โฮสต์ใดที่ไม่มีข้อมูลตำแหน่งจะอ้างใช้ตำแหน่งของโมไบล์โฮสต์ที่เก็บข้อมูลตำแหน่งเป็นตำแหน่งโมไบล์โฮสต์ของตน ในแพคเกจใดที่ระบุเป้าหมายเป็นโมไบล์โฮสต์ที่ไม่มีข้อมูล

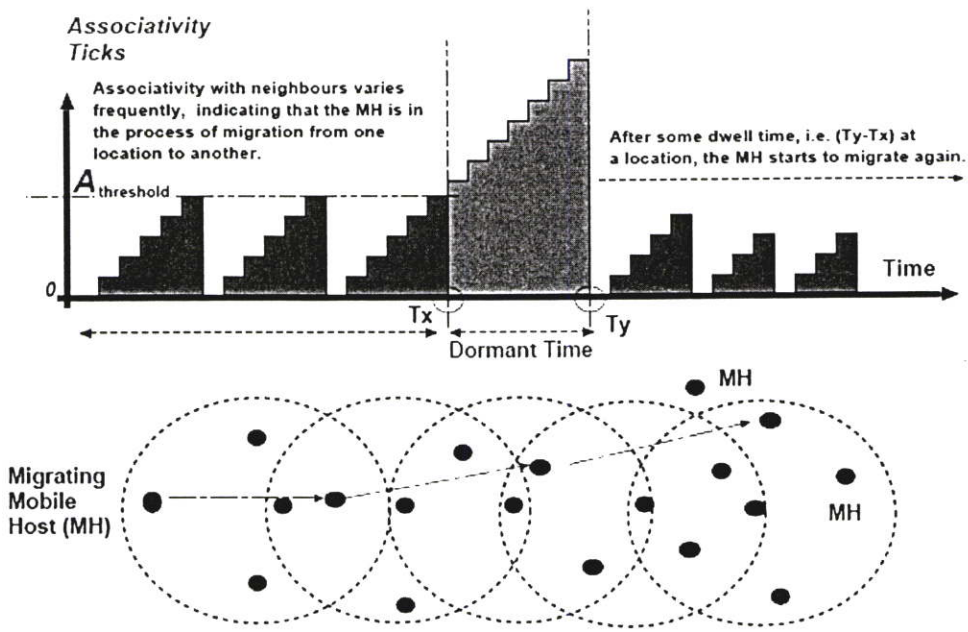
ตำแหน่ง จะส่งผ่านแพคเกจเข้ามายังพรอกซี่แล้วจึงใช้วิธีของ Distance Vector ในการส่งต่อแพคเกจไปให้โมไบล์โฮสต์ปลายทางซึ่งมีข้อมูลตำแหน่ง ในขั้นต่อไป

บทที่ 4
งานวิจัยที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงโปรโตคอลที่เกี่ยวกับงานวิจัยซึ่งเป็นต้นแบบที่นำมาพิจารณาและปรับปรุงประสิทธิภาพ โดยจะพิจารณาข้อดีข้อเสียของแนวคิดทั้งสองโปรโตคอลคือโปรโตคอลเอบีอาร์(Associativity-Based Routing Protocol: ABR) และโปรโตคอลแอลเออาร์ (Location-Aided Routing Protocol: LAR)

4.1 โปรโตคอล เอบีอาร์ (Associativity-Based Routing Protocol: ABR)

โปรโตคอลเอบีอาร์เป็นโปรโตคอลที่มีการเก็บข้อมูลเส้นทางก็ต่อเมื่อมีความต้องการ การค้นหาเส้นทางจะเริ่มต้นจากโมไบล์โฮสต์ต้นทางไปยังปลายทาง หากเส้นทางเดิมเสียหาย(Route Reconstruction) จะลบเส้นทางเก่าออกทั้งหมดจะไม่คงส่วนใดของเส้นทางเก่าเอาไว้ เพราะต้องการให้การเลือกเส้นทางเป็นไปตามแนวคิดของโปรโตคอล โดยปล่อยให้การเลือกเส้นทางเป็นหน้าที่ของโมไบล์โฮสต์ปลายทางจะเป็นผู้เลือกเส้นทางทั้งหมดจากหลายเส้นทางที่เป็นไปได้และใช้อัลกอริทึมพิจารณาเส้นทาง (Route Selection Algorithm) ซึ่งผลที่ได้จากการเลือกเส้นทางด้วยอัลกอริทึมนี้จะทำให้ได้เส้นทางที่มีเสถียรภาพคาดว่าจะอยู่ได้นาน (Long-Lived) เนื่องจากคุณสมบัติของ Associativity



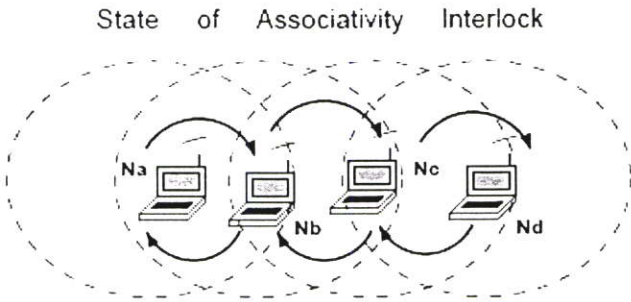
รูปที่ 4.1 แสดงถึงค่าความสัมพันธ์ (tick) ในช่วงที่เกิด Dormant

4.1.1 กฎของความสัมพันธ์ (Associativity Rule)

กฎความสัมพันธ์นี้หมายถึงค่าความสัมพันธ์ระหว่างคู่โมไบล์โฮสต์ใดๆ ซึ่งแสดงพฤติกรรมการเคลื่อนที่หรือการคงที่ของโมไบล์โฮสต์ที่อยู่รอบข้างกัน (Neighboring Mobile Host) ในช่วงเวลาหนึ่งซึ่งเรียกว่า Associativity-ticks จำนวนของ ticks นี้เมื่อมีค่าถึงค่าหนึ่งซึ่งมากกว่าค่าเทรชโฮล (Threshold) ที่กำหนด ลิงค์จะเปลี่ยนสถานะการเชื่อมเป็นช่วงคงที่ (Dormant Interval) ซึ่งแสดงถึงเสถียรภาพของลิงค์ระหว่างคู่โมไบล์โฮสต์ว่ามีเสถียรภาพและเหมาะสมในการใช้เป็นเส้นทางเชื่อมต่อ เป็นลิงค์ที่คาดว่าจะอยู่ได้นาน (Long-Lived) ดังที่แสดงไว้ในรูปที่ 4.1 ticks จะถูกปรับปรุงโดยโมไบล์โฮสต์ที่ดาต้าลิงค์เลเยอร์ จากการส่งบีคอน (Beacon Message) ออกไปทุกๆช่วงเวลาเพื่อแจ้งสถานะการมีอยู่ของโมไบล์โฮสต์และให้ทุกโมไบล์โฮสต์รอบข้างที่ได้รับเมสเสจบีคอนทำการปรับปรุงค่า ticks ซึ่งเป็นค่าความสัมพันธ์ระหว่างคู่โมไบล์โฮสต์ที่อยู่ใกล้กัน (สถานะการเชื่อมต่อหรือลิงค์ระหว่างคู่โมไบล์โฮสต์)

4.1.2 คุณสมบัติความสัมพันธ์ (Property of Associativity)

โมไบล์โฮสต์จะอยู่ในสถานะการเคลื่อนที่สูง ลิงค์มีเสถียรภาพต่ำหาก Associativity-tick มีค่าน้อยหรือต่ำกว่าค่าเทรชโฮล ในกรณีกลับกันถ้ามีค่ามากจะแสดงถือสภาวะความมีเสถียรภาพของลิงค์ระหว่างคู่โมไบล์โฮสต์ ซึ่งวิธีนี้เป็นแนวคิดสำคัญในการเลือกเส้นทางที่เหมาะสม ในกรณีทุกโมไบล์โฮสต์ระหว่างเส้นทางมีค่าความสัมพันธ์สูงทั้งหมด อาจเกิดสภาพปรากฏการณ์อินเตอร์ล็อค ซึ่งเป็นลักษณะที่ค่าความสัมพันธ์จะเพิ่มขึ้นถ้าคู่โมไบล์โฮสต์ยังอยู่ในระยะติดต่อกันได้ ดังในรูปที่ 4.2 ซึ่งค่า Associativity-tick จะถูกรีเซตต่อเมื่อโมไบล์โฮสต์ตนเองหรือโมไบล์โฮสต์รอบข้างเคลื่อนที่ออกไปนอกระยะทำการ ซึ่งไม่ได้เกิดจากเหตุเพราะเส้นทางถูกยกเลิกหรือสิ้นสุดการใช้งานเส้นทาง



รูปที่ 4.2 ปรากฏการณ์ Interlock

4.1.3 พารามิเตอร์ที่ใช้ในโปรโตคอล

โดยทั่วไปค่าพารามิเตอร์ที่ถูกใช้ในกระบวนการค้นหาเส้นทางจะประกอบด้วย

- เวลาที่ใช้ในการแก้ไขเส้นทาง(Recovery Time)
- จำนวนฮอพสั้นที่สุดที่ไปยังปลายทาง(Minimum Hop Count)

- ดีเลย์ (Propagation Delay)
- ขนาดลิงค์ (Link Capacity)
- พารามิเตอร์การหลีกเลี่ยงลูป (Loop Avoidance)

สำหรับโปรโตคอลเอบีอาร์ได้กำหนดพารามิเตอร์ประกอบด้วย

- ค่าความสัมพันธ์ของเส้นทาง(Route Longevity)
- โหลดระหว่างเส้นทาง (Relaying load)
- ขนาดของลิงค์ที่ถูกนำมาเป็นเส้นทาง (Route Capacity)

จุดสำคัญที่สุดของโปรโตคอลเอบีอาร์คือพารามิเตอร์ความสัมพันธ์ (Longevity) ซึ่งถ้าค่าความสัมพันธ์น้อยการรับส่งข้อมูลอาจมีการสะดุดหรือถูกขัดจังหวะบ่อยเนื่องจากการเคลื่อนที่ของโมไบล์โฮสต์ภายในเครือข่าย หรือการจราจรของข้อมูลซึ่งคับระหว่างเส้นทาง

4.1.4 อัลกอริทึมการเลือกเส้นทาง (Route Selection Algorithm)

กำหนดให้เซต A เป็นกลุ่มของเส้นทางต่างๆที่สามารถเชื่อมต่อจากโมไบล์โฮสต์ต้นทาง SRC ไปยังโมไบล์โฮสต์ปลายทาง DEST ได้ หากเส้นทางใดที่มีค่าความสัมพันธ์ระหว่างเส้นทางสูง เส้นทางนั้นจะถูกเลือกเป็นเส้นทางสื่อสารข้อมูลโดยโมไบล์โฮสต์ DEST โดยไม่พิจารณาค่าจำนวนของฮอพ แต่หากมีเส้นทางหลายเส้นทางที่มีค่าความสัมพันธ์ของเส้นทางสูงจะพิจารณาเลือกเส้นทางเหล่านั้นจากค่าจำนวนฮอพ แต่หากมีเส้นทางหลายเส้นทางที่ค่าความสัมพันธ์และรวมถึงค่าจำนวนฮอพเท่ากัน โมไบล์โฮสต์ DEST จะสามารถเลือกเส้นทางใดก็ได้ในเซตนี้ อัลกอริทึมต่อไปนี้ใช้ในการพิจารณาเลือกเส้นทางโดยโมไบล์โฮสต์ DEST

ให้ S_i เป็นเซตของเส้นทางที่เป็นไปได้จากโมไบล์โฮสต์ต้นทางไปยังปลายทาง โดยที่ i = 1, 2, 3...

ให้ RL^i_j เป็นภาระโหลดที่เกิดขึ้นในโมไบล์โฮสต์ j ซึ่งอยู่ในเส้นทาง S_i ใดๆ โดยที่ j = 1, 2, 3...

ให้ RL_{max} เป็นจำนวนภาระโหลดมากที่สุดเท่าที่โมไบล์โฮสต์แต่ละตัวจะรับได้

ให้ $A_{THREASHOLD}$ เป็นค่าความสัมพันธ์น้อยสุดที่ถือว่ามีเสถียรภาพ

ให้ AT^i_j เป็นค่าความสัมพันธ์ระหว่างโมไบล์โฮสต์ j ของเส้นทางใดๆใน S_i

ให้ H_i เป็นค่าความมีเสถียรภาพรวมของเส้นทางใดๆใน S_i

ให้ L_i เป็นค่าความไม่มีเสถียรภาพรวมของเส้นทางใดๆใน S_i

ให้ H_{iave} เป็นค่าเฉลี่ยเสถียรภาพของเส้นทางใดๆใน S_i

ให้ L_{iave} เป็นค่าเฉลี่ยของความไม่มีเสถียรภาพของเส้นทางใดๆใน S_i

ให้ Y_i แทนจำนวนโมไบล์โฮสต์ของเส้นทางใดๆใน S_i ที่สามารถรองรับภาระโหลดได้

ให้ U_i แทนจำนวนโมไบล์โฮสต์ของเส้นทางใดๆใน S_i ที่ไม่สามารถรองรับภาระโหลดได้

ให้ Y_{iave} แทนจำนวนค่าอัตราส่วนเฉลี่ยที่สามารถรับภาระโหลดได้

ให้ U_{iave} แทนจำนวนค่าอัตราส่วนเฉลี่ยที่ไม่สามารถรับภาระโหลดได้

Begin

For each Route in S_i

Begin

a <- 0

For each node j in Route S_i

Begin

If $(AT^i_j \geq A_{THREASHOLD})$ H_i++; Else L_i++;

If$(RL^i_j \geq RL_{max})$ U_i++; Else Y_i++;

a++;

End

$H_{iave} = H_i/a$; $L_{iave} = L_i/a$;

$U_{iave} = U_i/a$; $Y_{iave} = Y_i/a$;

End

//Best Route computation

Let the set of acceptable Route with $U_{iave} = 0$ and $H_{iave} \neq 0$ be P_l, where $P_l \subseteq S_i$

Begin

// Find Route with highest degree of association stability

Compute a Route k with $H_{kave} > H_{lave}$, $\forall$ l$\neq$k

Or if a set of Routes K_n exists such that $H_{k1ave} = H_{k1ave} = ...H_{kpave}$

Where n = {1, 2, 3, ...p}

Begin

//Compute minimum Hop Route without violating relaying load

Compute a Route Kk with Min{Kk} < Min{Km}, $\forall$k$\neq$m

Or if a set of Routes Ko exists such that Min{K1} = Min{K2} =...Min{Kq},

Where o = {1, 2,..q}

Begin

Multiple same Associativity & Minimum Hop Routes exists

Arbitrary select a minimum Hop Route Kk from Ko

End

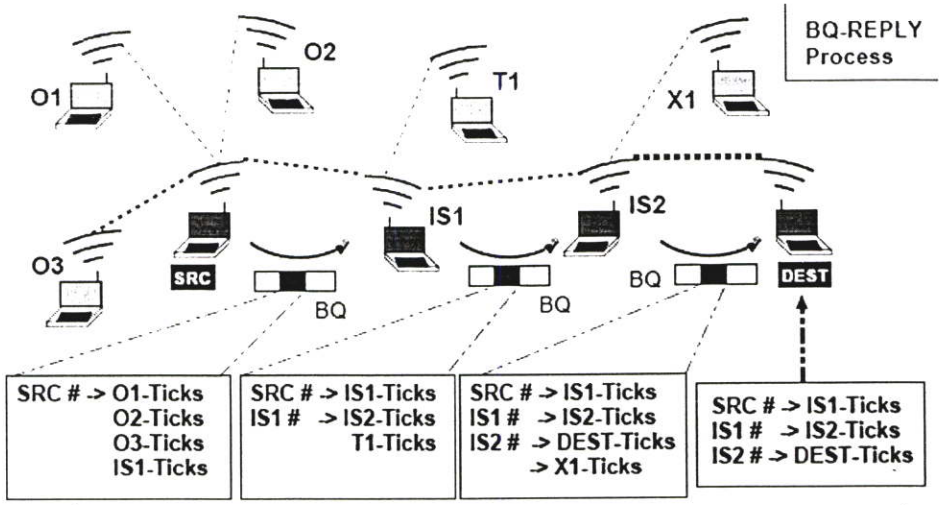
End

End

End

4.1.5 การค้นหาเส้นทาง (Route Discovery Phase)

กระบวนการค้นหาเส้นทางประกอบด้วยสองส่วนคือบีคิว (Broadcast Query: BQ) และรีพลาย (Reply) ซึ่งเรียกช่วงเวลาทั้งหมดของการบรอดคาสเมสเสจค้นหาเส้นทางและรอเมสเสจตอบกลับว่า ช่วงเวลา BQ-Reply ซึ่งเริ่มต้นจากทุกๆโมไบล์โฮสต์ ยกเว้นโมไบล์โฮสต์ข้างเคียงกับโมไบล์โฮสต์ปลายทาง ต้องการติดต่อไปยังปลายทางแต่ไม่มีเส้นทางไปยังโมไบล์โฮสต์ปลายทาง โมไบล์โฮสต์ที่ต้องการเส้นทางไปยังโมไบล์โฮสต์ปลายทางจะบรอดคาสต์เมสเสจร้องขอเส้นทางออกไปยังเครือข่าย โดยเมสเสจจะถูกส่งออกไปพร้อมกับการแนบหมายเลขลำดับหมายเลขลงไปด้วย ซึ่งใช้เพื่อระบุความแตกต่างของแต่ละบีคิวเมสเสจที่ถูกส่งออกมาเพื่อค้นหาเส้นทาง ในครั้งแรกที่เมสเสจบีคิวได้ถูกบรอดคาสต์โดยโมไบล์โฮสต์ต้นทาง ทุกๆโมไบล์โฮสต์ที่อยู่ระหว่างเส้นทางจะรับเมสเสจบีคิวและตรวจสอบว่าเคยได้รับเมสเสจนี้มาก่อนหรือไม่ ถ้าเคยได้รับเมสเสจนี้มาก่อนหน้าแล้ว จะดรอบเมสเสจนั้น แต่หากยังไม่เคยได้รับเมสเสจบีคิวมาก่อนโมไบล์โฮสต์จะตรวจสอบว่าตนเองเป็นโมไบล์โฮสต์ปลายทางหรือไม่ ถ้าไม่ใช่โมไบล์โฮสต์ระหว่างเส้นทางจะต่อท้ายแอดเดรสของโมไบล์โฮสต์ตนเองลงในปฟิลด์ IN IDs ของบีคิวเมสเสจและบรอดคาสต์เมสเสจบีคิวต่อไปยังเครือข่าย สำหรับค่าความสัมพันธ์ทั้งหมดกับโมไบล์โฮสต์รอบข้างก็จะถูกแนบไปพร้อมกันด้วยรวมทั้งค่าโหลด ดีเลย์และจำนวนฮอพ หลังจากนั้นโมไบล์โฮสต์ใดๆที่ได้รับเมสเสจบีคิวมา จะลบค่าความสัมพันธ์ที่ส่งมาทั้งหมดให้เหลือแค่เพียงส่วนที่เป็นความสัมพันธ์ระหว่างโมไบล์โฮสต์ตนเองกับโมไบล์โฮสต์ที่ได้ส่งเมสเสจบีคิวมาให้ตนเท่านั้น วิธีการแนบและการตัดทอนค่าความสัมพันธ์ต่างๆจะเป็นดังตัวอย่างในรูปที่ 4.4



รูปที่ 4.3 การแนบและตัดทอนค่าความสัมพันธ์ระหว่างเมสเสจถูกบรอดคาสต์

TYPE	SRC ID	DEST ID	LIVE	IN IDs	METRICS	SEQ NO.	CRC

รูปที่ 4.4 ข้อมูลฟิลด์ต่างๆในเมสเสจร้องขอเส้นทางบีคิว

โมไบล์โฮสต์ต้นทางจะรอคอยอยู่ช่วงเวลาหนึ่งเพื่อเก็บเส้นทางต่างๆ เพื่อเลือกเส้นทางที่ดีที่สุดจากเส้นทางเหล่านี้และเมื่อเลือกเส้นทางโดยใช้อัลกอริทึมการเลือกเส้นทางแล้ว เมสเสจรีพลายจะถูกส่งกลับไปหาโมไบล์โฮสต์ต้นทาง ระหว่างเส้นทางที่เมสเสจตอบกลับเส้นทางผ่านโมไบล์โฮสต์ใดที่ได้รับเมสเสจตอบกลับและพบว่าตนเองมีชื่ออยู่ในเส้นทางนั้นด้วยก็จะบันทึกไว้ว่าตนเป็นเส้นทางผ่านระหว่างคู่โมไบล์โฮสต์ต้นทางและปลายทางใด แต่หากไม่เป็นเช่นนั้น โมไบล์โฮสต์นอกเหนือเส้นทางได้รับเมสเสจรีพลายก็จะดรอบเมสเสจตอบกลับนั้นทิ้งเพื่อเป็นการป้องกันการเดินทางของแพคเกจข้อมูลซ้ำซ้อนกัน เมื่อโมไบล์โฮสต์ต้นทางได้รับเมสเสจรีพลายแล้วจะนำข้อมูลส่วนของเส้นทางมาบันทึกลงในตารางเส้นทางและใช้ข้อมูลเส้นทางนี้เป็นเส้นทางการเชื่อมต่อรับส่งข้อมูลไปยังโมไบล์โฮสต์ปลายทาง

TYPE	SRC ID	DEST ID	INs IDs	SEQ ID	ROUTE QUALITIES	CRC

รูปที่ 4.5 ข้อมูลฟิลด์ต่างๆในเมสเสจตอบกลับเส้นทาง

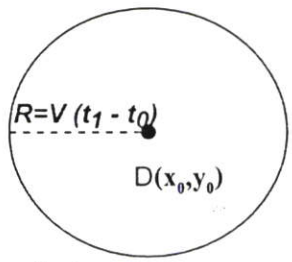
4.1.6 Route Reconstruction Phase

สำหรับส่วนที่สองนี้มีหน้าที่แก้ปัญหาเมื่อเส้นทางที่ใช้สื่อสารข้อมูลนั้นเกิดปัญหาทำให้ไม่สามารถสื่อสารข้อมูลต่อไปได้ ซึ่งแบ่งออกเป็น 3 ส่วนคือ การค้นหาเส้นทางโดยยังคงเส้นทางเดิมบางส่วน (Partial Route Discovery) ใช้ในการหาเส้นทางใหม่โดยยังคงเส้นทางเก่าบางส่วนที่ยังใช้ได้ โดยเมื่อการเชื่อมต่อเสียหาย โมไบล์โฮสต์ที่อยู่ในเส้นทางจะพยายามค้นหาเส้นทางไปยังโมไบล์โฮสต์ปลายทางเอง ซึ่งถ้าหาไม่พบก็จะย้อนกลับมายังต้นทางเพื่อทำกระบวนการค้นหาเส้นทางโดยโมไบล์โฮสต์ต้นทางอีกครั้ง ข้อดีคือเป็นการลดดีเลย์ที่จะเกิดขึ้นในการหาเส้นทางใหม่ทั้งหมด ซึ่งวิธีการนี้เรียกว่าการค้นหาเส้นทางแบบโลคอล (Local Query) ส่วนที่สองคือการลบเส้นทางที่ไม่ได้ใช้งานแล้ว (Invalid Route Erasure) เป็นการส่งเมสเสจเพื่อแจ้งให้โมไบล์โฮสต์ในเส้นทางถึงรู้ถึงสภาพเส้นทางที่มีปัญหา และลบเส้นทางนั้นออกจากความจำของโมไบล์โฮสต์ เนื่องจากเป็นเส้นทางที่ไม่ได้ใช้แล้ว ส่วนที่สามคือการบันทึกเพื่อระบุว่าตนเป็นเส้นทางการเชื่อมต่อใหม่(Valid Route Update) เมื่อการค้นหาเส้นทางแบบโลคอล สามารถหาเส้นทางได้สำเร็จ ก็จะปรับสถานะของโมไบล์โฮสต์เพื่อระบุให้ทราบว่าโมไบล์โฮสต์ใดบ้างเป็นเส้นทางการเชื่อมต่อและให้สามารถทำการเชื่อมต่อได้

4.2 โปรโตคอลแอลเออาร์ (Location-Aided Routing Protocol: LAR)

ได้นำเสนอวิธีการจำกัดการบรอดคาสต์เมสเสจการร้องขอเส้นทาง โดยสมมติว่าโมไบล์โฮสต์มีข้อมูลตำแหน่งของตนเองและเรียกใช้ได้ตลอดเวลาซึ่งเป็นข้อมูลที่ใหม่อยู่เสมอ โดยข้อมูลตำแหน่งอาจได้มาจากโมดูลจีพีเอส (Global Positioning Systems: GPS) ซึ่งฝังอยู่ในอุปกรณ์

เคลื่อนที่ไร้สายมีอยู่ในปัจจุบัน โดยโปรโตคอลแอลเออาร์ ได้นำเสนอพื้นที่สองส่วนที่เรียกว่า พื้นที่การส่งต่อเมสเสจร้องขอเส้นทาง (Request Zone) และพื้นที่คาดหวัง (Expected Zone) เมื่อโมไบล์โฮสต์ต้นทาง ต้องการติดต่อไปยังโมไบล์โฮสต์ปลายทาง D โดยสมมติว่าโมไบล์โฮสต์ต้นทางมีข้อมูลตำแหน่งของโมไบล์โฮสต์ D เมื่อเวลา t_0 ที่ผ่านมา สมมติเวลาปัจจุบันเป็น t_1 ดังนั้นเราจะสามารถคำนวณหาพื้นที่คาดหวังของโมไบล์โฮสต์ D ณ เวลา t_1 ในมุมมองของโมไบล์โฮสต์ต้นทางได้ โดยสมมติให้ v เป็นความเร็วของโมไบล์โฮสต์ D ดังนั้นโมไบล์โฮสต์ D จะมีพื้นที่คาดหวังอยู่ในพื้นที่รัศมีคือ $v(t_1-t_0)$ โดยมีศูนย์กลางอยู่ที่จุด Dx_0,Dy_0 เมื่อเวลา t_0 ซึ่งพื้นที่คาดหวังนี้จะเป็นส่วนที่คาดหวังว่าโมไบล์โฮสต์ปลายทาง D จะเคลื่อนที่อยู่ภายในพื้นที่ดังที่แสดงไว้ในรูปที่ 4.6

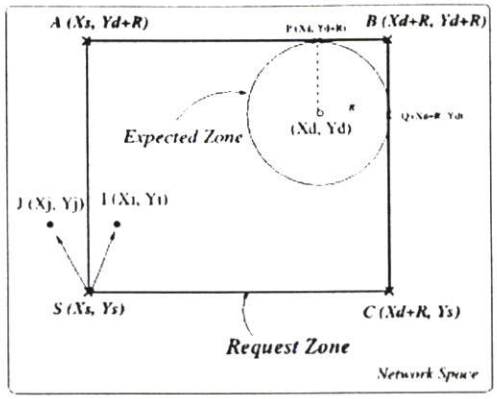


รูปที่ 4.6 พื้นที่คาดหวัง (Expected Zone)

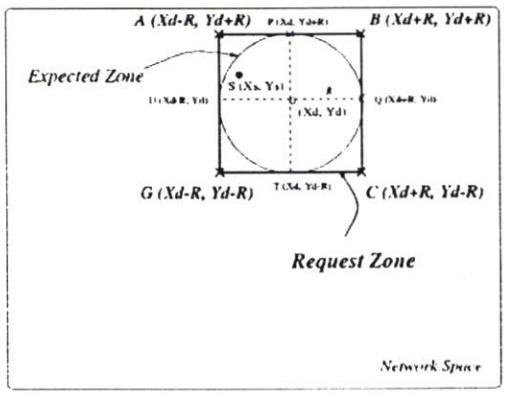
หลังจากที่ได้คำนวณหาพื้นที่คาดหวัง ต่อไปจะเป็นการหาพื้นที่การส่งต่อเมสเสจการร้องขอเส้นทางซึ่งเป็นพื้นที่ทั้งหมดที่สามารถบรอดคาสต์เมสเสจการร้องขอเส้นทางได้ ดังนั้นพื้นที่ส่งต่อเมสเสจร้องขอเส้นทงควรล้อมส่วนของพื้นที่คาดหวังไว้

หากโมไบล์โฮสต์ระหว่างทาง (IM) ใดๆ ได้รับเมสเสจการร้องขอเส้นทางก็จะตรวจสอบตำแหน่งของโมไบล์โฮสต์ตนเองและตรวจสอบว่าตำแหน่งของโมไบล์โฮสต์อยู่ภายในพื้นที่การส่งต่อเมสเสจการร้องขอเส้นทางหรือไม่ หากพบว่าตำแหน่งของโมไบล์โฮสต์อยู่ภายนอกพื้นที่การส่งต่อเมสเสจการร้องขอเส้นทางจะดรอบเมสเสจการร้องขอโดยไม่มีการส่งต่อไปยังโมไบล์โฮสต์รอบข้างอีก ซึ่งเป็นส่วนสำคัญในการจำกัดไม่ให้มีการส่งต่อเมสเสจการร้องขอเส้นทางออกไปทั้งเครือข่าย

ในการเลือกพื้นที่การส่งต่อเมสเสจการร้องขอเส้นทางจะต้องพิจารณาสองส่วน คือความเหมาะสมระหว่างพื้นที่ของการส่งต่อและการเกิดโอเวอร์เฮดที่เกิดจากการขยายพื้นที่ของพื้นที่ส่งต่อ หากพื้นที่การส่งต่อเมสเสจร้องขอเส้นทางมีขนาดเล็กเกินไป อาจทำให้ไม่สามารถพบเส้นทางไปยังโมไบล์โฮสต์ปลายทางได้ ซึ่งต้องมีการขยายพื้นที่การส่งต่อเมสเสจให้ใหญ่มากขึ้น สำหรับพื้นที่ในการส่งต่อเมสเสจร้องขอเส้นทาง โปรโตคอล แอลเออาร์ได้นำเสนอสองวิธี คือ



(ก)



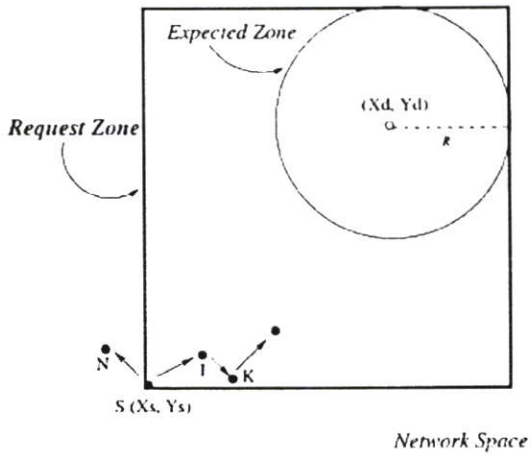
(ข)

รูปที่ 4.7 พื้นที่การส่งต่อเมสเสจซึ่งแยกจากพื้นที่คาดหวัง และรวมกับพื้นที่คาดหวัง

4.2.1 LAR แบบที่ 1

จะเลือกพื้นที่การส่งต่อเมสเสจเป็นสี่เหลี่ยมผืนผ้าหรือจตุรัสตามลำดับดังแสดงในรูปที่ 4.7โดยแบ่งออกเป็นสองกรณีคือกรณีโมไบล์โฮสต์ต้นทางอยู่ภายนอกพื้นที่คาดหวังและโมไบล์โฮสต์ต้นทางอยู่ภายในพื้นที่คาดหวังตามลำดับ สมมติว่าโมไบล์โฮสต์ต้นทาง S รู้ตำแหน่งของโมไบล์โฮสต์ปลายทาง D ว่าเคยอยู่ ณ ตำแหน่ง X_d,Y_d เมื่อเวลา t_0 ที่ผ่านมา เมื่อเวลา t_1 โมไบล์โฮสต์ต้นทาง S ได้เริ่มทำการค้นหาเส้นทางไปยังโมไบล์โฮสต์ปลายทาง D โดยสมมติว่าโมไบล์โฮสต์ S รู้ความเร็วของโมไบล์โฮสต์ D ว่าเคลื่อนที่ด้วยความเร็ว v ดังนั้นเมื่อเวลา t_1 โมไบล์โฮสต์ต้นทาง S จะสามารถหาพื้นที่คาดหวังของโมไบล์โฮสต์ปลายทาง D ได้เป็นพื้นที่มีรัศมีเป็น $R = v(t_1-t_0)$ โดยมีศูนย์กลางอยู่ที่ X_d,Y_d สำหรับพื้นที่การส่งต่อเมสเสจจะเป็นลักษณะสี่เหลี่ยมผืนผ้าโดยมีโมไบล์โฮสต์ต้นทาง S รวมภายในพื้นที่การส่งต่อเมสเสจด้วย เมื่อโมไบล์โฮสต์ต้นทาง S เริ่มทำการส่งต่อเมสเสจการร้องขอเส้นทาง ออกไปยังเครือข่าย โมไบล์โฮสต์ใดๆ ที่ได้รับเมสเสจก็จะตรวจสอบตำแหน่งของตนเองว่าอยู่ภายในพื้นที่การส่งต่อเมสเสจหรือไม่ ถ้าตำแหน่งของโมไบล์โฮสต์ที่ไม่อยู่ภายในพื้นที่การส่งต่อเมสเสจโมไบล์โฮสต์จะทำการดรอบเมสเสจการร้องขอเส้นทางนั้น แต่ถ้าโมไบล์โฮสต์พบว่าตนเองอยู่ภายในพื้นที่การส่งต่อเมสเสจ

โมไบล์โฮสต์จะทำการเพิ่มค่าเลขแอดเดรสโมไบล์โฮสต์ของตนและค่าความสัมพันธ์ระหว่างโมไบล์โฮสต์ลงในเมสเสจก่อนส่งต่อเมสเสจไปยังทิศทางของพื้นที่คาดหวังต่อไป ดังตัวอย่างที่แสดงในรูปที่ 4.7 โมไบล์โฮสต์ J ซึ่งมีตำแหน่ง X_j,Y_j ได้รับเมสเสจร้องขอเส้นทางและพบว่าตำแหน่งของโมไบล์โฮสต์อยู่นอกพื้นที่การส่งต่อเมสเสจจึงครอบเมสเสจร้องขอเส้นทางดังกล่าว แต่สำหรับโมไบล์โฮสต์ระหว่างทาง I ซึ่ง



รูปที่ 4.8 LAR Box-Scheme

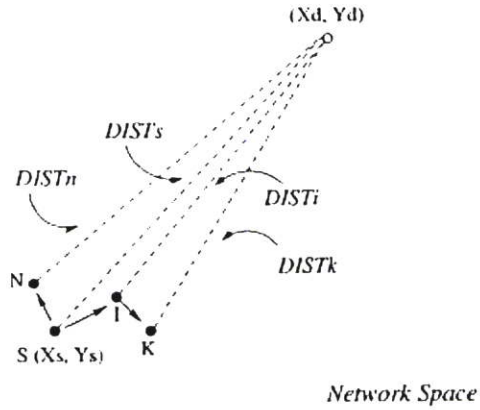
มีตำแหน่ง เป็น X_i,Y_i ตรวจพบว่าตำแหน่งโมไบล์โฮสต์ตนเองอยู่ภายในพื้นที่การส่งต่อเมสเสจจึงทำการเพิ่มค่าเลขประจำโมไบล์โฮสต์และค่าความสัมพันธ์ระหว่างโมไบล์โฮสต์ลงในเมสเสจแล้วทำการส่งต่อเมสเสจออกไปยังเครือข่ายอีกครั้งในรูปที่ 4.8

4.2.2 LAR แบบที่ 2

เป็นวิธีการส่งต่อเมสเสจการร้องขอเส้นทางโดยพิจารณาทิศทางให้ลู่เข้าใกล้โมไบล์โฮสต์ปลายทาง เมื่อโมไบล์โฮสต์ต้นทาง S ทำการส่งเมสเสจร้องขอเส้นทางออกมายังเครื่อข่ายโมไบล์โฮสต์ I ซึ่งอยู่ในเครือข่ายได้รับเมสเสจการร้องขอเส้นทางจาก S ก็จะคำนวณระยะทางจากตำแหน่งโมไบล์โฮสต์ของตน $I(X_i,Y_i)$ไปยังโมไบล์โฮสต์ปลายทาง $D(X_d,Y_d)$ ออกมาเป็นระยะทาง $DIST_i$ และโมไบล์โฮสต์ I จะทำการส่งต่อเมสเสจการร้องขอเส้นทางนี้ก็ต่อเมื่อ

$$\alpha(DIST_s)+\beta \geq DIST_i \qquad (4.1)$$

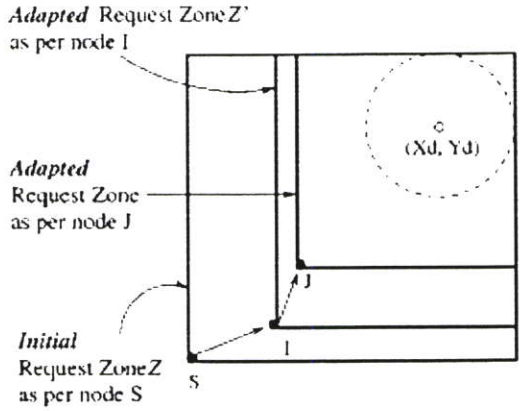
หากไม่เช่นนั้นโมไบล์โฮสต์ I จะครอบเมสเสจร้องขอเส้นทางดังแสดงในรูปที่ 4.9 ให้ α=1, β=0 ซึ่งสมมติว่าค่าตำแหน่งที่อ่านได้จากอุปกรณ์นั้นไม่มีการผิดพลาด สำหรับแอลเออาร์นั้นสามารถใส่ข้อมูลตำแหน่งของโมไบล์โฮสต์ลงไปในแพคเกจต่างๆได้หลายประเภท เพื่อให้มีข้อมูลตำแหน่งที่ใหม่อยู่เสมอ



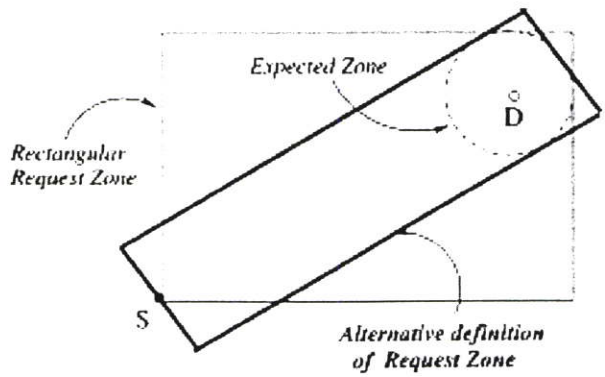
รูปที่ 4.9 พื้นที่การส่งต่อเมสเสจร้องขอเส้นทาง (Request zone) แบบที่ 2

4.2.3 รูปแบบอื่นๆของพื้นที่การส่งต่อเมสเสจร้องขอเส้นทาง

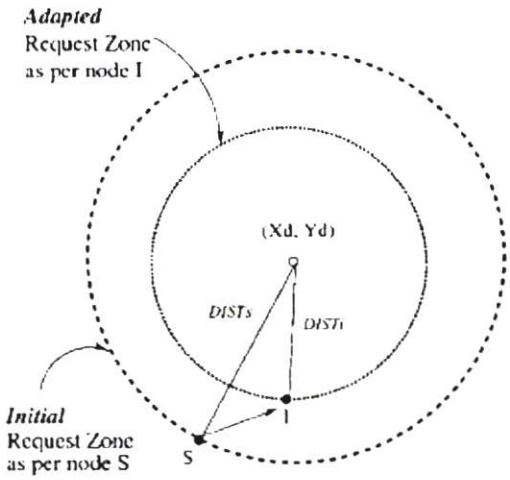
นอกจากลักษณะที่นำเสนอในรูปของสี่เหลี่ยมผืนผ้าแบบ LAR-Box ดังรูปที่ 4.8 และ LAR-Step ดังรูปที่ 4.9 แล้วยังมีการนำเสนอรูปแบบของพื้นที่การส่งต่อเมสเสจร้องข้อเส้นทางอื่นๆอีกดังรูปที่ 4.10-4.13



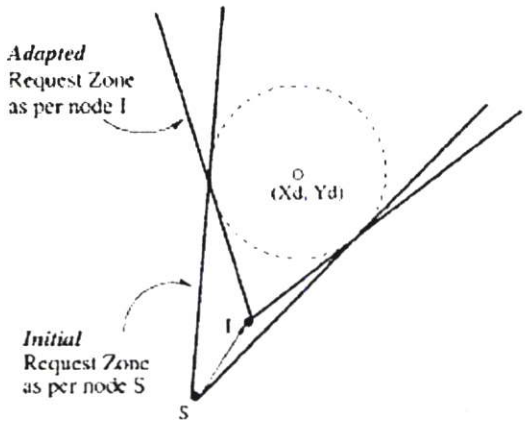
รูปที่ 4.10 LAR-Step แบบกรอบสี่เหลี่ยมลู่เข้า



รูปที่ 4.11 LAR-Box แบบแทยง



รูปที่ 4.12 LAR-Step แบบวงกลมลู่เข้า



รูปที่ 4.13 LAR-Step แบบกรวยลู่เข้า

บทที่ 5
การอิมพลีเมนต์โปรโตคอลเอ็มเอบีอาร์

ในบทนี้กล่าวถึงการออกแบบโปรโตคอลค้นหาเส้นทางเอบีอาร์ (Associativity-Based Routing Protocol: ABR) และโปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์ (Associativity-Based Routing Protocol: ABR) ลงในโปรแกรมจำลองการทำงานเอ็นเอสทู ทั้งสองโปรโตคอลเป็นโปรโตคอลค้นหาเส้นทางที่อยู่ในเลเยอร์ Network ของเลเยอร์ OSI

5.1 การจำลองการทำงานของโปรโตคอลที่นำเสนอเอ็มเอบีอาร์ (MABR)

ในส่วนนี้จะเป็นการออกแบบโปรโตคอลค้นหาเส้นทางเอบีอาร์และเอ็มเอบีอาร์ซึ่งอิมพลีเมนต์ลงในโปรแกรมจำลองการทำงานเอ็นเอสทูซึ่งประกอบด้วยส่วนงานหลัก สำหรับโปรโตคอลค้นหาเส้นทางเอบีอาร์จะประกอบด้วยส่วนสำคัญสองส่วนคือกระบวนการค้นหาเส้นทาง และกระบวนการจัดการต่อข้อมูลความสัมพันธ์ระหว่างโมไบล์โฮสต์ ขั้นตอนการค้นหาเส้นทางหรือช่วงที่เรียกว่า (BQ-Reply Duration) ประกอบด้วยอีเวนท์และกระบวนการตอบสนองต่ออีเวนท์ประกอบด้วย แพคเกจร้องขอเส้นทางบีคิว (Broadcast Query Packet) แพคเกจตอบกลับเส้นทาง (Reply Packet) อัลกอริทึมจัดการต่ออีเวนท์การร้องขอเส้นทางโดยโมไบล์โฮสต์ต้นทาง (Source) ระหว่างทาง(Intermediate) และปลายทาง (Destination) อัลกอริทึมจัดการต่ออีเวนท์การตอบกลับเส้นทางโดยโมไบล์โฮสต์ปลายทาง ระหว่างทาง และต้นทาง อัลกอริทึมการเลือกเส้นทาง (Route Selection Algorithm) ขั้นตอนการรับส่งและเก็บข้อมูลความสัมพันธ์ระหว่างโมไบล์โฮสต์ (Beaconing) ประกอบด้วย แพคเกจส่งข้อมูลบีคอน อัลกอริทึมในการจัดการอีเวนท์บีคอน การอิมพลีเมนต์กระบวนการสำคัญสำหรับโปรโตคอลเอ็มเอบีอาร์ซึ่งได้ปรับปรุงจากโปรโตคอลค้นหาเส้นทางต้นแบบเอบีอาร์ แบ่งออกเป็นสองส่วนเช่นกัน ประกอบด้วยกระบวนการค้นหาเส้นทางโดยใช้ข้อมูลตำแหน่ง และกระบวนการจัดการต่อข้อมูลความสัมพันธ์ระหว่างโมไบล์โฮสต์ ขั้นตอนเพิ่มเติมและปรับปรุงจากโปรโตคอลค้นหาเส้นทางเอบีอาร์สำหรับโปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์ที่นำเสนอเฉพาะในช่วงเวลาของการค้นหาเส้นทาง (BQ-Reply Duration) ประกอบด้วย แพคเกจร้องขอเส้นทางที่แนบข้อมูลตำแหน่ง แพคเกจตอบกลับเส้นทางที่แนบข้อมูลตำแหน่ง อัลกอริทึมจัดการต่อข้อมูลตำแหน่งที่แนบลงในอีเวนท์การร้องขอเส้นทางโดยโมไบล์โฮสต์ต้นทาง ระหว่างทาง และปลายทาง และเงื่อนไขการส่งต่อแพคเกจร้องขอเส้นทาง อัลกอริทึมจัดการต่ออีเวนท์การตอบกลับเส้นทางที่แนบข้อมูลตำแหน่งโดยโมไบล์โฮสต์ปลายทาง ระหว่างทาง และต้นทาง อัลกอริทึมการเลือกเส้นทาง (Route Selection Algorithm) อัลกอริทึมจัดการต่อข้อมูลตำแหน่งภายในโมไบล์โฮสต์ ขั้นตอนการรับส่งและเก็บ

ข้อมูลความสัมพันธ์ระหว่างโมไบล์โฮสต์ในเอ็มเอบีอาร์ซึ่งยังคงอัลกอริทึมเดิมของโปรโตคอลต้นแบบเอบีอาร์เอาไว้ ประกอบด้วย แพคเกจส่งข้อมูลบีคอน อัลกอริทึมในจัดการอีเวนท์บีคอน

5.1.1 การอิมพลีเมนต์โปรโตคอลค้นหาเส้นทางเอบีอาร์

ในการอิมพลีเมนต์โปรโตคอลแบ่งออกเป็นการออกแบบแพคเกจ และกระบวนการจัดการกับแพคเกจที่เป็นอีเวนท์ ซึ่งในโปรโตคอลเอบีอาร์มีแพคเกจที่เป็นส่วนสำคัญประกอบด้วย BQ แพคเกจ Reply แพคเกจและ Beacon แพคเกจ และส่วนอัลกอริทึมเพื่อตอบสนองต่อแพคเกจที่เป็นอีเวนท์ประกอบด้วย อัลกอริทึมจัดการต่ออีเวนท์การร้องขอเส้นทาง อัลกอริทึมจัดการต่ออีเวนท์การตอบกลับเส้นทาง อัลกอริทึมการเลือกเส้นทาง และอัลกอริทึมในการจัดการอีเวนท์บีคอน

5.1.1.1 รูปแบบแพคเกจร้องขอเส้นทาง (ABR Broadcast Query Packet)

รูปแบบแพคเกจร้องขอเส้นทางบีคิว (Broadcast Query Packet: BQ) จะอิงกับงานวิจัยที่เสนอใน [8] ซึ่งมีโครงสร้างดังรูปที่ 5.1 โดยแต่ละฟิลด์จะมีความหมายตามตารางที่ 5.1

Type	SRC ID	DEST ID	LIVE	IM IDs	METRICS	SEQ NO.	CRC

รูปที่ 5.1 ข้อมูลฟิลด์ต่างๆในแพคเกจร้องขอเส้นทางบีคิว

ตารางที่ 5.1 แสดงความหมายของแต่ละฟิลด์ในแพคเกจร้องขอเส้นทาง

TYPE	ชนิดของแพคเกจ (ระบุเป็น BQ Packet)
SRC ID	ID ของโมไบล์โฮสต์ต้นทาง
DEST ID	ID ของโมไบล์โฮสต์ปลายทาง
LIVE	ค่าจำนวนฮอพก่อนที่จะถูกครอบจากเครือข่าย
IM IDs	หมายเลข ID ของโมไบล์โฮสต์ต่างๆที่เป็นเส้นทางเชื่อมต่อ
METRICS	ค่าพารามิเตอร์เพื่อใช้วัดคุณภาพของเส้นทาง เช่นค่าความสัมพันธ์ระหว่างเส้นทาง
SEQ NO.	หมายเลขซีเควนซ์ของแพคเกจ
CRC	โค๊ดเพื่อตรวจสอบความผิดพลาดในการส่งข้อมูล

จากข้อมูลในตารางที่ 5.1 นำมาสร้างเป็นแพคเกจข้อมูลโดยไม่ได้สร้าง CRC ลงในแพคเกจเนื่องจากเพื่อความง่ายในการออกแบบซึ่งสมมติว่าการส่งแพคเกจ ข้อมูลที่ได้รับจะไม่มี Error จะได้การโค๊ตในภาษา C++ ได้ดังนี้ โดย u_intX_t แทนชนิดข้อมูลไม่เก็บเครื่องหมายขนาด X บิท nsaddr_t เป็นโครงสร้างข้อมูลของการเก็บแอดเดรสในโมไบล์โฮสต์ขนาด 32 บิท

Broadcast Query Packet

```
typedef struct hdr_abr_bc {
        u_int8_t   bc_type;                                  // Type
        nsaddr_t   bc_src_addr;                              // SRC ID
        nsaddr_t   bc_dst_addr;                              // DEST ID
        u_int16_t  bc_live;                                  // LIVE
        nsaddr_t   bc_hostAddresses[ABR_MAX_SIZE];           // IN IDs
        u_int16_t  bc_associativity[ABR_MAX_SIZE];           // METRICS
        u_int16_t  bc_noOfNeighbors;
        nsaddr_t   bc_neighborAddresses[ABR_MAX_SIZE];
        u_int16_t  bc_neighborTick[ABR_MAX_SIZE];
        u_int32_t  bc_seqno;                                 // SEQ NO.
        ......
        ......
  }
} hdr_abr_bc;
```

5.1.1.2 รูปแบบแพคเกจตอบกลับเส้นทาง (ABR Reply Packet)

รูปแบบแพคเกจตอบกลับเส้นทาง (Reply Packet) อิงกับงานวิจัยที่เสนอจากโปรโตคอลเอบีอาร์ ซึ่งมีโครงสร้างดังรูปที่ 5.2 โดยแต่ละฟิลด์จะมีความหมายตามตารางที่ 5.2

TYPE	SRC ID	DEST ID	IM IDs	SEQ ID	ROUTE QUALITIES	CRC

รูปที่ 5.2 ข้อมูลฟิลด์ต่างๆในแพคเกจตอบกลับเส้นทาง

ตารางที่ 5.2 แสดงความหมายของแต่ละฟิลด์ในแพคเกจตอบกลับเส้นทาง

TYPE	ชนิดของแพคเกจ (ระบุเป็น Reply Packet)
SRC ID	ID ของโมไบล์โฮสต์ต้นทาง (ต้นทางของแพคเกจ Reply)
DEST ID	ID ของโมไบล์โฮสต์ปลายทาง (ปลายทางของแพคเกจ Reply)
IM IDs	หมายเลข ID ของโมไบล์โฮสต์ต่างๆที่เป็นเส้นทางเชื่อมต่อ
SEQ NO.	หมายเลขซีเควนซ์ของแพคเกจ
Route Qualities	ตอบกลับคุณภาพของเส้นทางที่เลือก (สำหรับโมไบล์โฮสต์ต้องการทราบคุณภาพของเส้นทาง)
CRC	โค๊ดเพื่อตรวจสอบความผิดพลาดในการส่งข้อมูล

จากข้อมูลในตารางที่ 5.2 นำมาสร้างเป็นแพคเกจข้อมูลโดยจะข้าม Route Qualities และ CRC ออกไปเพื่อง่ายในการ จะได้โค๊ตในภาษา C++ ได้ดังนี้

Reply Packet

```
typedef struct hdr_abr_reply {
        u_int8_t   re_type;                          // TYPE
        nsaddr_t   re_src_addr;                      // SRC ID
        nsaddr_t   re_dst_addr;                      // DEST ID
        nsaddr_t   re_inx[ABR_MAX_SIZE];             // IM IDs
        ......
        ......
  }
} hdr_abr_reply;
```

5.1.1.3 รูปแบบแพคเกจบีคอน (ABR Beacon Packet)

รูปแบบแพคเกจบีคอน (Beacon Packet) อิงกับงานวิจัยที่เสนอจากโปรโตคอลเอบีอาร์ แต่ไม่มีข้อมูลแพคเกจที่ชัดเจน ซึ่งข้อมูลของแพคเกจบีคอนต้องมีขนาดเล็กและมี ID ของโมไบล์โฮสต์ต้นทางที่ส่งแพคเกจบีคอนเป็นสำคัญ จึงออกแบบให้มีโครงสร้างดังรูปที่ 5.3 โดยแต่ละฟิลด์จะมีความหมายตามตารางที่ 5.3

TYPE	SRC ID

รูปที่ 5.3 ข้อมูลฟิลด์ต่างๆในแพคเกจตอบกลับเส้นทาง

ตารางที่ 5.3 ความหมายของแต่ละฟิลด์ในแพคเกจบีคอน

TYPE	ชนิดของแพคเกจ (ระบุเป็น Beacon Packet)
SRC ID	ID ของโมไบล์โฮสต์ต้นทาง (โมไบล์โฮสต์ที่ส่ง beacon Packet)

จากข้อมูลในตารางที่ 5.3 นำมาสร้างเป็นแพคเกจข้อมูลจะได้โค๊ตในภาษา C++ ได้ดังนี้

Beacon Packet

```
typedef struct hdr_abr_tick {
        u_int8_t ti_type;                    // TYPE
        nsaddr_t ti_source_addr;             // SRC ID
        ......
```

.....

} hdr_abr_tick;

5.1.1.4 อัลกอริทึมการร้องขอเส้นทาง (ABR Route Discovery Algorithm)

สำหรับส่วนที่สองที่มีหน้าที่ตอบสนองต่ออีเวนท์ประกอบด้วย อีเวนท์การร้องขอเส้นทาง อัลกอริทึมจัดการต่ออีเวนท์การตอบกลับเส้นทาง อัลกอริทึมการเลือกเส้นทาง และอัลกอริทึมในการจัดการอีเวนท์บีคอน อีเวนท์การร้องขอเส้นทางและการตอบกลับเส้นทางสามารถเขียนได้เป็นอัลกอริทึมแยกเป็นโมไบล์โฮสต์ต้นทาง ระหว่างทาง และปลายทางได้ดังนี้

Algorithm for Intermediate Mobile Host (ABR)

If intermediate I gets route request Packet (BQ). Then

- Rebroadcast route request Packet

Else If intermediate I gets reply Packet (Reply). Then

If self mobile host's ID is one of a reply route. Then

- Make valid route to route cache.
- Forward reply Packet backward to upper-stream node referred from a reply route.

Else

- Drop Packet

End If

End If

Algorithm for Source Mobile Host (ABR)

If source S wants to communicates to destination D, Then

- Broadcast route request message uses normal flooding mode

End If

Algorithm for Destination Mobile Host (ABR)

If destination D gets the route request Packet (BQ). Then

- Select a route uses route **selection Algorithm**
- Send reply Packet back to the source using selected path

End If

ในการออกแบบโปรโตคอลได้แยกเมธอดการทำงานในส่วนของอัลกอริทึมการจัดการอีเวนท์ร้องขอเส้นทางและอีเวนท์ตอบกลับเส้นทาง เป็นเมธอดการส่ง BQ (Source) ฟังก์ชันการตอบสนองต่อแพคเกจ BQ(Intermediate, Destination) เมธอดการส่ง Reply (Destination) และเมธอดการตอบสนองต่อแพคเกจ reply แสดงเป็นโค๊ตในภาษา C++ ซึ่งประกอบด้วยส่วนการส่งแพคเกจร้องขอเส้นทางออกไปยังเครือข่ายจากโมไบล์โฮสต์ต้นทาง ด้วยเมธอด send_BC() และมีการตั้งเวลา Time-out ด้วยเมธอด RouteDiscoveryTime() ซึ่งจะถูกกำหนดเวลาไว้เป็นค่าคงที่เวลาหนึ่ง ในตัวอย่างนี้กำหนดเป็นค่าคงที่ ABR_ROUTE_DISCOVERY_TIMEOUT หลังจากเวลาผ่านไป ABR_ROUTE_DISCOVERY_TIMEOUT แล้วเมธอด RouteDiscoveryTimer::expire() จะถูกเรียกขึ้นมาทำงานเพื่อตรวจสอบว่าพบเส้นทางหรือยัง ถ้ายังจะกลับมาเรียกเมธอด send_BC() เพื่อส่งแพคเกจร้องขอเส้นทางไปยังเครือข่ายอีกครั้ง (Retransmission) จำนวนครั้งของการลองส่งใหม่จะไม่เกินค่าคงที่ ABR_MAX_REQUESTS ครั้ง สำหรับโค๊ตตัวอย่างถูกแสดงเอาไว้พอเป็นที่สังเขปยังไม่ครบถ้วนสมบูรณ์ เพื่อให้เห็นเฉพาะการโปรแกรมในจุดสำคัญ สำหรับในตัวโปรแกรมที่ใช้จำลองการทำงานจริงอาจมีรายละเอียดมากกว่านี้

Sending BQ (Source) Method

```
void ABR::send_BC(nsaddr_t dest, u_int16_t ttl) {
        Packet *Query    = allocpkt();              // Allocate Query Packet
        hdr_abr_bc *bc;                             // Packet type
        bc->bc_type       = ABR_PT_BC;              // Type = BQ
        bc->bc_src_addr    = index;                 // SRC = self ID (sending Node)
        bc->bc_dst_addr    = dest;                  // DEST
        bc->bc_seqno       = seqno++;               // SEQ NO.
        bc->bc_live        = 0;                     // LIVE
        bc->bc_noOfNeighbors = 0;                   // ..METRICS
  // Add neighbor information to BQ (Associativity added to Metrics)
 neighbor_entry *entry = s_table;          // s_table stores ass-tick in mobile host
 while(entry != NULL) {
        bc->bc_neighborAddresses[bc->bc_noOfNeighbors] = entry->addr;
        bc->bc_neighborTick[bc->bc_noOfNeighbors] = entry->associativity_ticks;
        bc->bc_noOfNeighbors += 1;
    entry = entry->next;
 }
        sentBCs++;                  // count number of sent BQ
```

```
 // Send the BQ
 Scheduler::instance().schedule(target_, Query, 0);
 // Start BQ timer for RouteDiscovery time-out and Retransmission
    RouteDiscoveryTimer *timer = new RouteDiscoveryTimer(this, dest);
          ......
 }
/****************** Route Discovery Timer *********************/

RouteDiscoveryTimer::RouteDiscoveryTimer(ABR* abr, nsaddr_t dest) {
 this->destaddr = dest;
 this->sourceaddr = this->abr->index;
 this->distance_DEST = -1;
 this->ttl = ABR_MAX_TTL;                                    //Live
  // Schedule this timer
 resched(ABR_ROUTE_DISCOVERY_TIMEOUT);
}
/****************** Time-out Handler ****************/
void RouteDiscoveryTimer::expire(Event*) {
 // Check for a valid route
 abr_route_entry *find = ABR_lookupRouteEntry(abr->r_table, sourceaddr, destaddr,
PRIMARY_ROUTE);
  // Update number of times this timer has been scheduled
 times += 1;
 if (find != NULL) {      // if route founded no retransmits bq again
          delete this;
          return;
 } // Route not found try until reached to limited
    if (times == ABR_MAX_REQUESTS) {
          delete this;
          return;
    }
    else {
          // Try to Query again
```

```
        abr->send_BC(destaddr);
        // Reschedule myself
        resched(ABR_ROUTE_DISCOVERY_TIMEOUT);
        return;
    }
}
```

เมธอด handle_QUERY() จะเป็นเมธอดในการสนองต่อแพคเกจ BQ ที่เป็นอีเวนท์ถูกส่งมาจากโมไบล์โฮสต์ต้นทางด้วยเมธอด send_BC() และเมื่อแพคเกจมาถึงโมไบล์โฮสต์ระหว่างทางหรือปลายทางแล้ว เมธอด handle_QUERY() จะถูกเรียกขึ้นมาทำงานเพื่อเลือกว่าจะส่งต่อเมสเสจ(หากเป็นโมไบล์โฮสต์ระหว่างทาง)หรือจะตอบกลับด้วยเมสเสจตอบกลับ(reply กรณีที่เป็นโมไบล์โฮสต์เป้าหมาย) ซึ่งหากโมไบล์โฮสต์ที่ได้รับแพคเกจ BQ เป็นโมไบล์โฮสต์ปลายทางจะทำการคำนวณค่าคุณภาพของเส้นทางต่างๆด้วยเมธอด ABR_calculateRouteMetrics() หลังจากนั้นจะเรียอกเมธอด send_REPLY(); เพื่อเลือกเส้นทางที่ดีที่สุดจากที่คำนวณไว้ก่อนหน้าแล้วตอบเส้นทางนั้นกลับไปด้วยเมสเสจรีพลายย้อนไปยังเส้นทางที่เลือกไว้ แต่หากโมไบล์โฮสต์ที่ได้รับแพคเกจ BQ เป็นโมไบล์โฮสต์ระหว่างทางก็จะส่งต่อเมสเสจร้องขอเส้นทางนั้นต่อไป แต่หากค่า LIVE ของเมสเสจหมดลง แพคเกจจะถูกดรอบไม่ส่งต่อไปอีก

Handle Query (BQ) Method for Intermediate and Destination

```
void ABR::handle_QUERY(Packet *p) {
        nsaddr_t sender  = ip->saddr();          // get sender address
        hdr_abr_bc *bc = p;                      // pointer bc points to received BQ Packet
        nsaddr_t source  = bc->bc_src_addr;      // set source address from Packet to variable
        nsaddr_t dest   = bc->bc_dst_addr;       // set deste address from Packet to variable
        u_int32_t seqno = bc->bc_seqno;
        u_int16_t *hops = &(bc->bc_live);        // LIVE
        u_int16_t * noOfNeighbors  = &(bc->bc_noOfNeighbors);          // METRICS..
        nsaddr_t * hostAddresses = &(bc->bc_hostAddresses[0]);
        u_int16_t *hostTicks = &(bc->bc_associativity[0]);
        nsaddr_t *neighborAddresses = &(bc->bc_neighborAddresses[0]);
        u_int16_t * neighborTicks    = &(bc->bc_neighborTick[0]);
    // Check that we haven't handled this request before. If we are the
    // destination host, we are interested in processing the Query no matter
    // what in order to find the "best" route alternative
```

```
// Find our associativity tick in list
            u_int16_t assoTick = 0;
            for (int i = 0; i < *noOfNeighbors; i++) {
              if (neighborAddresses[i] == index) {
          assoTick = neighborTicks[i];
          break;
              }
            }
  // Am I destination? So calculate route quality
    if (index == dest) {
    buffered_route *br = ABR_calculateRouteMetrics(*hops, hostAddresses, hostTicks);
   send_REPLY(source, br);                    // send reply back to source
    }
   // Insert our neighbor information
 // Current host is not destination and the ttl is zero, drop it!
          if ((dest != index) && (ip->ttl_ == 0)) {
                  drop(p, DROP_RTR_TTL);
                  return;
          }
 // send out BQ Packet to neighbor
 Scheduler::instance().schedule(target_, p, 0.01 * Random::uniform());
}
```

เมื่อแพคเกจ BQ ถูกส่งมาถึงโมไบล์โฮสต์ปลายทางแล้วซึ่งเมธอด handle_QUERY() จะถูกเรียกขึ้นมาทำงาน โมไบล์โฮสต์ปลายทางจะเลือกเส้นทางและตอบกลับเส้นทางด้วยเมธอด ABR_getBestRoute() และ send_REPLY() ตามลำดับ

Sending Reply from Destination

```
void ABR::send_REPLY(nsaddr_t source, buffered_route *br) {
 // Select the best route
 buffered_route *selected = br;
 if (br == NULL) {
   selected = ABR_getBestRoute(bufferedRoutes[source]);
 }
```

```
// Allocate REPLY Packet
 Packet *reply = allocpkt();

 hdr_abr_reply *re;
 re->re_type      = ABR_PT_REPLY;
 re->re_src_addr  = index;
 re->re_dst_addr  = source;
 re->re_hopCnt    = selected->hops;
 // Copy the route into the REPLY Packet
 for (int i = 0; i < selected->hops; i++) {
  re->re_inx[i] = selected->route[i];
 }
 // Send the reply Packet
 Scheduler::instance().schedule(target_, reply, 0.);
}
```

เมื่อโมไบล์โฮสต์ระหว่างทางได้รับเมสเสจรีพลายจะเรียกเมธอด handle_REPLY() และตรวจสอบว่าตนเป็นหนึ่งในเส้นทางการเชื่อมต่อระหว่างโมไบล์โฮสต์ปลายทางและต้นทางก็จะปรับปรุงค่าในตารางเส้นทางให้ตรงกันด้วยเมธอด ABR_insertRoute() และส่งต่อแพคเกจตอบกลับย้อนไปหาโมไบล์โฮสต์ต้นทาง แต่หากโมไบล์โฮสต์ระหว่างทางไม่ได้เป็นหนึ่งในเส้นทางการเชื่อมต่อ เมสเสจตอบกลับจะถูกดรอบ แต่หากโมไบล์โฮสต์ต้นทางเป็นผู้ได้รับแพคเกจตอบกลับ เมธอด handle_REPLY() จะถูกเรียกขึ้นมาเพื่อปรับปรุงตารางเส้นทางและดรอบเมสเสจนั้นไม่ส่งต่อไปอีก

Handle Reply for Intermediate and Source

```
void ABR::handle_REPLY(Packet *p) {
 hdr_abr_reply * re;
 nsaddr_t reply_dest  = re->re_dst_addr;
 nsaddr_t reply_src   = re->re_src_addr;
 // Find our position in the route (Intermediate check whether itself is one of the route in reply?)
// If I'm intermediate and get reply Packet
 int re_index = -1;
 for (int i = (re->re_hopCnt - 1); i >= 0; i--) {
  if (re->re_inx[i] == index) {
```

```
    re_index = i;
    break;
   }
  }
 assert(re_index != -1);
    nsaddr_t next;
    if (index == reply_dest) {
         next = index;
    }
    else {
         next  = re->re_inx[re_index - 1];
    }
    // Update routing table (Valid Route Action in Intermediate)
    abr_route_entry *newentry = ABR_insertRoute(&r_table, reply_dest, reply_src, next,
sender, PRIMARY_ROUTE);
    newentry->expires_at = ABR_CURRENT_TIME + ABR_ROUTETIMEOUT;
    newentry->dest_hops = re->re_hopCnt - re_index;
    newentry->source_hops = re_index;
    //If I'm Destination of reply Packet (source)
    if (index == reply_dest) {
         // delete Route Discovery  timer
          RouteDiscoveryTimer *timer = look->second;
          timer->force_cancel();
          delete timer;
         Packet::free(p);                    // Send no further , I'm destination of reply Packet
    }
    else { // Send this update further along the route , if I'm not the destination of reply Packet
         Scheduler::instance().schedule(target_, p, 0);
    }
}
```

5.1.1.5 อัลกอริทึมการทำบีคอน (ABR Beaconing Algorithm)

เมธอด send_TICK() มีหน้าที่ในการส่งแพคเกจบีคอนออกไปยังโมไบล์โฮสต์รอบข้าง โดยแพคเกจนี้จะไม่มีการส่งต่อ และการส่งจะมีเพียงข้อมูลไอดีของโมไบล์โฮสต์ที่ส่ง จึงทำให้

แพคเกจมีขนาดเล็ก และมีเมธอด expire() เป็นตัวกำหนดคาบเวลาในการเรียกเมธอด send_TICK() ขึ้นมาทำงานทุกๆคาบเวลา

Send Associativity Tick (Send Beacon)

```
void ABR::send_TICK() {
 // Send another tick to the neighbors
 Packet *tick = allocpkt();
  hdr_abr_tick *ti;
 ti->ti_type       = ABR_PT_TICK;
 ti->ti_source_addr = index;
p->saddr() = index;
 Scheduler::instance().schedule(target_, tick, 0);
}
/***************************Tick Timer ************************/
void TickTimer::expire(Event*) {
 // Send a TICK message for the neighbors
 abr->send_TICK();
 // Reschedule timer
 resched(ABR_AT_INTERVAL + ABR_JITTER);
}
```

เมธอด handle_TICK() ใช้สำหรับการจัดการแพคเกจบีคอนที่ได้รับจากการส่งของโมไบล์โฮสต์อื่นๆด้วยเมธอด send_TICK() เมธอด handle_TICK() จะทำหน้าที่ในการนับจำนวนบีคอนที่ได้รับจากโมไบล์โฮสต์ใดๆเอาไว้

Handle Tick (Beacon counting)

```
void ABR::handle_TICK(Packet *p) {
   hdr_abr_tick *ti;
 nsaddr_t neighbor = ti->ti_source_addr;
 // count neighbor's Ticks ; s_table store list of  Associativity-ticks of  mobilehosts
 ABR_tickNeighbor(&s_table, neighbor, ABR_CURRENT_TIME);
 Packet::free(p);
}
```

5.1.2 การอิมพลีเมนต์โปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์

เป็นการแก้ไขต่อจากการอิมพลีเมนต์โปรโตคอลค้นหาเส้นทางเอบีอาร์ดังในหัวข้อที่ 5.1.2 ดังในรูปที่ 5.4 5.5และ 5.6 เป็นอัลกอริทึมที่เพิ่มเติมจากโปรโตคอลเอบีอาร์เดิม สังเกตในส่วนที่เป็นบล็อคสีเข้มจะเป็นส่วนที่เพิ่มเติมเข้าไป ซึ่งได้เขียนเป็นโฟลวของอัลกอริทึมแบ่งออกเป็น 3 ส่วน คือ โมไบโฮสต์ต้นทาง ปลายทาง และระหว่างทาง ดังนั้นจึงแก้ไขปรับปรุงอัลกอริทึมที่แสดงในรายละเอียดของโปรโตคอลเอบีอาร์เป็นดังนี้

Algorithm for Intermediate Mobile Host(MABR)

If intermediate I gets route request Packet (BQ). Then
- *If* position information available in request Packet (BQ). Then
  - Update source and destination positions with timestamps to its PIT.
  - Compute expected zone and request zone by source, destination positions and timestamps
  - Get position itself
    - *If* position itself I is in request zone. Then
      - Rebroadcast route request Packet
    - *Else*
      - Drop Packet
    - *End If.*
- *Else*
  - Rebroadcast route request Packet
- *End If*

Else If intermediate I gets reply Packet (Reply). Then
- *If* position information available in Packet. Then
  - Update destination position with timestamp to its PIT.
- *End If*
- *If* its mobile host's ID is one of a reply route. Then
  - Make valid route to route cache.
  - Forward reply Packet backward to upper-stream host referred from a reply route.
- *Else*
  - Drop Packet
- *End If*

End if

Algorithm for Source Mobile Host (MABR)

If source S wants to communicates to destination D, Then
- Lookup destination position in PIT.
- *If* destination position is empty and run in normal ABR mode. Then
  - Broadcast route request Packet uses normal flooding mode.
- *Else If* destination position is empty. Then
  - Append its position and timestamp to route request Packet. (BQ)
  - Broadcast BQ to the Network.
- *Else If* destination position isn't empty. Then
  - Append its position, destination position, and timestamps to route request Packet
  - Broadcast BQ to the Network.
- *End If*

End If

Algorithm for Destination Mobile Host (MABR)

If destination D gets the route request Packet. Then
- *If* position information available in Packet. Then
  - Update source and destination positions with timestamps to its PIT.
  - Select a route uses route selection Algorithm

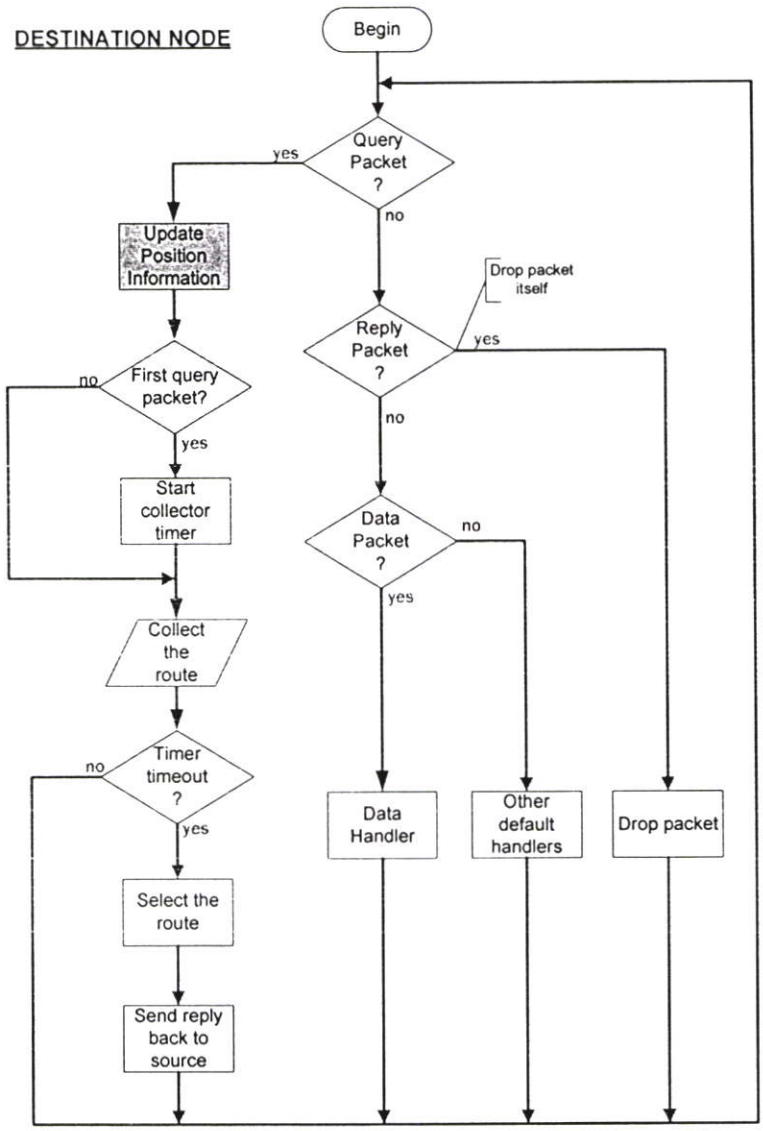
- Include position and timestamp itself to reply Packet
- Send reply Packet back to the source using selected path

Else

- Select a route uses route selection Algorithm
- Send reply Packet back to the source using selected path

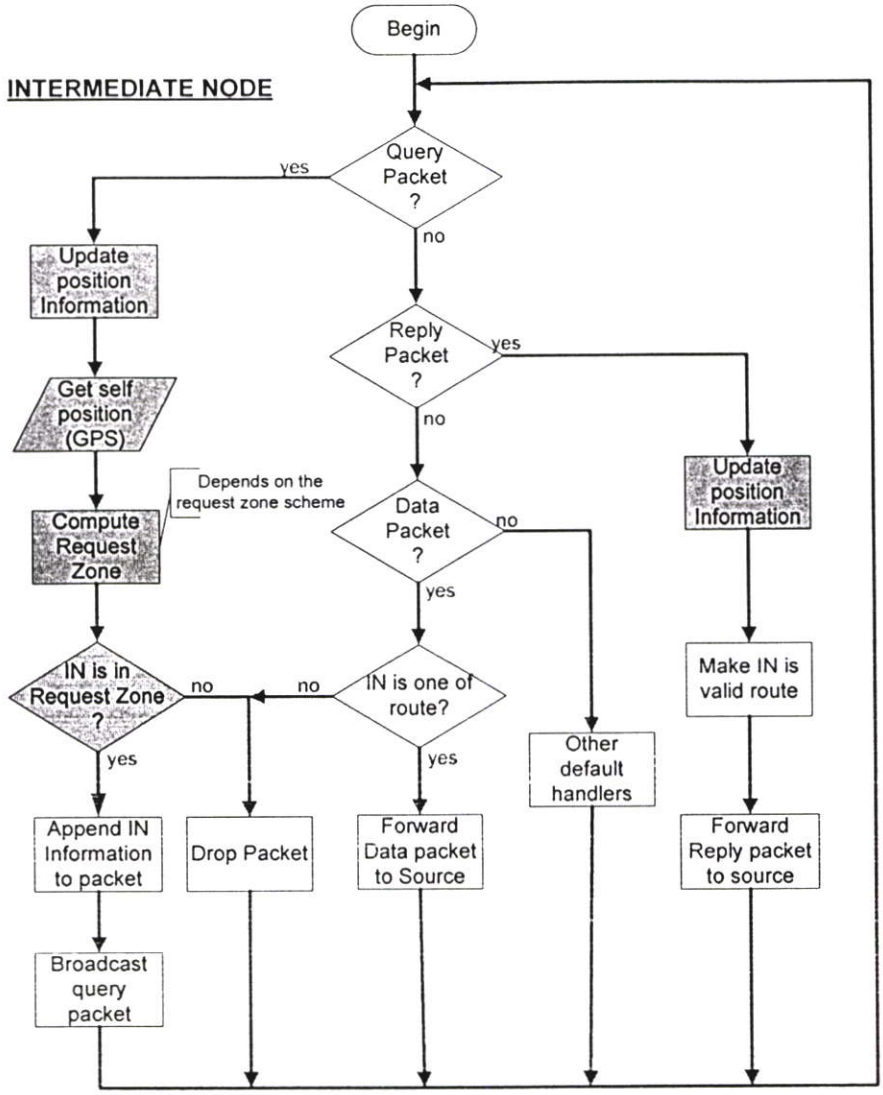
End If

End If



รูปที่ 5.4 โปรโตคอลเอ็มเอบีอาร์ที่เพิ่มเติมจากเอบีอาร์(โมไบล์โฮสต์ปลายทาง)

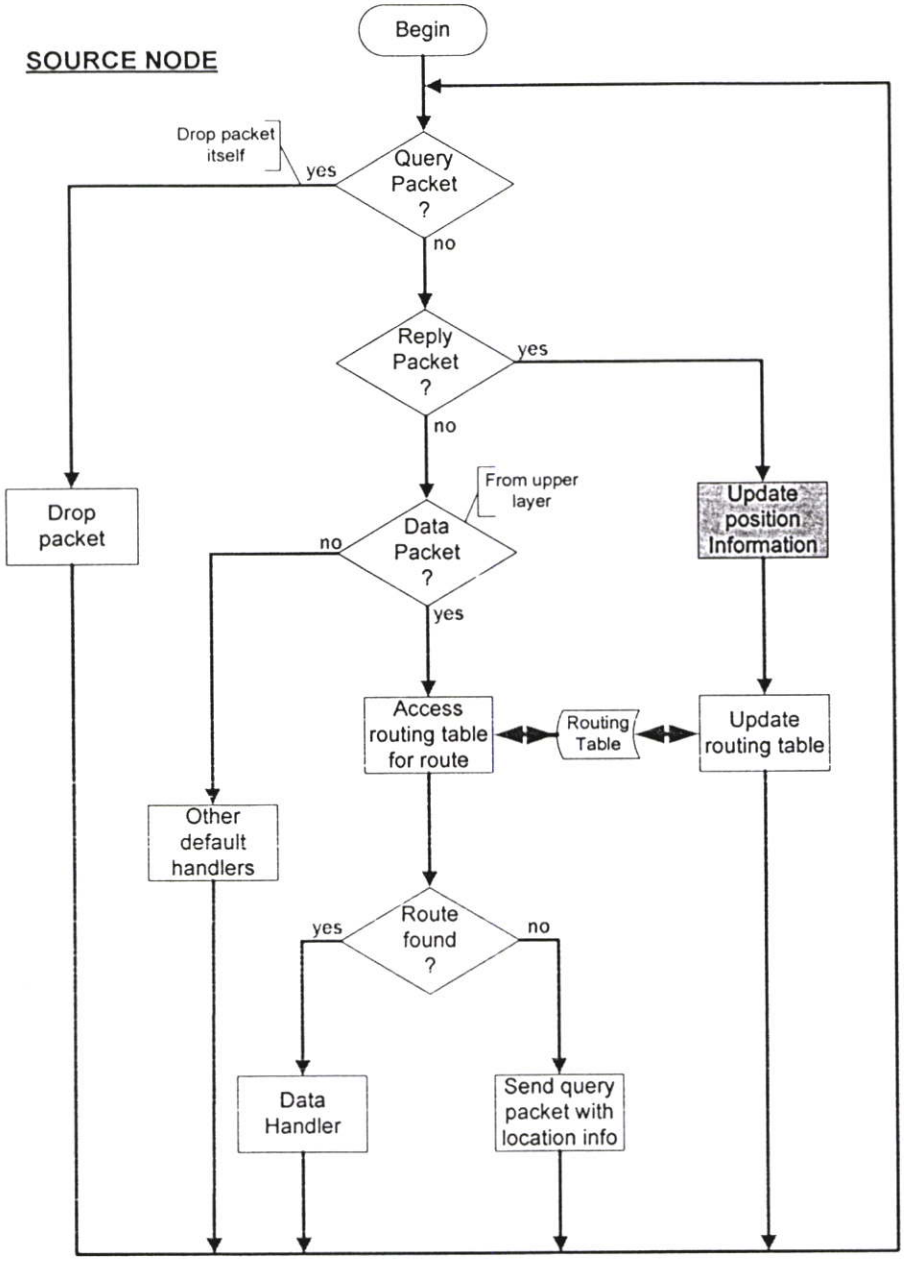
พิจารณาในส่วนของโปรโตคอลแอลเออาร์ซึ่งนำมาใช้ในการปรับปรุงโปรโตคอลเอบีอาร์แบ่งเป็นพื้นที่การส่งต่อเมสเสสร้องขอเส้นทาง (Request Zone) และพื้นที่คาดหวังของโมไบล์โฮสต์ปลายทาง (expected Zone) เป็นพื้นที่ซึ่งถูกคำนวณขึ้นโดยโมไบล์โฮสต์ต้นทาง ซึ่งต้องการหาคาดเดาเพื่อหาตำแหน่งของโมไบล์โฮสต์ปลายทาง สำหรับพื้นที่คาดหวังสามารถคำนวณได้โดยมีสมมติฐานดังนี้



รูปที่ 5.5 โปรโตคอลเอ็มเอบีอาร์ที่เพิ่มเติมจากเอบีอาร์(โมไบล์โฮสต์ระหว่างทาง)

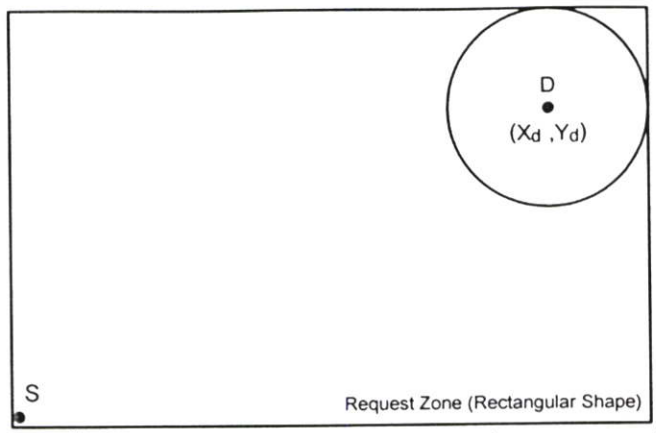
- สมมติค่าความเร็วของโมไบล์โฮสต์ในเครือข่ายเป็นค่าความเร็วคงที่ v โดยค่าดังกล่าวสามารถแทนค่าเฉลี่ยการเคลื่อนของโมไบล์โฮสต์ต่างๆในเครือข่าย หรือสามารถใช้ค่าความเร็วการเคลื่อนที่สูงสุดของโมไบล์โฮสต์ที่กำหนดขึ้นเอง เพื่อให้เหมาะสมกับสภาพของเครือข่าย
- สมมติว่าโมไบล์โฮสต์ทราบตำแหน่งปัจจุบันของตน และเรียกใช้งานได้ตลอดเวลาโดยไม่มีค่าความผิดพลาด

สำหรับโปรโตคอลเอ็มเอบีอาร์ได้เลือกใช้การคำนวณพื้นที่การส่งต่อสองแบบคือเลือกใช้พื้นที่การส่งต่อแบบ(LAR Scheme I) ซึ่งเป็นพื้นที่สี่เหลี่ยมผืนผ้าล้อมรอบพื้นที่คาดหวังของโมไบล์โฮสต์ปลายทางตามรูปที่ 5.7 และ5.8 สำหรับขั้นตอนในการคำนวณหาพื้นที่การส่งต่อเมสเสจร้องขอเส้นทางทั้งสองรูปแบบทั้งในรูปที่ 5.7 และ 5.8 แต่ละโมไบล์โฮสต์จะมีการคำนวณพื้นที่การส่ง

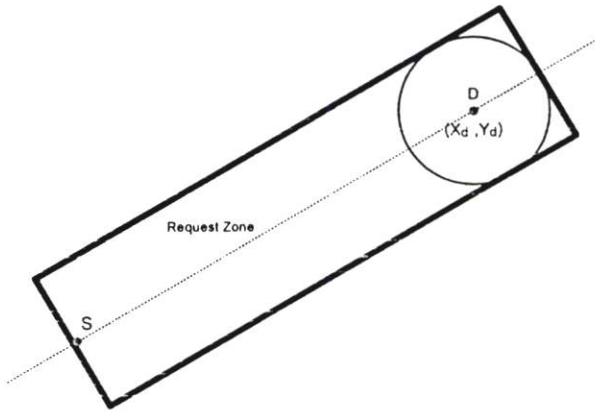


รูปที่ 5.6 โปรโตคอลเอ็มเอบีอาร์ที่เพิ่มเติมจากเอบีอาร์(โมไบล์โฮสต์ต้นทาง)

ต่อ และเปรียบเทียบกับตำแหน่งของโมไบล์โฮสต์ตนว่าอยู่ในพื้นที่ดังกล่าวหรือไม่ โดยงานวิจัยนี้ได้พิจาณากระบวนการค้นหาเส้นทางเป็นสามส่วน ในครั้งแรกจะค้นหาเส้นทางโดยใช้รูปแบบพื้นที่การส่งต่อตามรูปที่ 1 หากไม่สามารถค้นหาเส้นทางได้พบ จะเริ่มการค้นหาครั้งที่สองโดยใช้พื้นที่การส่งต่อเป็นดังในรูปที่ 5.7 และถ้ายังค้นหาเส้นทางไม่พบ จะทำการบรอดคาสต์เมสเสจร้องหาเส้นทางออกไปทั้งเครือข่าย

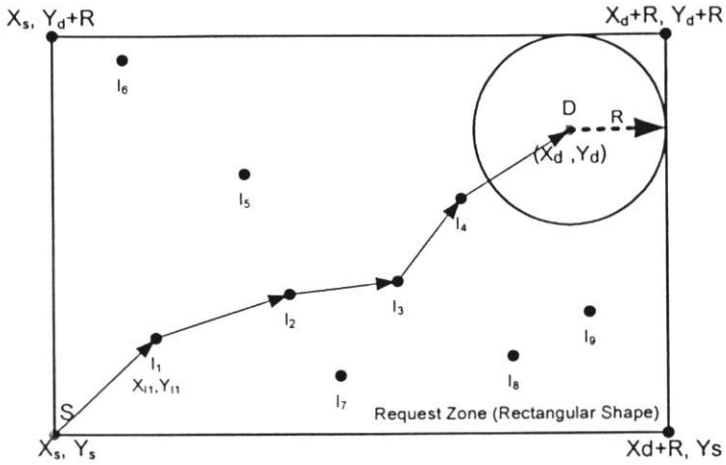


รูปที่ 5.7 พื้นที่การส่งต่อเมสเสจร้องขอเส้นทางที่เลือกใช้แบบที่ 1

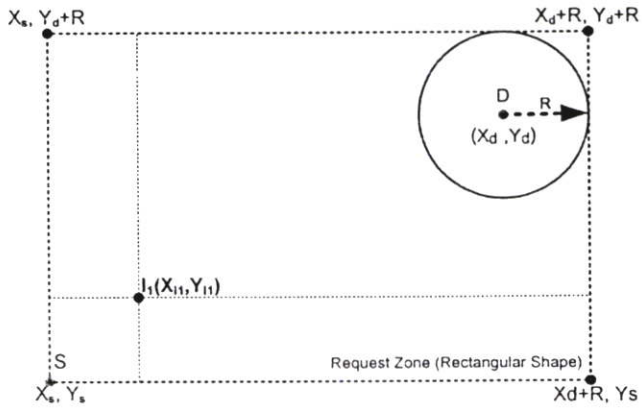


รูปที่ 5.8 พื้นที่การส่งต่อเมสเสจร้องขอเส้นทางที่เลือกใช้แบบที่ 2

การคำนวณหาพื้นที่การส่งต่อในรูปแบบที่ 5.8 ซึ่งแต่ละโมไบล์โฮสต์จะคำนวณจุดสำคัญ 4 ให้ทราบทั้ง 4 ตำแหน่งได้แก่ (X_s,Y_s) (X_s, Y_d+R) (X_d+R,Y_d+R) และ (X_d+R,Y_s) โดย R คือรัศมีการกระจัดของโมไบล์โฮสต์ปลายทาง D จากสมการ $R = v(t_1 - t_0)$ แต่ละโมไบล์โฮสต์เมื่อคำนวณพื้นที่ทั้งสองคือพื้นที่คาดหวังและพื้นที่การส่งต่อได้แล้วจะตรวจสอบพื้นที่การส่งต่อกับตำแหน่งของตนก่อนตัดสินใจว่าจะส่งต่อหรือครอบเมสเสจร้องขอเส้นทาง โมไบล์โฮสต์ระหว่างทางใดๆ แทนด้วย$(I_1, I_2, I_3,..,I_n)$ จะทำการคำนวณพื้นที่การส่งต่อ โดยนำข้อมูลตำแหน่งที่แนบมากับเมสเสสร้องขอเส้นทาง ซึ่งประกอบด้วยข้อมูลตำแหน่งของโมไบล์โฮสต์ต้นทาง, โมไบล์โฮสต์ปลายทางและเวลาที่ได้ทำการบันทึกตำแหน่ง ได้กำหนดเป็นตัวแปรเก็บค่าดังกล่าวซึ่งถอดออกมาคำนวณจากเมสเสสร้องขอเส้นทาง โดยกำหนด (X_s,Y_s,T_s) สำหรับโมไบล์โฮสต์ต้นทาง S และ (X_d,Y_d,T_d) สำหรับโมไบล์โฮสต์ปลายทาง D X และ Y ถูกกำหนดให้เป็นตัวแปรสำหรับเก็บค่าตำแหน่งในแนวแกนนอน, แกนตั้งและ T คือเวลาที่ได้ทำการบันทึกค่าของตำแหน่ง



รูปที่ 5.9 การคำนวณพื้นที่การส่งต่อเมสเสจร้องขอเส้นทางที่เลือกใช้แบบที่ 1

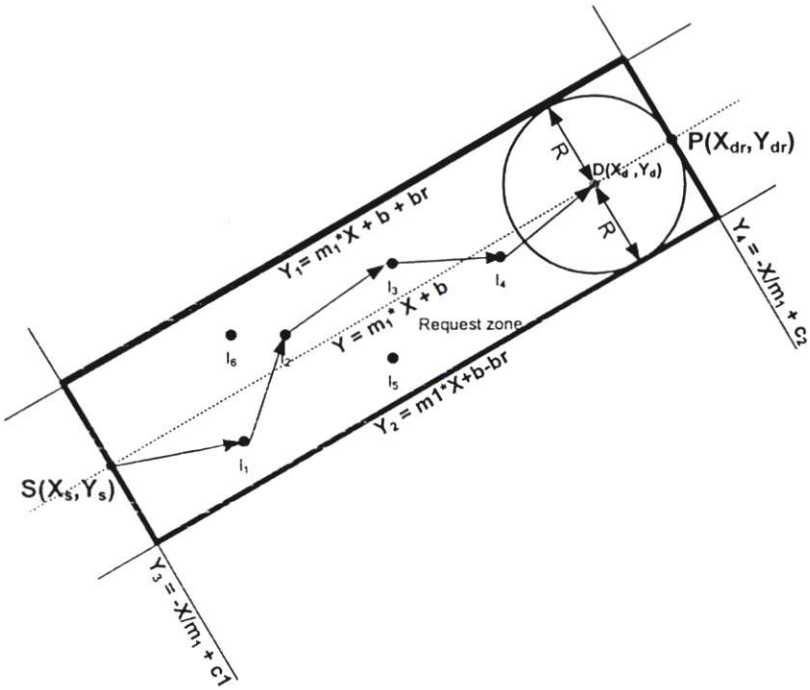


รูปที่ 5.10 การตัดสินใจส่งต่อหรือดรอบแพคเกจในพื้นที่การส่งต่อแบบที่ 1

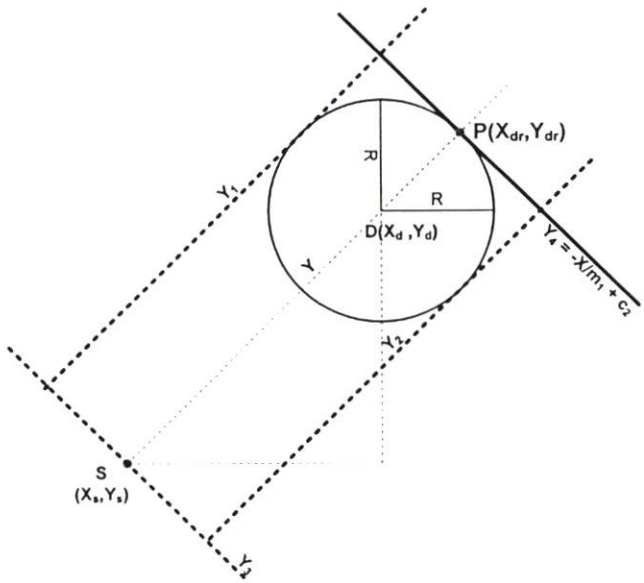
สำหรับขั้นตอนการคำนวณเพื่อตัดสินใจส่งต่อหรือดรอบเมสเสสร้องขอเส้นทางนั้น มีขั้นตอนต่อไปนี้

1. เมื่อโมไบล์โฮสต์ระหว่างทางได้รับเมสเสสร้องขอเส้นทางก็จะถอดเอาข้อมูลตำแหน่งเก็บลงในตัวแปร X_s,Y_s,T_s, X_d,Y_d,T_d
2. คำนวณหาพื้นที่คาดหวัง R โดยหาได้จาก $R = V(T_s - T_d)$
3. คำนวณจุดสำคัญ 4 จุดซึ่งจะเป็นมุมของสี่เหลี่ยมผืนผ้าที่ตีกรอบพื้นที่การส่งต่อประกอบด้วย (Xs,Ys), (Xs, Yd+R), (Xd+R, Yd+R), (Xd+R, Ys)
4. หาค่าตำแหน่งปัจจุบันของโมไบล์โฮสต์ระหว่างทางที่กำลังคำนวณเพื่อหาพื้นที่การส่งต่อลงในตัวแปร Xi,Yi
5. ขั้นตอนสุดท้ายจะนำตำแหน่งของโมไบล์โฮสต์ตนเอง Xi,Yi มาตรวจสอบว่าอยู่ในพื้นที่การส่งต่อเมสเสสการร้องขอเส้นทางหรือไม่ โดยมีโค๊ดเป็นโปรแกรมภาษา C++ ดังนี้

```
bool normalRect(....)
{       ......
        ......
        if( Xi<Xs || Xi>Xd+R || Yi<Ys || Yi>Yd+R  ) return false;
        else return true;
}
```



รูปที่ 5.11 การคำนวณพื้นที่การส่งต่อเมสเสจร้องขอเส้นทางที่เลือกใช้แบบที่ 2



รูปที่ 5.12 การหาค่าจุด $P(X_{dr},Y_{dr})$

หลังจากฟังก์ชั่น normalRect() ได้คำนวณค่าต่างๆแล้วก็จะรีเทิร์นค่าผลของการคำนวณที่ได้ออกมาเป็นจริง (true) หรือเท็จ(false) โดยผลที่ได้จากฟังก์ชั่นนี้จะถูกนำไปตัดสินใจว่าโมไบล์โฮสต์ระหว่างทางจะครอบเมสเสสร้องขอเส้นทางที่ได้รับหรือจะทำการส่งต่อเมสเสสร้องขอเส้นทาง รูปแบบการคำนวณพื้นที่การส่งต่อเมสเสสร้องขอเส้นทางในแบบที่ 2 รูปที่ 5.8 จะมีการคำนวณดังนี้ สำหรับโมไบล์โฮสต์ระหว่างทางใดๆ แทนด้วย($I_1, I_2, I_3,..,I_n$) จะทำการคำนวณพื้นที่การส่งต่อ โดยนำข้อมูลตำแหน่งที่แนบมากับเมสเสสร้องขอเส้นทาง ซึ่งประกอบด้วยข้อมูลตำแหน่งของโมไบล์โฮสต์ต้นทาง, โมไบล์โฮสต์ปลายทางและเวลาที่ได้ทำการบันทึกตำแหน่ง ได้กำหนดเป็นตัวแปรเก็บค่าดังกล่าวซึ่งถอดออกมาคำนวณจากเมสเสสร้องขอเส้นทาง โดยกำหนด (X_s,Y_s,T_s) สำหรับโมไบล์โฮสต์ต้นทาง S และ (X_d,Y_d,T_d) สำหรับโมไบล์โฮสต์ปลายทาง D ตัวแปร X และ Y ถูกกำหนดให้เป็นตัวแปรสำหรับเก็บค่าตำแหน่งในแนวแกนนอน, แกนตั้งและ T คือเวลาที่ได้ทำการบันทึกค่าของตำแหน่ง พื้นที่การส่งต่อเมสเสสร้องขอเส้นทาง จะถูกตีกรอบให้อยู่ในพื้นที่ของสมการเส้นตรงสี่เส้น คือ Y_1, Y_2, Y_3 และ Y_4 โดยเริ่มคำนวณสมการเส้นตรง Y_1, Y_2, Y_3 และ Y_4 ซึ่งล้อมรอบพื้นที่เป็นพื้นที่การส่งต่อเมสเสสร้องขอเส้นทาง จากสมการ Y, Y_1, Y_2, Y_3 และ Y_4 จะมีขั้นตอนการคำนวณดังนี้ ให้สมการเส้นตรง Y เป็นเส้นตรงที่ลากผ่านตำแหน่งของโมไบล์โฮสต์ต้นทาง $S(X_s,Y_s)$ ไปยังโมไบล์โฮสต์ปลายทาง $D(X_d,Y_d)$คำนวณหาค่าความชันสมการเส้นตรง

$$m_1 = \frac{Y_d - Y_s}{X_d - X_s} \tag{5.1}$$

จากสมการเส้นตรง $Y = m_1X + b$
เมื่อแทนจุด $S(X_s,Y_s)$ เพื่อสร้างเส้นตรง Y จะได้ค่าคงที่ $b = Y_s - m_1X_s$ ทำให้สร้างสมการเส้นตรง Y ได้เป็น

$$Y = Y_s + m_1(X - X_s) \tag{5.2}$$

เมื่อ Y เป็นสมการเส้นตรงที่มีความชัน m_1 ซึ่งลากผ่านตำแหน่ง $S(X_s,Y_s)$
ให้ $Dist_{SD}$ เป็นระยะทางระยะทางที่สั้นที่สุดระหว่างโมไบล์โฮสต์ต้นทาง S ไปยังโมไบล์โฮสต์ปลายทาง D จะคำนวณหา $Dist_{SD}$ ได้จาก

$$Dist_{SD} = \sqrt{(X_d - X_s)^2 + (Y_d - Y_s)^2} \tag{5.3}$$

เนื่องจาก Y_1 จะมีระยะห่างตั้งฉากกับเส้นตรง Y เป็นระยะ R หรือเท่ากับรัศมีที่คำนวณได้จากพื้นที่คาดหวัง จะได้ค่าคงที่ b_r ซึ่งทำให้เส้นตรง Y และ Y_1 ขนานกันโดยมีระยะห่างตั้งฉากเป็น R คือ $b_r = R \times \frac{|Y_d - Y_s|}{Dist_{SD}}$

$$Y_1 = Y_s + m_1(X - X_s) + R \times \frac{|Y_d - Y_s|}{Dist_{SD}} \quad (5.4)$$

และ Y_2 เป็นเส้นตรงที่ขนานกับ Y ซึ่งจะมีระยะห่างตั้งฉากกับเส้นตรง Y เป็นระยะ R เช่นเดียวกับเส้นตรง Y_1 ในด้านตรงข้ามจะได้สมการเป็น

$$Y_2 = Y_s + m_1(X - X_s) - R \times \frac{|Y_d - Y_s|}{Dist_{SD}} \quad (5.5)$$

เส้นตรง Y_3 เป็นเส้นตรงที่มีความชันตั้งฉากกับเส้นตรง Y, Y_1 และ Y_2 ลากผ่านตำแหน่งของโมไบล์โฮสต์ $S(X_s,Y_s)$ จากความสัมพันธ์ของเส้นที่ตั้งฉากกับเส้นตรง Y จะได้ความชันของเส้นตั้งฉากเป็น $m_2 = -1/m_1$ หรือจะได้สมการเป็น

$$Y_3 = \frac{-X}{m_1} + c_1 \quad (5.4)$$

เมื่อแทนสมการด้วยตำแหน่งของโมไบล์โฮสต์ต้นทาง $S(X_s,Y_s)$ จะได้ค่า $c_1 = Y_s + \frac{X_s}{m_1}$

ดังนั้นจะได้สมการเส้นตรง Y_3

$$Y_3 = Y_s + \frac{X_s - X}{m_1} \quad (5.5)$$

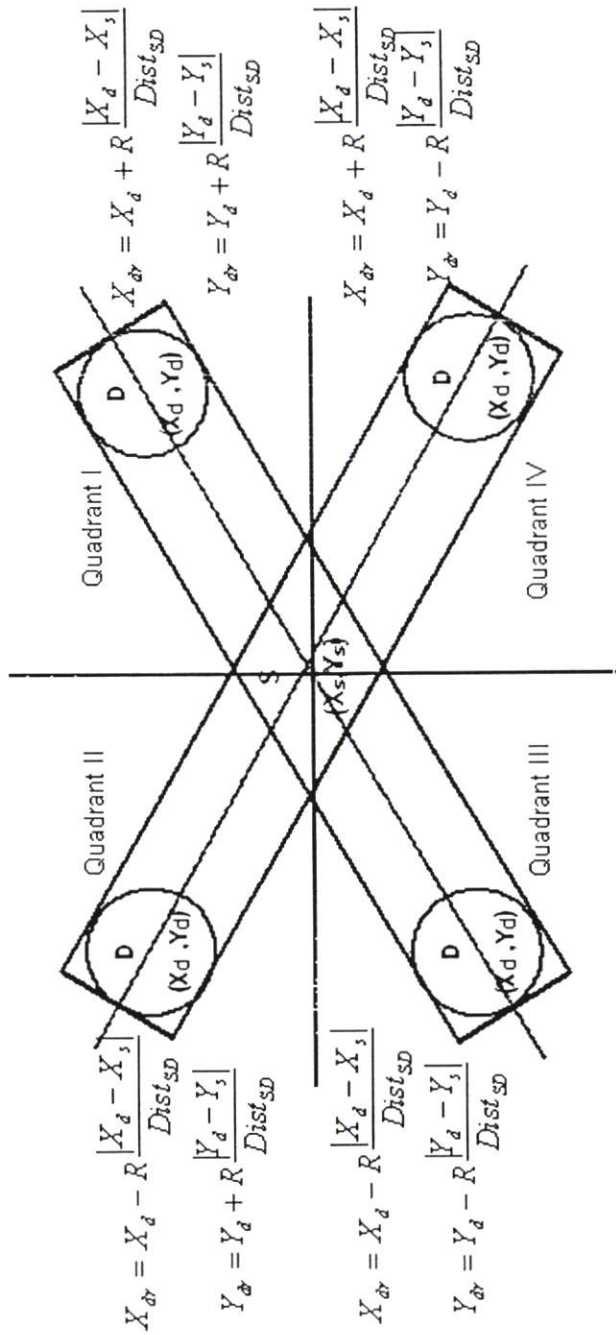
และสมการ Y_4 ซึ่งมีความชันเท่ากับสมการ Y_3 ดังนั้นจึงสร้างสมการเส้นตรงที่จุด $P(X_{dr},Y_{dr})$ โดยการคำนวณหาจุด X_{dr},Y_{dr} จะมีการคำนวณได้ 4 แบบ หรือ 4 Quadrants ตามแต่ลักษณะตำแหน่งของโมไบล์โฮสต์ โดยผลที่ได้จะแสดงไว้ดังรูปที่ 5.13

สุดท้ายเราสามารถสร้างสมการเส้นตรง Y_4 ได้เป็น

$$Y_4 = Y_{dr} + \frac{X_{dr} - X}{m_1} \quad (5.6)$$

สำหรับขั้นตอนการคำนวณเพื่อตัดสินใจการส่งต่อหรือครอบเมสเสสร้องขอเส้นทางนั้น มีขั้นตอนต่อไปนี้

1. เมื่อโมไบล์โฮสต์ระหว่างทางได้รับเมสเสสร้องขอเส้นทางก็จะถอดเอาข้อมูลตำแหน่งเก็บลงในตัวแปร X_s,Y_s,T_s, X_d,Y_d,T_d
2. คำนวณหาพื้นที่คาดหวัง R โดยหาได้จาก $R = V \times (T_s - T_d)$
3. หาค่าตำแหน่งปัจจุบันของโมไบล์โฮสต์ระหว่างทางที่กำลังคำนวณเพื่อหาพื้นที่การส่งต่อลงในตัวแปร X_i,Y_i

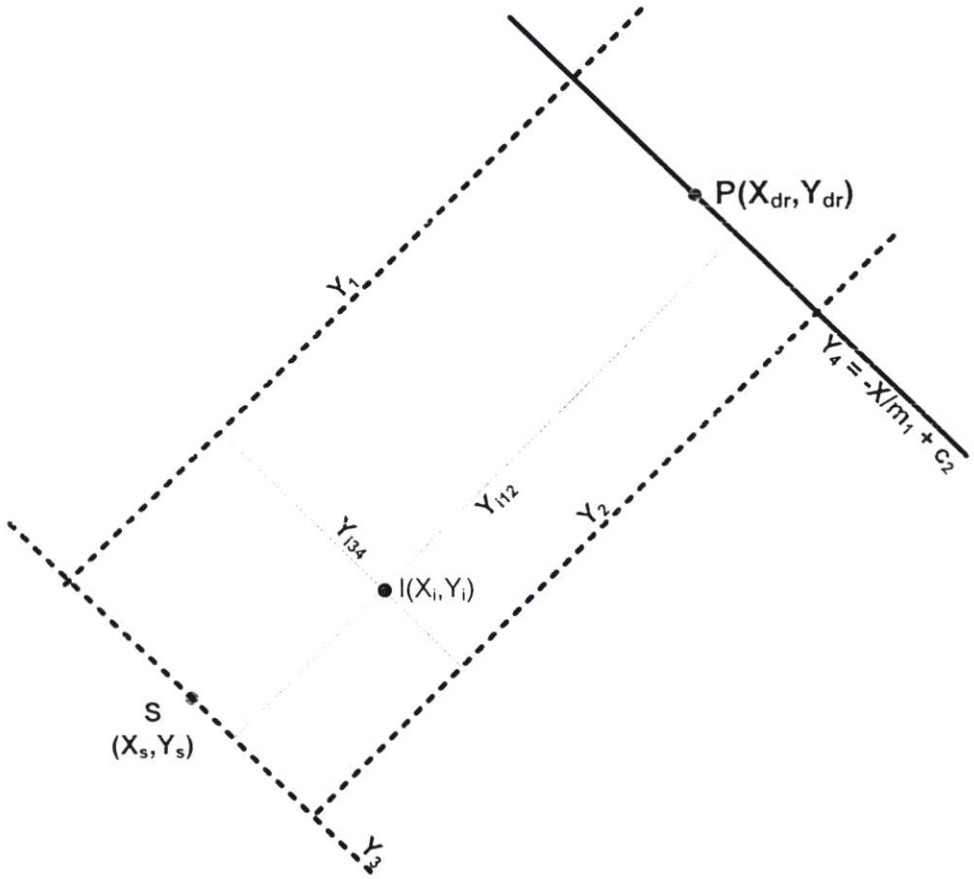


รูปที่ 5.13 ค่า Xdr และ Ydr ที่ใช้ในการคำนวณพื้นที่การส่งต่อในแต่ละควอแดรนต์

4. สร้างสมการเส้นตรงสองเส้นคือ Y_{i12} ซึ่งเป็นเส้นตรงขนานกับ Y1 และ Y2 และสร้างเส้นตรง Y_{i34} ซึ่งเป็นเส้นตรงขนานกับ Y_3 และ Y_4 โดยจะได้

$$Y_{i12} = Y_i + m_1(X - X_i) \tag{5.7}$$

$$Y_{i34} = Y_i + \frac{X_i - X}{m_1} \tag{5.8}$$



รูปที่ 5.14 การตัดสินใจส่งต่อหรือดรอบแพคเกจในพื้นที่การส่งต่อแบบที่ 2

5. จากสมการ Y1, Y2, Y3 และ Y4 จะเป็นสมการเส้นตรง 4เส้นซึ่งตีกรอบพื้นที่ที่สามารถส่งต่อเมสเสสร้องขอเส้นทาง และจากสมการ Y_{i12} และ Y_{i34} นำสมการทั้งหมด แทนค่าด้วย X_s, Y_s, X_d, Y_d, X_i, Y_i และกำหนดให้ $X = 0$
6. ผลที่ได้จะเหลือเป็นค่าคงที่ $Y_{1|x=0}$, $Y_{2|x=0}$, $Y_{3|x=0}$, $Y_{4|x=0}$, $Y_{i12|x=0}$ และ $Y_{i34|x=0}$ ตามลำดับ
7. ถ้า $Y_{i12|x=0} \le Y_{1|x=0}$ และ $Y_{i12|x=0} \ge Y_{2|x=0}$ และ $Y_{i34|x=0} \le Y_{4|x=0}$ และ $Y_{i34|x=0} \ge Y_{3|x=0}$ แสดงว่าโมไบล์โฮสต์ระหว่างทางจะอยู่ในพื้นที่การส่งต่อเมสเสสร้องขอเส้นทาง ซึ่งโค๊ดภาษา C++ จะเป็นดังนี้

```
bool diagonalRect(....)
{       ......
        ......
        if( yi12<= y1 && yi12>=y2 && yi34<=y4 && yi34>=y3)
                return true;
        return false;
}
```

หลังจากฟังก์ชั่น diagonalRect() ได้คำนวณค่าต่างๆแล้วก็จะรีเทิร์นค่าผลของการคำนวณที่ได้ออกมาเป็นจริง (true) หรือเท็จ(false) โดยผลที่ได้จากฟังก์ชั่นนี้จะถูกนำไปตัดสินใจว่าโมไบล์โฮสต์ระหว่างทางจะครอบเมสเสสร้องขอเส้นทางที่ได้รับหรือจะทำการส่งต่อเมสเสสร้องขอเส้นทาง ซึ่งจะนำฟังก์ชันทั้งสองนี้ normalRect() และ diagonalRect() ไปใส่เพิ่มเติมในส่วนอัลกอริทึมการตัดสินใจส่งต่อเมสเสจร้องข้อเส้นทาง

สำหรับส่วนที่ถูกแทรกในบรรทัดจะเป็นตัวอักษรเอียงเพื่อให้เห็นส่วนเพิ่มเติมหรือแก้ไขในโปรโตคอลเดิมชัดเจนขึ้น

5.1.2.1 รูปแบบแพคเกจร้องขอเส้นทาง (MABR Route Discovery Packet)

รูปแบบแพคเกจร้องขอเส้นทางบีคิว (Broadcast Query Packet: BQ) จะอิงกับรูปแบบเดิมที่ได้อิมพลีเมนต์ลงในโปรโตคอลเอบีอาร์ มีโครงสร้างดังรูปที่ 5.15 โดยแต่ละฟิลด์จะมีความหมายตามตารางที่ 5.4

Type	SRC ID	DEST ID	LIVE	IM IDs	METRICS	SEQ NO.	*SRC POS INFO*	*DEST POS INFO*	CRC

รูปที่ 5.15 ข้อมูลฟิลด์ต่างๆในแพคเกจร้องขอเส้นทางบีคิว(เอ็มเอบีอาร์)

ตารางที่ 5.4 แสดงความหมายของแต่ละฟิลด์ในแพคเกจร้องขอเส้นทาง

TYPE	ชนิดของแพคเกจ (ระบุเป็น BQ Packet)
SRC ID	ID ของโมไบล์โฮสต์ต้นทาง
DEST ID	ID ของโมไบล์โฮสต์ปลายทาง
LIVE	ค่าจำนวนฮอพก่อนที่จะถูกครอบจากเครือข่าย
IM IDs	หมายเลข ID ของโมไบล์โฮสต์ต่างๆที่เป็นเส้นทางเชื่อมต่อ
METRICS	ค่าพารามิเตอร์เพื่อใช้วัดคุณภาพของเส้นทาง เช่นค่าความสัมพันธ์ระหว่างเส้นทาง
SEQ NO.	หมายเลขซีเควนซ์ของแพคเกจ
SRC POS INFO	*ตำแหน่งของโมไบล์โฮสต์ต้นทาง X, Y, Z และเวลาเก็บข้อมูล* t_s
DEST POS INFO	*ตำแหน่งของโมไบล์โฮสต์ต้นทาง X, Y, Z และเวลาเก็บข้อมูล* t_d

CRC	โค๊ดเพื่อตรวจสอบความผิดพลาดในการส่งข้อมูล

จากข้อมูลในตารางที่ 5.4 นำมาสร้างเป็นแพคเกจข้อมูลโดยไม่ได้สร้าง CRC ลงในแพคเกจเนื่องจากเพื่อความง่ายในการออกแบบ สำหรับส่วนที่เพิ่มเติมเข้าไปเป็นข้อมูลตำแหน่งของโมไบล์โฮสต์ต้นทางและปลายทางรวมถึงเวลาการบันทึกข้อมูล (Timestamp) เพื่อง่ายต่อการจำลองการทำงาน การคำนวณจึงใช้แต่ข้อมูลในแกน X และ Y ในแนวระนาบเท่านั้น

Broadcast Query Packet

```
typedef struct hdr_mabr_bc {
        u_int8_t   bc_type;                                  // Type
        nsaddr_t   bc_src_addr;                              // SRC ID
        nsaddr_t   bc_dst_addr;                              // DEST ID
        u_int16_t  bc_live;                                  // LIVE
        nsaddr_t   bc_hostAddresses[MABR_MAX_SIZE];          // IN IDs
        u_int16_t  bc_associativity[MABR_MAX_SIZE];          // METRICS
        u_int16_t  bc_noOfNeighbors;
        nsaddr_t   bc_neighborAddresses[MABR_MAX_SIZE];
        u_int16_t  bc_neighborTick[MABR_MAX_SIZE];
        u_int32_t  bc_seqno;                                 // SEQ NO.
        // source position details
        float Xs;                   // x axis
        float Ys;           // y axis
        //float Zs;                 // z axis
        double ts;                  // timestamp
        // destination position details
         float Xd;
         float Yd;
         //float Zd;
         double td;                  // timestamp
        ......
        ......
  }
} hdr_mabr_bc;
```

5.1.2.2 รูปแบบแพคเกตตอบกลับเส้นทาง (MABR Route Reply Packet)

รูปแบบแพคเกจตอบกลับเส้นทาง (Reply Packet) อิงกับงานวิจัยที่เสนอจากโปรโตคอลเอบีอาร์ ซึ่งมีโครงสร้างดังรูปที่ 5.16 โดยแต่ละฟิลด์จะมีความหมายตามตารางที่ 5.5

TYPE	SRC ID	DEST ID	IM IDs	SEQ ID	ROUTE QUALITIES	*DEST POS INFO*	CRC

รูปที่ 5.16 ข้อมูลฟิลด์ต่างๆในแพคเกจตอบกลับเส้นทาง

ตารางที่ 5.5 แสดงความหมายของแต่ละฟิลด์ในแพคเกจตอบกลับเส้นทาง

TYPE	ชนิดของแพคเกจ (ระบุเป็น Reply Packet)
SRC ID	ID ของโมไบล์โฮสต์ต้นทาง (ต้นทางของแพคเกจ Reply)
DEST ID	ID ของโมไบล์โฮสต์ปลายทาง (ปลายทางของแพคเกจ Reply)
IM IDs	หมายเลข ID ของโมไบล์โฮสต์ต่างๆที่เป็นเส้นทางเชื่อมต่อ
SEQ NO.	หมายเลขซีเควนซ์ของแพคเกจ
Route Qualities	ตอบกลับคุณภาพของเส้นทางที่เลือก (สำหรับโมไบล์โฮสต์ต้องการทราบคุณภาพของเส้นทาง)
DEST POS INFO	*ตำแหน่งของโมไบล์โฮสต์ต้นทาง X, Y, Z และเวลาเก็บข้อมูล* t_d
CRC	โค๊ดเพื่อตรวจสอบความผิดพลาดในการส่งข้อมูล

จากข้อมูลในตารางที่ 5.5 นำมาสร้างเป็นแพคเกจข้อมูลโดยจะข้าม Route Qualities และ CRC ออกไปเพื่อง่ายในการโปรแกรม จะได้โค๊ดในภาษา C++ ได้ดังนี้

Reply Packet

```
typedef struct hdr_mabr_reply {
        u_int8_t   re_type;                          // TYPE
        nsaddr_t   re_src_addr;                      // SRC ID
        nsaddr_t   re_dst_addr;                      // DEST ID
        nsaddr_t   re_inx[MABR_MAX_SIZE];            // IM IDs
        // destination position details
         float Xd;
         float Yd;
         //float Zd;
         double td;              // timestamp
```

```
    ......
    ......
  }
} hdr_mabr_reply;
```

5.1.2.3 รูปแบบแพคเกจบีคอน (MABR Beacon Packet)

สำหรับรูปแบบแพคเกจบีคอน (Beacon Packet) ไม่มีส่วนของการแก้ไขปรับปรุงใดๆ

5.1.2.4 อัลกอริทึมค้นหาเส้นทาง (MABR Route Discovery Algorithm)

เมธอด send_BC() ในโปรโตคอลเอ็มเอบีอาร์มีการปรับปรุงโดยแนบข้อมูลตำแหน่งลงในแพคเกจร้องขอเส้นทางโดยดึงข้อมูลตำแหน่งโมไบล์โฮสต์ปลายทางจาก PositionInformationTable ที่ใช้เก็บข้อมูลตำแหน่งของโมไบล์โฮสต์ต่างๆ และดึงข้อมูลของโมไบล์โฮสต์ตนเอง (โมไบล์โฮสต์ต้นทาง) จากฟังก์ชัน *SelfNode->getLoc(&Xs, &Ys, &Zs)* โดยที่ SelfNode เป็นพอยน์เตอร์หมายถึงโมไบล์โฮสต์ปัจจุบัน

Sending BQ (Source) Method

```
void MABR::send_BC(nsaddr_t dest, u_int16_t ttl) {
    Packet *Query    = allocpkt();            // Allocate Query Packet
    hdr_mabr_bc *bc;                          // Packet type
    bc->bc_type       = MABR_PT_BC;           // Type = BQ
    bc->bc_src_addr   = index;                // SRC = self ID (sending Node)
    bc->bc_dst_addr   = dest;                 // DEST
    bc->bc_seqno      = seqno++;              // SEQ NO.
    bc->bc_live       = 0;                    // LIVE
    bc->bc_noOfNeighbors = 0;                 // ..METRICS
    MobileNode* SelfNode = (MobileNode*)MobileNode::get_node_by_address(index);
    double Xs = 0.0, Ys = 0.0, Zs = 0.0;
    double Xd = 0.0, Yd = 0.0, Zd =0.0, td=0.0;
    SelfNode->getLoc(&Xs, &Ys, &Zs);          // get current mobile host position
    int  lookuppos = LookupPos(PositionInformationTable, dest);
    if( lookuppos != NotFound)
    {       Xd = PositionInformationTable[lookuppos].X;
            Yd = PositionInformationTable[lookuppos].Y;
           // Zd = PositionInformationTable[lookuppos].Z;
```

```
                td = PositionInformationTable[lookuppos].timestamp;

        }
        else
        { // if no information record found then using ABR
        send_BC_original(dest, ttl);        // call ABR send_BC() method
                return;
        }

  }
        .....
        // Similar to original ABR code
        .....
 }
/****************** Route Discovery Timer **********************/

RouteDiscoveryTimer::RouteDiscoveryTimer(MABR* mabr, nsaddr_t dest) {
        // Similar to original ABR code
        .....
}
/****************** Time-out Handler ****************/
void RouteDiscoveryTimer::expire(Event*) {
        // Similar to original ABR code
        .....
 }
```

เมธอด handle_QUERY() ในโปรโตคอลเอ็มเอบีอาร์จะดึงข้อมูลตำแหน่งที่ได้รับจากแพคเกจร้องขอเส้นทาง BQ ที่ส่งจากเมธอด send_BC() ซึ่งจะได้ข้อมูลตำแหน่งของโมไบล์โฮสต์ต้นทางและปลายทาง หลังจากนั้นโมไบล์โฮสต์จะดึงข้อมูลตำแหน่งของตนเองมาเปรียบเทียบโดยส่งข้อมูลตำแหน่งทั้งหมดไปคำนวณในฟังก์ชัน *lar->forward();* ซึ่งฟังก์ชันนี้จะไปเรียกฟังก์ชันรูปแบบพื้นที่การส่งต่อโดยเลือกระหว่างพื้นที่รูปแบบที่ 1 และพื้นที่รูปแบบที่ 2 (ระหว่าง normalRect และ diagonalRect) ซึ่งการเลือกแล้วแต่การจำลองการทำงานในแต่ละครั้ง หรือขึ้นกับครั้งในการบรอดคาสต์เพื่อค้นหาเส้นทาง จากเมธอดนี้หากว่าโมไบล์โฮสต์นี้ไม่ใช่โมไบล์โฮสต์ปลายทางของแพคเกจ และฟังก์ชัน lar->forward() รีเทิร์นค่ากลับมาเป็น false แพคเกจร้องขอ

เส้นทางจะถูกครอบตามเงื่อนไขที่โมไบล์โฮสต์ระหว่างทางได้รับเมสเสจร้องขอเส้นทางแต่ตนเองอยู่นอกพื้นที่การส่งต่อเมสเสจร้องขอเส้นทาง

Handle Query (BQ) Method for Intermediate and Destination

```
void MABR::handle_QUERY(Packet *p) {
 double Xs, Ys, Zs, ts;
 double Xd, Yd, Zd, td;
        nsaddr_t sender  = ip->saddr();          // get sender address
        hdr_mabr_bc *bc = p;                     // pointer bc points to received BQ Packet
        nsaddr_t source  = bc->bc_src_addr;      // set source address from Packet to variable
        nsaddr_t dest   = bc->bc_dst_addr;       // set deste address from Packet to variable
        u_int32_t seqno = bc->bc_seqno;
        u_int16_t *hops = &(bc->bc_live);        // LIVE
        u_int16_t * noOfNeighbors  = &(bc->bc_noOfNeighbors);          // METRICS..
        nsaddr_t * hostAddresses = &(bc->bc_hostAddresses[0]);
        u_int16_t *hostTicks = &(bc->bc_associativity[0]);
        nsaddr_t *neighborAddresses = &(bc->bc_neighborAddresses[0]);
        u_int16_t * neighborTicks    = &(bc->bc_neighborTick[0]);
        // modified include position info
        Xs = (double)bc_m->Xs;
        Ys = (double)bc_m->Ys;
//      Zs = 0.0;
        ts = (double)bc_m->ts;
        Xd = (double)bc_m->Xd;
        Yd = (double)bc_m->Yd;
//      Zd = 0.0;
        td = (double)bc_m->td;
        //get current mobile host position
        MobileNode* SelfNode = (MobileNode*)MobileNode::get_node_by_address(index);
        double Xi = 0.0, Yi = 0.0, Zi = 0.0;
        SelfNode->getLoc(&Xi, &Yi, &Zi);
        // create lar scheme
        LAR *lar = new LAR(speed);
```

```
//set up type of request zone normalRect() or diagonalRect()
        lar->SetScheme(scheme);
        int lookupdstinfo = LookupPos(PositionInformationTable,dest);
// forwarding decision by LAR
        bool forwardD = lar->forward(Xs,Ys,ts, bc_m->Xd,bc_m->Yd,bc_m->td, Xi, Yi);

// modified MABR::if newer destination location stored in intermediatenode then replace newer
//locationinfo to the Packet being forward.....
        //update newer position table
        InsertPos(PositionInformationTable, source, Xs, Ys, 0.0, ts);
        InsertPos(PositionInformationTable, dest, bc_m->Xd, bc_m->Yd, 0.0, bc_m->td);

        // if intermediate mobile host is outside of request zone, Drop it!.
        if( forwardD == false && index!=dest) {
                Packet::free(p);
                return;
        }
  }
  // Check that we haven't handled this request before. If we are the
  // destination host, we are interested in processing the Query no matter
  // what in order to find the "best" route alternative
  // Find our associativity tick in list
.....
// Similar to original ABR code
.....
}
```

เมื่อแพคเกจ BQ ถูกส่งมาถึงโมไบล์โฮสต์ปลายทางแล้วซึ่งเมธอด handle_QUERY() จะถูกเรียกขึ้นมาทำงาน โมไบล์โฮสต์ปลายทางจะเลือกเส้นทางและตอบกลับเส้นทางด้วยเมธอด ABR_getBestRoute() และ send_REPLY() ตามลำดับ

Sending Reply from Destination

```
void MABR::send_REPLY(nsaddr_t source, buffered_route *br) {
  // Select the best route
```

```
buffered_route *selected = br;
if (br == NULL) {
  selected = ABR_getBestRoute(bufferedRoutes[source]);
}
// Allocate REPLY Packet
Packet *reply = allocpkt();
hdr_mabr_reply *re;
re->re_type      = MABR_PT_REPLY_M;
re->re_src_addr  = index;
re->re_dst_addr  = source;
re->re_hopCnt    = selected->hops;
MobileNode* SelfNode = (MobileNode*)MobileNode::get_node_by_address(index);
double Xd = 0.0, Yd = 0.0, Zd =0.0, td=0.0;
// get current mobile host's position and attach position information  to reply Packet
SelfNode->getLoc(&Xd, &Yd, &Zd);
re->Xd          = (float)Xd;
re->Yd    = (float)Yd;
//re->Zd              = (float)Zd;
re->td    = MABR_CURRENT_TIME;
re->re_type      = MABR_PT_REPLY;
re->re_src_addr  = index;
.....
// Similar to original ABR code
.....
}
```

เมื่อโมไบล์โฮสต์ระหว่างทางหรือโมไบล์โฮสต์ต้นทางได้รับเมสเสจรีพลายจะดึงข้อมูลตำแหน่งของโมไบล์โฮสต์ปลายทางมาปรับปรุงข้อมูลตำแหน่งในตารางข้อมูลตำแหน่งของตน ก่อนบันทึกลงในตารางเส้นทางกรณีที่เป็นโมไบล์โฮสต์ต้นทาง หรือส่งต่อในกรณีที่เป็นโมไบลโฮสต์ระหว่างทาง

Handle Reply Packet for Intermediate and Source

```
void MABR::handle_REPLY(Packet *p) {
  hdr_mabr_reply * re;
```

```
nsaddr_t reply_dest  = re->re_dst_addr;
nsaddr_t reply_src   = re->re_src_addr;
//added
double Xd            = (double)re->Xd;
double Yd            = (double)re->Yd;
double Zd          = 0.0;
double td          = re->td;
        //update position information table(PIT) from recieved reply Packet
        ::InsertPos(PositionInformationTable, reply_src, Xd, Yd, Zd, td);
//end
// Find our position in the route (Intermediate check whether itself is one of the route in reply?)
// If I'm intermediate and get reply Packet
.....
// Similar to original ABR code
.....
```

5.1.2.5 อัลกอริทึมการทำบีคอน(MABR Beaconing Algorithm)

อัลกอริทึมนี้โปรโตคอลเอ็มเอบีอาร์ไม่มีการแก้ไขจากโปรโตคอลเดิม

5.2 ผลที่ได้จากการจำลองการทำงาน

ผลการจำลองการทำงานของโปรโตคอลเอบีอาร์จะเก็บผลที่ได้จากการจำลองการทำงานไว้ในไฟล์เทรซ (Trace File) ซึ่งประกอบด้วยผลการจำลองการทำงานแบบอนิเมชันสำหรับใช้กับการแสดงผลการจำลองการทำงานซึ่งเห็นลักษณะโครงข่าย การเคลื่อนที่ของโฮสต์ ข้อมูลและช่วงเวลาที่มีการสื่อสารข้อมูล และผลการจำลองการทำงานแบบละเอียดจะบันทึกอีเวนท์ต่างๆทุกอย่างที่เกิดขึ้นภายในเลเยอร์ MAC, Routing (Network Layer) เป็นต้น รวมถึงบอกตำแหน่งของโมไบล์โฮสต์ ซึ่งข้อมูลแบบละเอียดเหล่านี้จะนำมาวิเคราะห์ผลจากการจำลองการทำงาน รูปแบบของเทรซไฟล์ในเอบีอาร์และเอ็มเอบีอาร์ได้กำหนดเอาไว้มีรูปแบบดังตารางที่ 5.6 และตารางที่ 5.7

ตัวอย่างรูปแบบของข้อมูลเทรซของโปรโตคอลเอบีอาร์

```
s 0.000001000 _4_ RTR --- 0 ABR 25 [0 0 0 0] ------- [4:25 -1:255 32 0] (ABR_TICK)
d 0.000001000 _4_ RTR --- 0 ABR 25 [0 0 0 0] ------- [4:25 -1:255 32 0] (ABR_BC)
r 0.000001000 _4_ RTR --- 0 ABR 25 [0 0 0 0] ------- [4:25 -1:255 32 0] (ABR_REPLY)
```

ตารางที่ 5.6 แสดงข้อมูลเทรซของเครือข่ายไร้สาย

Event	Abbreviation	Type	Value
Wireless Event	s: Send r: Receive d: Drop f: Forward	%.9f %d (%5.2f %5.2f) %3s %4s %d %s %d [%x %x %x %x]	
		%.9f _%d_ %3s %4s %d %s %d [%x %x %x %x]	
		double	Time
		int	Node ID
		double	X Coordinate (If Logging Position)
		double	Y Coordinate (If Logging Position)
		string	Trace Name
		string	Reason
		int	Event Identifier
		string	Packet Type
		int	Packet Size
		hexadecimal	Time To Send Data
		hexadecimal	Destination MAC Address
		hexadecimal	Source MAC Address
		hexadecimal	Type (ARP, IP)

ตารางที่ 5.7 แสดงข้อมูลเทรซของโปรโตคอลเอบีอาร์และเอ็มเอบีอาร์

ABR,MABR	------- [%d:%d %d:%d %d %d] %s	
	int	Source IP Address
	int	Source Port Number
	int	Destination IP Address
	int	Destination Port Number
	int	TTL Value
	int	Next Hop Address, If Any
	string	(ABR_TICK),(ABR_BC),(ABR_REPLY), (MABR_TICK),(MABR_BC),(MABR_REPLY)

ตัวอย่างรูปแบบของเอ็มเอบีอาร์เทรซ (MABR trace format)

s 0.000001000 _4_ RTR --- 0 MABR 25 [0 0 0 0] ------- [4:25 -1:255 32 0] (MABR_TICK)

r 0.000001000 _4_ RTR --- 0 MABR 25 [0 0 0 0] ------- [4:25 -1:255 32 0] (MABR_BC)

d 0.000001000 _4_ RTR --- 0 MABR 25 [0 0 0 0] ------- [4:25 -1:255 32 0] (MABR_REPLY)

บทที่ 6

การจำลองการทำงานของระบบ

ในบทนี้จะกล่าวถึงการนำผลที่ได้จากการจำลองการทำงานของโปรโตคอลเอบีอาร์และเอ็มเอบีอาร์ซึ่งเก็บในรูปของเทรซไฟล์มาทำการรวบรวมเพื่อประมวลผลของประสิทธิภาพเครือข่ายเพื่อเปรียบเทียบการทำงานในการพัฒนาโปรโตคอลค้นหาเส้นทางทั้งสองโปรโตคอล

6.1 การประมวลผลประสิทธิภาพของเครือข่ายจากการจำลองการทำงาน

ในการวัดผลประสิทธิภาพของการทำงานของโปรโตคอลเอ็มเอบีอาร์เทียบกับโปรโตคอลต้นแบบเอบีอาร์ มีพารามิเตอร์ที่สนใจ 5 พารามิเตอร์ ซึ่งประกอบด้วย คอนโทรลโอเวอร์เฮด ดีเลย์ในการรับส่งแพคเกจข้อมูล อัตราการรับส่งข้อมูล ภาระโหลดที่เกิดขึ้นต่อโมไบล์โฮสต์ในเครือข่าย และค่าประสิทธิภาพในการรับส่งแพคเกจข้อมูล โดยพิจารณาดังนี้

6.1.1 โอเวอร์เฮดที่ใช้ในการค้นหาเส้นทางทั้งหมด (Control Overhead: CO)

โอเวอร์เฮดที่ใช้ในการค้นหาเส้นทาง คือปริมาณข้อมูลในเลเยอร์ Routing (RTR) ซึ่งมีหน้าที่ในการค้นหาเส้นทางให้กับเลเยอร์ที่สูงกว่าขึ้นไป ซึ่งเราจะนำเฉพาะข้อมูลในเทรซ RTR มาคำนวณโดยรวมปริมาณขนาดแพคเกจทั้งหมดที่เป็นแพคเกจ ABR_TICK ABR_BC ABR_REPLY สำหรับโปรโตคอลเอบีอาร์ และรวมแพคเกจทั้งหมดที่เป็นแพคเกจคอนโทรล MABR_TICK MABR_BC MABR_REPLY แล้วหารด้วยเวลาในการจำลองการทำงานจะได้ผลลัพธ์ออกมาเป็นจำนวนปริมาณโอเวอร์เฮดในการค้นหาเส้นทาง มีหน่วยเป็นบิทต่อวินาที

$$\text{Average control overhead} = \frac{\sum \text{all control packets size in the network (RTR)}}{\text{Simulation Time}} \qquad (6.1)$$

6.1.2 ดีเลย์ในการส่งข้อมูล (Data Packet Delay: Delay)

ค่าดีเลย์เฉลี่ยในการส่งแพคเกจข้อมูล สามารถคำนวณหาได้จากเวลารวมที่แพคเกจข้อมูล (data packet) ทั้งหมด ถูกส่งจากต้นทางจนกระทั่งไปถึงยังผู้รับปลายทาง หารด้วยจำนวนแพคเกจข้อมูลที่ถูกส่งไปยังปลายทาง

$$\text{Average End - to - End Delay} = \frac{\sum \text{all data packets delay}}{\text{Amount of Transfered data packets}} \qquad (6.2)$$

6.1.3 อัตราการรับส่งข้อมูล (Packet Delivery Fraction: PDF)

เป็นการหาอัตราส่วนที่แพคเกจถูกส่งไปถึงผู้รับต่อจำนวนแพคเกจที่ส่งไปทั้งหมด คำนวณได้จาก

$$\text{PDF} = \frac{\text{Amount of packets were successfully sent to the taget}}{\text{Amount of packets were sent to the network}} \quad (6.3)$$

6.1.4 ภาระโหลดที่มีต่อโมไบล์โฮสต์ (Mobile Host Load)

เป็นค่าของโหลดที่เกิดขึ้นจากการทำงานเฉพาะที่เกิดจากแพคเกจในกระบวนการค้นหาเส้นทางและการหาค่าความสัมพันธ์ระหว่างโมไบล์โฮสต์ในเครือข่าย ซึ่งคำนวณได้จาก

$$\text{Mobile Host Load} = \frac{\sum \text{size of all data packets} + \sum \text{size of all routing packets}}{\text{Amount of mobile host in the network}} \quad (6.4)$$

6.1.5 ประสิทธิภาพในการสื่อสารข้อมูล (Data Throughput)

คือปริมาณข้อมูลที่สามารถสื่อสารได้ทั้งเครือข่ายต่อช่วงเวลาการจำลองการทำงาน คำนวณได้จาก

$$\text{Data Throughput} = \frac{\sum \text{size of all data packets}}{\text{Simulation Time}} \quad (6.5)$$

6.2 ผลการจำลองการทำงานในสภาพแวดล้อมต่างๆ

งานวิจัยนี้ได้จำลองการทำงานของเครือข่ายออกเป็น 2 สภาวะแวดล้อมซึ่งมีพารามิเตอร์ดังตารางที่ 6.1 และ 6.2 เพื่อหาค่าพารามิเตอร์ประสิทธิภาพจาก 6.1 โดยในการทำงานสภาพแวดล้อมแรกเพื่อศึกษาผลของความเร็วที่มีผลต่อประสิทธิภาพระหว่างโปรโตคอลค้นหาเส้นทางเอบีอาร์และโปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์

ตารางที่ 6.1 การจำลองการทำงานในสภาพแวดล้อมที่ 1

พื้นที่จำลองการทำงาน	1000เมตร x 1000เมตร
จำนวนโมไบล์โฮสต์	50
เวลาหยุดพัก (pause time)	5 วินาที
ความเร็วการเคลื่อนที่ของโมไบล์โฮสต์	5, 10, 15, 20, 25, 30 เมตรต่อวินาที ±10%
จำนวนคู่การสื่อสาร	20

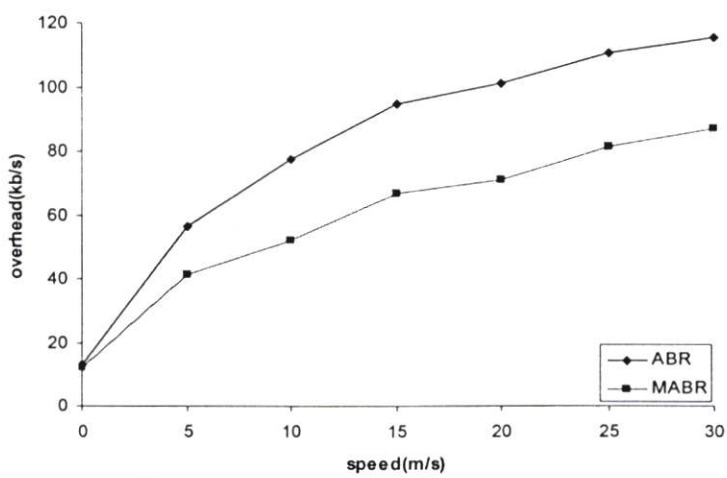
อัตราการส่งแพคเกจ	4 แพคเกจต่อวินาที
รูปแบบการกระจายและการเคลื่อนที่	แรนดอมเวย์พอยน์ ,แรนดอมยูนิฟอร์ม
เวลาการจำลองการทำงาน (Simulation time)	1800 วินาที
แบนด์วิธ	2 ล้านบิทต่อวินาที
รัศมีทำการของโมไบล์โฮสต์	250 เมตร

ตารางที่ 6.2 การจำลองการทำงานในสภาพแวดล้อมที่ 2

พื้นที่จำลองการทำงาน	1000เมตร x 1000เมตร
จำนวนโมไบล์โฮสต์	50
เวลาหยุดพัก (pause time)	5 วินาที
ความเร็วการเคลื่อนที่ของโมไบล์โฮสต์	3, 20, 50, 80 กิโลเมตรต่อชั่วโมง ±10%
จำนวนคู่การสื่อสาร	1, 2, 4, 8, 12, 16, 20, 25, 30, 35, 40, 45, 50
อัตราการส่งแพคเกจ	2 แพคเกจต่อวินาที
รูปแบบการกระจายและการเคลื่อนที่	แรนดอมเวย์พอยน์ ,แรนดอมยูนิฟอร์ม
เวลาการจำลองการทำงาน (Simulation time)	1800 วินาที
แบนด์วิธ	2 ล้านบิทต่อวินาที
รัศมีทำการของโมไบล์โฮสต์	250 เมตร

6.2.1 โอเวอร์เฮดในสภาพแวดล้อมแบบที่ 1

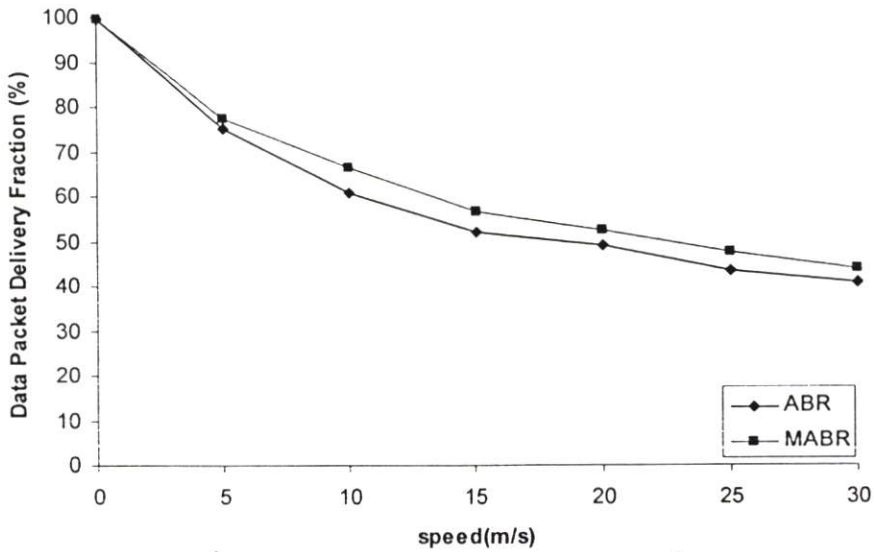
กราฟเปรียบเทียบความเร็วซึ่งส่งผลต่อปริมาณคอนโทรลโอเวอร์เฮด ผลการทดลองพบว่า ค่าโอเวอร์เฮดที่ได้จากกราฟ เห็นได้ว่าปริมาณโอเวอร์เฮดของโปรโตคอลที่ปรับปรุงเอ็มเอบีอาร์มีประสิทธิภาพที่ดีกว่าเมื่อเทียบกับโปรโตคอลเดิม โดยสามารถลดโอเวอร์เฮดในการบรอดคาสเมสเสจร้องข้อเส้นทางในเครือข่ายลงได้ถึง 30-40%



รูปที่ 6.1 กราฟ COในสภาพแวดล้อมแบบที่ 1

6.2.2 อัตราการรับส่งข้อมูล ในสภาพแวดล้อมแบบที่ 1

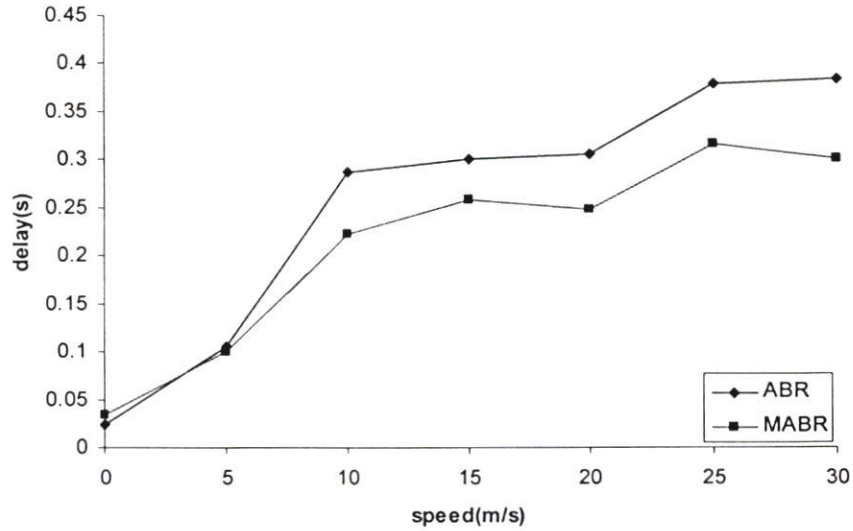
กราฟแสดงอัตราการรับส่งข้อมูลเปรียบเทียบกับความเร็วของโมไบล์โฮสต์ในเครือข่าย แสดงให้เห็นถึงประสิทธิภาพการรับส่งข้อมูลที่ดีขึ้น ของโปรโตคอลเอ็มเอบีอาร์ที่ดีกว่าโปรโตคอลเอบีอาร์สูงสุดถึงประมาณ 10% อันเนื่องมาจากการจำกัดพื้นที่บรอดคาสต์ลดการรบกวนเส้นทางการส่งข้อมูลในเครือข่ายลง



รูปที่ 6.2 กราฟ PDF ในสภาพแวดล้อมแบบที่ 1

6.2.3 ดีเลย์การรับส่งข้อมูล ในสภาพแวดล้อมแบบที่ 1

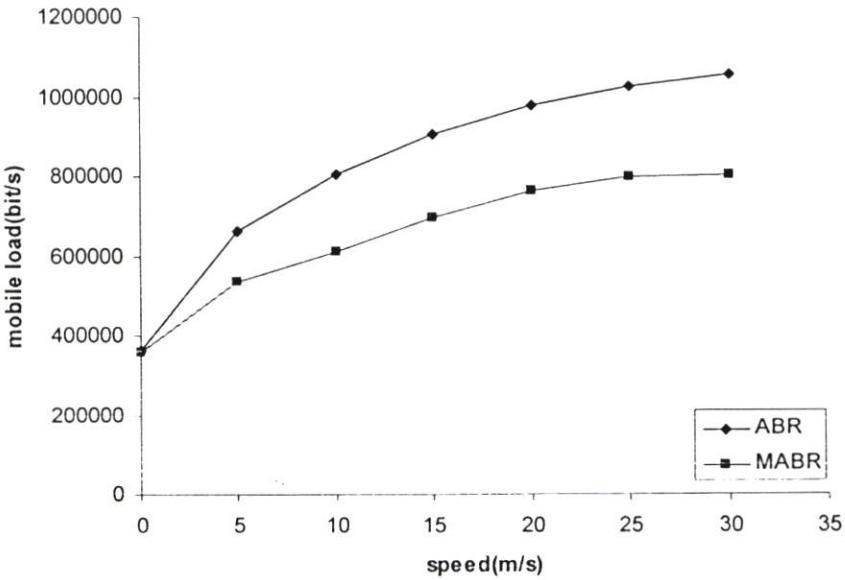
กราฟเปรียบเทียบความเร็วกับค่าดีเลย์ จากผลการทดลองแสดงให้เห็นว่าโปรโตคอลค้นหาเส้นทางเอบีอาร์มีดีเลย์ที่สูงกว่าโปรโตคอลที่ได้ทำการปรับปรุงสูงสุดถึงประมาณ 25%



รูปที่ 6.3 กราฟ Delay ในสภาพแวดล้อมแบบที่ 1

6.2.4 ภาระโหลด ในสภาพแวดล้อมแบบที่ 1

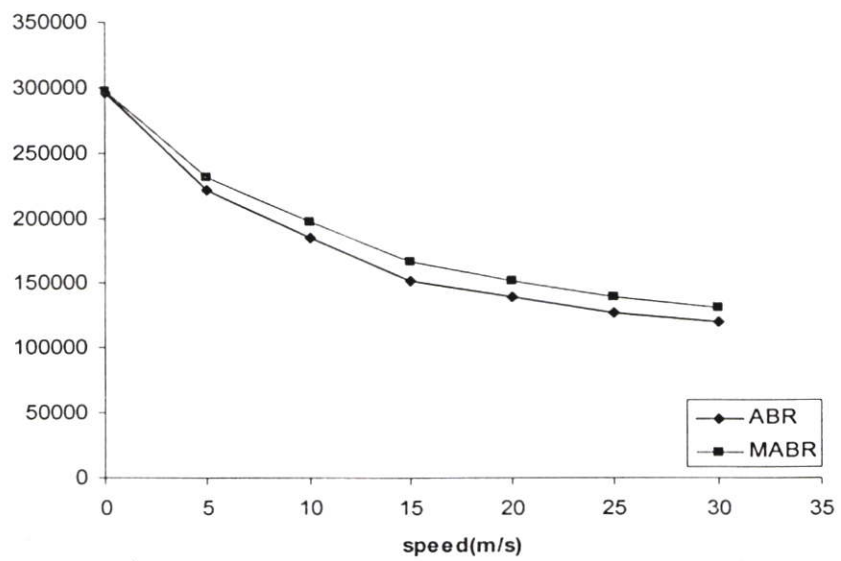
เปรียบเทียบค่าภาระของโมไบล์โฮสต์กับความเร็วของโมไบล์โฮสต์ การลดโอเวอร์เฮด ในการค้นหาเส้นทางลงทำให้ปริมาณแพคเกจในเครือข่ายลดลง โปรโตคอลเอ็มเอบีอาร์สามารถ ปรับปรุงประสิทธิภาพจากโปรโตคอลค้นหาเส้นทางเอบีอาร์ได้สูงถึงประมาณ 30%



รูปที่ 6.4 กราฟ Mobile Load ในสภาพแวดล้อมแบบที่ 1

6.2.5 ประสิทธิภาพการรับส่งข้อมูล ในสภาพแวดล้อมแบบที่ 1

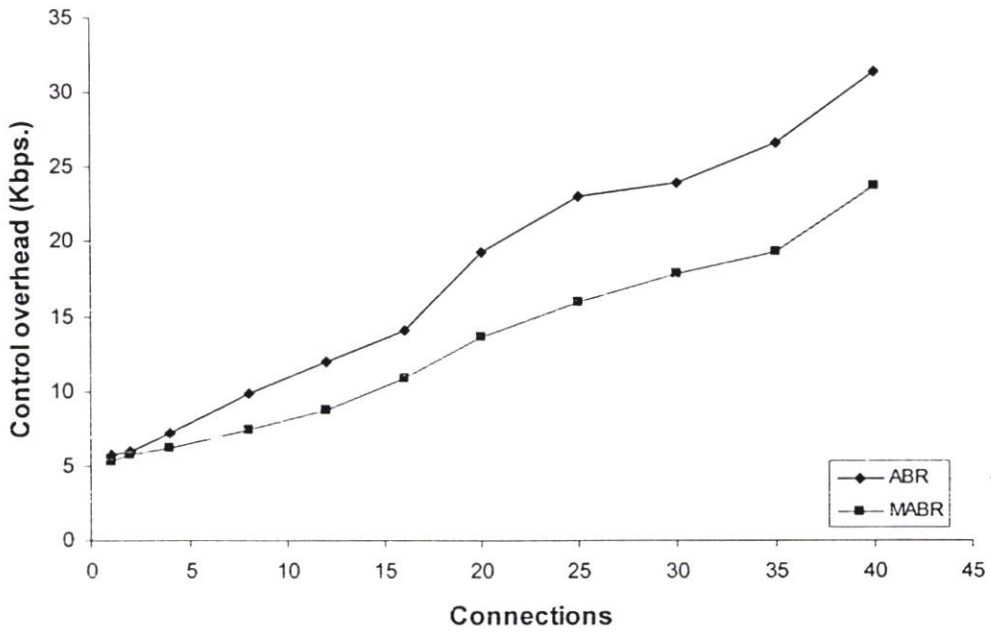
กราฟเปรียบเทียบประสิทธิภาพการรับส่งข้อมูลเปรียบเทียบกับความเร็วในการเคลื่อนที่ ของโมไบล์โฮสต์ หรือ Throughput ของการรับส่งข้อมูลมีหน่วยเป็น บิทต่อวินาที



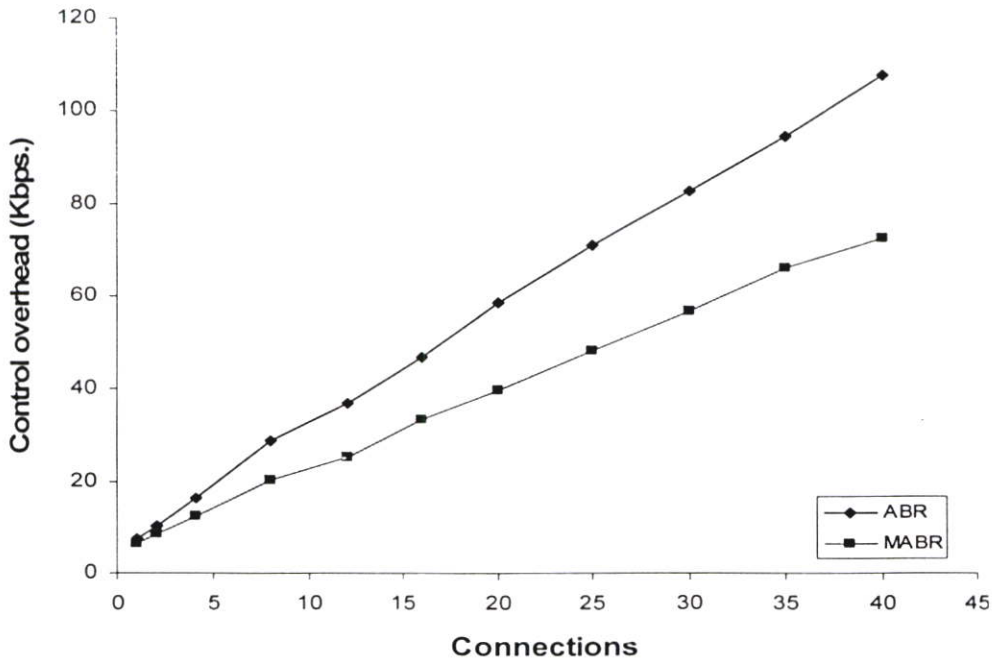
รูปที่ 6.5 กราฟ Data Throughput ในสภาพแวดล้อมแบบที่ 1

6.2.6 โอเวอร์เฮดในสภาพแวดล้อมที่ 2 ที่ความเร็ว 3, 20, 50, 80km/hr

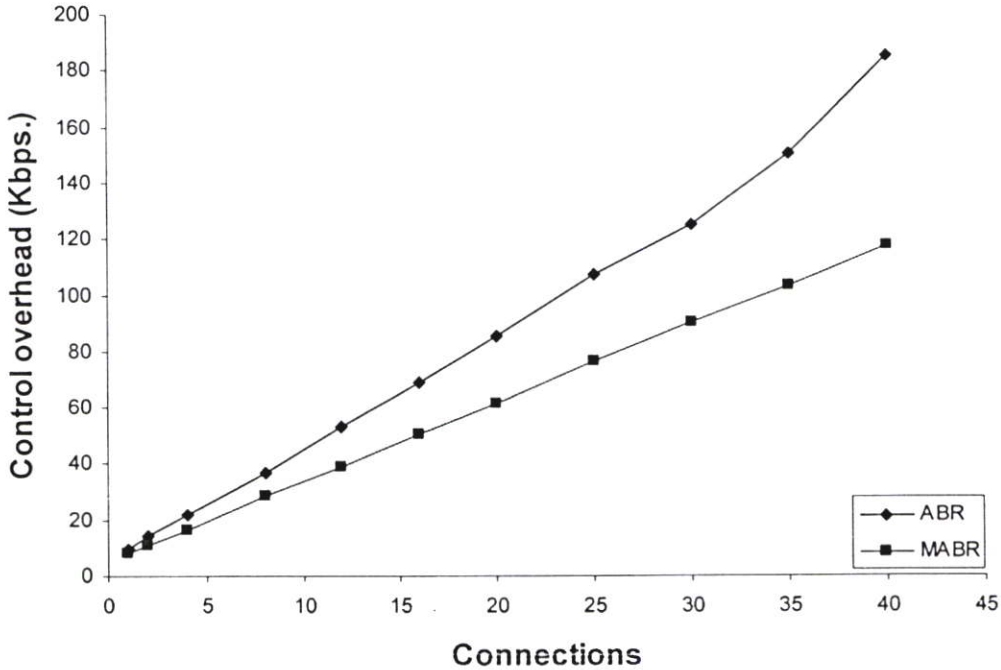
แสดงค่าประสิทธิภาพของจำนวนคู่การสื่อสารข้อมูล ซึ่งส่งผลต่อค่าโอเวอร์เฮดในการค้นหาเส้นทางภายในเครือข่ายจะเห็นได้ว่า รูปที่ 6.7-6.9 มีการเปลี่ยนแปลงทันใดที่ปริมาณการเชื่อมต่อ 40คู่ แต่จะพบว่าโปรโตคอลเอ็มเอบีอาร์ยังสามารถลดโอเวอร์เฮดต่ำกว่าโปรโตคอลเอบีอาร์ต้นแบบ



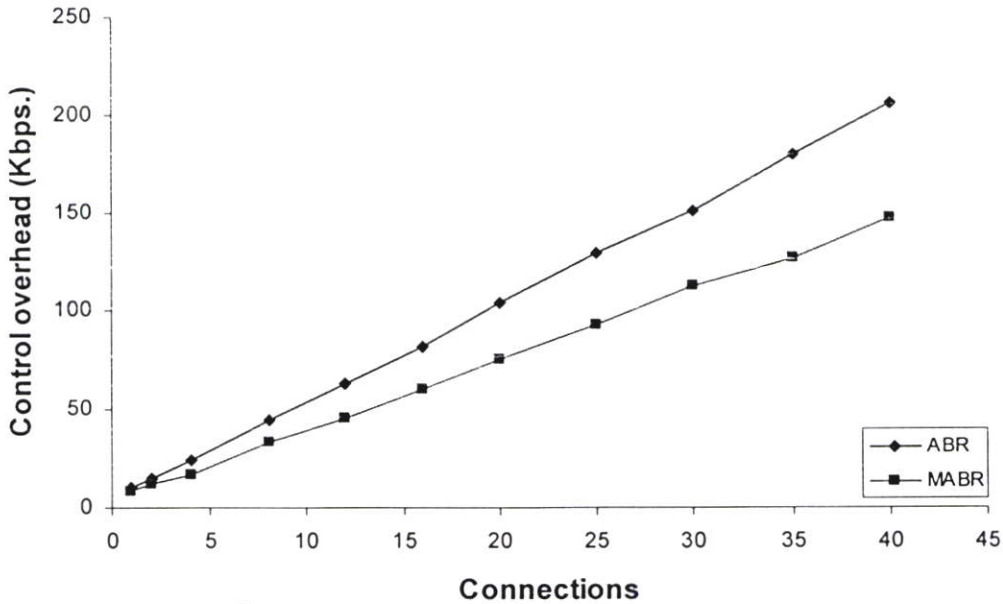
รูปที่ 6.6 กราฟ COในสภาพแวดล้อมแบบที่ 2 (3km/hr)



รูปที่ 6.7 กราฟ COในสภาพแวดล้อมแบบที่ 2 (20km/hr)



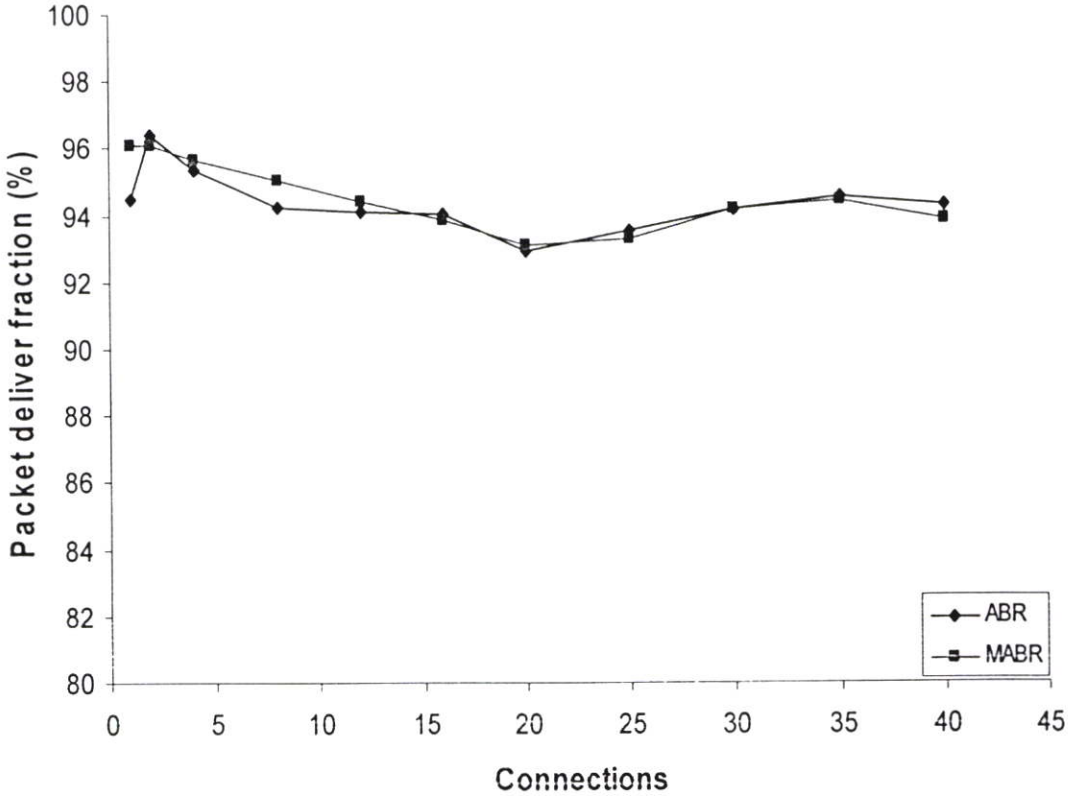
รูปที่ 6.8 กราฟ COในสภาพแวดล้อมแบบที่ 2 (50km/hr)



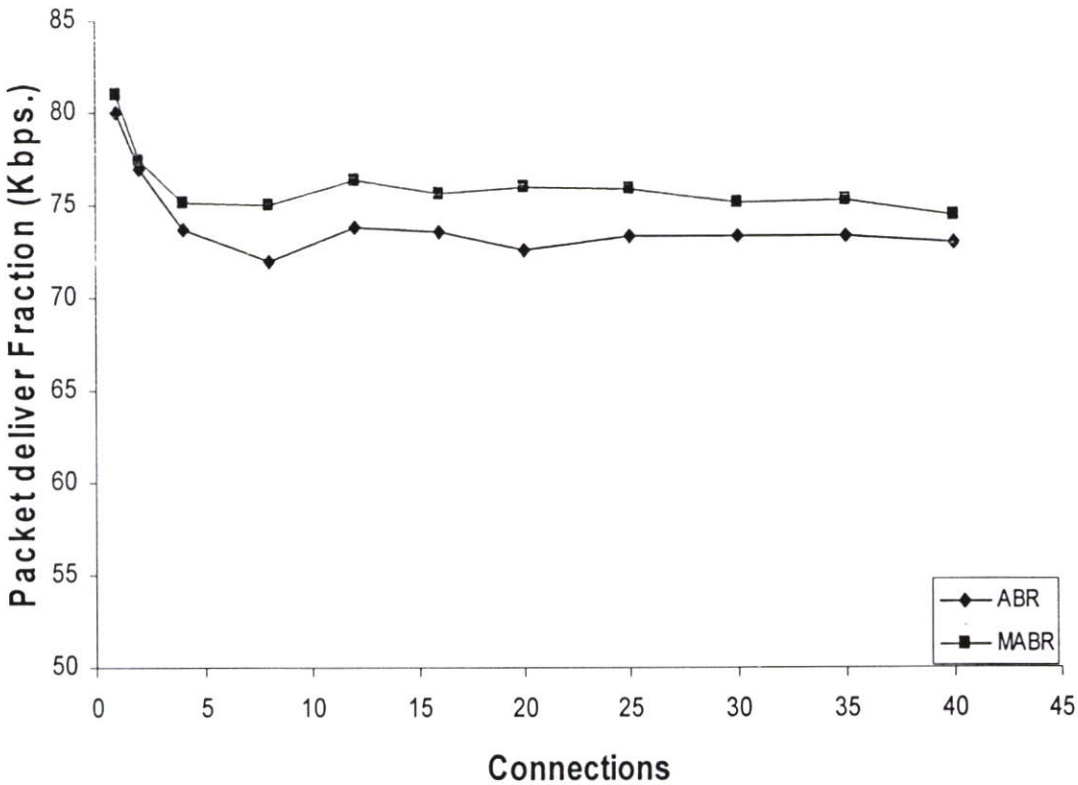
รูปที่ 6.9 กราฟ COในสภาพแวดล้อมแบบที่ 2 (80km/hr)

6.2.7 อัตราการรับส่งข้อมูล ในสภาพแวดล้อมที่ 2 ที่ความเร็ว 3, 20 50 80km/hr

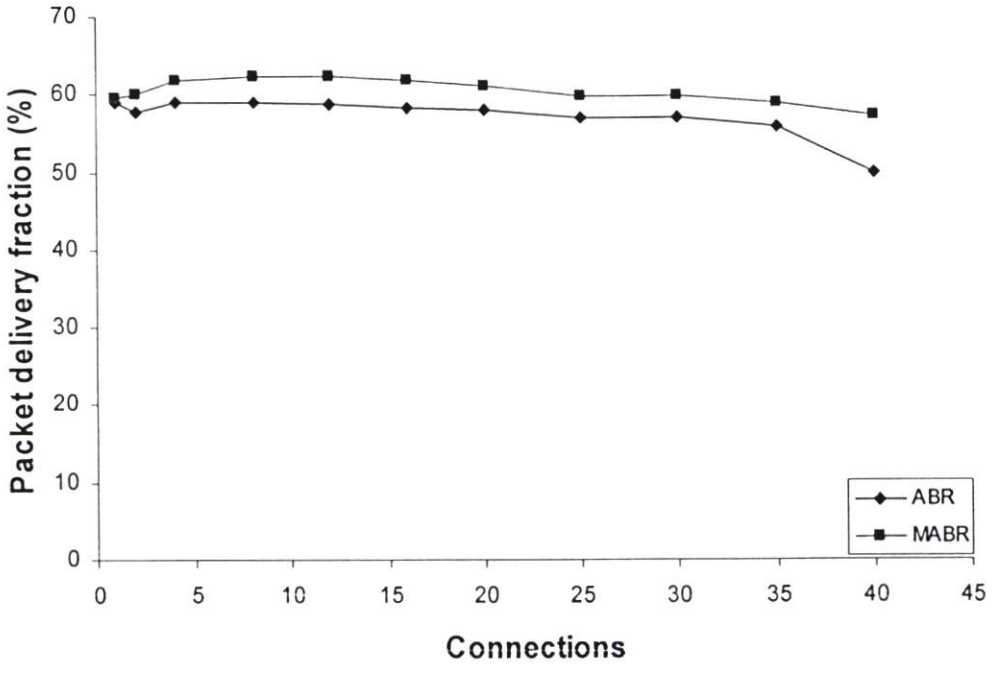
กราฟแสดงค่าประสิทธิภาพในการรับส่งข้อมูลเทียบกับปริมาณคู่การสื่อสารที่ความเร็วต่างๆกัน ผลที่ได้แสดงให้เห็นว่าเมื่อความเร็วสูงขึ้น และคู่การสื่อสารเพิ่มขึ้น ค่า PDF ของโปรโตคอลเอ็มเอบีอาร์จะมีค่าสูงกว่าโปรโตคอลเอ็มเอบีอาร์ เมื่อความเร็วและคู่การสื่อสารเพิ่มขึ้น



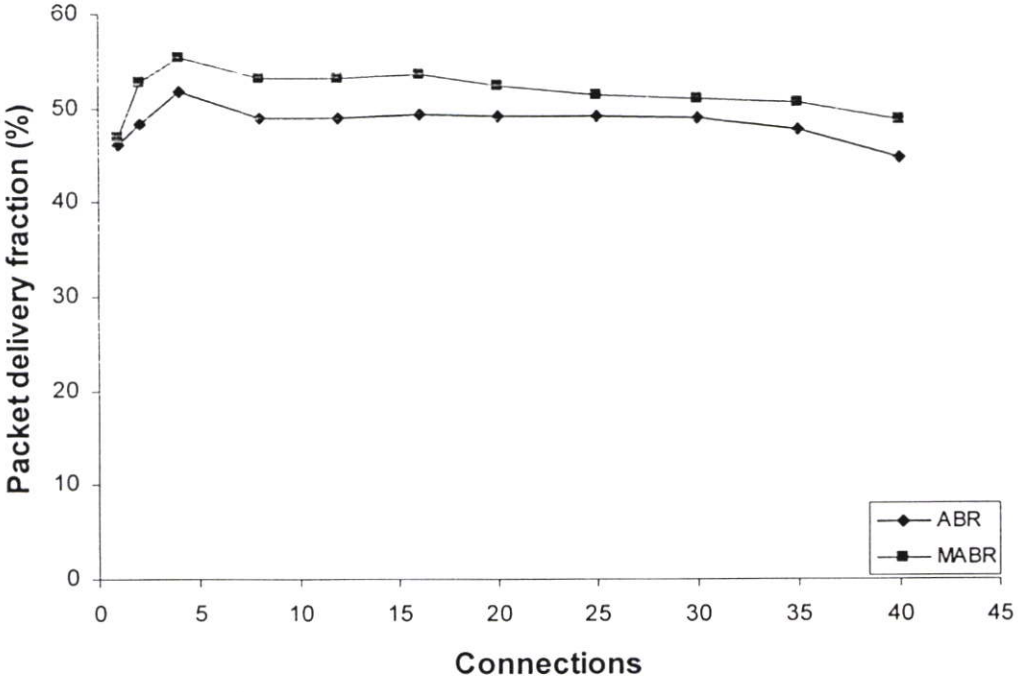
รูปที่ 6.10 กราฟ PDF ในสภาพแวดล้อมแบบที่ 2 (3km/h)



รูปที่ 6.11 กราฟ PDF ในสภาพแวดล้อมแบบที่ 2 (20km/h)



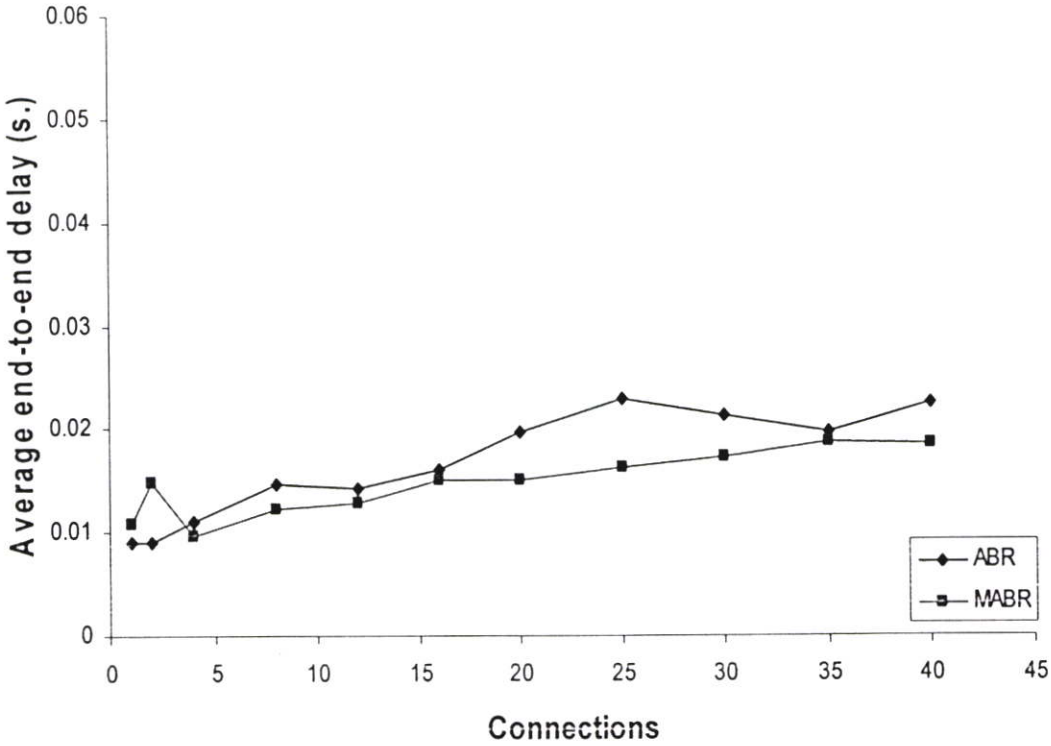
รูปที่ 6.12 กราฟ PDF ในสภาพแวดล้อมแบบที่ 2 (50km/h)



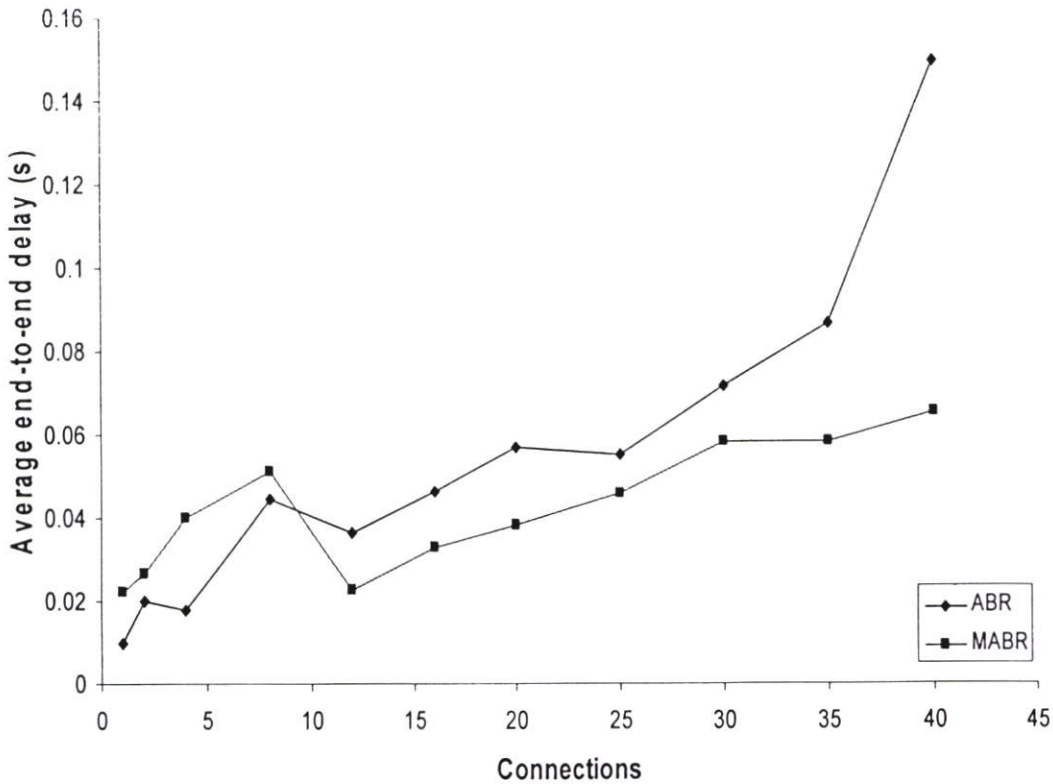
รูปที่ 6.13 กราฟ PDF ในสภาพแวดล้อมแบบที่ 2 (80km/h)

6.2.8 ดีเลย์การรับส่งข้อมูล ในสภาพแวดล้อมที่ 2

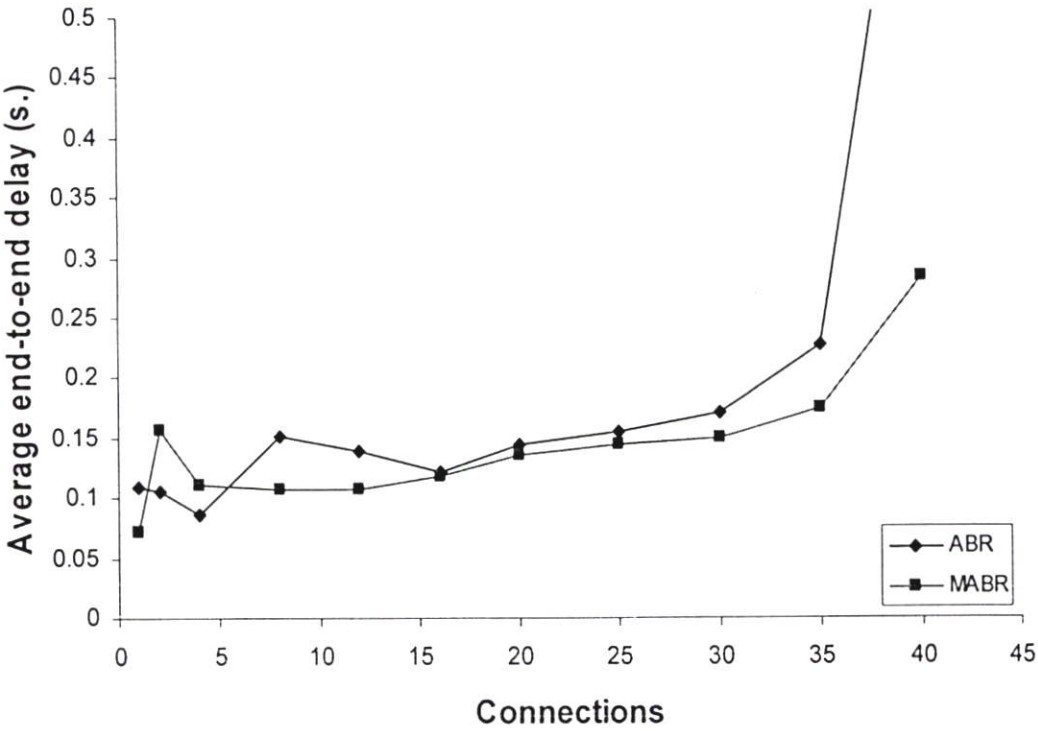
แสดงค่าเฉลี่ยดีเลย์ที่เกิดขึ้นจากเวลาที่แพคเกจข้อมูลใช้เวลาเดินทางจากต้นทางไปยังปลายทาง จะเห็นได้ว่าเมื่อความเร็วของโมไบล์โฮสต์สูงขึ้นและเมื่อปริมาณคู่การสื่อสารเพิ่มขึ้น โปรโตคอลเอ็มเอบีอาร์จะมีดีเลย์ที่ต่ำกว่าโปรโตคอลเอบีอาร์



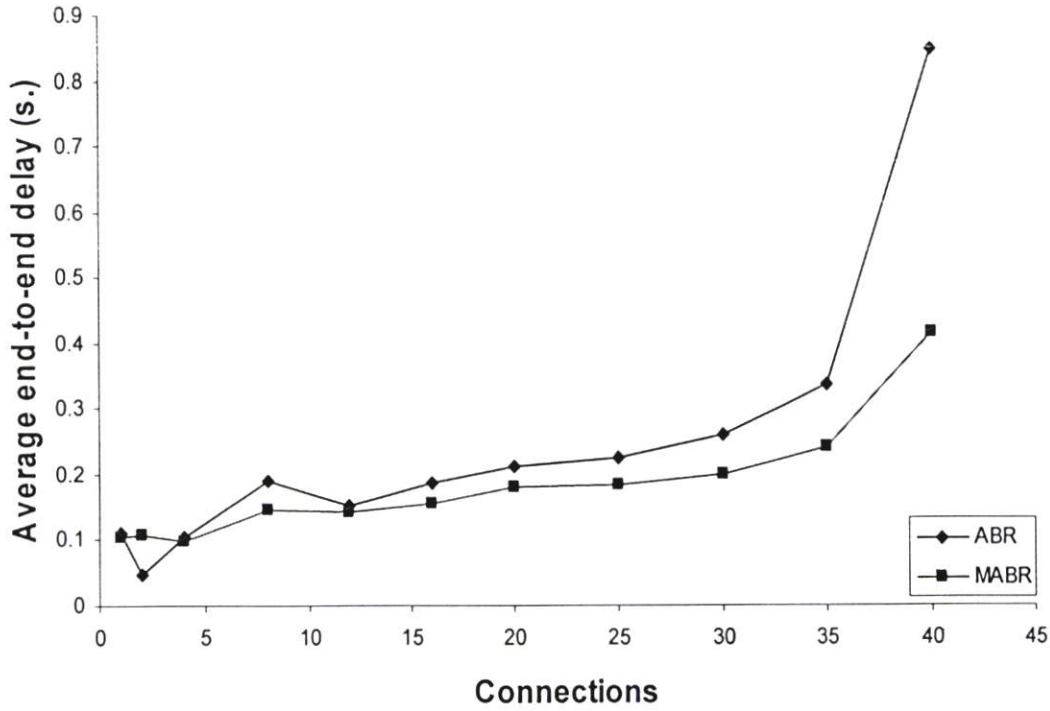
รูปที่ 6.14 กราฟ Delay ในสภาพแวดล้อมแบบที่ 2 (3km/hr)



รูปที่ 6.15 กราฟ Delay ในสภาพแวดล้อมแบบที่ 2 (20km/hr)



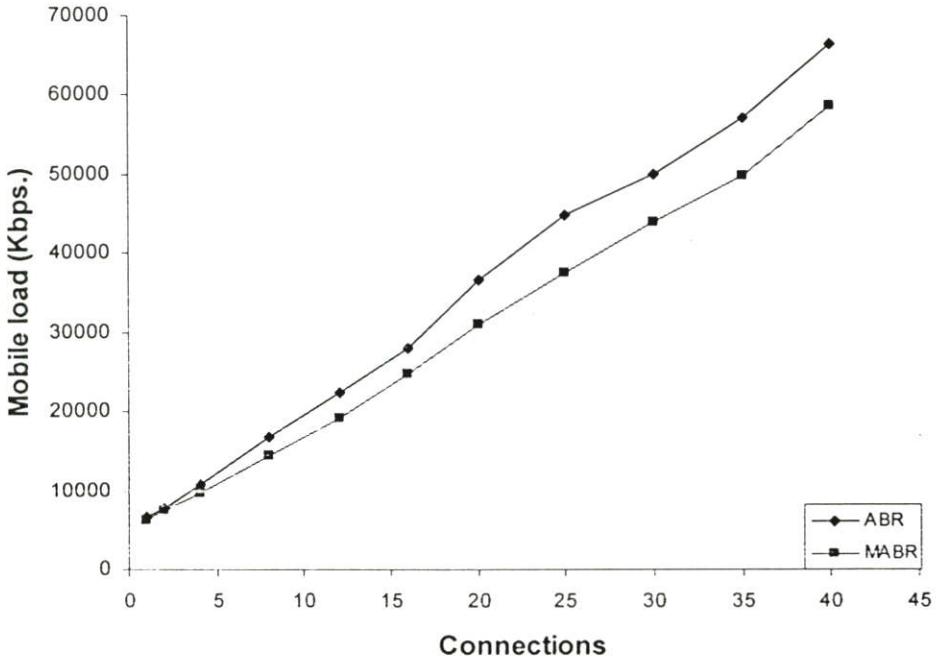
รูปที่ 6.16 กราฟ Delay ในสภาพแวดล้อมแบบที่ 2 (50km/hr)



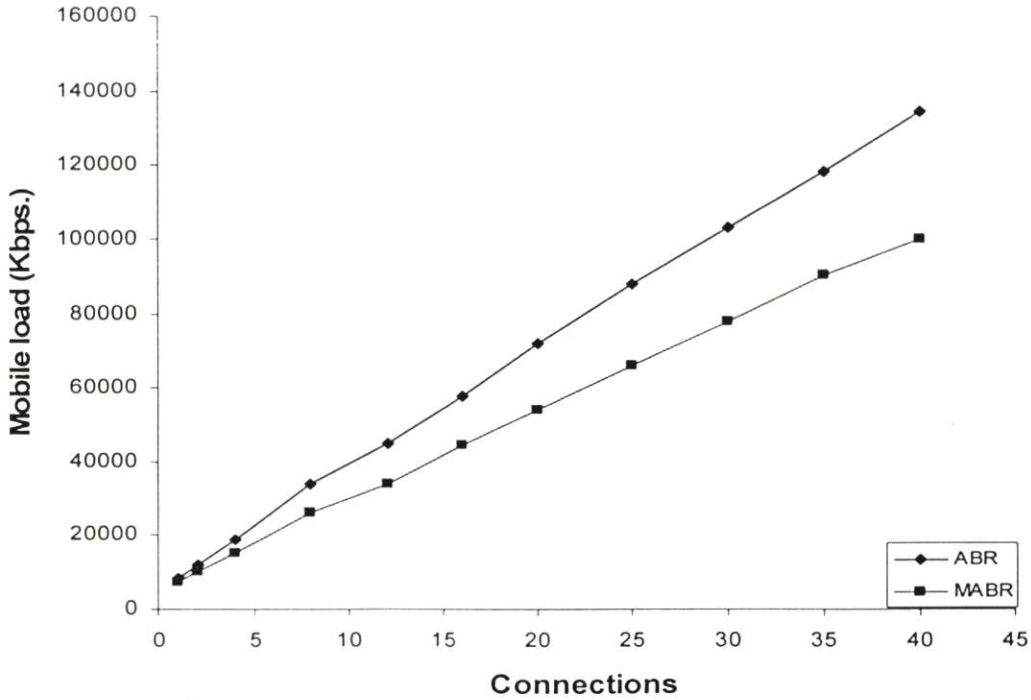
รูปที่ 6.17 กราฟ Delay ในสภาพแวดล้อมแบบที่ 2 (80km/hr)

6.2.9 ภาระโหลดในสภาพแวดล้อมแบบที่ 2

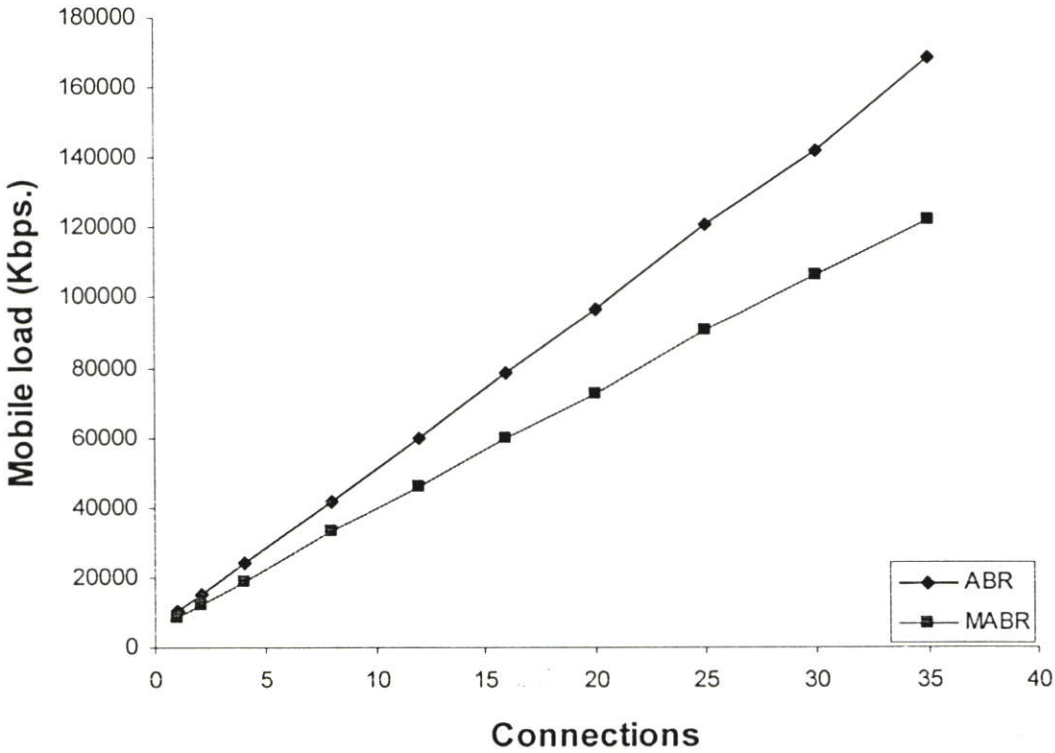
แสดงค่าภาระของโหลดที่เกิดขึ้นในแต่ละโมไบล์โฮสต์โดยเฉลี่ยภายในเครือข่าย เปรียบเทียบกับจำนวนคู่ของการสื่อสาร ในความเร็วที่แตกต่างกัน จะเห็นได้อย่างชัดเจนว่าจำนวนคู่ของการสื่อสารมีผลต่อภาระที่เกิดขึ้นต่อโมไบล์โฮสต์



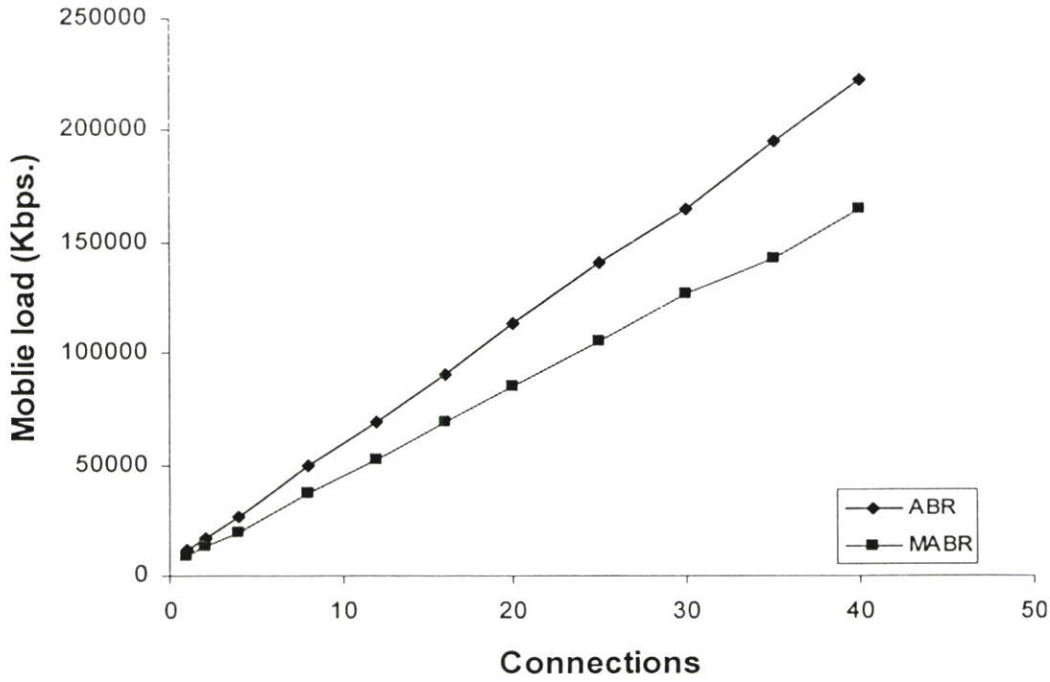
รูปที่ 6.18 กราฟ Mobile Load ในสภาพแวดล้อมแบบที่ 2 (3km/hr)



รูปที่ 6.19 กราฟ Mobile Load ในสภาพแวดล้อมแบบที่ 2 (20km/hr)



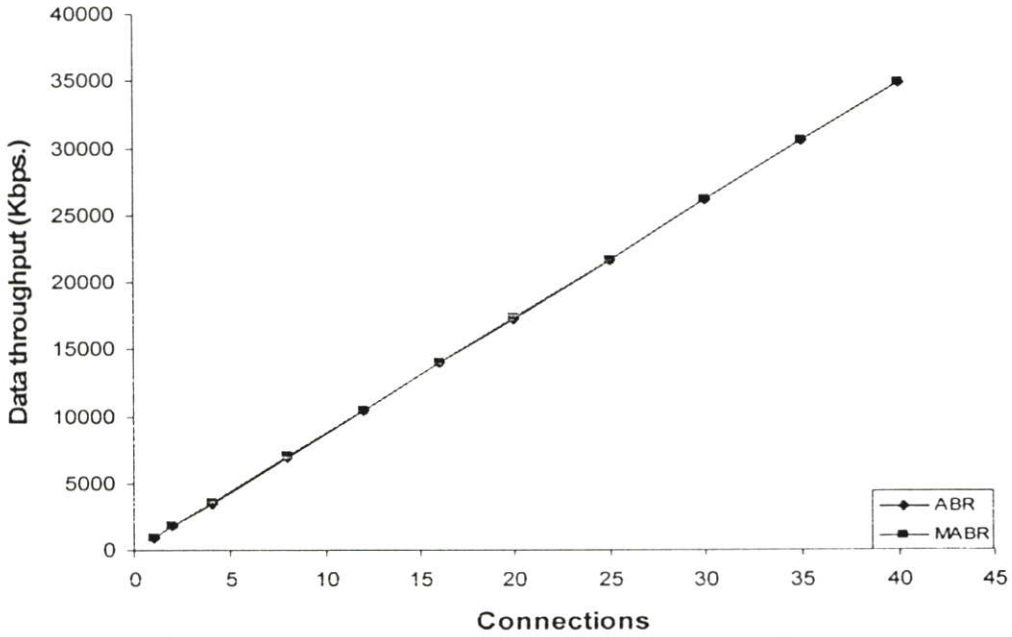
รูปที่ 6.20 กราฟ Mobile Load ในสภาพแวดล้อมแบบที่ 2 (50km/hr)



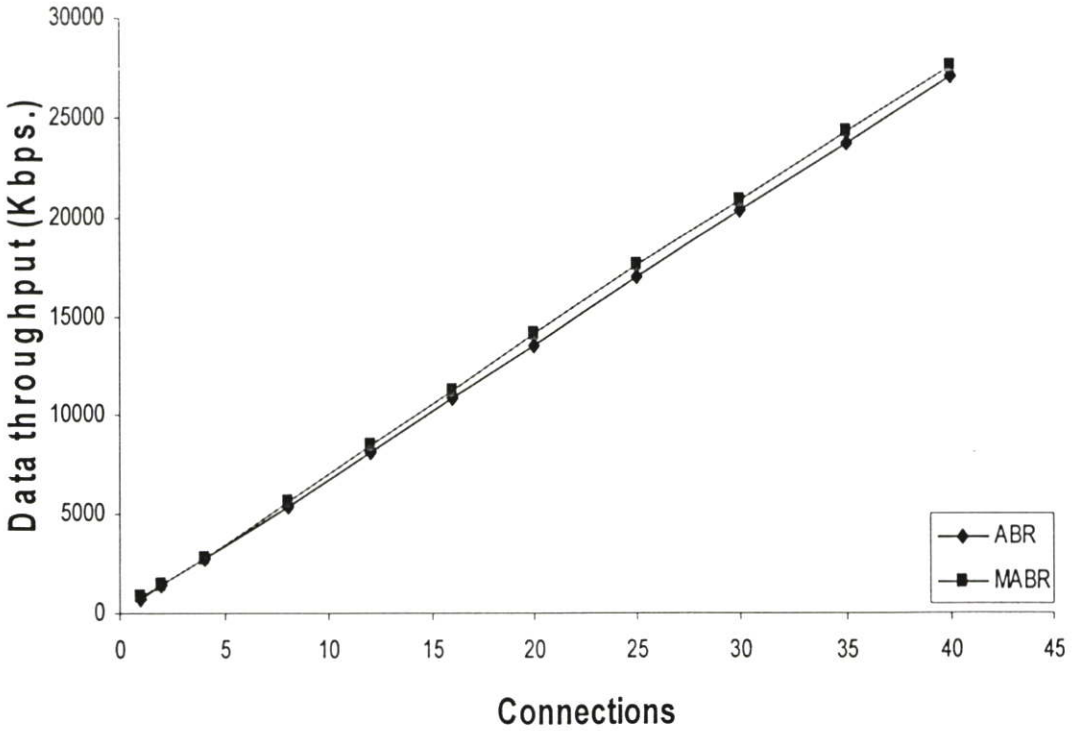
รูปที่ 6.21 กราฟ Mobile Load ในสภาพแวดล้อมแบบที่ 2 (80km/hr)

6.2.10 ประสิทธิภาพการรับส่งข้อมูลในสภาพแวดล้อมแบบที่ 2

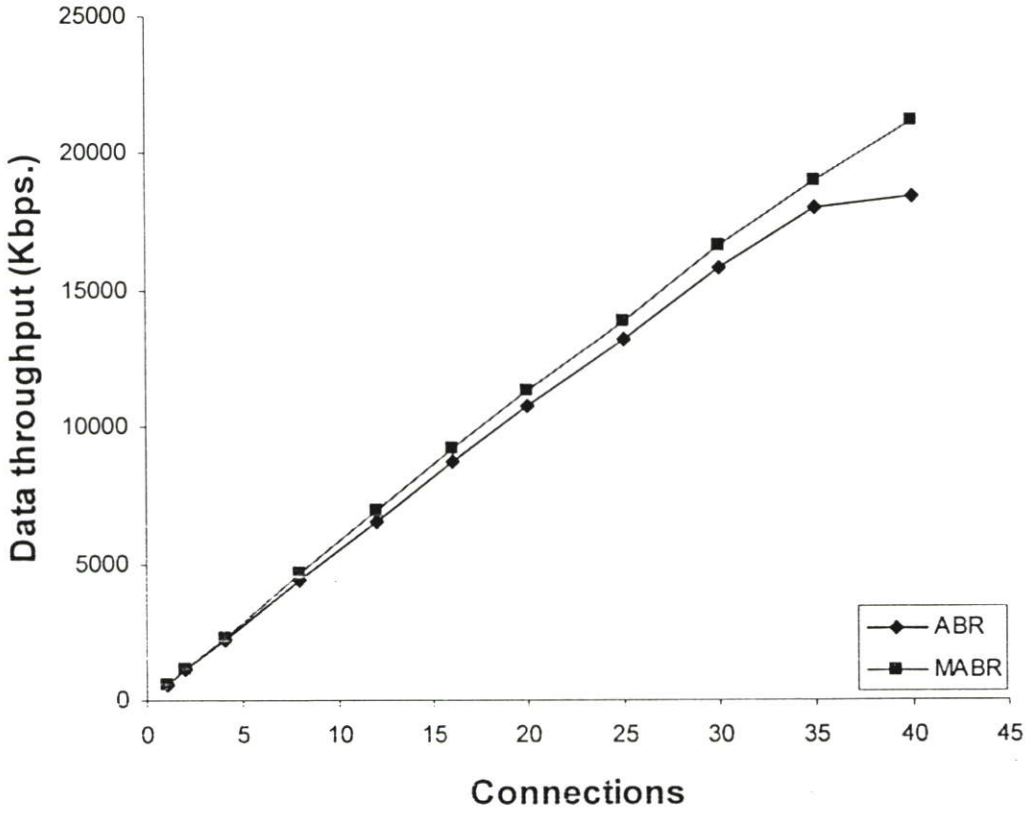
กราฟแสดงประสิทธิภาพในการรับส่งข้อมูลในเครือข่าย(Data Throughput) จากรูปที่ 6.22-6.25 โปรโตคอลเอบีอาร์และโปรโตคอลเอ็มเอบีอาร์จะมีประสิทธิภาพของการรับส่งข้อมูลใกล้เคียงกัน เมื่อความเร็วของโมไบล์โฮสต์และจำนวนคู่การสื่อสารเพิ่มขึ้น โปรโตคอลเอ็มเอบีอาร์จะมีประสิทธิภาพที่ดีกว่าเล็กน้อย



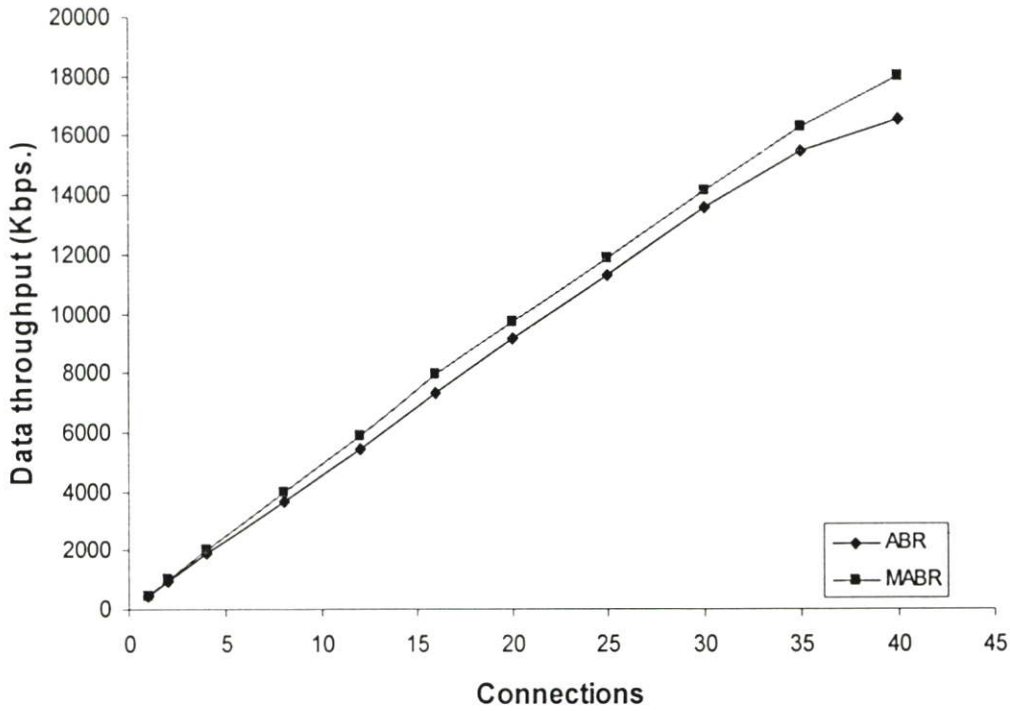
รูปที่ 6.22 กราฟ Data Throughput ในสภาพแวดล้อมแบบที่ 2 (3km/hr)



รูปที่ 6.23 กราฟ Data Throughput ในสภาพแวดล้อมแบบที่ 2 (20km/hr)



รูปที่ 6.24 กราฟ Data Throughput ในสภาพแวดล้อมแบบที่ 2 (50km/hr)



รูปที่ 6.25 กราฟ Data Throughput ในสภาพแวดล้อมแบบที่ 2 (80km/hr)

6.3 วิเคราะห์ผลที่ได้จากการจำลองการทำงาน

ผลที่ได้จากการจำลองการทำงานในหัวข้อ 6.2 แบ่งออกเป็นการทดลอง 2 สภาพแวดล้อม ในสภาพแวดล้อมที่ 1 ตารางที่ 6.1 โดยหาค่าพารามิเตอร์ที่สำคัญ 5 พารามิเตอร์ ประกอบด้วย คอนโทรลโอเวอร์เฮด อัตราการรับส่งข้อมูล ดีเลย์ ภาระโหลดแต่ละโมไบล์โฮสต์และค่าประสิทธิภาพของเครือข่าย ทั้งหมดเป็นการหาค่าพารามิเตอร์เพื่อหาผลกระทบและสังเกตสภาพแวดล้อมที่มีความเหมาะสม หรือสภาพแวดล้อมที่เป็นข้อด้อยโดยเทียบจากการเปลี่ยนแปลงของความเร็วโมไบล์โฮสต์ในเครือข่าย และการเปลี่ยนแปลงปริมาณข้อมูลการเชื่อมต่อ (Traffic)

6.3.1 ประสิทธิภาพในการจำลองการทำงานสภาพแวดล้อมที่ 1 เพื่อหาผลกระทบของประสิทธิภาพต่อความเร็วของโมไบล์โฮสต์ในเครือข่าย

ผลที่ได้จากการจำลองการทำงานในสภาพแวดล้อมที่ 1 จาก [20], [21] ในรูปที่ 6.1 เมื่อโมไบล์โฮสต์หยุดนิ่งจะพบว่าปริมาณโอเวอร์เฮดที่เกิดขึ้นจากการค้นหาเส้นทางในโปรโตคอลเอ็มเอบีอาร์มีค่าใกล้เคียงกับโปรโตคอลค้นหาเส้นทางต้นแบบเอบีอาร์ เนื่องจาก เมื่อเริ่มจำลองการทำงานโปรโตคอลเอ็มเอบีอาร์ แต่ละโมไบล์โฮสต์ยังไม่มีข้อมูลตำแหน่งของโมไบล์โฮสต์อื่นๆ เอ็มเอบีอาร์จะทำการบรอดคาสต์หาเส้นทางด้วยวิธีเดียวกับโปรโตคอลเอบีอาร์ ด้วยวิธีการบรอดคาสต์เมสเสจร้องขอเส้นทางไปทั้งเครือข่าย หลังจากนั้นตารางข้อมูลตำแหน่งของแต่ละโมไบล์โฮสต์จะเริ่มเก็บข้อมูลตำแหน่งจากแพคเกจร้องขอเส้นทางและแพคเกจตอบกลับเส้นทาง (request/reply) เนื่องจากแต่ละโมไบล์โฮสต์ซึ่งเคลื่อนที่ด้วยความเร็วต่ำมากรวมทั้งกรณีการหยุดนิ่งของโมไบล์โฮสต์ เส้นทางจึงสามารถคงอยู่ได้นานและมีอัตราการเสียหายของเส้นทางต่ำ ส่งผลให้ค่าโอเวอร์เฮดที่ใช้ในกระบวนการค้นหาเส้นทางมีปริมาณน้อย เมื่อโมไบล์โฮสต์เริ่มเคลื่อนที่ด้วยความเร็วสูงขึ้นจะพบความแตกต่างของการใช้พื้นที่การส่งต่อเมสเสจร้องข้อเส้นทางในโปรโตคอลเอ็มเอบีอาร์อย่างชัดเจนเนื่องจาก การเคลื่อนที่ของโมไบล์โฮสต์ที่สูงขึ้นส่งผลต่อโอเวอร์เฮดในการค้นหาเส้นทางเนื่องจากการเพิ่มขึ้นของอัตราความเสียหายต่อเส้นทาง จากแนวคิดจำกัดพื้นที่การบรอดคาสต์ให้มีพื้นที่เล็กลงสามารถทำให้สามารถลดพื้นที่การส่งต่อเมสเสจร้องขอเส้นทางหรือมีผลต่อโอเวอร์เฮดจากการค้นหาเส้นทางโดยตรง

ในรูปที่ 6.2 จะเห็นได้ว่าค่าอัตราการรับส่งข้อมูลจะค่อยๆมีแนวโน้มต่ำลงเมื่อความเร็วและจำนวนคู่การสื่อสารเพิ่มสูงขึ้น สาเหตุหลักเนื่องมาจากปัญหาความเสียหายของเส้นทางที่เกิดขึ้นเมื่อโมไบล์โฮสต์เคลื่อนที่ด้วยความเร็วสูงขึ้น เนื่องจากการค้นหาเส้นทางในโปรโตคอลเอ็มเอบีอาร์สามารถทำได้อย่างรวดเร็ว ผลจากการจำลองการทำงานโปรโตคอลเอ็มเอบีอาร์จึงสามารถเพิ่มค่าอัตราการรับส่งข้อมูลให้สูงขึ้นได้เมื่อเทียบกับโปรโตคอลเอบีอาร์

สำหรับรูปที่ 6.3 แสดงให้เห็นแนวโน้มการเพิ่มขึ้นของดีเลย์ในการรับส่งข้อมูลเนื่องจากการเพิ่มขึ้นของความเร็วและคู่การสื่อสาร เนื่องจากการหยุดชะงักในการตอบสนองต่อเมสเสจ

ร้องขอเส้นทาง ซึ่งเมื่อความเร็วสูงขึ้นยิ่งส่งผลต่อความเสียหายต่อเส้นทางอยู่บ่อยครั้ง การส่งข้อมูลจึงหยุดชะงักเนื่องจากโมไบล์โฮสต์ต้องทำการตอบสนองต่อเมสเสจร้องขอเส้นทางซึ่งมีค่าความสำคัญ(priority) สูงกว่าแพคเกจข้อมูลโดยทั่วไป ดังนั้นแพคเกจข้อมูลจึงเข้าคิวอยู่ในโมไบล์โฮสต์นั้นจนกว่าโมไบล์โฮสต์จะตอบสนองต่อเมสเสจร้องขอเส้นทางได้เรียบร้อย ซึ่งหากเมสเสจร้องขอเส้นทางมีจำนวนมากอาจทำให้แพคเกจข้อมูลที่อยู่ในคิวถูกดรอบ ซึ่งทำให้เกิดดีเลย์ในการรับส่งข้อมูลเพิ่มขึ้นอย่างมาก จากการจำลองการทำงานเราพบว่า ความเร็วเป็นตัวแปรสำคัญต่อพารามิเตอร์ดีเลย์ สำหรับรูปที่ 6.3 แสดงให้เห็นว่าโปรโตคอลเอ็มเอบีอาร์ยังสามารถปรับลดค่าดีเลย์ในการรับส่งข้อมูลให้ต่ำลงได้

สำหรับรูปที่ 6.4 แสดงถึงค่าภาระโหลดที่เกิดขึ้นต่อโมไบล์โฮสต์ที่อยู่ในเครือข่าย ซึ่งรวมทั้งข้อมูลที่ถูกส่งผ่านไปยังเครือข่ายและข้อมูลอื่นๆในเลเยอร์ต่างๆทั้งหมด ซึ่งจะแสดงให้เห็นภาระทั้งหมดที่โมไบล์โฮสต์จะต้องรับภาระซึ่งจะเห็นได้ว่า เมื่อความเร็วที่สูงขึ้นและประมาณคู่การสื่อสารที่มากขึ้นส่งผลให้เกิดภาระโหลดต่อโมไบล์โอสต์สูงขึ้นอย่างรวดเร็วโดย ณ ความเร็วสูงโอเวอร์เฮดจะเป็นภาระโหลดส่วนใหญ่ของโมไบล์โฮสต์ เนื่องจากความสามารถในการลดโอเวอร์เฮดดังกล่าว จึงทำให้ภาระโหลดที่เกิดจากการลดโอเวอร์เฮดของโปรโตคอลเอ็มเอบีอาร์มีค่าต่ำกว่าโปรโตคอลค้นหาเส้นทางเอบีอาร์

ในรูปที่ 6.5 ค่าประสิทธิภาพของการรับส่งข้อมูลของโปรโตคอลเอ็มเอบีอาร์มีแนวโน้มลดลงเมื่อความเร็วและปริมาณคู่การสื่อสารเพิ่มขึ้น ในรูปที่ 6.5 แสดงให้เห็นว่าโปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์สามารถเพิ่มความสามารถในเทอมของประสิทธิภาพการรับส่งข้อมูลได้ เนื่องจากเมสเสจร้องขอเส้นทางเป็นคอนโทรลเมสเสจที่มีค่าความสำคัญมากกว่าแพคเกจข้อมูล ดังนั้นเมื่อโมไบล์โฮสต์ใดๆที่กำลังสื่อสารข้อมูลอยู่หรือกำลังส่งต่อข้อมูลโดยที่เป็นโมไบล์โฮสต์ที่อยู่ในเส้นทางการสื่อสารข้อมูล จะต้องหยุดการสื่อสารข้อมูลและตอบสนองต่อเมสเสจร้องขอเส้นทางที่เข้ามาก่อนกลับไปทำการสื่อสารข้อมูลอีกครั้ง ทำให้สามารถลดดีเลย์ที่เกิดจากการดรอบของคิวเนื่องจากการรอและลดปริมาณภาระโหลดที่เกิดจากโอเวอร์เฮดในการบรอดคาสต์หาเส้นทางได้อย่างชัดเจน ซึ่งจะเห็นได้ว่าเมื่อโมไบล์โฮสต์ในเครือข่ายมีการเคลื่อนที่สูงขึ้น โปรโตคอลค้นหาเส้นทางเอบีอาร์จะให้ประสิทธิภาพของเครือข่ายที่ดีขึ้นเมื่อเปรียบเทียบกับโปรโตคอลค้นหาเส้นทางเอบีอาร์

6.3.2 ประสิทธิภาพในการจำลองการทำงานสภาพแวดล้อมที่ 2 เพื่อหาผลกระทบของประสิทธิภาพต่อความเร็วและปริมาณข้อมูล (Traffic)ในเครือข่าย

ผลที่ได้จากการจำลองการทำงานในสภาพแวดล้อมที่ 2 แสดงการจำลองการทำงาน ในรูปที่ 6.6-6.25 โดยพิจารณาผลกระทบของความเร็วการเคลื่อนที่ของโมไบล์โฮสต์และปริมาณคู่การสื่อสารในเครือข่ายที่เพิ่มขึ้น(Traffic) โดยแบ่งความเร็วออกเป็น 4 ระดับคือความเร็ว 3 กิโลเมตรต่อชั่วโมงอยู่ในช่วงความเร็วของการเดินหรือวิ่งมนุษย์ ความเร็ว 20 กิโลเมตรต่อชั่วโมง

อยู่ในช่วงความเร็วของยานพาหนะที่ความเร็วต่ำ 50 กิโลเมตรในช่วงความเร็วปานกลาง และความเร็ว 80 กิโลเมตรต่อชั่วโมงเทียบกับความเร็วของยานพาหนะในความเร็วสูง การทดสอบการจำลองการทำงานโดยการเพิ่ม Traffic ให้กับเครือข่ายจนกระทั่งเครือข่ายอิ่มตัว เราพบว่าเครือข่ายยังคงทำงานได้ที่โหลด Traffic 40 คู่ ผลการจำลองการทำงานแสดงให้เห็นว่า หากมีการเพิ่ม Traffic ให้กับเครือข่ายส่งผลให้โปรโตคอลเอ็มเอบีอาร์สามารถลดโอเวอร์เฮดได้เช่นเดียวกับการทดลองในสภาพแวดล้อมที่ 1 มากขึ้นเมื่อเทียบกับโปรโตคอลเอบีอาร์ อันเนื่องมาจากจำนวนคู่การสื่อสารที่เพิ่มขึ้นจะเพิ่มจำนวนของกระบวนการค้นหาเส้นทางให้มากขึ้นตามไปด้วย

ในรูปที่ 6.6-6.9 แสดงให้เห็นแนวโน้มการเพิ่มขึ้นของโอเวอร์เฮดที่เกิดจากการเคลื่อนที่และจำนวนคู่การสื่อสาร ซึ่งมีแนวโน้มไปในรูปแบบเดียวกับการทดลองในสภาพแวดล้อมที่ 1 การทดลองแสดงให้เห็นว่าผลของพารามิเตอร์โอเวอร์เฮดมีค่าเฉลี่ยความสามารถในการลดโอเวอร์เฮดในระบบลงราว 30-35% ซึ่งแสดงให้เห็นชัดเจนในการใช้พื้นที่การส่งต่อ (request zone) เพื่อลดปริมาณการบรอดคาสต์เมสเสจร้องขอเส้นทาง

รูปที่ 6.10-6.13 เป็นกราฟแสดงความสามารถในการรับส่งข้อมูล ในส่วนผลการทดลองของอัตราการรับส่งข้อมูลพบว่าผลของอัตราการรับส่งข้อมูลที่ความเร็วต่ำ (3km/hr) การทำงานของโปรโตคอลเอ็มเอบีอาร์มีค่าใกล้เคียงกับโปรโตคอลเอบีอาร์ Traffic ที่เพิ่มขึ้นมีผลต่ออัตราการรับส่งข้อมูลไม่มากนัก แต่เมื่อความเร็วสูงขึ้นจะเห็นความชัดเจนของอัตราการส่งข้อมูลที่แตกต่างกันโดยที่โปรโตคอลเอ็มเอบีอาร์สามารถการรับส่งข้อมูลให้สูงขึ้น ซึ่งเกิดจากผลกระทบของความเร็วเป็นสำคัญ

สำหรับกราฟดีเลย์ซึ่งแสดงไว้ในรูปที่ 6.14-6.17 ค่าดีเลย์มีผลกระทบจากทั้งปริมาณ Traffic และความเร็วในการเคลื่อนที่ของโมไบล์โฮสต์ ซึ่งจะเห็นความแตกต่างได้ชัดเจนจากการเปลี่ยนแปลงของ Traffic และความเร็วการเคลื่อนที่ของโมไบล์โฮสต์ ซึ่งมีแนวโน้มเป็นแบบเดียวกับการจำลองการทำงานในสภาพแวดล้อมที่ 1 คือค่าดีเลย์จะเพิ่มสูงขึ้นเมื่อความเร็วเพิ่มขึ้นและค่าดีเลย์จะสูงขึ้นเมื่อคู่การสื่อสารเพิ่มขึ้นด้วย ซึ่งจากการจำลองการทำงานชี้ให้เห็นว่า ปริมาณแพคเกจข้อมูลในครอบจาก interface queue เป็นจำนวนเกินกว่าครึ่งของแพคเกจข้อมูลที่ถูกดรอบทั้งหมด เนื่องจากโมไบล์โฮสต์ต้องตอบสนองต่อเมสเสจร้องขอเส้นทางซึ่งมีความสำคัญมากกว่าแพคเกจข้อมูล เมื่อมีปัจจัยด้านความเร็วและคู่การสื่อสารที่สูงขึ้นทำให้การรับส่งข้อมูลภายในโมไบล์โฮสต์เกิดการหยุดชะงักเพื่อค้นหาเส้นทางทดแทน และเป็นสาเหตุให้แพคเกจข้อมูลดรอบออกจาก interface queue ทำให้ต้องมีการ retransmission แพคเกจข้อมูลซ้ำซึ่งเป็นกระบวนการที่ก่อให้เกิดดีเลย์สูงขึ้นอย่างมาก

รูปที่ 6.18-21 แสดงภาระโหลดที่เกิดขึ้นต่อโมไบล์โฮสต์ซึ่งรวมข้อมูลและคอนโทรลโอเวอร์เฮดจากทุกๆเลเยอร์ซึ่งจะเห็นได้ว่า แนวโน้มของภาระโหลดต่อโมไบล์โฮสต์เพิ่มขึ้นอย่างรวดเร็วเมื่อมีการเคลื่อนที่ด้วยความเร็วที่สูงขึ้น จากการจำลองพบว่าปริมาณโอเวอร์เฮดที่เกิดขึ้นจากการค้นหาเส้นทางทดแทนเส้นทางที่มีความเสียหาย มีปริมาณสูงขึ้นอย่างรวดเร็ว ซึ่งใน

ความเร็วจากการทดลองที่ 80km/hr ที่จำนวนคู่การสื่อสาร 40 คู่ จะพบว่าปริมาณภาระของโมไบล์โฮสต์ถูกใช้งานเต็มแบนด์วิธ ซึ่งเป็นโอเวอร์เฮด ถึง 1.77Mbps สำหรับโปรโตคอลต้นแบบเอบีอาร์และ 1.32Mbps สำหรับโปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์ จากแบนด์วิธทั้งหมด 2Mbps แสดงให้เห็นว่าที่ความเร็วสูงและจำนวนคู่การสื่อสารที่มากนั้นส่งผลให้ปริมาณโอเวอร์เฮดที่วิ่งอยู่ในเครือข่ายอาจสูงถึงประมาณ 65-90%

รูปที่ 6.22-6.25 กราฟแสดงค่าประสิทธิภาพในการรับส่งข้อมูลซึ่งจากการจำลองการทำงานได้ส่งข้อมูลทดสอบเป็นอัตราคงที่ 2 แพคเกจต่อวินาทีในแต่ละคู่การสื่อสาร กราฟที่ได้จึงแสดงความสามารถในการรับส่งแพคเกจข้อมูลในเครือข่ายซึ่งเห็นได้ว่ามีความแตกต่างกันไม่มากนักระหว่างโปรโตคอลเอบีอาร์และโปรโตคอลเอ็มเอบีอาร์ จากความเร็วที่ 50 และ 80km/hr ที่คู่การสื่อสาร 40 คู่โปรโตคอลเอบีอาร์จะแสดงประสิทธิภาพการรับส่งข้อมูลที่ลดลงอย่างรวดเร็วจากการจำลองการทำงาน เราสังเกตได้ว่าเมื่อความเร็วหรือคู่การสื่อสารที่มากขึ้นจะส่งผลให้โปรโตคอลเอบีอาร์ถึงจุดอิ่มตัวรวดเร็วกว่าโปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์ ซึ่งหมายถึงว่าหากคู่ปริมาณการสื่อสารเพิ่มมากขึ้นหรือความเร็วเพิ่มสูงขึ้นจะทำให้ความสามารถของโปรโตคอลเอบีอาร์ลดลงอย่างรวดเร็วเมื่อถึงจุดหนึ่งเนื่องจากการอิ่มตัวของแบนด์วิธ ในขณะที่โปรโตคอลเอ็มเอบีอาร์มีการใช้ทรัพยากรได้อย่างมีประสิทธิภาพมากขึ้น ทำให้โปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์สามารถรองรับการทำงานที่ความเร็วและปริมาณคู่การสื่อสารได้มากขึ้นกว่าโปรโตคอลค้นหาเส้นทางเอบีอาร์ แต่จากการทดสอบนี้เราได้แสดงผลให้เห็นในส่วนของการจำลองการทำงานซึ่งทั้งโปรโตคอลเอ็มเอบีอาร์และเอบีอาร์ยังทำงานได้โดยที่ยังไม่ถึงจุดอิ่มตัว ซึ่งจะเห็นได้ว่าสภาวะการทำงานโดยทั่วไปโปรโตคอลเอ็มเอบีอาร์ยังสามารถปรับปรุงประสิทธิภาพในการรับส่งข้อมูลเมื่อเทียบกับโปรโตคอลเอบีอาร์ได้แต่ยังไม่มากนัก

6.3.3 สรุปผลการวิเคราะห์ที่ได้จากสภาพแวดล้อมการจำลองการทำงานทั้ง 2 สภาพแวดล้อม

ผลที่ได้จากการจำลองการทำงานทั้งสองการทดลอง ทำให้ทราบว่าโปรโตคอลที่ได้ทำการปรับปรุง เอ็มเอบีอาร์สามารถทำงานได้ดีขึ้นเมื่ออยู่ในสภาพการเคลื่อนที่ และรองรับปริมาณคู่การสื่อสารได้มากขึ้นเมื่อเทียบกับโปรโตคอลค้นหาเส้นทางเอบีอาร์ดังแสดงให้เห็นจากผลประสิทธิภาพในแง่ของโอเวอร์เฮด ดีเลย์ อัตราการส่งข้อมูล ประสิทธิภาพการส่งข้อมูล และปริมาณภาระที่เกิดขึ้นภายในโมไบล์โฮสต์

ในการพิจารณาการทำงานของโปรโตคอลเอ็มเอบีอาร์พบว่า เอ็มเอบีอาร์สามารถทำงานได้ดีขึ้นเมื่อปริมาณจำนวนโมไบล์โฮสต์ต่อพื้นที่มีค่าสูง เนื่องจากการส่งเมสเสจร้องขอไปยังทิศทางตำแหน่งโมไบล์โฮสต์ปลายทาง หากภายในเครือข่ายมีโมไบล์โฮสต์น้อยจะทำให้จำนวนเส้นทางที่ได้มีจำนวนน้อย และนอกจากนี้ยังอาจก่อให้เกิดปัญหาการค้นหาเส้นทางเดียวกับโปรโตคอลค้นหาเส้นทางกรีดี Greedy ในบทที่ 3 รูปที่ 3.6 และทำให้เกิดดีเลย์ที่สูงขึ้นเนื่องจาก

การค้นหาเส้นทางในพื้นที่การส่งต่อเมสเสจร้องขอเส้นทางไม่มีเส้นทาง หรือเป็นเส้นทางที่ไม่เหมาะสม

บทที่ 7
สรุปผลการวิจัย และข้อเสนอแนะ

ปัจจุบันเครื่องมือสื่อสารข้อมูลในแบบไร้สายเป็นสิ่งที่อำนวยความสะดวกเป็นอย่างมาก และด้วยราคาที่ลดลงของเทคโนโลยีทางด้านการผลิตทำให้ความนิยมใช้เครือข่ายไร้สายเพิ่มมากขึ้น สำหรับเครือข่ายไร้สายแอดฮอคเป็นอีกหนึ่งทางเลือกที่ให้ความสะดวกเนื่องจากความอิสระในการเคลื่อนที่ของโฮสต์ที่อยู่ในเครือข่าย และสามารถนำไปใช้ในพื้นที่โดยไม่ต้องทำการติดตั้งสถานีฐาน เช่นงานด้านทางทหาร การสำรวจ ซึ่งพื้นที่การใช้งานไม่เหมาะสมหรือไม่สะดวกต่อการติดตั้งสถานีฐานได้

เนื่องจากโปรโตคอลค้นหาเส้นทางเป็นโปรโตคอลที่เสนอขึ้นโดยแนวคิดของการลดโอเวอร์เฮดและดีเลย์ที่เป็นผลเกิดจากกระบวนการค้นหาเส้นทาง โดยเลือกเส้นทางที่เป็นเส้นทางมีเสถียรภาพสูง หากเส้นทางใดมีค่าความสัมพันธ์ของโมไบล์โฮสต์ต่างๆในเส้นทางมาก จะทำให้เส้นทางนี้เป็นเส้นทางที่คาดหวังว่าคือเส้นทางที่อยู่ได้นาน เป็นการเลือกเส้นทางที่มีอัตราการเสียหายของเส้นทางต่ำ เส้นทางที่ได้จึงอาจจะไม่ใช่เส้นทางที่สั้นที่สุด แต่อยู่ได้นาน ข้อด้อยสำหรับโปรโตคอลค้นหาเส้นทางเอบีอาร์คือการบรอดคาสต์เพื่อค้นหาเส้นทาง เมื่อโปรโตคอลทำงานอยู่ในเครือข่ายที่มีการเคลื่อนที่ความเร็วสูง ความน่าจะเป็นที่เส้นทางจะเสียหายเนื่องจากการเคลื่อนที่ของโมไบล์โฮสต์ต่างๆในเครือข่าย จะเพิ่มสูงขึ้น การบรอดคาสต์เพื่อค้นหาเส้นทางทดแทนเส้นทางเดิม หรือการค้นหาเส้นทางใหม่ จะก่อให้เกิดโอเวอร์เฮดต่อเครือข่ายจำนวนมาก ซึ่งเมื่อโมไบล์โฮสต์เคลื่อนที่ด้วยความเร็วสูงมากค่าหนึ่ง เราจะพบว่าแพคเกจที่วิ่งอยู่ในเครือข่ายเกือบทั้งหมดเป็นแพคเกจที่ส่งมาเพื่อค้นหาเส้นทางเป็นส่วนใหญ่ การลดพื้นที่การบรอดคาสต์เป็นการช่วยลดปริมาณแพคเกจเพื่อการค้นหาเส้นทางได้เป็นอย่างดี

ในวิทยานิพนธ์ฉบับนี้ได้นำเสนอการปรับปรุงโปรโตคอลค้นหาเส้นทางเอบีอาร์โดยใช้ข้อมูลตำแหน่งเพื่อช่วยให้การบรอดคาสต์เมสเสจร้องขอเส้นทางเป็นไปในพื้นที่จำกัดที่กำหนดไว้ตามแนวคิดเพื่อลดพื้นที่การบรอดคาสต์จากโปรโตคอลค้นหาเส้นทางแอลเออาร์ โดยสมมติว่าโมไบล์โฮสต์ทุกตัวในเครือข่ายทราบข้อมูลตำแหน่งของตนเอง และสามารถประมาณค่าความเร็วเฉลี่ยหรือความเร็วสูงสุดของโมไบล์โฮสต์ในเครือข่ายได้ เมื่อนำแนวคิดของเอบีอาร์ที่เลือกเส้นทางจากความสัมพันธ์ระหว่างโมไบล์โฮสต์และการจำกัดการบรอดคาสต์เมสเสจร้องขอเส้นทาง ทำให้โปรโตคอลเอ็มเอบีอาร์สามารถจำกัดพื้นที่เพื่อลดโอเวอร์เฮดจากการบรอดคาสต์ค้นหาเส้นทางรวมถึงลดโอเวอร์เฮดที่จะเกิดจากปัญหาเส้นทางเสียหาย เนื่องจากโปรโตคอลเอบีอาร์มีการเลือกเส้นทางที่คาดหวังว่าจะอยู่ได้นาน หลังจากการปรับปรุงโปรโตคอลค้นหาเส้นทางเอบีอาร์ซึ่งใช้ชื่อว่า โปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์ ได้วัดประสิทธิภาพการทำงานของเครือข่ายเปรียบเทียบกับโปรโตคอลเดิม พบว่าค่าพารามิเตอร์ประสิทธิภาพ โปรโตคอลเอบีอาร์

สามารถลดค่าโอเวอร์เฮดจากการค้นหาเส้นทาง สามารถลดดีเลย์ในการส่งข้อมูล ลดปริมาณภาระของโมไบล์โฮสต์ต่างๆในเครือข่ายให้ลดลง และสามารถเพิ่มอัตราการรับส่งข้อมูลรวมถึงเพิ่มค่าประสิทธิภาพในการใช้แบนด์วิธเพื่อการรับส่งข้อมูลได้ดียิ่งขึ้น นอกจากนี้ยังพบว่าโปรโตคอลเอ็มเอบีอาร์จะเหมาะสมกับเครือข่ายที่มีการเคลื่อนที่ของโมไบล์โฮสต์บ่อยครั้งและเหมาะสมสำหรับเครือข่ายที่มีโมไบล์โฮสต์ต่อพื้นที่หนาแน่นซึ่งสามารถลดโอเวอร์เฮดของระบบได้อย่างมาก รวมถึงสามารถรองรับจำนวนคู่การสื่อสารข้อมูลได้มากขึ้นเมื่อเทียบกับโปรโตคอลค้นหาเส้นทางเอบีอาร์เดิม

วิธีการที่นำเสนอในวิทยานิพนธ์เป็นวิธีในการค้นหาเส้นทางเพื่อจำกัดบริเวณการค้นหาเส้นทางให้อยู่เฉพาะในพื้นที่จำกัด แต่เนื่องจากการจำกัดพื้นที่ดังกล่าว ทำให้คุณภาพของเส้นทางที่ได้เมื่อเทียบกับโปรโตคอลเอบีอาร์เดิมนั้นมีค่าลดลง เนื่องจากการจำกัดพื้นที่ ทำให้เส้นทางอื่นๆที่เหมาะสมกว่าแต่อยู่ภายนอกพื้นที่ซึ่งได้จำกัดไว้ โปรโตคอลจึงเลือกเส้นทางที่ดีที่สุดในพื้นที่เฉพาะที่กำหนดไว้ได้เท่านั้น ทำให้ต้องมีการค้นหาเส้นทางบ่อยครั้งกว่าโปรโตคอลต้นแบบโปรโตคอลเอ็มเอบีอาร์จึงให้ผลของประสิทธิภาพลดลง เมื่อถูกนำไปใช้กับการทำงานของเครือข่ายที่มีความเร็วต่ำหรือจำนวนโมไบล์โฮสต์ต่อพื้นที่มีจำนวนเบาบาง

บรรณานุกรม

[1] R.E. Bellman, **Dynamic Programming**, Princeton: Princeton Univ. Press 1957.

[2] C.E. Perkins and P.Bhagwat, "**Highly Dynamic Destination-Sequenced Distance Vector Routing (DSDV) for Mobile Computers**," Proc. ACM SIGCOMM'94, London, U.K., Sep. 1994, pp. 234-44.

[3] S. Basagni, I. Chalamtac, V.R. Syrotiuk, and B.A. Woodward, "**A distance Routing effect algorithm for mobility (DREAM)**," in Proceeding of the AMC/IEEE International Comference on Mobile Computing and Networking (Mobicom), 1998, pp. 76-84.

[4] P. Jacquet *et al.*, "**Optimized Link State Routing Protocol**," draft-ietf-manetolsr- 05.txt, Internet Draft, IETF MANET Working Group, Nov. 2000.

[5] C. E. Perkins, E. M. Royer, and S. Das, **Ad Hoc On Demand Distnace Vector (AODV) Routing**, IETF RFC, No. 3561, Jul. 2003.

[6] D. B. Johnson, D. A. Maltz, and Y.-C. Hu, **The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks**, IETF Internet-Draft, draft-ietfmanet-dsr-00.txt, Jul. 2004.

[7] V. D. Park and M.S. Corson, "**A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks**," *Proc. IEEE INFOCOM '97*, Kobe, Japan, Apr. 1997, pp. 1405–13.

[8] C.-K. Toh, "**Associativity-Based Routing For Ad Hoc Mobile Networks**," *WL Pers. Commun. J.*, Special Issue on *Mobile Networking and Computing Systems*, Kluwer, vol. 4, no. 2, Mar. 1997, pp. 103–39.

[9] Y.-B. Ko and N. H. Vaidya, "**Location-aided Routing(LAR) in Mobile Ad Hoc Networks**," *ACM/IEEE Int'l. Conf. Mobile Comp. Net.*, 1998, pp. 66–75.

[10] J. Li *et al.*, "**A Scalable Location Service for Geographic Ad Hoc Routing**," *Proc. 6th Annual ACM/IEEE Int'l. Conf. Mobile Comp. Net.*, Boston, MA, 2000, pp. 120–30.

[11] Z. J. Haas and B. Liang, "**Ad Hoc Mobility Management with Uniform Quorum Systems**," *IEEE/ACM Trans. Net.*, vol. 7, no. 2, Apr. 1999, pp. 228–40.

[12] I. Stojmenovic, "**Home Agent Based Location Update and Destination Search Schemes in Ad Hoc Wireless Networks**," Tech. rep. TR-99-10, Comp. Science, SITE, Unive. Ottawa, Sept. 1999.

[13] B. Karp and H. T. Kung, "**Greedy Perimeter Stateless Routing for Wireless Networks**," *Proc. 6th Annual ACM/IEEE Int'l. Conf. Mobile Comp. Net.*,Boston, MA, Aug. 2000, pp. 243–54.

[14] L. Blazevic *et al.*, "**Self-Organization in Mobile Ad Hoc Networks: The Approach of TermiNodes**," *IEEE Commun. Mag.*, 2001.

[15] The grid project homepage, http://www.pdos.lcs.mit.edu/grid

[16] S. McCanne and S. Floyd. (2003, December) ns2 network simulator 2. [Online]. Available: http://www.isi.edu/nsnam/ns

[17] UCLA Parallel Comp. Lab. and Wireless Adaptive Mobility Lab., "**GloMoSim: A Scalable Simulation Environment for Wireless and Wired Network System,**" http://pcl.cs.ucla/edu/projects/domains/glomosim.html

[18] OPNET Contributed Model Depot: http://www.opnet.com/services/depot/home.html

[19] Qualnet simulator, available at http://www.qualnet.com

[20] M. Kummakasikit, S. Thipchaksurat, R. Varakulsiripunth, "**Modification of Associativity -Based Routing Protocol Based on Mobile Host's Location in Ad-Hoc Wireless Networks,**" The 9th Nation Computer Science and Engineering Conference (NCSEC'2005), pp. 21-29, University of Thai Chamber of Commerce, Bangkok Thailand, October 27-28, 2005.

[21] M. Kummakasikit, S. Thipchaksurat, R. Varakulsiripunth, "**Performance Improvement of Associativity-Based Routing Protocol for Mobile Ad-Hoc Networks,**" Fifth International Conference on Information, Communications and Signal Processing, The Grand Hotel, Bangkok, Thailand, December 6-9, 2005.

ภาคผนวก

ภาคผนวก ก.

การจำลองการทำงาน

ในส่วนนี้จะกล่าวถึงการจำลองการทำงานของโปรแกรมจำลองการทำงานที่ถูกนำมาใช้ในงานวิจัยซึ่งแสดงรายละเอียดของโปรแกรมจำลองการทำงานที่ได้อิมพลีเมนต์โปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์

- **การจำลองการทำงานของซิมูเลเตอร์**

ซิมูเลเตอร์หรือโปรแกรมจำลองการทำงานในปัจจุบันส่วนใหญ่จะเป็นลักษณะ Discrete Event และเป็นการทำงานโดยใช้ลักษณะของอีเวนท์เป็นตัวกำหนดลักษณะของการทำงานหรือที่เรียกว่าเป็น Event Driven นอกจากการจำลองการทำงานที่สามารถนำมาวิเคราะห์ได้อย่างง่ายดาย ยังมีเครื่องมืออื่นๆ ไม่ว่าจะแสดงภาพ อนิเมชันหรือคำนวณค่าพารามิเตอร์ที่ใช้วัดประสิทธิภาพของเครือข่าย

- **ชนิดของซิมูเลเตอร์**

เนทเวิร์คซิมูเลเตอร์หรือโปรแกรมเพื่อการจำลองการทำงานของเครือข่ายในปัจจุบันจะเป็นแบบ Discrete Event สามารถจำลองเลเยอร์ต่างๆพร้อมทั้งอิมพลีเมนต์โปรโตคอลต่างๆลงในแต่ละเลเยอร์ OSI

- **NS-2 (Network Simulator 2) [16]**

NS-2 เป็นโปรแกรมจำลองการทำงานแบบ Discrete Event พัฒนาโดยมหาวิทยาลัยเคลิฟอเนีย เบิร์กเลย์ ซึ่งสามารถรองรับรูปแบบการจำลองการทำงานได้หลากหลายเนื่องจากมีโปรโตคอลหลายหลายถูกจำลองขึ้นบน OSI เลเยอร์ และยังให้ผู้ใช้สามารถเพิ่มเติมโปรโตคอลในแบบที่ต้องการเข้าไปได้โดยการสร้างเอเจนท์ (Agent) ซึ่งเป็นคลาสถ่ายทอดมาจากคลาสหลักที่มีให้ใน NS-2 ปัจจุบันสามารถนำมาจำลองเครือข่ายได้ทั้งแบบไร้สายและแบบเครือข่ายสายทั่วไป ซึ่งถูกสร้างขึ้นจากภาษา C++ โดยใช้ OTCL เป็นฟรอนต์แอนด์ ผลที่ได้ของการจำลองการทำงานไม่ว่าจะเป็นการเคลื่อนที่ของโมไบล์โฮสต์ การเดินทางของแพคเกจต่างๆ และผลลัพธ์ในรูปอื่นๆ ที่ได้จากการทดลอง จะถูกเก็บลงในไฟล์ที่เรียกว่าเป็นเทรคซ์ไฟล์ (Trace File) นอกจากนั้น NS-2 ยังมีเครื่องมือที่ช่วยอำนวยความสะดวกเพื่อวิเคราะห์ผลของการจำลองการทำงานเช่น Xgraph ใช้ในการสร้างกราฟผลของการจำลองการทำงาน NAM หรือ Network Animator เป็นโปรแกรมที่บันทึกผลรวมถึงลักษณะทางกายภาพของเครือข่ายที่สามารถนำผลของการจำลองการทำงานมาแสดงให้เห็นแบบอนิเมชันบนหน้าจอ

- **GloMoSim [17]**

GloMoSim เป็นโปรแกรมการจำลองการทำงานอีกประเภทซึ่งถูกนำมาใช้ในการจำลองการทำงานของเครือข่ายไร้สาย พัฒนาโดยมหาวิทยาลัยเคลิฟอเนีย ลอสเองเจลิส ซึ่งได้จำลองโปรโตคอลต่างๆในแต่ละเลเยอร์เอาไว้ ให้ผู้ใช้สามารถเลือกใช้โปรโตคอลต่างๆในแต่ละเลเยอร์ได้ นอกจากนี้โกลโมซิมยังมี API ซึ่งช่วยให้ผู้ใช้สามารถเพิ่มโปรโตคอลในแบบที่ต้องการเข้าไปในโปรแกรมได้อย่างง่าย พารามิเตอร์ที่ใช้กำหนดสภาพแวดล้อมของเครือข่าย ลักษณะข้อมูลในเครือข่ายและโปรโตคอลที่ใช้สามารถกำหนดลงในคอนฟิกูเรชันไฟล์ โดยมีอนิเมชันแสดงสถานะการจำลองการทำงานที่สามารถตรวจสอบหรือสั่งพักชั่วคราวหรือสั่งเริ่มต้นการจำลองการทำงานใหม่โดยการใช้เครื่องมือจำลองที่เขียนขึ้นโดยภาษาจาวา หลังจากการจำลองการทำงานสิ้นสุดลง ผลของการจำลองการทำงานจะแสดงเป็นลักษณะของข้อมูลตัวเลขของค่าพารามิเตอร์ต่างๆ

- **OPNET [18]**

OPNET Modeler เป็นโปรแกรมซิมูเลเตอร์ที่พัฒนาโดยสถาบันเทคโนโลยีแมสซาชูเสท โปรแกรม OPNET สามารถออกแบบและจำลองการทำงานของเครือข่าย อุปกรณ์และโปรโตคอลต่างๆ ผลของโปรแกรมจะสามารถแสดงสถานะของการจำลองการทำงานแบบอนิเมชัน กราฟของการจำลองการทำงาน แต่การคอนฟิกูเรชันเพื่อกำหนดสภาวะแวดล้อมเป็นไปอย่างซับซ้อน

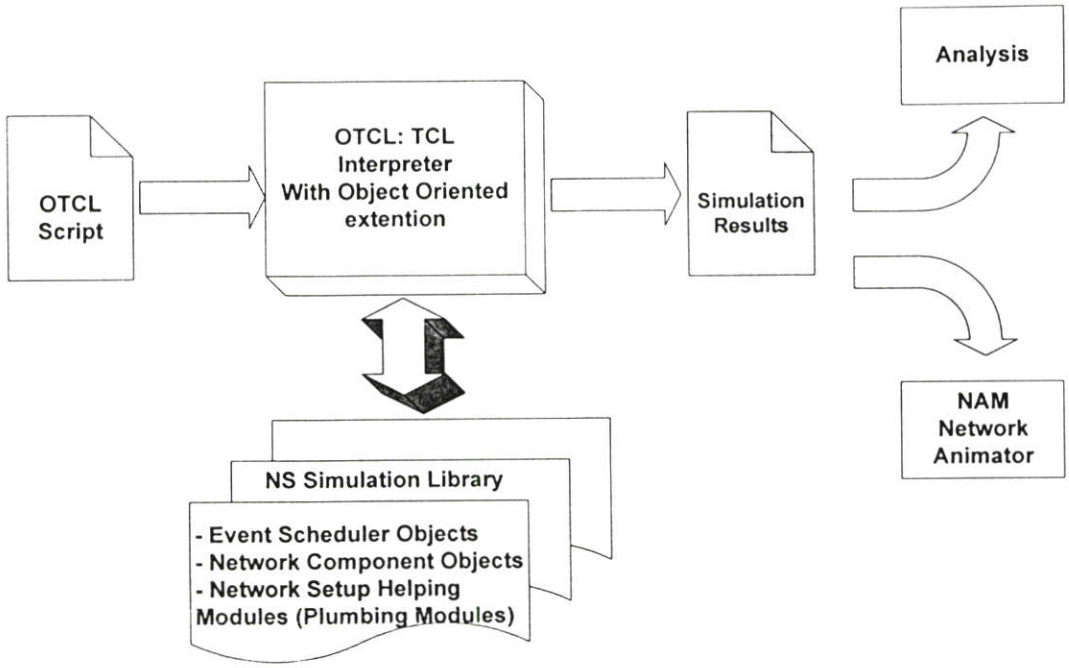
- **Qualnet [19]**

QualNet เป็นเครื่องมือสำหรับเครือข่ายทั้งแบบสายและไร้สาย เป็นโปรแกรมจำลองการทำงานแบบคอมเมอเชียลที่อ้างว่าสามารถทำงานได้เร็วที่สุดสำหรับการออกแบบ Realtime QualNet Animator เป็นส่วนของการแสดงภาพการจำลองการทำงานแบบกราฟฟิก QualNet Designer เป็นเครื่องมือที่สามารถสร้างออโตเมต้าแบบจำกัดสถานะที่ใช้ในการอธิบายคุณลักษณะของเครือข่าย สามารถทำงานได้ทั้งในระบบปฏิบัติการวินโดวส์และลินุกซ์

- **ซิมูเลเตอร์เอ็นเอสทู (NS-2 Simulator)**

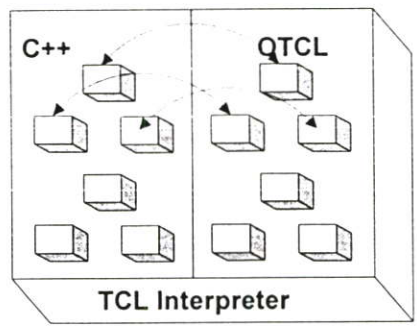
เอ็นเอส (Network Simulator: NS) เป็นโปรแกรมจำลองการทำงานที่พัฒนาขึ้นเพื่อใช้ในการ จำลองการทำงานของเครือข่าย ในรูปแบบที่หลากหลาย โดยได้มีการอิมพลีเมนต์โปรโตคอลต่างๆสำหรับเครือข่ายเช่น TCP, UDP, FTP, Telnet, Web, CBR queue ของเราเตอร์เช่น Drop tail, RED, CBQ อัลกอริทึมการค้นหาเส้นทางเช่น Dijkstra เป็นต้น นอกจากนี้เอ็นเอสยังอิมพลีเมนต์การทำมัลติคาสต์และเพิ่มโปรโตคอลในชั้นเลเยอร์ MAC สำหรับการจำลองการทำงานเครือข่ายภายใน (Local Area Networks: LAN) ปัจจุบันเอ็นเอสเป็นส่วนหนึ่งของโปรเจค VINT ซึ่งมีการ

พัฒนาเครื่องมือในการจำลองการทำงานของเครือข่าย ให้สามารถแสดงผลการจำลองการทำงานแบบอนิเมชัน รวมถึงเครื่องมือที่ใช้เพื่อวิเคราะห์ข้อมูลจากผลของการจำลองการทำงาน ปัจจุบันเอ็นเอสเวอชัน 2 ได้ถูกสร้างขึ้นโดยใช้ภาษา C++ และ OTCL ซึ่งภาษาทีซีแอล มีความสามารถในการทำงานแบบเชิงวัตถุ (Object-Oriented) ใช้เป็นฟรอนท์เอ็น เอ็นเอสเป็นอินเทอพรีเตอร์ (Interpreter) ซึ่งมีการจำลองการทำงานแบบตามลำดับอีเวนท์ (Event) รูปที่ ก.1 เอ็นเอสจะเป็นโปรแกรมแบบเชิงวัตถุโดยผ่านอินเทอรพรีเตอร์ทีซีแอล ซึ่งมีการเรียงลำดับอีเวนท์ ไลบรารีของอุปกรณ์เครือข่ายถูกใช้ในการกำหนดลักษณะเครือข่าย (Plumbing) ในการจำลองการทำงานโดยการจำลองการทำงานเริ่มจากการสร้างสคริปต์ทีซีแอลเพื่อกำหนดลักษณะของเครือข่าย ผู้ใช้จะเขียนสคริปต์เพื่อให้เกิดอีเวนท์ และระบุลักษณะโครงสร้างเครือข่ายโดยการใช้ออบเจ็คต่างๆของเครือข่ายที่มีอยู่ในเอ็นเอส และระบุถึงแหล่งข้อมูลต้นทาง เมื่อมีการเริ่มหรือหยุดเหตุการรับส่งข้อมูลโดยผ่านตัวจัดลำดับอีเวนท์(Event Scheduler) ในการกำหนดลักษณะต่างๆของเครือข่ายหรือ Plumbing จะเป็นการกำหนดค่าต่างๆในเครือข่ายซึ่งจะหาระยะเส้นทางระหว่างออบเจ็คในเครือข่ายด้วยวิธีการใช้พอยเตอร์ของออบเจ็คที่อยู่ใกล้เคียงซึ่งชี้ไปยังออบเจ็คที่เหมาะสม เมื่อผู้ใช้ต้องการสร้างออบเจ็คใหม่ขึ้นในเครือข่ายจะสามารถสร้างได้ทั้งการเขียนออบเจ็คใหม่จากการรวมออบเจ็คหลายออบเจคจากไลบรารีที่มีให้เข้าด้วยกันและหาเส้นทางสื่อสารระหว่างออบเจ็คต่างๆ ซึ่งขั้นตอนการหาเส้นทางข้อมูลต่างๆระหว่างออบเจ็คทั้งหมดค่อนข้างเป็นวิธีการที่ยุ่งยาก แต่กระบวนการทั้งหมดจะทำผ่านโมดูลโอทีซีแอลซึ่งทำให้ขั้นตอนง่ายขึ้น แต่ละอีเวนท์ที่ถูกนำไปจัดเรียงในตัวจัดการอีเวนท์นั้นจะเป็นไอดีของแพคเกจซึ่งแบ่งแยกความแตกต่างของแพคเกจต่างๆด้วยเวลาที่จัดเรียงและพอยน์เตอร์ที่ชี้ไปยังไปยังออบเจ็คที่สามารถสนองตอบต่อชนิดของอีเวนท์นั้นๆ ตัวจัดการอีเวนท์ในเอ็นเอสจะมีการติดตามเวลาการจำลองการทำงานและเลือกอีเวนท์ที่อยู่ในลำดับอีเวนท์เพื่อเรียกไปยังส่วนประกอบอื่นๆของเครือข่ายและฟังชันงาน ที่สามารถตอบสนองกับอีเวนท์ต่อแพคเกจ ส่วนประกอบต่างๆในเครือข่ายจะสื่อสารกันโดยมีการส่งผ่านข้อมูลเป็นแพคเกจโดยไม่ส่งผลต่อเวลาในการจำลองการทำงาน(Simulation Time) แต่ละอุปกรณ์ในเครือข่ายจำเป็นต้องใช้เวลาส่วนหนึ่งไปในการตอบสนองต่อแพคเกจ เช่นในรูปของดีเลย์ที่เกิดขึ้น โดยเวลานี้ทำได้โดยการที่ตัวจัดการอีเวนท์ได้รับอีเวนท์ล่วงหน้าและรอจนกระทั่งอีเวนท์ถูกส่งย้อนกลับมายังตัวจัดการอีเวนท์ ก่อนที่ส่งจะไปยังกระบวนการตอบสนองต่อแพคเกจต่อไป ตัวอย่างเช่น เนทเวิร์คสวิชต์ใช้เวลาในการจำลองการทำงานเป็นดีเลย์ 20ไมโครวินาที เนทเวิร์คสวิชต์จะส่งอีเวนท์ย้อนกลับไปยังตัวจัดการอีเวนท์ที่มีเวลาเป็น 20ไมโครวินาทีถัดไป หลังจาก 20 ไมโครวินาทีผ่านไป ตัวจัดการอีเวนท์จะดึงอีเวนท์ส่งไปยังเนทเวิร์คสวิชต์อีกครั้งและทำให้แพคเกจถูกเราท์ไปยังลิงค์เอาท์พุตที่เหมาะสม นอกจากนี้ตัวจัดการอีเวนท์ยังจัดการเกี่ยวกับตัวนับเวลา (Timer) ในวิธีเดียวกับดีเลย์จะต่างกันเพียงเวลาจะไม่ขึ้นกับอุปกรณ์ใดในเครือข่ายแต่จะพิจารณาเวลาของแพคเกจเป็นหลัก ตัวอย่างเช่นส่วน Retransmission ของโปรโตคอล TCP ที่จำเป็นต้องมีตัวนับเวลาเพื่อกำหนดเวลา Time-Out ในการส่งแพค



รูปที่ ก.1 ภาพโครงสร้างโปรแกรมจำลองการทำงานเอ็นเอส

เอ็นเอสถูกสร้างขึ้นจากภาษา C++ และ OTCL โดยได้แยกส่วนที่เกี่ยวกับดาต้าออกจากส่วนของการควบคุม เพื่อจะลดเวลาแพคเกจและปริมาณอีเวนท์ที่ใช้ในการประมวลผล (Processing Time) ซึ่งจะต่างจากเวลาจำลองการทำงาน (Simulation Time) ตัวจัดการอีเวนท์และออบเจ็คต่างๆในเครือข่ายถูกสร้างและคอมไพล์ด้วยภาษา C++ หลังจากการคอมไพล์ออบเจ็คเหล่านี้จะถูกควบคุม



รูปที่ ก.2 ภาพการแมพออบเจ็คระหว่างภาษา C++ และ OTCL

โดยการแมพกันโดย OTCL Linkage ทำให้สามารถแมพออบเจ็คระหว่างโอทีซีแอลออบเจ็คและ C++ ออบเจ็คได้ ทำให้โอทีซีแอลสามารถเรียกใช้ฟังก์ชันต่างๆหรือตั้งค่า แก้ไขค่าใดๆบางอย่างในออบเจค C++ ได้ ดังในรูปที่ ก.2 ได้แสดงให้เห็นลักษะโครงสร้างซึ่งสามาถแมพออบเจคเข้าหากันระหว่างออบเจคที่คอมไพล์แล้วซึ่งถูกเขียนขึ้นจาก C++ ไปยัง ออบเจคที่อยู่ในด้านอินเทอร์พรีเตอร์โอทีซีแอลได้

- **ภาษาโอทีซีแอล (OTCL Language)**

โอทีซีแอลเป็นอินเทอร์พรีเตอร์ที่ถูกใช้เป็นฟรอนต์เอนด์เพื่อใช้ในการควบคุมการจำลองการทำงาน ในส่วนนี้จะแสดงให้เห็นถึงลักษณะของภาษาโอทีซีแอลที่นำมาใช้ในเอ็นเอส ในตัวอย่างรูปที่ ก.3 แสดงสคริปต์ภาษาโอทีซีแอลซึ่งสร้างฟังชัน (Procedure) และทดสอบการเรียกฟังก์ชัน

```
# Writing a procedure called "test"
proc test {} {
    set a 43
    set b 27
    set c [expr $a + $b]
    set d [expr [expr $a - $b] * $c]
    for {set k 0} {$k < 10} {incr k} {
        if {$k < 5} {
            puts "k < 5, pow = [expr pow($d, $k)]"
        } else {
            puts "k >= 5, mod = [expr $d % $k]"
        }
    }
}

# Calling the "test" procedure created above
test
```

รูปที่ ก.3 ตัวอย่างที่ 1 ภาษาโอทีซีแอล

ในภาษาโอทีซีแอลจะใช้คำสั่ง proc ในการกำหนดฟังชัน ตามด้วยชื่อของฟังก์ชันงานและสามารถใส่อากิวเมนต์ (argument) ใดๆลงในปีกกาเพื่อเข้าไปคำนวณยังฟังก์ชัน set แทนด้วยการกำหนดค่าให้กับตัวแปร ตามด้วยชื่อตัวแปรและค่าที่ต้องการกำหนดลงในตัวแปร [expr ...] เป็นการให้อินเทอร์พรีเตอร์คำนวณสมการที่อยู่ข้างใน สังเกตว่าการเรียกใช้ตัวแปรหลังจากการกำหนดค่าจะเห็นได้ว่ามีการใส่เครื่องหมายดอลล่าห์ไซน์ $ นำหน้าชื่อของตัวแปรเสมอ ยกเว้นคำสั่งในการกำหนดค่า คำสั่ง puts ใช้ในการแสดงผลที่ต้องการออกในหน้าจอซึ่งตามด้วยข้อความหรือ ค่าที่ต้องการแสดงผลอยู่ภายในเครื่องหมาย " " ซึ่งในตัวอย่างแรกจะได้ผลลัพท์เป็น

k < 5, pow = 1.0

k < 5, pow = 1120.0

k < 5, pow = 1254400.0

k < 5, pow = 1404928000.0

k < 5, pow = 1573519360000.0

k >= 5, mod = 0

k >= 5, mod = 4

k >= 5, mod = 0

k >= 5, mod = 0

k >= 5, mod = 4

ในตัวอย่างที่สองจะแสดงการสร้างสคริปต์ให้เป็นแบบเชิงวัตถุ ซึ่งได้แสดงให้เห็นวิธีการสร้างออบเจ็คโดยโอทีซีแอลซึ่งออบเจ็ค ดังในรูปที่ ก.4ในตัวอย่างที่สอง โอทีซีแอลได้สร้างออบเจ็คคลาสชื่อ mom และ kid โดย kid เป็นคลาสลูกของคลาสแม่ mom และมีฟังก์ชันสมาชิกชื่อ greet แต่ละคลาสของตัวเอง หลังจากที่คลาสถูกกำหนดแล้ว แต่ละออบเจ็คที่ถูกสร้างขึ้นแล้ว (Instance) จะประกาศตัวแปร age เป็น 45 และ 15 ให้กับ mom และ kid ตามลำดับ คำว่า Class ใช้ในการแทนความหมายเพื่อสร้างคลาสของออบเจ็คและ instproc แทนฟังก์ชันสมาชิกที่อยู่ในคลาสการถ่ายทอดคลาสทำได้โดยการประกาศ –superclass ในการกำหนดฟังก์ชันสมาชิกนั้น

```
# add a member function call "greet"
Class mom
mom instproc greet {} {
    $self instvar age_
    puts "$age_ year old mom say:
    How are you doing?"
}

# Create a child class of "mom" called "kid"
# and overide the member function "greet"
Class kid -superclass mom
kid instproc greet {} {
    $self instvar age_
    puts "$age_ year old kid say:
    What's up, dude?"
}

# Create a mom and a kid object, set each age
set a [new mom]
$a set age_ 45
set b [new kid]
$b set age_ 15

# Calling member function "greet" of each object
$a greet
$b greet
```

รูปที่ ก.4 ตัวอย่างที่ 2 ภาษาโอทีซีแอล

$self จะแทนความหมายของพอยน์เตอร์ this ในความหมายเดียวกับภาษา C++ ซึ่งชี้มาที่ออบเจ็คของตัวเอง instvar จะใช้ตรวจสอบว่ามีการใช้ชื่อตัวแปรนี้ในคลาสตนเองหรือคลาสแม่ ถ้าตัวแปรถูกกำหนดไว้ก่อนหน้าแล้วจะเป็นการอ้างถึงตัวแปรนั้น แต่ถ้าไม่เช่นนั้นจะสร้างตัวแปรใหม่ขึ้น ในการสร้างอินสแตนซ์ของออบเจ็คทำได้โดยใช้ คำสั่ง new ดังในตัวอย่างซึ่งผลที่ได้จากตัวอย่างที่สองจะเป็น

45 year old mom say:

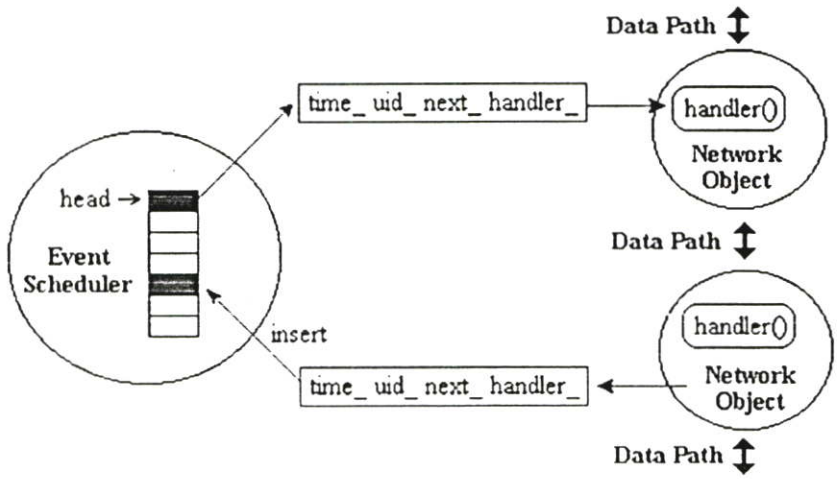
How are you doing?

15 year old kid say:

What's up, dude?

- **ตัวจัดการอีเวนท์ (Event Scheduler)**

ตัวจัดการอีเวนท์จะมีการเรียกไปยังออบเจ็คเครือข่าย (Network Objects) เพื่อให้ตอบสนองต่ออีเวนท์ที่เกิดขึ้น ในรูปที่ ก.5 ออบเจ็คจะส่งอีเวนท์มาให้กับตัวจัดการอีเวนท์ และในทางกลับกัน เมื่อถึงเวลาที่กำหนด ตัวจัดการอีเวนท์ก็จะส่งอีเวนท์ออกไปให้กับออบเจ็คเครือข่ายเพื่อให้ตอบสนองต่ออีเวนท์ที่เข้าไปในออบเจ็คได้



รูปที่ ก.5 ความสัมพันธ์ระหว่างตัวจัดการอีเวนท์กับเนทเวิร์คออบเจ็ค

สังเกตว่าเส้นทางการส่งผ่านของข้อมูล (Data Path) และเส้นทางการผ่านของอีเวนท์จะต่างกัน โดยทั่วไปแล้ว ตัวจัดการอีเวนท์จะแบ่งออกเป็นสองลักษณะคือ Realtime และ non-Realtime ในแบบที่เป็น non-Realtime จะแบ่งเป็น List, Heap และ Calender แม้ว่าการทำงานทั้งสามจะดูเหมือนกันแต่สร้างขึ้นเพื่อให้รองรับกับเอ็นเอสเวอร์ชันเก่า Calender non-Realtime จะถูกตั้งค่าให้เป็นค่าดีฟอลท์ สำหรับ Realtime จะใช้ในการทำอีมูเลชัน (Emulation) ซึ่งให้โปรแกรมสามารถจำลองการทำงานไปพร้อมกับเครือข่ายจริง สำหรับปัจจุบันการทำอีมูเลชันยังอยู่ในขั้นการพัฒนา โดยสามารถเลือกลักษณะของตัวจัดการอีเวนท์เป็นรูปแบบอื่นได้ดังนี้

```
...
set ns [new Simulator]
$ns use-Scheduler Heap
...
```

นอกจากนี้แล้วตัวจัดการอีเวนท์ยังจัดการเรื่องของเวลาในการประมวลผลการจำลองการทำงานด้วยเช่น เวาลาในการเริ่มต้นหรือหยุดการสื่อสารโปรโตคอล FTP หรือกำหนดเวลาในการจำลองการทำงาน ตัว

จัดการอีเวนท์จะมีฟังชันสมาชิกที่จัดการเช่น at time "string" เป็นการส่งอีเวนท์พิเศษที่เรียกว่า AtEvent ดังตัวอย่าง

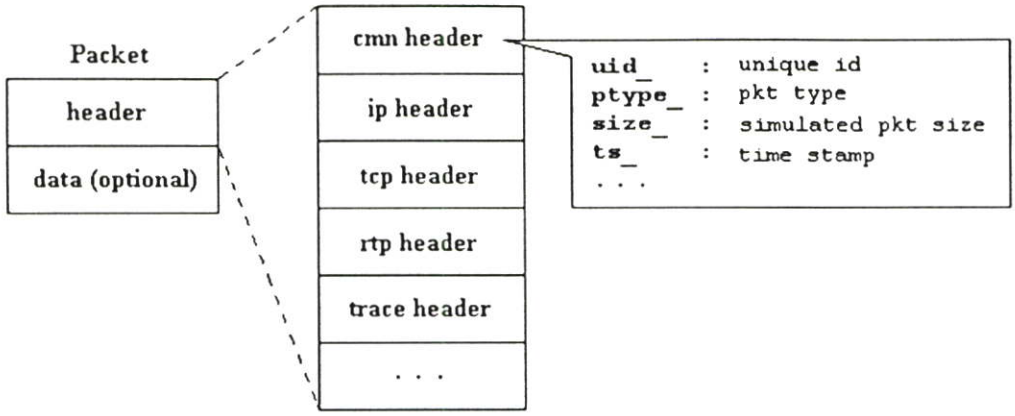
```
...
set ns [new Simulator]
$ns use-Scheduler Heap
$ns at 300.5 "complete_sim"
...
proc complete_sim{} {
        ...
}
```

ดังตัวอย่างจะเป็นการกำหนดเวลาในการจำลองการทำงาน เมื่อถึงเวลาที่ 300.5 ตัวจัดการอีเวนท์จะเรียกใช้ฟังก์ชัน complete_sim นอกจากนี้ตัวจัดการอีเวนท์ยังมีฟังก์ชันสมาชิกอื่นๆอีกดังนี้

```
Simulator instproc now          # รีเทิร์นค่าเวลาจำลองการทำงานปัจจุบัน
Simulator instproc at args      # กำหนดเวลาเพื่อเรียกฟังก์ชันมาทำงาน
Simulator instproc at-now args  # กำหนดให้เรียกฟังก์ชันมาทำงานทันที
Simulator instproc after n args # กำหนดให้เรียกฟังก์ชันมาทำงานเมื่อผ่านไป n วินาที
Simulator instproc run args     # เริ่มต้นการทำงานตัวจัดการอีเวนท์
Simulator instproc halt         # หยุดตัวจัดการอีเวนท์
```

- **แพคเกจอีเวนท์**

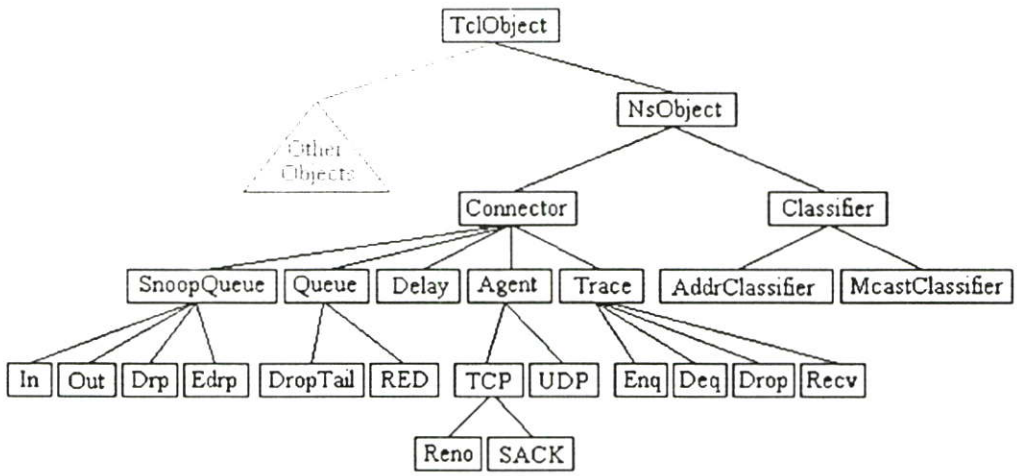
ในเอ็นเอส แพคเกจต่างๆถูกสร้างขึ้นจากการรวมกันของชั้นเฮดเดอร์ แพคเกจเฮดเดอร์ชนิดต่างๆจะถูกสร้างขึ้นหลังจากออบเจ็ค Simulator ซึ่งเฮดเดอร์เหล่านี้อาจประกอบได้ด้วยเฮดเดอร์ IP, TCP, RTP และแต่ละเฮดเดอร์จะถูกอ้างถึงโดยการใช้ค่าออฟเซต (Offset) ดังในรูปที่ ก.6 โดยทั่วไปแล้วแพคเกจประกอบด้วยส่วนของเฮดเดอร์เป็นหลักถึงแม้ว่าแพคเกจจะสามารถแนบข้อมูลลงในชั้นของดาต้า แต่โปรโตคอลต่างๆไม่ได้ถูกอิมพลีเมนต์มาให้มีการรองรับ เนื่องจากไม่มีประโยชน์ต่อการจำลองการทำงานในระบบ non-Realtime



รูปที่ ก.6 แพคเกจในเอ็นเอส

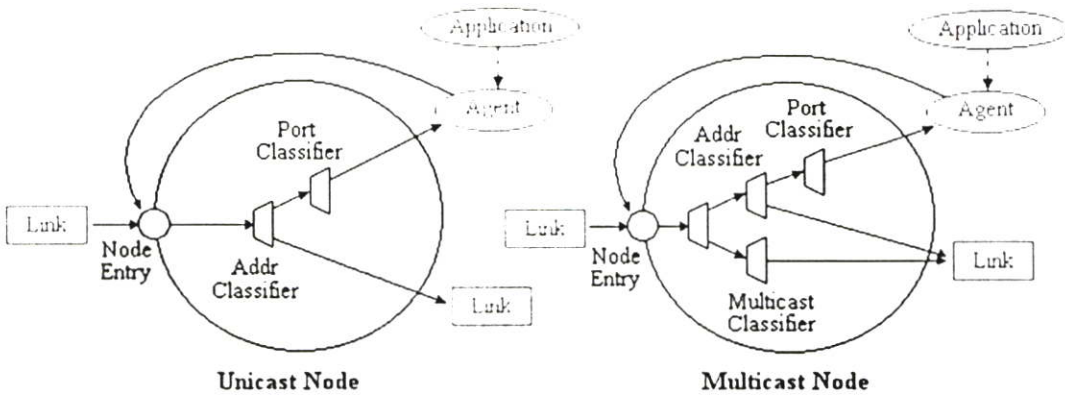
- **ส่วนประกอบในเครือข่าย (Network compenents)**

จากรูปที่ ก.7 แสดงโครงสร้างของออบเจ็คต่างๆในเอ็นเอส ที่รากของกราฟจะเป็นออบเจ็คของ TclObject ซึ่งเป็นคลาสแม่ของทุกๆคลาสในไลบราลีของโอทีซีแอลไม่ว่าจะเป็นตัวจัดการอีเวนท์ ส่วนประกอบในเครือข่าย ตัวนับเวลาและออบเจ็คต่างๆที่รวมอยู่ในโปรแกรมแสดงอนิเมชัน NAM คลาสลูกของ NsObject เป็นคลาสแม่ของส่วนประกอบต่างๆในเครือข่ายที่จะ



รูปที่ ก.7 โครงสร้างของโปรแกรมจำลองการทำงานเครือข่ายเอ็นเอส (บางส่วน)

ตอบสนองต่อแพคเกจ ซึ่งอาจจะนำมาประกอบรวมกันเป็นโหนดหรือลิงค์ โดยแบ่งเป็นคลาสย่อย Connector และ Classifier ขึ้นกับจำนวนของเอาต์พุต ออบเจ็คเครือข่ายอย่างง่ายโดยทั่วไปจะมีเพียงเอาต์พุตเดียว ซึ่งจะเป็นคลาสลูกของ Connector และออบเจ็คที่มีหลายเอาต์พุตจะเป็นคลาสลูกของคลาส Classifier



รูปที่ ก.8 โครงสร้างโหนดแบบ Unicast และ Multicast

โหนดสร้างขึ้นจากการรวมกันของออบเจ็คต่างๆประกอบด้วยออพเจ็คที่เป็นอินพุตของโหนด (Node Entry) และ Classifier ดังรูปที่ ก.8 ในเอ็นเอสปกติจะแบ่งโหนดออกเป็นสองลักษณะ Unicast โหนดประกอบด้วย Classifier ของแอดเดรสและพอร์ต สำหรับโหนดแบบ Multicast จะมี Multicast Classifier สามารถกระจายออกได้หลายเอาต์พุต โดยทั่วไปในเอ็นเอสจะเป็นโหนดแบบ Unicast ซึ่งเป็นค่าดีฟอลท์ ในการสร้างโหนดแบบมัลติคาสต์ผู้ใช้จำเป็นต้องระบุชนิดของโหนดลงในโอทีซีแอลสคริปต์ ดังนี้

Unicast

$ns rtproto type

type: Static, Session, DV, cost, multi-path

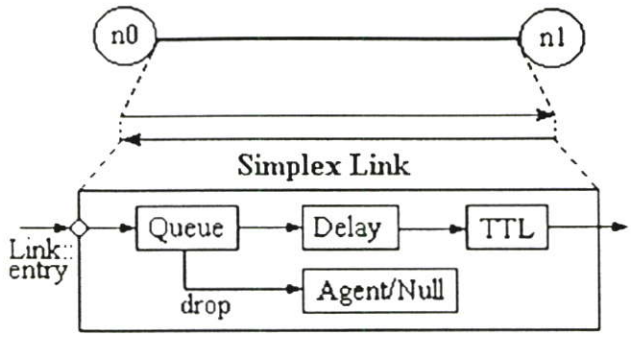
Multicast

ns Multicast (right after set $ns [new Scheduler])

$ns mrtproto type

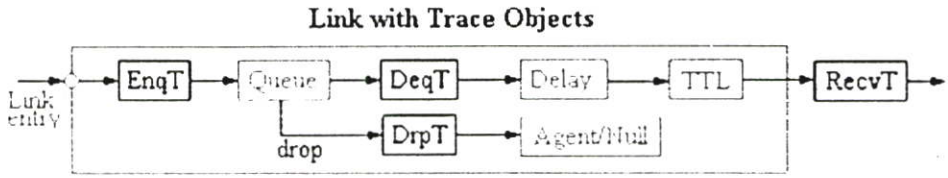
type: CtrMcast, DM, ST, BST

ลิงค์ (Link) สร้างจากการรวมกันของออบเจ็คเช่นเดียวกับโหนด ในตัวอย่างดังแสดงไว้ในรูปที่ ก.9 เป็นโครงสร้างของลิงค์ที่เป็นแบบ Duplex-Link สังเกตว่าเอาต์พุตคิวของโหนดจะถูกอิมพลีเมนต์เป็นส่วนหนึ่งของ Simplex Link แพคเกจจะถูกดึงจากคิวและถูกส่งไปยังออพเจ็ค Delay เพื่อจำลองการทำงานของดีเลย์ที่เกิดขึ้นในลิงค์ และแพคเกจที่ถูกดรอบจากคิวจะถูกส่งไปยัง Null Agent ส่วน TTL จะคำนวณค่าพารามิเตอร์ Time-to-Live ของแต่ละแพคเกจและปรับปรุงค่า TTL ของแพคเกจ



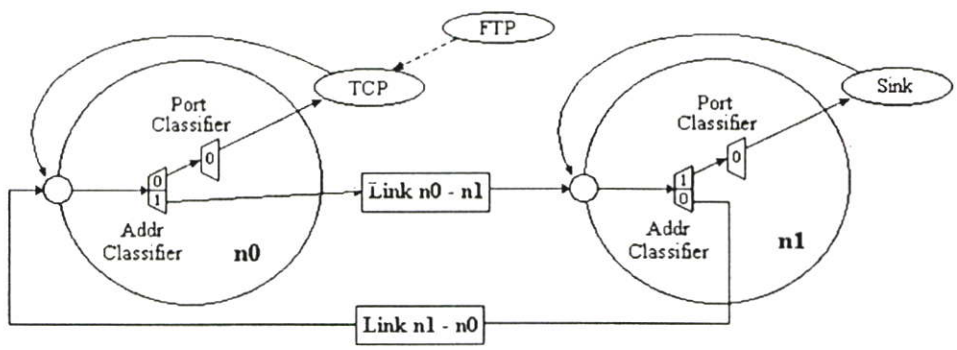
รูปที่ ก.9 โครงสร้างโหนดแบบ Unicast และ Multicast

การติดตามหรือ (Tracing) จะคั่นอยู่กับลิงค์แบบ Simplex ดังในรูปที่ ก.10 เพื่อเก็บข้อมูลที่เกิดขึ้นภายในลิงค์ (ดังที่ระบุในโอทีซีแอลสคริปต์ $ns Trace-all *File* หรือ $ns namTrace-all *File*)



รูปที่ ก.10 โครงสร้างลิงค์ที่มีออบเจ็คเทรซเพื่อเก็บข้อมูล

เมื่อเทรซออบเจ็คถูกเพิ่มเข้าไปในลิงค์เช่น (EnqT, DeqT, DrpT และ RecvT) เมื่อมีแพคเกจผ่านเข้ามาเทรซออบเจ็คจะเก็บข้อมูลบันทึกลงไฟล์โดยกระบวนการนี้ไม่ส่งผลต่อเวลาจำลองการทำงาน ตัวอย่างของการส่งผ่านแพคเกจในรูปที่ ก.11 แสดงให้เห็นการเชื่อมต่อของโหนดสองตัวเชื่อมต่อกันผ่านลิงค์ โดยสมมติโหนด n0 มีแอดเดรสคือ 0 และโหนด n1 มีแอดเดรสคือ 1 TCP Agent จะใช้พอร์ต 0 ของโหนด n0 เพื่อเชื่อมต่อแอพพลิเคชัน FTP ผ่าน TCP กับพอร์ต 0 ของโหนด n1



รูปที่ ก.11 โครงสร้างลิงค์ที่มีออบเจ็คเทรซเพื่อเก็บข้อมูล

- **การเชื่อมต่อระหว่างโอทีซีแอลกับ C++**

ในการแมพออบเจ็คต่างๆที่เขียนขึ้นในภาษา C++ และโอทีซีแอลเข้าด้วยกันทั้งแมพคลาส หรือแมพเฉพาะตัวแปลบางตัว ดังตัวอย่างในรูปที่ ก.12 เป็นการแสดงการแมพออบเจ็คคลาสของ C++ เข้ากับออบเจ็คคลาสของโอทีซีแอล โดยในตัวอย่างได้สร้างเอเจนท์ขึ้นมาชื่อว่า MyAgent ซึ่งสืบทอดมาจากคลาส Agent และสร้าง Linkage จาก C++ ไปยังโอทีซีแอลชื่อ Agent/MyAgentOTCL โดยอ้างออบเจ็คของโอทีซีแอลให้ไปเรียกออบเจ็คของ C++มาทำงานอีกต่อหนึ่งโดยผ่านฟังก์ชันชื่อว่า create ซึ่งผลลัพธ์ที่ได้จากการสร้างออบเจ็คในโอทีซีแอลจะทำให้เกิดออบเจ็คใน C++ ขึ้น

เมื่อมีการเรียกออบเจ็ค "new Agent/MyAgentOTCL" จะทำให้อินสแตนซ์ออบเจ็ค MyAgent ถูกสร้างขึ้นโดยมีพอยน์เตอร์อ้างจากโอทีซีแอลได้

สำหรับการแมพออบเจ็คที่เป็นตัวแปรจาก C++ ไปยังโอทีซีแอลอาจทำได้โดยตัวอย่างในรูปที่ 13 ซึ่งเป็นการแมพตัวแปรในออบเจ็คคลาสชื่ my_var1 และ my_var2 เข้ากับ my_var1_OTCL และ my_var2_OTCL ตามลำดับโดยผ่านฟังก์ชั่น bind() ซึ่งเอ็นเอสยอมให้มีการแมพด้วยฟังก์ชันต่อไปนี้

bind(): real or integer variables

bind_Time(): Time variable

bind_bw(): bandwidth variable

bind_bool(): boolean variable

```
class MyAgent : public Agent {
public:
        MyAgent();
protected:
        int command(int argc, const char*const* argv);
private:
        int    my_var1;
        double my_var2;
        void   MyPrivFunc(void);
};
```

```
static class MyAgentClass : public TclClass {
public:
        MyAgentClass() : TclClass("Agent/MyAgentOtcl") {}
        TclObject* create(int, const char*const*) {
                return(new MyAgent());
        }
} class_my_agent;
```

รูปที่ ก.12 การแมพคลาสออบเจ็ค MyAgent เข้ากับออบเจ็ค Agent/MyAgentOTCL

```
MyAgent::MyAgent() : Agent(PT_UDP) {
        bind("my_var1_otcl", &my_var1);
        bind("my_var2_otcl", &my_var2);
}
```

รูปที่ ก.13 แสดงการแมพตัวแปรของ โอทีซีแอลไปยัง C++

การแมพระหว่างเมธอด (Method) ของคลาสออบเจ็คใน C++ กับโอทีซีแอลทำได้โดยผ่านฟังก์ชัน ชื่อ command() ดังแสดงในตัวอย่างที่ ก.14 ซึ่งจะเห็นได้ว่าโอทีซีแอลสามารถเรียกเมธอดของ ออบเจ็ค C++ ได้เช่น

set myAgent [new Agent_MyAgentOTCL]

$myAgent call-my-priv-func

```
int MyAgent::command(int argc, const char*const* argv) {
      if(argc == 2) {
            if(strcmp(argv[1], "call-my-priv-func") == 0) {
                  MyPrivFunc();
                  return(TCL_OK);
            }
      }
      return(Agent::command(argc, argv));
}
```

รูปที่ ก.14 แสดงการแมพเมธอดของโอทีซีแอลไปยัง C++

การแมพคำสั่งจาก C++ ไปยังโอทีซีแอลสามารถทำได้ดังตัวอย่างรูปที่ ก.15 เป็นการส่งแสดงค่า จาก C++ ไปยังโอทีซีแอลอินเทอร์พรีเตอร์

```
void MyAgent::MyPrivFunc(void) {
      Tcl& tcl = Tcl::instance();
      tcl.eval("puts \"Message From MyPrivFunc\"");
      tcl.evalf("puts \"      my_var1 = %d\"", my_var1);
      tcl.evalf("puts \"      my_var2 = %f\"", my_var2);
}
```

รูปที่ ก.15 แสดงการแมพคำสั่งจาก C++ ไปยังโอทีซีแอล

หากสั่งทดสอบสคริปต์โอทีซีแอลจากการแมพด้วยสคริปต์

set myAgent [new Agent/MyAgentOTCL]

$myAgent set my_var1_OTCL 2

$myAgent set my_var2_OTCL 3.14

จะได้เอาต์พุตเป็น

Message From MyPrivFunc

my_var1 = 2

my_var2 = 3.140000

- **วิเคราะห์ผลที่ได้จากการจำลองการทำงาน (Trace File Analysis)**

หลังจากการจำลองการทำงานเทรซ (Trace File) จะเก็บข้อมูลระหว่างการจำลองการทำงานลงในไฟล์ซึ่งมีข้อมูลหลายเลเยอร์แต่ละเลเยอร์จะถูกแยกออกจากกันโดยมีรูปแบบเทรซของเครือข่ายสายตามตารางที่ โดยมีรูปแบบการเก็บข้อมูลดังนี้

การเก็บข้อมูลแบบเครือข่ายสายทั่วไปประกอบด้วยฟิลด์ต่างๆในตารางที่ ก.1 และตารางที่ ก.3 ตัวอย่างเช่นข้อมูล IP ก็จะต่อท้ายตารางที่ ก.1 ด้วย "------- [%d:%d %d:%d %d %d]" จะได้เป็น

(r, d, e, +, -) %g %d %d %s %d %s %d %d.%d %d.%d %d %d ------- [%d:%d %d:%d %d %d]

สำหรับรูปแบบของเครือข่ายไร้สายจะเป็นดังตารางที่ ก.2 และข้อมูลการเคลื่อนที่ของโหนดกับพลังงานของโหนดจะเป็นไปดังตารางที่ ก.3 และ ก.4 ตามลำดับ

ตารางที่ ก.1 แสดงรูปแบบของไฟล์เทรซของแพคเกจข้อมูลทั่วไปสำหรับเครือข่ายสาย

อีเวนท์	ตัวย่อ	ชนิดข้อมูล	ข้อมูลที่เก็บ
Normal Event	r:Receive d:Drop e:Error +:Enqueue -:Dequeue	%g %d %d %s %d %s %d %d.%d %d.%d %d %d	
		double	Time
		int	Source Node
		int	Destination Node
		string	Packet Name
		int	Packet Size
		string	Flags
		int	Flow ID
		int	Source Address
		int	Destination Address
		int	Sequence Number
		int	Unique Packet ID

ตารางที่ ก.2 แสดงรูปแบบของไฟล์เทรซของแพคเกจข้อมูลทั่วไปของเครือข่ายไร้สาย

อีเวนท์	ตัวย่อ	ชนิดข้อมูล	ข้อมูลที่เก็บ
Wireless Event	s:Send r:Receive d:Drop f:Forward	%.9f %d (%6.2f %6.2f) %3s %4s %d %s %d [%x %x %x %x]	
		%.9f _%d_ %3s %4s %d %s %d [%x %x %x %x]	
		double	Time
		int	Node ID
		double	X Coordinate (If Logging Position)
		double	Y Coordinate (If Logging Position)
		string	Trace Name
		string	Reason
		int	Event Identifier
		string	Packet Type
		int	Packet Size
		hexadecimal	Time To Send Data
		hexadecimal	Destination MAC Address
		hexadecimal	Source MAC Address
		hexadecimal	Type (ARP, IP)

ตารางที่ ก.3 แสดงรูปแบบของไฟล์เทรซของแพคเกจข้อมูลแต่ละเลเยอร์

อีเวนท์	ชนิดข้อมูล	ข้อมูลที่เก็บ
ARP Trace	------- [%s %d/%d %d/%d]	
	string	Request or Reply
	int	Source MAC Address
	int	Source Address
	int	Destination MAC Address
	int	Destination Address
IP Trace	------- [%d:%d %d:%d %d %d]	

	int	Source IP Address
	int	Source Port Number
	int	Destination IP Address
	int	Destination Port Number
	int	TTL Value
	int	Next Hop Address, If Any
TCP Trace	[%d %d] %d %d	
	int	Sequence Number
	int	Acknowledgment Number
	int	Number Of Times Packet Was Forwarded
	int	Optimal Number Of Forwards
CBR Trace	[%d] %d %d	
	int	Sequence Number
	int	Number Of Times Packet Was Forwarded
	int	Optimal Number Of Forwards
IMEP Trace	[%c %c %c 0x%04x]	
	char	Acknowledgment Flag
	char	Hello Flag
	char	Object Flag
	hexadecimal	Length

ตารางที่ ก.4 แสดงรูปแบบเทรซในการเคลื่อนที่และค่าพลังงานของโหนดไร้สาย

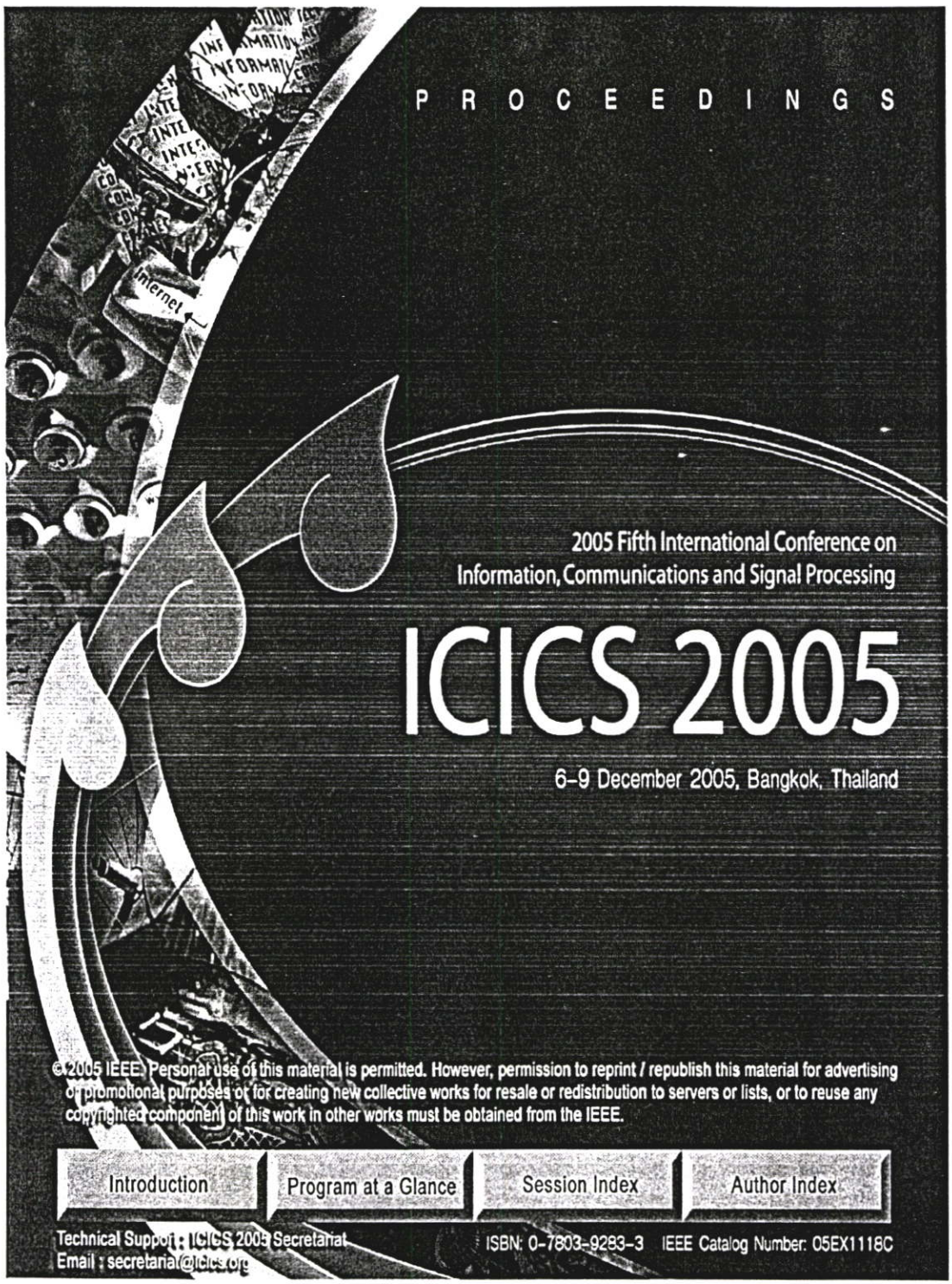
อีเวนท์	ตัวย่อ	ชนิดข้อมูล	ข้อมูลที่เก็บ
Mobile Node Movement	M	%.5f %d (%.2f, %.2f, %.2f), (%.2f, %.2f), %.2f	
		double	Time
		int	Address (Node ID?)
		double	X Coordinate
		double	Y Coordinate

		double	Z Coordinate
		double	Destination X Coordinate
		double	Destination Y Coordinate
		double	Movement Speed
Mobile Node Energy	N	-t %f -n %d -e %f	
		double	Time
		int	Address (Node ID?)
		double	Energy

ภาคผนวก ข.

ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่

1. M. Kummakasikit, S. Thipchaksurat, R. Varakulsiripunth, "Modification of Associativity-Based Routing Protocol Based on Mobile Host's Location in Ad-Hoc Wireless Networks," The 9th Nation Computer Science and Engineering Conference (NCSEC'2005), pp. 21-29, University of Thai Chamber of Commerce, Bangkok Thailand, October 27-28, 2005.
2. M. Kummakasikit, S. Thipchaksurat, R. Varakulsiripunth, "Performance Improvement of Associativity-Based Routing Protocol for Mobile Ad-Hoc Networks," Fifth International Conference on Information, Communications and Signal Processing, The Grand Hotel, Bangkok, Thailand, December 6-9, 2005.


PROCEEDINGS
2005 Fifth International Conference on
Information, Communications and Signal Processing
ICICS 2005
6-9 December 2005, Bangkok, Thailand
©2005 IEEE. Personal use of this material is permitted. However, permission to reprint / republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.
Introduction
Program at a Glance
Session Index
Author Index
Technical Support : ICICS 2005 Secretariat
Email : secretariat@icics.org
ISBN: 0-7803-9283-3 IEEE Catalog Number: 05EX1118C

W1A.4

Performance Improvement of Associativity-Based Routing Protocol for Mobile Ad Hoc Networks

Manoon KUMMAKASIKIT, Sakchai THIPCHAKSURAT and Ruttikorn VARAKULSIRIPUNTH
Faculty of Engineering and Research Center for Communications and Information Technology (ReCCIT)
King Mongkut's Institute of Technology Ladkrabang,
Chalongkrung Road, Ladkrabang, Bangkok, Thailand 10520
e-mail : joesk29na786@yahoo.com, {ktsakcha, kvruttik}@kmitl.ac.th

***Abstract*—Ad hoc network is a set of mobile nodes that each mobile host is able to move freely and the network is unpredictable topologies transformation. Therefore, the searching for transmission path becomes difficult and complicated. Because each mobile host in ad hoc network acts as a switching node that routes and forward packets to its neighboring mobile hosts (or nodes), hop by hop, or multi-hop. Unlike the conventional mobile wireless networks that all nodes must communicate via the base-station as single-hop. Many proposed routing protocols have been developed for ad hoc network in order to maximize the benefit of bandwidth and network utilization. The Associativity-Based Routing (ABR) is an effective routing protocol that becomes interested. Thus, in this paper, we proposed the modification and improvement of ABR protocol by including the position information of mobile hosts into its algorithm. This approach can reduce routing overhead of route discovery process and increase more beneficial performance for mobile ad hoc networks.**

Keywords*—*ABR, Ad Hoc Network, MABR, GPS

I. Introduction

In mobile ad hoc networks (MANET), each mobile host can move freely around the area of the network. All mobile hosts are connected each other by the radio range. Communication between mobile hosts is performed directly via the selected path. The path or route is a sequence of mobile hosts in the network that forward packets to the destination hop by hop without passing to the base-station.

Many routing algorithms for ad hoc networks have been proposed with the different idea in the classification of source-initiated on-demand routing protocol such as Associativity-Based Routing (ABR) [1], Position-Aided Routing (LAR) [2], Dynamic Source Routing (DSR) [3] and Ad hoc On-Demand Distance Vector (AODV) routing [4]. Most of them attempted to improve and utilize the networks performance appropriate to the desired environments and limited radio bandwidth.

We have chosen ABR as a case study. This is because ABR focuses on the link stability and has fewer route-reconstructions that can reduce some fractions of routing overhead. Associativity-ticks are performed to express the stability of links by means of beaconing. Beacon messages are periodically sent to the neighbor to inform the link stability between node itself and neighbors. As an example in Fig. 1, each node periodically sends beacon messages to the neighbors and receives beacon messages to keep track the connectivity between node itself and neighbors. The received beacon messages are stored in associativity-table called associativity-tick. As demonstrated in Fig. 2, when MH2 (MH: Mobile Host) moves out from radio range of MH1, associativity-tick inside the table of MH1 and MH2 are reset. This stability information is included in route request message and sent out to all nodes i.e., mobile hosts, in the network. Any intermediate node that received this message will insert its ID and associativity-tick with the neighbors to the route request message. When the messages arrived at the destination, the destination will select a suitable route from a set of many routes that route request messages have passed. The associativity-tick information that was inserted into route request message by intermediate nodes is used as a stability of each route's selection algorithm at the destination. After a route is selected, destination sends reply message back to the sender via the selected path. The duration of finding a route is called route request-reply process.

When route request messages are broadcasted to the entire network in the route discovery processes, all reachable nodes must forward received route request message to find a route, although they are handling for data packets from the different routes. When a node has higher mobility, and/or more data traffic, probability of dropping the data packets is increased because of waiting for a route and/or waiting for the medium to be idle.

Where all nodes share the medium in the overlapped transmission ranges. These problems brought us to an idea of applying position information of mobile hosts in routing algorithm that proposed in LAR.

In the first stage of route discovery process, firstly LAR computes two regions called expected zone and request zone, where expected zone is an expected area that has the destination node, and request zone is an area that the route request messages are broadcasted and should include an expected zone. LAR tried to limits the searching of route request message by means of request zone. All route request messages are only broadcasted in this area. If any nodes outside of the request zone get the route request messages, the messages will be dropped. LAR algorithm has been also im-

0-7803-9282-5/05/$20.00 ©2005 IEEE

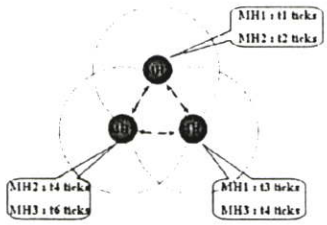


Figure 1. Example of associativity-tick operation

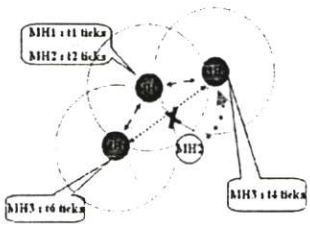


Figure 2. Associativity-tick when MH2 moves out from radio range of MH1

plemented in the DSR protocol such that all IDs of node in request message are a set of all nodes that was kept while the request message passed to the nodes. This information will be future used for selecting a route. When route request message reached at the destination, the shortest path or path with shortest delay will be selected as a route. During the route discovery process, some position parameters are used for request zone and expected zone computations. A little increasing of overhead due to position information is performed compares to routing overhead due to broadcasting route request messages.

Thus this paper we considered the integration of advantages of ABR and other mentioned routing protocols by modifying them. The objective is to optimize some bandwidth utilization used by routing agent and to minimize the increased overhead occurs when position information is performed. Our approach is "Modified Associativity-Based Routing (MABR) protocol.

II. MABR Protocol

One assumption of LAR is that all nodes known its current positions, by getting the information from well-known device, i.e., Global Positioning System (GPS). If the approximate speed of the desired ad hoc networks is V, when source node S wants to communicates with destination node D at time t_1 and node S knows the position of node D in the network at time-stamp t_0. Then the displacement of the destination node D (within the radius of an expected zone) is $R=V\times(t_1-t_0)$. Therefore, the expected zone at time t_1 is a circular region that node D is the center point as shown in Fig. 3. For the request zone, LAR proposed two schemes of routing broadcasting decision. Scheme (I) depends on the shape of selected area (call "LAR-box") and scheme (II) depends on the distance (call "LAR-step) between each intermediate node and expected zone. If LAR-step does not cover to the destination, messages will be dropped. For MABR we selected scheme (I) as a rectangular shape as shown in Fig. 4.

In MABR, the additional "Position Information Tables" (PIT) are included in each node. The table is updated all others nodes position information by learning from the position information included in the message passed through this node. Two parameters i.e., nodes positions and position's time-stamps are inserted in routing messages for using to determine the expected zone and request zone. In a route discovery process, the positions of source and destination nodes, and position's time-stamps are added to the route request message so that any intermediate nodes that received these route request messages will update the position information of source and destination in its PIT. After destination node has selected a route, it sends the reply message with position and timestamp of the destination node to source node. Any intermediate nodes that received the route request messages or reply messages will update its PIT for future use as described in the following algorithm.

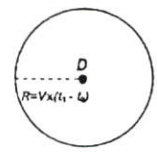


Figure 3. Expected zone

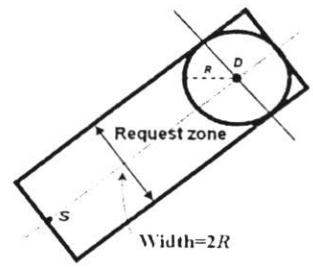


Figure 4. Broadcasting area of route request message when source node is outside of the expected zone

A. Intermediate node:

Following is an algorithm which intermediate node will update PIT when it received a route discovery message or reply message in the duration of route request-reply.

Algorithm for Intermediate Node

if intermediate node I gets route request message

if position information is available in message

- Update source and destination positions with timestamps to its PIT if available in the message.
- Compute expected zone and request zone by using source-destination positions and timestamps.
- Get position itself.

if position node I is in request zone. Then

- Rebroadcast route request message.

else

- Drop message.

end if

else

- Rebroadcast route request message.

end if

else if intermediate node I gets reply message.

if position information is available in message.

- Update destination position with timestamp to its PIT.

end if

if its node's ID is one of a reply route.

- Make valid route and insert to route cache.
- Forward reply message backward to upper-stream node referred from a reply route.

else

- Drop message.

end if

end if

B. Source Node:

When source node S wants to communicate with destination node D. Firstly, node S checks whether does it has the destination position, if yes, it includes this position information of node D, position of itself and both timestamps to the route request message.

Algorithm for Source Node

if source node S wants to communicate with destination node D

- Lookup destination position in PIT.

if destination position in PIT is empty and S wants to run original ABR.

- Broadcast route request message by using normal flooding mode.

else if destination position in PIT is not empty, run MABR.

- Include its position and timestamp to route request message.
- Broadcast route request message inside request zone.

else

- Include its position, destination position, and timestamps in route request message.
- Broadcast route request message inside request zone.

end if

end if

C. Destination Node:

When the destination node receives the route discovery messages. It will collect all path information and select only one route and reply the message back to the source via selected path.

Algorithm for Destination Node

if destination node D received the route request message.

if position information is available in message.

- Update source and destination positions with timestamps in its PIT.
- Collect/Select a route by using route selection algorithm.
- Include position and timestamp itself in reply message.
- Send reply message back to the source via selected path.

else

- Collect/Select a route by using route selection algorithm.
- Send reply message back to the source via selected path.

end if

end if

In the duration of request-reply, if route request message has included the position information, then the destination node will update its PIT before handles the route request messages and uses the route selection algorithm similar to ABR protocol. The destination node only includes its current

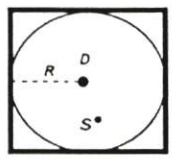


Figure 5. Broadcasting area of route request message when source node is inside of the expected zone

time and position into the reply message if the route request messages has position information and sends reply back to the source node via selected path.

III. Knowledge Extraction

In case of position information is very up-to-date, expected zone and request zone will be bounded as the very small area. For example, when time t_1=101 node S wants to communicate with node D when node S knows that node D is at the position (100,200) at time t_0=100. Here assume that the transmission speed in ad hoc network V = 10.0m/s, and radius of expected zone is only $R=V\times(t_1-t_0)$ = 10m, then the width of request zone which equal to the diameter of the expected zone is 2x10=20m. Thus, we can define some appropriate constant value to be a minimum default value of the expected zone's radius.

In reverse case that if the position information is very old, the request zone will be very large. Then MABR acts as normal flooding mode that is, entire network able to broadcast route request messages. In the simulation, we found that if we selected the minimum of default radius constant of expected zone too small, it will result in reduction of route searching probability, and vice versa. But if we selected it too high this will increase more routing overhead.

IV. Simulation Model

For the simplicity, we assumed that all position information has an error free. The simulation model and implemented code were based on ns-2 [5] with the wireless extensions of CMU project [6]. IEEE802.11 standard with the Distributed Coordination Function (DCF) was introduced in MAC layer. The radio model was commercial radio interface (Wavelan [9]) with bit-rate of 2Mb/s and radio range of 250 meters. The capacity of interface queue was 64 packets. There were 20 CBR (Constant Bit-Rate) traffic sources with sending rate of 4 packets/s, where each packet size was 64 bytes. Mobility model was random waypoint model with 50 nodes and all nodes were able to move around with in the area of 1000x1000m2. We had used "mobgen" [6] generator instead of "setdest" to generate many different mobility scenario for the test. Speeds of mobile node were 0, 5, 10, 15, 20, 25 and 30m/s with pause time of 5s before migration again. In our test we ran multiple times of 1800s in the network model time.

V. Simulation Results

Overhead of control message is a parameter that expresses the utilization of bandwidth. When all nodes are stationary, some of route discoveries are performed and about 10 Kb/s of overhead are constantly consumed by beacon messages that are sent periodically.

TABLE I Simulation Parameters

Mobile Host	50
Area	1000x1000 m^2
Moving pattern	Random waypoint
Speed	0-30 m/s
Simulation time	1,800 s.
Interface Queue	64 packets
CBR connections	20 pairs
Packet size	64 bytes
CBR sending rate	4 packets/s.
Pause time	5 s.
Link Bandwidth	2Mbps

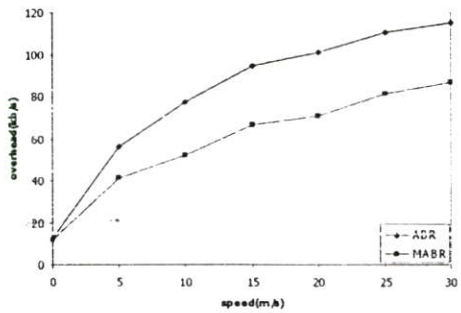


Figure 6. Overhead of control messages

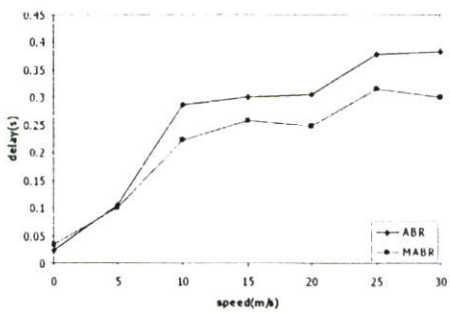


Figure 7. Average End-to-End delay

When all nodes start moving, the route discovery processes are often invoked due to link failure. Also many of route discovery processes generate more request messages to the network. While the protocols are running in the very high speed of mobility and most of the packets in the network are request messages, which are generated by route discovery processes. The result in Fig. 6 demonstrated that the request zone in MABR reduced some overhead bandwidth when mobility was increasing. And the overall overhead of MABR is lower than ABR.

We evaluated the average end-to-end delay of data packets as shown in Fig. 7. When all nodes were stationary, only some of route discovery processes were performed. MABR has shown the lower end-to-end delays than ABR. This is because the stability of the route in MABR is less than ABR and the additional control overhead from position and timestamp information are included in the route request messages of MABR. When nodes start moving, the simulation result shows that ABR has larger end-to-end delays, due to higher control overhead and queuing delay. We observed that the congestion of control messages and data messages in ABR are increased during all nodes are moving with high speed. This is because the route discovery processes of ABR tried to broadcast the route request messages to entire of the network. But MABR has limited an area of broadcasting route request messages, thus the congestion problem was reduced.

VI. Conclusion

Mobile Ad Hoc network (MANET), is different from normal wireless network because it can be applied to many situations without the centralized stations and administrator. However, the main problems of MANET are route searching and maximizing the utilization of the limited radio bandwidth. Many proposed routing protocols like, ABR, LAR, DSR and AODV are the examples of routing protocols. Most of them are based on the classification of on demand routing and source-initiate routing protocol, which a route is searched whenever the source needs a route.

In this paper, we proposed "Modified Associativity-Based Routing" (MABR) protocol with the application of position information of mobile hosts. The MABR attempts to reduce overhead fractions in the network by limiting the area for sending the route request message in the route discovery processes. We have shown that MABR provided the better performance in the terms of overhead and end-to-end delay comparing with the traditional ABR protocol.

References

[1] C.-K. Toh, "Associativity-Based Routing For Ad Hoc Mobile Networks," Wireless Pers. Commun. J., Special Issue on Mobile Networking and Computing Systems, Kluwer Academic, vol. 4, no. 2,Mar. 1997, pp 103-39.

[2] Y-B Ko and N. H. Vaidya. Position-Aided Routing (LAR) in MobileAd Hoc Networks. ACM/Baltzer Wireless Networks (WINET) journal,Vol.6-4, 2000.

[3] D.Johnson and D.Maltz, "Dynamic source routing in ad hoc wireless networks," in Mobile Computing, T.Imielinsky and H.Korth, Eds., pp. 153–181. Kluwer Academic Publishers, 1996.

[4] C.E. Perkins and E. M. Royer. Ad hoc On-Demand Distance Vector Routing. Ad Hoc Networking, edited by C. E. Perkins, Addison-Wesley, 2001.

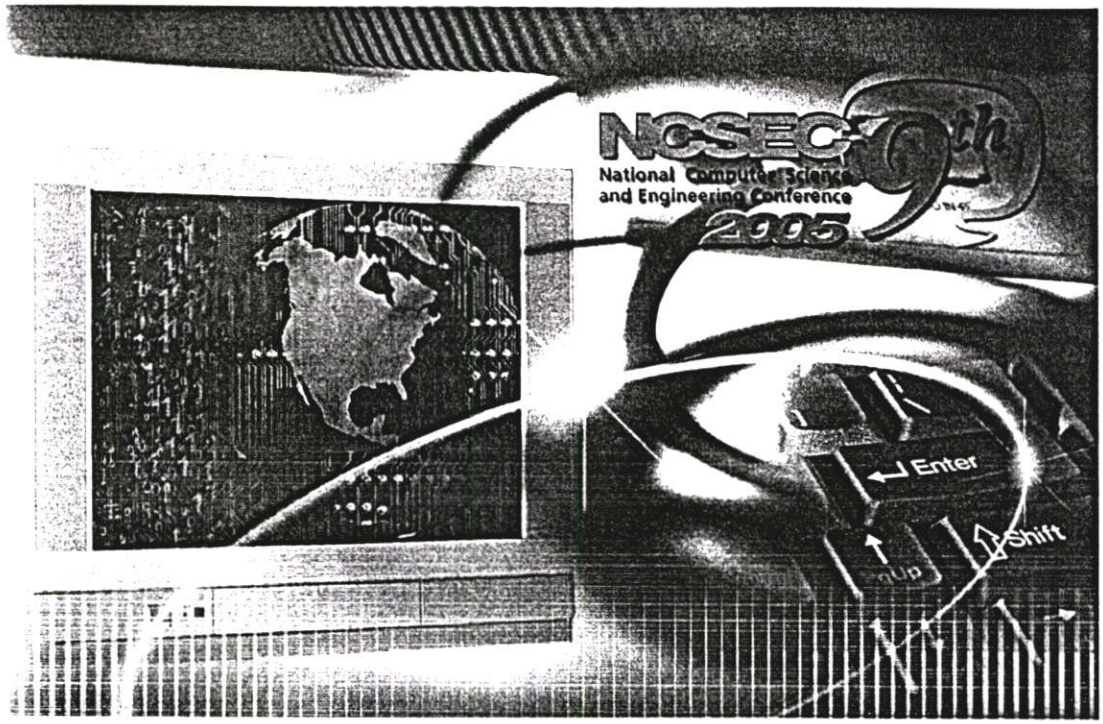
[5] The Network Simulator ns-2, available online at http://www.isi.edu/nsnam/ns/ J. Wang, "Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication)," IEEE J. Quantum Electron., submitted for publication.

[6] The Monarch Group at Rice University, project website http://www.monarch.cs.rice.edu/.

[7] T. Camp, J. Boleng, and V. Davies, A Survey of Mobility Models for Ad Hoc Network Research, Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, pp. 483-502, 2002.M. Young, The Technical Writers Handbook. Mill Valley, CA: University Science, 1989.

[8] T. Camp, J. Boleng, B. Williams, L. Wilcox, and W. Navidi, Performance Comparision of Two Position Based Routing Protocols for Ad Hoc Networks, Proceedings of the IEEE INFOCOM, pp. 1678-1687, 2002. S. Chen, B. Mulgrew, and P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," IEEE Trans. Neural Networks, vol. 4, pp. 570–578, July 1993.

[9] Bruce Tuch. Development of WaveLAN, an ISM band wireless LAN. AT&T Technical Journal, 72(4):27-33, July/Aug 1993.S. P. Bingulac, "On the compatibility of adaptive controllers (Published Conference Proceedings style)," in Proc. 4th Annu. Allerton Conf. Circuits and Systems Theory, New York, 1994, pp. 8-16.



The 9th National Computer Science and Engineering Conference

October 27-28, 2005

University of Thai Chamber of Commerce, Bangkok Thailand

Organized by:

Department of Computer Engineering, School of Engineering,
University of Thai Chamber of Commerce

In Cooperation with:

Electrical Engineering; Electronics, Computer, Telecommunications and Information Technology Association of Thailand (ECTI)

IEEE Communications Society, Thailand Chapter

Sponsored by:

University of Thai Chamber of Commerce

National Electronics and Computer Technology Center (NECTEC)

Sun Microsystems (Thailand)

CS Loxinfo Public Company Limited

Pearson Education Indochina Limited

The OGA Group

The 9th National Computer Science and Engineering Conference

NCSEC 9th
National Computer Science and Engineering Conference
2005

October 27-28, 2005
University of Thai Chamber of Commerce,
Bangkok Thailand

Organized by:
Department of Computer Engineering, School of Engineering,
University of Thai Chamber of Commerce

In Cooperation with:
Electrical Engineering; Electronics, Computer, Telecommunications and Information Technology Association of Thailand (ECTI)

IEEE Communications Society, Thailand Chapter

Sponsored by:
University of Thai Chamber of Commerce

National Electronics and Computer Technology Center (NECTEC)

Sun Microsystems (Thailand)

CS Loxinfo Public Company Limited

Pearson Education Indochina Limited

The OGA Group

ISBN 974-677-541-3

Printing: New UM Ad. Co., Ltd. 39/4 Vibhavadee Rangsit Rd., Jatujak, Bangkok 10900

Contents

Computer Networks

การปรับปรุงโปรโตคอลการค้นหาเส้นทางเอบีอาร์ด้วยข้อมูลตำแหน่งของโมไบล์โฮสต์ในเครือข่ายไร้สายแอดฮอค

Modification of Associativity-Based Routing Protocol based on Mobile Host's Location in Ad-Hoc Wireless Networks

มนูญ คัมมกสิกิจ
ศักดิ์ชัย ทิพย์จักษุรัตน์
รัตติกร วราภูลศิริพันธุ์
คณะวิศวกรรมศาสตร์ และ สำนักวิจัยการสื่อสารและเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
Email: joesk29na786@msn.com ktsakcha@kmitl.ac.th kvruttik@kmitl.ac.th

การปรับปรุงโปรโตคอลการค้นหาเส้นทางเอบีอาร์ด้วยข้อมูลตำแหน่งของโมไบล์โฮสต์ในเครือข่ายไร้สายแอดฮอก

Modification of Associativity-Based Routing Protocol based on Mobile Host's Location in Ad-Hoc Wireless Networks

มนูญ ค้มมกสิกิจ ศักดิ์ชัย ทิพย์จักษุรัตน์ รัตติกร วราภูลศิริพันธุ์
คณะวิศวกรรมศาสตร์ และ สำนักวิจัยการสื่อสารและเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
Email: joesk29na786@msn.com, {ktsakcha, kvruttik}@kmitl.ac.th

บทคัดย่อ

งานวิจัยนี้เราได้นำเสนอวิธีการปรับปรุงโปรโตคอลค้นหาเส้นทางเอบีอาร์ในเครือข่ายไร้สายแอดฮอก โดยการนำตำแหน่งของโมไบล์โฮสต์มาประกอบการพิจารณาในการค้นหาเส้นทาง ทั้งนี้เพื่อจัดขอบเขตของการส่งแพ็คเกตร้องขอเส้นทางให้จำกัดเฉพาะกลุ่มของโมไบล์โฮสต์ที่ต้องการ ซึ่งเราได้นำเสนอโปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์ ในงานวิจัยนี้ จากวิธีการที่ได้นำเสนอ เราพบว่าโปรโตคอลเอ็มเอบีอาร์สามารถที่จะเพิ่มอัตราการส่งข้อมูลให้สูงขึ้น นอกจากนี้ยังสามารถลดค่าโอเวอร์เฮดของการส่งข้อมูลลงด้วย เมื่อเปรียบเทียบกับโปรโตคอลเอบีอาร์

Abstract

We propose the performance improvement of Associativity-Based Routing (ABR) protocol by considering the position of nodes in ad-hoc wireless networks. In our proposed protocol, the route discovery message is sent to the expected group of mobile host as the result, the overhead can be reduced. In this paper, we proposed our protocol called Modified Associativity-Based Routing (MABR) protocol. We evaluate the performance of MABR comparing with ABR protocol by means of the simulation. The results show that MABR protocol provides higher data packet delivery fraction and lower delay than those of ABR protocol.

คำสำคัญ: แอดฮอก โมไบล์โฮสต์ เส้นทาง เอบีอาร์ แอดเอ็มเอบีอาร์

1. บทนำ

เครือข่ายไร้สายแอดฮอกประกอบด้วยโมไบล์โฮสต์ต่างๆที่เคลื่อนที่อย่างเป็นอิสระ การติดต่อสื่อสารของเครือข่ายแอดฮอก แต่ละโมไบล์โฮสจะทำหน้าที่เสมือนเป็นเราท์เตอร์ซึ่งสามารถรับและส่งต่อ(store and forward) ข้อมูลที่ได้รับจากโมไบล์โฮสต์ข้างเคียงได้ ดังนั้นในเครือข่ายแอดฮอกจึงไม่จำเป็นต้องมีตัวกลางในการติดต่อสื่อสาร แต่จะเป็นลักษณะที่มีการส่งต่อข้อมูลต่อๆกันจนถึงโมไบล์โฮสต์ปลายทางเรียกว่าเป็นการสื่อสารแบบ ฮอพบายฮอพ (hop-by- hop) แต่ละโมไบล์โฮสต์จึง

ต้องมีการเก็บข้อมูลเส้นทางต่างๆที่ใช้ในการสื่อสารไว้ในตารางเส้นทาง(routing table) จากคุณลักษณะของเครือข่ายแอดฮอคจะเห็นได้ว่า เมื่อแต่ละโมไบล์โฮสต์ในเครือข่ายสามารถเคลื่อนที่ได้อย่างเป็นอิสระต่อกันส่งผลให้การค้นหาเส้นทางจากต้นทางไปยังปลายทางนั้นซับซ้อนมากกว่าระบบเครือข่ายไร้สายที่พบเห็นทั่วไป ซึ่งการสื่อสารจะกระทำกันโดยผ่านตัวกลางที่เป็นสถานีฐาน(Base station) ก่อนเท่านั้น สำหรับข้อมูลเส้นทางการสื่อสารในเครือข่ายแอดฮอคนั้นจะมีการเปลี่ยนแปลงอยู่ตลอดเวลาตามสภาพของเส้นทางและโครงสร้างของเครือข่าย สำหรับวิธีการค้นหาเส้นทางนั้นจะขึ้นอยู่กับโปรโตคอลที่ได้นำมาพิจารณา

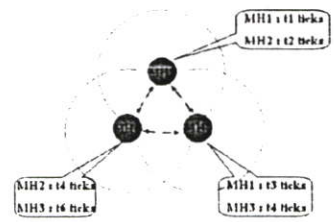
งานวิจัยนี้เราได้นำโปรโตคอลเอบีอาร์(Associativity-Based Routing, ABR)[1] ซึ่งเป็นโปรโตคอลในกลุ่มของโปรโตคอลแบบออนดีมานด์มาพิจารณาและปรับปรุงแก้ไขโดยใช้แนวคิดการนำข้อมูลตำแหน่งของโมไบล์โฮสต์เพื่อจำกัดพื้นที่การค้นหาเส้นทางซึ่งเป็นข้อด้อยข้อหนึ่งของโปรโตคอลแบบออนดีมานด์ที่กระบวนการค้นหาเส้นทางจะมีการบรอดคาสท์แมสเสสร้องขอเส้นทางไปทั่วเครือข่าย ทำให้สูญเสียแบนด์วิธในการค้นหาเส้นทางสูง สำหรับการนำโปรโตคอลเอบีอาร์(Associativity-Based Routing, ABR)[1] มาพิจารณานั้น เนื่องมาจากเอบีอาร์มีแนวคิดในการลดคอนโทรลโอเวอร์เฮด (Control Overhead)หรือปริมาณข้อมูลต่างที่ใช้สำหรับค้นหาเส้นทาง และโปรโตคอลเอบีอาร์ยังสามารถพิจารณาสภาพและ คุณภาพของเส้นทางได้ เช่นสามารถหลีกเลี่ยงเส้นทางที่มีการจราจรของข้อมูลอย่างคับคั่งแออัด

2. งานวิจัยที่เกี่ยวข้อง

งานวิจัยเราได้นำแนวคิดของโปรโตคอลในการค้นหาเส้นทางเพื่อลดโอเวอร์เฮดของปริมาณข้อมูลสำหรับใช้ค้นหาเส้นทางในเครือข่าย โปรโตคอลที่เราพิจารณาประกอบด้วยโปรโตคอลเอบีอาร์(Associativity-Based Routing protocol, ABR)[1] และโปรโตคอลแอลเออาร์ (Location-Aided Routing protocol, LAR)[2]

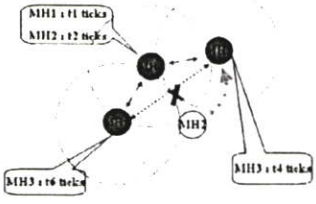
2.1 Associativity-Based Routing (ABR)

การทำงานของโปรโตคอลนี้จะใช้ค่าความสัมพันธ์ระหว่างโมไบล์โฮสต์ (associativity tick) ในการตัดสินใจเลือกเส้นทาง ค่าความสัมพันธ์นี้ เป็นข้อมูลที่เก็บอยู่ในแต่ละโมไบล์โฮสต์ โดยจะถูกนำมาใช้เพื่อแสดงสถานะของการเชื่อมต่อระหว่างโมไบล์โฮสต์ต่างๆที่อยู่รอบข้าง ทุกๆ คาบเวลาหนึ่ง แต่ละโมไบล์โฮสต์จะส่งแมสเสสออกมาเรียกว่า บีคอน (beacon) เพื่อบอกสถานะของตนแก่โมไบล์โฮสต์รอบข้าง เมื่อโมไบล์โฮสต์รอบข้างได้รับบีคอน จะทำการบันทึกจำนวนครั้งที่ได้รับอาจรวมถึงความแรงสัญญาณของโมไบล์โฮสต์ที่ได้รับเอาไว้ ดังแสดงในรูปที่ 2(a) ตัวอย่างเช่นเมื่อโมไบล์โฮสต์ MH2 มีการเคลื่อนที่ออกไปทำให้การเชื่อมต่อระหว่างโมไบล์โฮสต์ MH1 และโมไบล์โฮสต์ MH2 เสียหาย เมื่อถึงคาบเวลาของการทำบีคอน



รูปที่ 2(a) การทำงานของ associativity โดยบีคอนแมสเสส

ไมโบล์โฮสต์ MH1 และ MH2 ก็จะไม่สามารถแลกเปลี่ยนแมสเสสบีคอนซึ่งกันและกันได้ แสดงให้เห็นสถานะของสองโมไบล์โฮสต์นี้ว่าไม่สามารถเชื่อมต่อกันได้ หลังจากนั้นโมไบล์โฮสต์ที่เกี่ยวข้องก็จะลบค่าความสัมพันธ์ระหว่างโมไบล์โฮสต์ที่ไม่สามารถเชื่อมต่อกันได้ออกจากตารางความสัมพันธ์ระหว่างโมไบล์โฮสต์ จากการทำงานดังกล่าวจะเห็นว่าช่วงเวลาที่โมไบล์โฮสต์ MH2 และโมไบล์โฮสต์ MH3 ยังติดต่อกันอยู่ได้ตลอดเวลา แสดงให้เห็นว่าโมไบล์โฮส MH2 และโมไบล์โฮส MH3



รูปที่2(b) ตารางแสดง associativity โดยบีคอนเมสเสส เมื่อโมไบล์โฮสต์ MH2 เคลื่อนที่ออกจากระยะรัศมีทำการของโมไบล์โฮสต์ MH1

มีเสถียรภาพของการเชื่อมต่อที่ดีกว่า ซึ่งเอบีอาร์ได้ใช้วิธีการนับจำนวนบีคอนในการทำนายลักษณะเสถียรภาพการเชื่อมต่อระหว่างโมไบล์โฮสต์ สำหรับขนาดของเมสเสจบีคอนนี้เป็นข้อมูลที่มีขนาดเล็กมากเมื่อเทียบกับข้อมูลเมสเสสที่จะถูกส่งออกมาในกระบวนการค้นหาเส้นทาง

ในกระบวนการค้นหาเส้นทางจะมีหน้าที่ในการค้นหาเส้นทางและเลือกเส้นทาง เมื่อโมไบล์โฮสต์ต้นทางต้องการค้นหาเส้นทางไปยังโมไบล์โฮสต์ปลายทาง โมไบล์โฮสต์ต้นทางก็จะบรอดคาสท์เมสเสสการร้องขอเส้นทาง (Broadcast Query message; BQ) ออกไปสู่เครือข่าย ซึ่งโมไบล์โฮสต์ต่างๆที่ได้รับเมสเสสนี้ ถ้าไม่ใช่โมไบล์โฮสต์ปลายทางหรือเรียกว่าโมไบล์โฮสต์ระหว่างทาง (Intermediate Node, IN)ซึ่งสามารถส่งผ่านข้อมูลจากต้นทางไปยังปลายทางได้ จะส่งต่อ (forward) เมสเสสการร้องขอเส้นทางต่อๆ กันไปจนทั่วทั้งเครือข่ายจนกว่าจะหาโมไบล์โฮสต์ปลายทางได้พบ โดยระหว่างเส้นทาง โมไบล์โฮสต์ต่างๆจะเพิ่มเลขประจำโมไบล์โฮสต์ ของตนลงในเมสเสสร้องขอเส้นทาง BQ แล้วส่งต่อ BQ ออกไป เมื่อเมสเสส BQ ถูกส่งไปถึงโมไบล์โฮสต์ปลายทาง โมไบล์โฮสต์ปลายทางก็จะเลือกเส้นทางจากโดยพิจารณาค่าความสัมพันธ์ระหว่างโมไบล์โฮสต์จาก BQ ที่ได้รับมาทั้งหมด โดยเลือกเส้นทางที่เหมาะสมที่สุด แล้วจึงตอบกลับด้วยรีพลายเมสเสส (reply message) ที่มีค่าเลขประจำโมไบล์โฮสต์ต่างๆ ที่จะถูกใช้ส่งต่อข้อมูลหรือใช้เป็นเส้นทางในการสื่อสารข้อมูล ส่งย้อนกลับไปหาโมไบล์โฮสต์ต้นทาง เมื่อโมไบล์โฮสต์ต้นทางได้รับรีพลายเมสเสส ก็จะปรับปรุงข้อมูลตารางเส้นทางที่ได้ตามข้อมูลเส้นทางที่อยู่ในรีพลายเมสเสสเป็นอันเสร็จสิ้นกระบวนการการค้นหาเส้นทาง ซึ่งในงานวิจัยนี้จะพิจารณาถึงโอเวอร์เฮดที่จะถูกใช้ไปสำหรับช่วงการค้นหาเส้นทาง โดยพยายามลดปริมาณข้อมูลเมสเสส BQ ที่ใช้ในการค้นหาเส้นทาง โดยอาศัยตำแหน่งของโมไบล์โฮสต์มาช่วยตามแนวคิดของโปรโตคอลแอลเออาร์ Location-Aided Routing (LAR)[2] เพื่อให้การค้นหาเส้นทางมีประสิทธิภาพมากขึ้น

2.2 โปรโตคอลแอลเออาร์ (Location-Aided Routing protocol, LAR)[2]

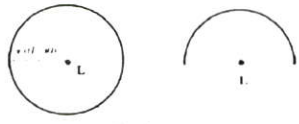
หลักการทำงานของโปรโตคอลแอลเออาร์คือจะทำการจำกัดพื้นที่การบรอดคาสท์เมสเสสการร้องขอเส้นทางให้น้อยลง โดยพิจารณาข้อมูลตำแหน่งของตนเอง ซึ่งข้อมูลนี้อาจจะได้มาจากอุปกรณ์ประเภทจีพีเอส (Global Positioning system, GPS) ที่มีอยู่ในปัจจุบัน โดยโปรโตคอลแอลเออาร์แบ่งการทำงานออกเป็น 2 ส่วนคือ

1) พื้นที่การส่งต่อเมสเสสร้องขอเส้นทาง (Request Zone)

2) พื้นที่คาดหวัง (Expected Zone)

2.2.1 พื้นที่คาดหวัง (Expected Zone)

เมื่อโมไบล์โฮสต์ต้นทาง S ต้องการติดต่อไปยังโมไบล์โฮสต์ปลายทาง L โดยสมมติว่าโมไบล์โฮสต์ต้นทาง S มีข้อมูลตำแหน่งของโมไบล์โฮสต์ L โดยพิจารณาจากเวลาที่ผ่านไป (t_0) กับเวลาปัจจุบัน(t_1) ดังนั้นเราจะสามารถคำนวณหาพื้นที่คาดหวังซึ่งเป็นตำแหน่งพื้นที่คาดหมายของโมไบล์โฮสต์ปลายทาง L เมื่อเวลา t_1 ในมุมมองของโมไบล์โฮสต์ต้นทาง S ได้ โดยสมมติให้ v เป็นความเร็วของโมไบล์โฮสต์ปลายทาง L ที่มีการเคลื่อนที่ ดังนั้นโมไบล์โฮสต์ L จะมีพื้นที่คาดหวังอยู่ในรัศมีคือ $v(t_1-t_0)$ โดยมีศูนย์กลางอยู่ที่ L พื้นที่คาดหวังนี้เป็นพื้นที่ที่เราคาดว่าโมไบล์โฮสต์ปลายทาง L จะอยู่ภายในพื้นที่คาดหวังนี้ ดังแสดงในรูปที่ 3



รูปที่ 3: พื้นที่คาดหวัง

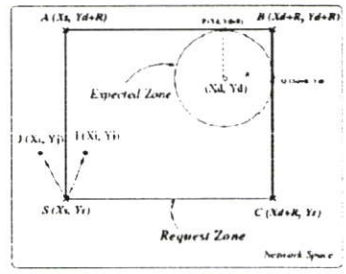
จากรูปที่ 3 เราจะเห็นพื้นที่คาดหวังในรูปที่ 3(a) ที่มีรัศมีเป็น $v(t_1-t_0)$ และมีศูนย์กลางที่โมไบล์โฮสต์ปลายทาง L จากรูป (b) โมไบล์โฮสต์ต้นทาง S รู้ว่าโมไบล์โฮสต์ปลายทาง L เคลื่อนที่ไปทางทิศเหนือ โมไบล์โฮสต์ต้นทาง S จะทำการกำหนดพื้นที่คาดหวังของโมไบล์โฮสต์ปลายทาง L เพียงแค่พื้นที่ครึ่งวงกลมเท่านั้น

2.2.2 พื้นที่การส่งต่อเมสเสสร้องขอเส้นทาง (Request Zone)

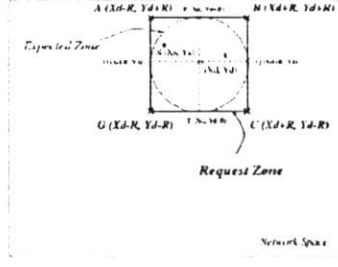
กระบวนการต่อไปจะเป็นการหาพื้นที่การส่งต่อเมสเสสร้องขอเส้นทาง (Request zone) ซึ่งเป็นพื้นที่ทั้งหมดที่จะถูกส่งต่อเมสเสสการร้องขอเส้นทาง โดยกำหนดให้พื้นที่คาดหวังอยู่ภายในพื้นที่การส่งต่อเมสเสสร้องขอเส้นทางด้วย ส่วนพื้นที่นอกเหนือจากพื้นที่การส่งต่อเมสเสสการร้องขอเส้นทาง (request zone) หากโมไบล์โฮสต์ระหว่างเส้นทาง ได้รับและพบว่าตนเองมีตำแหน่งของโมไบล์โฮสต์อยู่ภายนอกพื้นที่การส่งต่อเมสเสสร้องขอเส้นทางก็จะทิ้งเมสเสส ดังกล่าวไป ซึ่งส่วนนี้เป็นส่วนของการจำกัดไม่ให้มีการบรอดคาสท์เมสเสสการร้องขอเส้นทางออกไปทั้งเครือข่าย

การคำนวณหาพื้นที่การส่งต่อเมสเสสการร้องขอเส้นทาง (Request zone) ในรูปแบบที่นำมาพิจารณาดังที่แสดงในรูปที่ 4 ซึ่งได้เลือกพื้นที่การส่งต่อเมสเสสร้องขอเส้นทางเป็นกรอบสี่เหลี่ยมดังแสดงในรูป ที่ 4(a) และ 4(b) ตามลำดับ โดยที่แบ่งออกเป็นอีก 2 รูปแบบคือกรณีโมไบล์โฮสต์ต้นทางอยู่ภายนอกพื้นที่คาดหวัง (รูป ที่ 4(a)) และโมไบล์โฮสต์ต้นทางอยู่ภายในพื้นที่คาดหวังดังแสดงในรูป ที่ 4 (b)

สมมติว่าโมไบล์โฮสต์ต้นทาง S รู้ตำแหน่งของโมไบล์โฮสต์ปลายทาง D ว่าเคยอยู่ ณ ตำแหน่งเป็น X_d,Y_d เมื่อเวลา t_0 ที่ผ่านมา และเมื่อเวลา t_1 โมไบล์โฮสต์ต้นทาง S ได้ทำการค้นหาเส้นทางไปยังโมไบล์โฮสต์ปลายทาง D กำหนดให้โมไบล์โฮสต์ D เคลื่อนที่ด้วยความเร็ว v ดังนั้นเมื่อเวลา t_1 โมไบล์โฮสต์ต้นทาง S จะสามารถคำนวณหาพื้นที่คาดหวังของโมไบล์โฮสต์ปลายทาง D ได้โดยมีรัศมีเท่ากับ $R = v(t_1-t_0)$ โดยมีศูนย์กลางที่ X_d,Y_d สำหรับพื้นที่การส่งต่อเมสเสสจะเป็นลักษณะของสี่เหลี่ยมผืนผ้าโดยมีโมไบล์โฮสต์ต้นทาง S อยู่ภายในพื้นที่การส่งต่อเมสเสส



รูปที่ 4(a): Request zone scheme I, source node outside the expected zone



รูปที่4(b): Request zone scheme I, source node inside the expected zone

เมื่อโมไบล์โฮสต์ต้นทาง S ทำการบรอดคาสท์เมสเสสการร้องขอเส้นทาง โมไบล์โฮสต์ระหว่างทางที่ได้รับเมสเสสการร้องขอเส้นทาง จะตรวจสอบกับตำแหน่งของโมไบล์

โฮสต์ตนเองว่ามีตำแหน่งอยู่ภายในพื้นที่การส่งต่อแพสเสสหรือไม่ ถ้าไม่จะทำการทิ้งแพสเสสที่ได้รับ แต่ถ้าพบว่าตำแหน่งของตนอยู่ภายในพื้นที่การส่งต่อแพสเสสก็จะส่งต่อแพสเสสการร้องขอเส้นทางไปยังทิศทางของพื้นที่คาดหวังได้ ดังตัวอย่างในโมไบล์โฮสต์ J ซึ่งเป็นโมไบล์โฮสต์ระหว่างทาง มีตำแหน่ง เป็น X_j,Y_j เมื่อได้รับแพสเสสการร้องขอเส้นทางและพบว่าตำแหน่งของโมไบล์โฮสต์ตนอยู่นอกพื้นที่การส่งต่อแพสเสสการร้องขอเส้นทางก็จะไม่ส่งต่อแพสเสจที่ได้รับมา แต่สำหรับโมไบล์โฮสต์ระหว่างทาง I ซึ่งมีตำแหน่งเป็น X_i,Y_i พบว่าตำแหน่งของตนเองอยู่ภายในพื้นที่การส่งต่อแพสเสส จะทำการส่งต่อแพสเสสการร้องขอเส้นทางทิศทางไปยังพื้นที่คาดหวังต่อไป

จากวิธีของแอลเออาร์ดังกล่าวเราสามารถนำวิธีการค้นหาเส้นทางแบบแอลเออาร์มาปรับปรุงในกระบวนการค้นหาเส้นทางของโปรโตคอลเอบีอาร์เพื่อลดการใช้แบนด์วิธให้น้อยลง

3. อัลกอริทึมที่นำเสนอ

เมื่อพิจารณาโดยใช้ข้อมูลตำแหน่งของโมไบล์โฮสต์ในการจำกัดพื้นที่การค้นหาเส้นทางของเอบีอาร์ดังแบบโดยได้เพิ่มส่วนของแอลเออาร์เข้าไปโดยให้มีส่วนการคำนวณหาพื้นที่การส่งต่อแพสเสส (request zone) ในช่วงของการทำ BQ-Reply โดยแต่ละโมไบล์โฮสต์จะต้องมีตารางตำแหน่ง (Location Table) อยู่ที่แต่ละโมไบล์โฮสต์เพื่อใช้เก็บข้อมูลตำแหน่งของโมไบล์โฮสต์ต่างๆ ในเครือข่าย เราเรียกโปรโตคอลที่เรานำเสนอนี้ว่า โปรโตคอลเอ็มเอบีอาร์ (Modified-Associativity-Based Routing, MABR) สำหรับการทำงานในช่วง BQ และ Reply จะเป็นดังต่อไปนี้

Algorithm for Intermediate Node

If intermediate node I gets route request message. Then
  If location information available in packet. Then
    - Update source and destination positions with timestamps to its location information table if available in the packet
    - Compute expected zone and request zone by source, destination positions and timestamps
    - Get position itself
    If position node I is in request zone. Then
      - Rebroadcast route request message
    Else
      - Drop packet
    End If.
  Else
    - Rebroadcast route request message
  End If
Else If intermediate node I gets reply message. Then
  If location information available in packet. Then
    - Update destination position with timestamp to its location information table
  End If
  If its node's ID is one of a reply route. Then
    - Make valid route to route cache.
    - Forward reply packet backward to upper-stream node referred from a reply route.
  Else
    - Drop packet
  End If
End if

Algorithm for Source Node

If source node S wants to communicates to destination node D, Then
  - Lookup destination position in location information table
  If destination position is empty and want to run the original ABR. Then
    - Broadcast route request message uses normal flooding mode
  Else If destination position is empty, try MABR. Then
    - Include its position and timestamp to route request message
    - Broadcast route request message uses request zone
  Else
    - Include its position, destination position, and timestamps to route request message
    - Broadcast route request message uses request zone
  End If
End If

Algorithm for Destination Node

If destination node D gets the route request messages. Then
  If location information available in packet. Then

- Update source and destination positions with timestamps to its location information table
- Select a route uses route selection algorithm(ABR selection algorithm)
- Add position and timestamp itself to reply packet
- Send reply packet back to the source using selected path

Else

- Select a route uses route selection algorithm
- Send reply packet back to the source using selected path

End If

End If

4. การจำลองการทำงานของโปรโตคอลเอ็มเอบีอาร์

ผลการจำลองการทำงานนั้นได้ใช้ Network Simulator version 2.1b9a และ 2.27 (NS-2) [3] โดยจำลองการทำงานโปรโตคอลเดิมเอบีอาร์ (MABR) เทียบกับเอบีอาร์ (ABR)

4.1 สภาวะแวดล้อมในการทดสอบ

สภาพแวดล้อมที่ใช้ในการจำลองการทำงานได้แสดงไว้ในตารางที่ 1

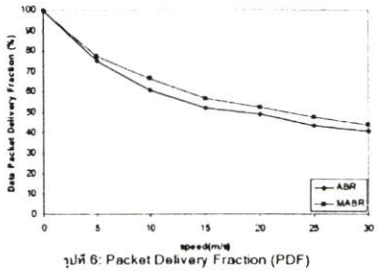
ตารางที่ 1: พารามิเตอร์ในการจำลองการทำงาน

จำนวนโมไบล์โฮสต์	50
พื้นที่	1000*1000 ตารางเมตร
ลักษณะการเคลื่อนที่	Uniform distribution
ความเร็วของโมไบล์โฮสต์	0 – 30 เมตรต่อวินาที
เวลาจำลองการทำงาน	1800 วินาที
หน่วยความจำชั่วคราว (Interface Queue)	64 แพคเกจ
จำนวนข้อมูลทดสอบ	20 คู่
ขนาดแพคเกจ	64 ไบท์
อัตราการส่งข้อมูลทดสอบ	4 แพคเกจต่อวินาที
เวลาหยุดพัก(pause time)	5 วินาที
แบนด์วิธของเครือข่าย	2 ล้านบิทต่อวินาที

สำหรับรูปแบบการเคลื่อนที่ของโมไบล์โฮสต์สร้างขึ้นโดยใช้ เจเนอเรเตอร์ mobgen[4] เพื่อจำลองการเคลื่อนที่ของเครือข่าย โดยพารามิเตอร์ที่พิจารณาแบ่งออกเป็นคอนโทรลโอเวอร์เฮด (Control Overhead) หรือปริมาณข้อมูลในการค้นหาเส้นทางและดีเลย์ (Average End-to-End Delay) ที่เกิดจากการรับ-ส่งข้อมูลทดสอบ

4.2 ผลจากการจำลองการทำงาน

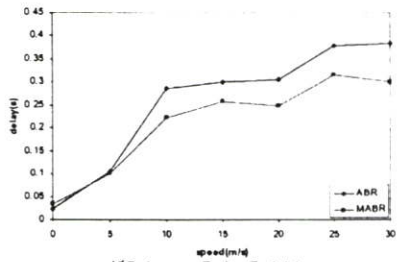
จากผลการจำลองการทำงานในรูปที่ 6 แสดงให้เห็นว่าเมื่อโมไบล์โฮสต์หยุดนิ่งไม่มีการเคลื่อนที่ การค้นหาเส้นทางถูกเรียกใช้เพียงช่วงเริ่มต้นของการเชื่อมต่อ



รูปที่ 6: Packet Delivery Fraction (PDF)

เนื่องจากโมไบล์โฮสต์ต่างๆไม่มีการเคลื่อนที่ เสถียรภาพการเชื่อมต่อระหว่างโมไบล์โฮสต์ค่อนข้างสูง การค้นหาเส้นทางอาจจะกระทำเพียงครั้งเดียว กราฟอัตราการรับส่งข้อมูลทดสอบนั้นจึงไม่ต่างกัน จะเห็นว่าอัตรา PDF มีค่าใกล้เคียง 100% ทุกความสามารถในการรับส่งข้อมูลต่างทดสอบเป็นไปได้อย่างราบรื่น แต่เมื่อโมไบล์โฮสต์มีการเคลื่อนที่ด้วยความเร็วที่สูงขึ้น ค่า PDF ก็จะลดลงเนื่องจากการเคลื่อนที่ทำให้เส้นทางการส่งข้อมูลเสียหายบ่อยมากขึ้น ทำให้ต้องเริ่มกระบวนการค้นหาเส้นทางใหม่บ่อยครั้ง ยิ่งความเร็วของโมไบล์โฮสต์สูงขึ้นก็จะยิ่งส่งผลให้โมไบล์โฮสต์ต่างๆใช้เวลาในการคอบสนองต่อการค้นหาเส้นทางเนื่องมาจากการเพิ่มขึ้นของความเสียหายที่เกิดขึ้นกับเส้นทาง ตามความเร็วของโมไบล์โฮสต์ แบนด์วิธส่วนหนึ่งที่ถูกใช้ไปในการค้นหาเส้นทางส่งผลให้อัตราการส่งข้อมูลต่างนั้นลดลง และการที่โมไบล์โฮสต์ต้องตอบสนองต่อเมสเสสการร้องขอเส้นทาง ทำให้โมไบล์โฮสต์ที่ได้รับเมสเสสนี้ต้องหยุดการสื่อสารข้อมูลที่กำลังติดต่ออยู่ชั่วคราว ข้อมูลต่างจึงอาจถูกดรอประหว่างการรอเส้นทางการส่งข้อมูลใหม่หรือตอบสนองต่อการค้นหาเส้นทาง

สำหรับเอ็มเอบีอาร์มีการจำกัดการค้นหาเส้นทางให้อยู่แต่ในพื้นที่การส่งต่อแพสเสสการร้องขอเส้นทางเท่านั้น ผลที่ได้ทำให้สามารถเพิ่มค่าอัตราการรับส่งข้อมูลมากขึ้นได้สูงถึงประมาณ 5%



รูปที่ 7: Average End-to-End delay

เมื่อโมไบล์โฮสต์หยุดนิ่งที่ความเร็วเป็นศูนย์ ดังที่แสดงในรูปที่ 7 ค่าเฉลี่ยดีเลย์ในการส่งข้อมูลทดสอบของโปรโตคอลเอบีอาร์และโปรโตคอลเอบีอาร์ที่เราได้ปรับปรุง มีค่าดีเลย์ใกล้เคียงกัน ต่อเมื่อโมไบล์โฮสต์เริ่มมีการเคลื่อนที่ด้วยความเร็วที่สูงขึ้น ทำให้จำนวนเส้นทางเกิดความเสียหายบ่อยขึ้นตามความเร็วของโมไบล์โฮสต์ โปรโตคอลทั้งเอบีอาร์และเอ็มเอบีอาร์ต้องมีการค้นหาเส้นทางใหม่อยู่ตลอดเวลาเพื่อทดแทนเส้นทางเดิม เนื่องจากเมื่อโมไบล์โฮสต์เคลื่อนที่ด้วยความเร็วที่สูงขึ้น ทำให้ปริมาณข้อมูลการร้องขอเส้นทางที่วิ่งอยู่ในเครือข่ายเพิ่มขึ้นตามความเร็วในการเคลื่อนที่ ทำให้ความสามารถในการส่งข้อมูลทดสอบลดลงเนื่องจากแบนด์วิธถูกใช้ไปเพื่อการรอและค้นหาเส้นทางใหม่ซึ่งก่อให้เกิดดีเลย์ที่สูงขึ้นซึ่งเรียกได้ว่าเป็น Queuing delay หลังจากที่เราได้ปรับปรุงโปรโตคอลเอ็มเอบีอาร์พบว่าสามารถจะลดดีเลย์ในการส่งข้อมูลทดสอบได้มากขึ้นถึง 30% เมื่ออยู่ที่ความเร็ว 30 เมตรต่อวินาที

5. สรุป

งานวิจัยนี้เราได้เสนอวิธีการการค้นหาเส้นทางเพื่อให้มีประสิทธิภาพสูงขึ้น โดยนำข้อมูลตำแหน่งของโมไบล์โฮสต์มาพิจารณาปรับปรุงโปรโตคอลค้นหาเส้นทางเอบีอาร์ (Associativity-Based Routing, ABR)[1] ในส่วนของการค้นหาเส้นทาง เนื่องจากกระบวนการค้นหาเส้นทางในเอบีอาร์จะทำการส่งข้อมูลเพื่อร้องขอเส้นทางออกไปทั่วเครือข่าย ทำให้โมไบล์โฮสต์ทั้งหมดในเครือข่ายที่ไม่มีความเกี่ยวข้องกับเส้นทาง ต้องหยุดการทำงานและคอยสนองต่อแพสเสสการร้องขอเส้นทาง ทำให้เกิดโอเวอร์เฮดและดีเลย์ที่สูงขึ้นและทำให้แบนด์วิธในการรับส่งข้อมูลลดลง เราเรียกวิธีการที่เรานำเสนอนี้ว่า โปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์ (Modified Associativity-Based Routing, MABR)

จากผลการจำลองการทำงาน เราพบว่าโปรโตคอลค้นหาเส้นทางเอ็มเอบีอาร์สามารถเพิ่มอัตราการส่งข้อมูลให้สูงขึ้น และทำให้ดีเลย์ในการรับส่งข้อมูลลดลงเมื่อเทียบกับโปรโตคอลค้นหาเส้นทางเอบีอาร์

6. เอกสารอ้างอิง

[1] C.-K. Toh, "Associativity-Based Routing For Ad Hoc Mobile Networks," Wireless Perl. Commun. J., Special Issue on Mobile Networking and Computing Systems, Kluwer Academic, vol. 4, no. 2, Mar. 1997, pp.103-39

[2] Y.-B. Ko and N.H. Vaidya, "Location-Aided Routing in Mobile Ad Hoc Networks",Tech. Rep. 98-012, CS Dept., Texas A&M University, June 1998.

[3] The Network Simulator ns-2, available online at http://www.isi.edu/nsnam/ns

[4] T. Camp, J. Boleng, and V. Davies, A Survey of Mobility Models for Ad Hoc Network Research, Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, pp. 483-502, 2002

ประวัติผู้เขียน

นายมนูญ คัมมกสิกิจ เกิดเมื่อวันที่ 4 พฤศจิกายน พ.ศ.2522 ที่จังหวัดราชบุรี สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้าคอมพิวเตอร์ จากภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าฯ พระนครเหนือ ในปีการศึกษา 2544 และเข้าศึกษาต่อในระดับปริญญาโท หลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีการศึกษา 2545 มีความสนใจด้านเครือข่ายไร้สาย และระบบปฏิบัติการลินุกซ์