การหาเส้นทางส่งข้อมูลในเครือข่ายไร้สายโดยวิธี Predicted Multipath Label Switching Protocol

นาวาอากาศตรี วิบูลย์ คูบูรณ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์)
คณะสถิติประยุกต์
สถาบันบัณฑิตพัฒนบริหารศาสตร์
2551

การหาเส้นทางส่งข้อมูลในเครือข่ายไร้สายโดยวิธี Predicted Multipath Label Switching Protocol

นาวาอากาศตรี วิบูลย์ คูบูรณ์

คณะสถิติประยุกต์

คณะกรรมการสอบวิทยานิพนธ์ ได้พิจารณาแล้วเห็นสมควรอนุมัติให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์)

ผู้ช่วยศาสตราจารย์ ..... ประธานกรรมการ
(ดร. โอม ศรนิล)

รองศาสตราจารย์ ..... กรรมการ และอาจารย์ที่ปรึกษาวิทยานิพนธ์
(ดร. พิพัฒน์ หิรัณย์วาณิชชากร)

ผู้ช่วยศาสตราจารย์ ..... กรรมการ
(ดร. อภิรักษ์ จิรายุสกุล)

รองศาสตราจารย์ ..... คณบดี
(ดร. สุรพงศ์ เอื้อวัฒนามงคล)
วันที่ 28 เดือน พฤษภาคม พ.ศ.2552

บทคัดย่อ

ชื่อวิทยานิพนธ์	การหาเส้นทางส่งข้อมูลในเครือข่ายไร้สายโดยวิธี Predicted Multipath Label Switching Protocol
ชื่อผู้เขียน	นาวาอากาศตรี วิบูลย์ คูบูรณ์
ชื่อปริญญา	วิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์)
ปีการศึกษา	2551

งานวิจัยนี้เป็นการนำเสนอโปรโตคอลการหาเส้นทางส่งข้อมูลในเครือข่ายไร้สาย ซึ่งใช้วิธีการของ Predicted Multipath Routing Protocol (PMP) ในการหาเส้นทางเนื่องจากเส้นทางที่ได้จากโปรโตคอลนี้มีความคงทนและประสิทธิภาพการส่งข้อมูลสูง และใช้หลักการของ Label Switching ในการนำส่งข้อมูลซึ่งจะทำให้การส่งข้อมูลทำได้รวดเร็วเนื่องจากช่วยทำให้ Header ของ Packet ข้อมูลมีขนาดสั้นลงอีกทั้งยังสนับสนุนการจัดการคุณภาพบริการการส่งข้อมูลได้อีกด้วย

จากการทดสอบประสิทธิภาพการทำงานของโปรโตคอลที่นำเสนอกับ PMP โดยเปรียบเทียบค่าสัดส่วนของปริมาณข้อมูลที่ปลายทางรับได้สำเร็จต่อปริมาณข้อมูลที่ส่งจากต้นทาง (Packet Delivery Ratio), ปริมาณข้อมูลที่หายไปในระหว่างการส่งข้อมูล (Number of Packets Dropped) และสัดส่วนของปริมาณข้อมูลที่ปลายทางรับได้สำเร็จต่อปริมาณข้อมูลที่ทุกโหนดในเครือข่ายส่งออกมา (Network Throughput) ผลการทดสอบชี้ให้เห็นว่าโปรโตคอลที่นำเสนอนั้นมีประสิทธิภาพสูงกว่า PMP

ABSTRACT

Title of Thesis	Predicted Multipath Label Switching Protocol for Wireless Ad Hoc Network
Author	Squadron Leader Wiboon Kuboon
Degree	Master of Science (Computer Science)
Year	2008

This research presents a routing protocol in wireless ad hoc network by using the Predicted Multipath Routing Protocol (PMP) for route discovery, because this protocol provides robustness and high effectiveness path. As for packet forwarding, Label switching technique is used to get high performance, because by using label in searching for route in router is faster than by using IP address, and header of sending packet is shorter than many other protocols. Furthermore, Label switching also supports Qos in data transfer.

Packet delivery ratio, number of packets dropped and network throughput can be shown as the results of this protocol efficiency, compares to PMP efficiency. The results show that Predicted Multipath Label Switching Protocol (PMLS) has better performance than PMP.

กิตติกรรมประกาศ

วิทยานิพนธ์เรื่องการหาเส้นทางส่งข้อมูลในเครือข่ายไร้สายโดยวิธี Predicted Multipath Label Switching Protocol นี้ สำเร็จลุล่วงได้ด้วยดี เนื่องมาจากบุคคลหลายท่านที่ได้กรุณาช่วยเหลือให้ข้อมูล ข้อเสนอแนะ คำปรึกษาแนะนำ ความคิดเห็นและกำลังใจ

ผู้เขียนขอกราบขอบพระคุณ รองศาสตราจารย์ ดร. พิพัฒน์ หิรัณย์วณิชชากร ที่ได้ให้คำแนะนำ ชี้แนะแนวทาง และตรวจสอบวิทยานิพนธ์ในทุกขั้นตอน

ขอขอบพระคุณ คุณ ศุภโชติ เลิศวรธรรม ที่ได้ชี้แนะและให้คำปรึกษาในการออกแบบโปรโตคอล PMP และสนับสนุนโปรแกรมการจำลองของโปรโตคอลดังกล่าวเพื่อใช้ในการเปรียบเทียบประสิทธิภาพการทำงาน

ขอขอบพระคุณคณาจารย์ของคณะสถิติประยุกต์ทุกท่าน ที่ประสิทธิ์ประสาทวิชาและถ่ายทอดความรู้ให้แก่ผู้ศึกษา และขอขอบคุณเจ้าหน้าที่คณะสถิติประยุกต์ ที่ได้ให้ความช่วยเหลือในการติดต่อประสานงานเป็นอย่างดี

ขอขอบพระคุณ ผู้บังคับบัญชาและผู้ร่วมงานของแผนกวิศวกรรมโยธา กองออกแบบและก่อสร้าง กองวิทยาการ กรมช่างโยธาทหารอากาศ กองทัพอากาศ ที่ได้ให้โอกาสและสนับสนุนการศึกษา

ขอขอบคุณนักศึกษาร่วมรุ่น รุ่นพี่ และรุ่นน้องในคณะสถิติประยุกต์ที่ได้ให้ความช่วยเหลือและประสานงานตลอดช่วงเวลาที่ได้ศึกษาอยู่ที่คณะสถิติประยุกต์

ท้ายสุดนี้ ขอกราบขอบพระคุณบิดา มารดา และญาติพี่น้องที่ได้ช่วยส่งเสริมสนับสนุนและเป็นกำลังใจให้แก่ผู้จัดทำวิทยานิพนธ์ตลอดมา

น.ต. วิบูลย์ คูบูรณ์

พฤษภาคม 2552

สารบัญ

หน้า

บทคัดย่อ (5)

ABSTRACT (6)

กิตติกรรมประกาศ (7)

สารบัญ (8)

สารบัญตาราง (10)

สารบัญภาพ (11)

บทที่ 1 บทนำ 1

1.1 ความสำคัญของปัญหา 1

1.2 วัตถุประสงค์ของการวิจัย 3

1.3 ขอบเขตของการวิจัย 3

1.4 ขั้นตอนการวิจัย 4

1.5 ประโยชน์ที่จะได้รับจากการวิจัย 4

บทที่ 2 การทบทวนวรรณกรรม 5

2.1 หลักการทำงานของโปรโตคอล PMP 5

2.1.1 โครงสร้างของ Route Table 6

2.1.2 โครงสร้างของ Packet ต่าง ๆ 6

2.1.3 กระบวนการหาเส้นทาง 8

2.1.4 กระบวนการนำส่งข้อมูล 9

2.1.5 กระบวนการบำรุงรักษาเส้นทาง 9

2.2 งานวิจัยและหลักการที่เกี่ยวข้องกับโปรโตคอล Label Switching 9

2.2.1 Multi-Protocol Label Switching (MPLS) 10

2.2.2 An On-Demand Routing Protocol in Ad Hoc Network Using Label Switching (ORAL) 10
2.2.3 Label Switching Multipath Routing (LSMR) 16
บทที่ 3 การทำงานของโปรโตคอล PMLS 25
3.1 หลักการของ PMLS 25
3.2 โครงสร้างของ Route Table 32
3.3 โครงสร้างของ Packet ต่าง ๆ 33
3.4 กระบวนการหาเส้นทาง 36
3.5 กระบวนการนำส่งข้อมูล 39
3.6 กระบวนการบำรุงรักษาเส้นทาง 40
3.7 ตัวอย่างกระบวนการทำงาน 42
บทที่ 4 การจำลองและการเปรียบเทียบประสิทธิภาพของการทำงาน 50
4.1 โปรแกรมจำลองการทำงาน 50
4.2 สภาวะแวดล้อมของการจำลองการทำงาน 51
4.3 ตัววัดเพื่อใช้ในการเปรียบเทียบประสิทธิภาพการทำงาน 52
4.4 ผลการทดสอบและเปรียบเทียบประสิทธิภาพการทำงาน 52
บทที่ 5 สรุปผลการวิจัยและข้อเสนอแนะ 59
5.1 สรุปผลการวิจัย 59
5.2 ข้อเสนอแนะ 61

บรรณานุกรม 62
ประวัติผู้เขียน 65

สารบัญตาราง

ตารางที่ **หน้า**

3.1 แสดง Route Table บางส่วนของโหนด ต่าง ๆ 45

3.2 แสดง Route Table ของโหนด 3 หลังจากการทำ Local Repair เสร็จสิ้น 48

3.3 แสดง Route Table ของโหนด 5 หลังจากการทำ Local Repair เสร็จสิ้น 48

3.4 แสดง Route Table ของโหนด 3 หลังจากการทำ Route Maintenance เสร็จสิ้น 49

3.5 แสดง Route Table ของโหนด 5 หลังจากการทำ Route Maintenance เสร็จสิ้น 49

สารบัญภาพ

ภาพที่ **หน้า**

2.1 โครงสร้าง Route Table ของโปรโตคอล PMP 6

2.2 โครงสร้าง Route Request Packet (RREQ) ของโปรโตคอล PMP 6

2.3 โครงสร้าง Route Reply Packet (RREP) ของโปรโตคอล PMP 7

2.4 โครงสร้าง Route Error Packet (RERR) ของโปรโตคอล PMP 7

2.5 โครงสร้าง Data Packet ของโปรโตคอล PMP 8

2.6 โครงสร้าง Route Table ของโปรโตคอล ORAL 11

2.7 โครงสร้างหลัก ๆ ของ RREQ, RREP, RUPP และ Data Packet ของโปรโตคอล ORAL 12

2.8 โครงสร้าง IDM Table และ ILM Table ของโปรโตคอล LSMR 17

2.9 โครงสร้าง PREQ, PREP และ Data Packet ของโปรโตคอล LSMR 18

2.10 การส่ง Path Request Packet (PREQ) และการสร้าง Table ของโปรโตคอล LSMR 20

2.11 การส่ง Path Reply Packet (PREP) และการสร้าง Table ของโปรโตคอล LSMR 21

2.12 กระบวนการทำ Local Repair ของโปรโตคอล LSMR 23

3.1 ความสัมพันธ์ระหว่าง RSSI และระยะทาง 27

3.2 การคาดคะเนความแรงของสัญญาณจากข้อมูลในอดีตในกรณีที่การเปลี่ยนแปลงความแรงของสัญญาณกำลังเพิ่มขึ้น 28

3.3 การคาดคะเนความแรงของสัญญาณจากข้อมูลในอดีตในกรณีที่การเปลี่ยนแปลงความแรงของสัญญาณกำลังลดลง 28

3.4 เส้นทางที่ไม่มีโหนดระหว่างทาง 31

3.5 เส้นทางที่มีโหนดระหว่างทางจำนวน 1 โหนด 31

3.6 โครงสร้าง Route Table ของโปรโตคอล PMLS 33

3.7 โครงสร้าง Route Request Packet (RREQ) ของโปรโตคอล PMLS 33

3.8 โครงสร้าง Route Reply Packet (RREP) ของโปรโตคอล PMLS 34
3.9 โครงสร้าง Data Packet ของโปรโตคอล PMLS 34
3.10 โครงสร้าง Route Error Packet (RERR) ของโปรโตคอล PMLS 35
3.11 โครงสร้าง Local Repair Request Packet (LREQ) ของโปรโตคอล PMLS 35
3.12 โครงสร้าง Local Repair Reply Packet (LREP) ของโปรโตคอล PMLS 36
3.13 แสดง Source Route Information, Signal Strength และ Packet Loss Ratio ที่ได้รับในแต่ละ Hop 43
3.14 แสดงรูปร่างเครือข่ายก่อนโหนด 8 เสียหาย 46
3.15 แสดงรูปร่างเครือข่ายหลังจากโหนด 8 เสียหาย 47
4.1 Packet Delivery Ratio ของ PMLS และ PMP 52
4.2 Number of Packets Dropped ของ PMLS และ PMP 53
4.3 Network Throughput ของ PMLS และ PMP 54
4.4 Packet Delivery Ratio ของ PMLS1และ PMLS2 55
4.5 Number of Packets Dropped ของ PMLS1และ PMLS2 55
4.6 Network Throughput ของ PMLS1และ PMLS2 56
4.7 Routing Overhead ของ PMLS1และ PMLS2 56
4.8 Packet Delivery Ratio ของ PMLS3 และ PMLS4 57
4.9 Number of Packets Dropped ของ PMLS3 และ PMLS4 57
4.10 Network Throughput ของ PMLS3 และ PMLS4 58

บทที่ 1

บทนำ

ในบทนี้จะกล่าวถึงความสำคัญของปัญหา วัตถุประสงค์ของการวิจัย ขอบเขตของการวิจัย ขั้นตอนการวิจัย และประโยชน์ที่จะได้รับจากการวิจัย โดยมีรายละเอียดดังต่อไปนี้

1.1 ความสำคัญของปัญหา

Mobile Ad Hoc Network (MANET) เป็นเครือข่ายไร้สายของอุปกรณ์เคลื่อนที่อาศัยการส่งข้อมูลต่อกันเป็นทอด ๆ แบบ Multi-Hop โดยโหนดแต่ละโหนดของการสื่อสารจะทำหน้าที่เป็นทั้งหน่วยประมวลผลและเราเตอร์ และไม่จำเป็นต้องมีระบบการจัดการจากศูนย์กลางทำให้มีความสะดวกในการทำงานเป็นอย่างมาก เหมาะสำหรับภารกิจที่ต้องการความคล่องตัวสูง เช่น การค้นหาและกู้ภัย หรือการโจมตีและการถอนตัวอย่างฉับพลันของหน่วยทหารในสมรภูมิ นอกจากนั้นในเชิงพาณิชย์ยังเหมาะสำหรับการจัดงานที่ต้องการความคล่องตัวสูง เช่น งานประกวดโชว์ตัวสินค้าหรืองานจัดมหกรรมแบบชั่วครั้งชั่วคราวต่าง ๆ เพราะสามารถติดตั้งและถอนการติดตั้งได้อย่างง่ายดาย จากข้อดีดังกล่าวทำให้มีการตื่นตัวมุ่งเน้นพัฒนาเทคโนโลยีของเครือข่าย MANET กันเป็นอย่างมาก และจากที่ทราบกันดีอยู่ว่าประสิทธิภาพของการสื่อสารภายในเครือข่ายใด ๆ นั้นจะสูงสุดหากข้อมูลถูกส่งไปยังเป้าหมายที่ต้องการตามเส้นทางที่ดีที่สุดในขณะเวลาหนึ่ง ๆ ซึ่งหากเป็นเครือข่ายที่มีสายแล้วการหาเส้นทางที่ดีที่สุด เพื่อใช้ในการส่งข้อมูลสามารถกระทำได้โดยง่าย รวมทั้งเส้นทางที่หามาได้นั้นยังมีโอกาสเสียหายน้อย แต่สำหรับ MANET แล้วเส้นทางที่ใช้ในการส่งข้อมูลนั้นมีโอกาสที่จะเสียหายได้มาก เนื่องจากการเชื่อมโยงซึ่งกันและกันของแต่ละโหนดภายในเครือข่ายจะมีการเปลี่ยนแปลงอยู่ตลอดเวลา นอกจากนั้นโหนดแต่ละโหนดยังมีข้อจำกัดในเรื่องของพลังงาน และความสามารถในการประมวลผลอีกด้วยจึงจำเป็นต้องมีโปรโตคอลในการหาเส้นทางและนำส่งข้อมูลที่มีประสิทธิภาพที่สร้างขึ้นเฉพาะเพื่อให้สอดคล้องเหมาะสมกับธรรมชาติของเครือข่ายไร้สาย สำหรับโปรโตคอลในการหาเส้นทางและนำส่งข้อมูลของเครือข่าย MANET ในปัจจุบันได้มีผู้คิดค้นและพัฒนาขึ้นเป็นจำนวนมากโดยสามารถแยกออกเป็นกลุ่มหลัก ๆ ได้สามกลุ่มตามลักษณะการทำงาน คือ

1. On-Demand หรือ Active จะเป็นโปรโตคอลที่จะหาเส้นทางก็ต่อเมื่อต้องการที่จะส่งข้อมูลเท่านั้น ได้แก่ (Johnson, Hu and Maltz, 2007: 1-107) โปรโตคอล Dynamic Source Routing (DSR) และ (Perkins, Belding-Royer and Das, 2003: 1-37) Ad Hoc On-Demand Distance Vector Routing (AODV) เป็นต้น

2. Proactive จะเป็นโปรโตคอลที่มีการปรับปรุงเส้นทางให้มีความทันสมัยอยู่เสมอเป็นระยะ ๆ แม้ว่าจะไม่ได้ใช้เส้นทางเหล่านั้นในการส่งข้อมูลก็ตาม ได้แก่ (Perkins and Bhagwat, 1994: 234-244) โปรโตคอล Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV), (Ogier, Templin and Lewis, 2004: 1-46) Topology Broadcast Based on Reverse Path Forwarding (TBRPF) และ (Clausen and Jacquet, 2003: 1-75) Optimized Link State Routing (OLSR) เป็นต้น

3. Hybrid จะเป็นโปรโตคอลที่ทำงานในลักษณะผสมผสานโดยใช้หลักการทำงานของทั้งสองกลุ่มดังที่ได้กล่าวมาแล้วมาปรับใช้ร่วมกัน ได้แก่ (Ramasubramanian, Hass and Sirer, 2003: 303-314) โปรโตคอล Hybrid Adaptive Routing Protocol (SHARP), (Haas, 1997: 562-566) Zone Routing Protocol (ZRP) และ (Joa-Ng and Lu, 1999: 1415-1425) Zone-Based Hierarchical Link State Protocol (ZHLS) เป็นต้น

เนื่องจากข้อจำกัดของเครือข่าย MANET ดังที่ได้กล่าวมาโปรโตคอลที่ทำงานแบบ On-Demand หรือ Active จึงได้รับความนิยมมากกว่าเนื่องจากสูญเสีย Overhead น้อยกว่าแบบ Proactive

นอกจากการหาเส้นทางที่ดีที่สุดในการส่งข้อมูลแล้ว ใน MANET ยังมีความต้องการที่จะส่งข้อมูลไปยังเป้าหมายเดียวกัน โดยใช้เส้นทางที่แตกต่างกันมากกว่าหนึ่งเส้นทางและให้เหมาะสมกับประเภทและอัตราการส่งของข้อมูลนั้น ๆ เช่น (Lee and Gerla, 2001: 3201-3205) โปรโตคอล Split Multipath Routing (SMR) และ (Marina and Das, 2001: 14-23) Ad Hoc On Demand Multipath Distance Vector Routing (AOMDV) เป็นต้น ซึ่งจะมีผลดีทำให้สามารถลดการแน่นขนัดของข้อมูลในเครือข่ายได้ ดังนั้นในช่วงเวลาหลายปีที่ผ่านมาจึงได้มีการพัฒนาโปรโตคอลสำหรับหาเส้นทางและนำส่งข้อมูลแบบหลายเส้นทางซึ่งทำงานตามลักษณะของ On-demand เพื่อนำมาใช้ในเครือข่าย MANET เป็นจำนวนมากโดยเรียกโปรโตคอลกลุ่มนี้ว่า On-Demand Multipath Ad Hoc Network Routing Protocol แต่โปรโตคอลที่กล่าวมายังไม่มีโปรโตคอลใดเลยที่สนใจถึงประสิทธิภาพของเส้นทางที่จะนำมาใช้ในการส่งข้อมูลทำให้การทำงานไม่มีประสิทธิภาพเท่าที่ควร อย่างไรก็ตามเมื่อไม่นานมานี้ (Supachote Lertvorratham and Pipat Hiranvanichakorn, 2007: 1-7) ได้นำเสนอโปรโตคอล Predicted Multipath Routing Protocol (PMP) ขึ้น ซึ่งโปรโตคอลดังกล่าวนี้

ได้มุ่งเน้นไปที่ประสิทธิภาพของเส้นทางที่จะนำมาใช้ในการส่งข้อมูลโดยใช้ค่าความคงทนต่อความเสียหายกับค่าประสิทธิภาพของการส่งข้อมูลของเส้นทางมาใช้ในการหาเส้นทางที่ดีที่สุด เพื่อนำมาใช้ในการส่งข้อมูล และผู้นำเสนอโปรโตคอลนี้ชี้ให้เห็นว่าเส้นทางที่ได้จากโปรโตคอล PMP นั้นมีประสิทธิภาพดีกว่าเส้นทางที่ได้จากโปรโตคอล Dynamic Source Routing (DSR) และโปรโตคอล Split Multipath Routing (SMR) อย่างชัดเจน แต่อย่างไรก็ตามถึงแม้ว่าเส้นทางที่ได้จากโปรโตคอล PMP จะมีประสิทธิภาพสูงก็ตาม Header ของข้อมูลที่จะส่งไปยังปลายทางนั้นยังต้องใช้ Source Route ซึ่งประกอบไปด้วยบรรดา IP Address ของโหนดระหว่างทางต่าง ๆ จนถึงปลายทางของเส้นทางนั้น ๆ เหมือนกับโปรโตคอล DSR ซึ่งทำให้ประสิทธิภาพของการส่งข้อมูลไม่สูงเท่าที่ควร ดังนั้นหากสามารถลดขนาดของ Header เหล่านี้ลงโดยใช้วิธีการ Label Switching ซึ่งเป็นเทคโนโลยีซึ่งนำเสนอโดย (Rosen, Viswanathan and Callon, 2001: 1-61) คือโปรโตคอล Multi-Protocol Label Switching (MPLS) ที่กำลังนิยมใช้ในเครือข่ายแบบมีสายนั้น การสับเปลี่ยนเส้นทางเพื่อนำส่งข้อมูลไปยังปลายทางโดยใช้ Label แทน IP Address ของโปรโตคอลนี้มีความรวดเร็วเป็นอย่างมาก ทั้งยังสามารถลดขนาดของ Header ของข้อมูลลงได้อีกด้วย ในงานวิจัยนี้จึงได้นำหลักการในการหาเส้นทางของโปรโตคอล PMP มาผนวกไว้กับวิธีการนำส่งข้อมูลโดยใช้หลักการของ Label Switching เพื่อสร้างเป็นโปรโตคอลซึ่งมีเส้นทางที่คงทนและประสิทธิภาพการส่งข้อมูลของเส้นทางสูงรวมทั้งมีการนำส่งข้อมูลที่รวดเร็ว และสามารถสนับสนุนการจัดการคุณภาพบริการการส่งข้อมูลได้ พร้อมทั้งเสนอกระบวนการบำรุงรักษาเส้นทางเพื่อเพิ่มความคงทนของเส้นทางส่งข้อมูลและประสิทธิภาพของการส่งข้อมูล

1.2 วัตถุประสงค์ของการวิจัย

เพื่อสร้างโปรโตคอลหาเส้นทางส่งข้อมูลในเครือข่ายไร้สายที่มีความคงทนของเส้นทางและประสิทธิภาพการส่งข้อมูลที่ดี

1.3 ขอบเขตของการวิจัย

1. พัฒนา และออกแบบโปรโตคอลในการหาเส้นทางส่งข้อมูลสำหรับเครือข่ายไร้สาย
2. จำลองการทำงานเพื่อทดสอบประสิทธิภาพโปรโตคอลที่นำเสนอ ด้วย JIST-SWANS
3. เปรียบเทียบประสิทธิภาพ และวิเคราะห์การทำงานของโปรโตคอลที่ออกแบบกับโปรโตคอล PMP

1.4 ขั้นตอนการวิจัย

ขั้นตอนการทำวิจัยแบ่งออกเป็น 6 ขั้นตอนดังนี้

1. ศึกษาเอกสารและงานวิจัยที่เกี่ยวข้อง ได้แก่
   - ศึกษางานวิจัยต่าง ๆ ที่เกี่ยวข้องกับการทำงานของ PMP
   - ศึกษางานวิจัยต่าง ๆ ที่เกี่ยวข้องกับหลักการของ Label switching
2. ศึกษาและวิเคราะห์เพื่อหาแนวคิดต่าง ๆ ที่เกี่ยวข้องเพื่อพัฒนาโปรโตคอล Predicted Multipath Label Switching (PMLS)
3. ออกแบบ และพัฒนาโปรโตคอล PMLS
4. ทดสอบและประเมินประสิทธิภาพการทำงานของโปรโตคอล PMLS กับ PMP โดยพัฒนาโปรแกรมเพื่อจำลองการทำงานของโปรโตคอลทั้งสองด้วย JIST-SWANS
5. เปรียบเทียบประสิทธิภาพการทำงานกับ PMP
6. วิเคราะห์ สรุปผลการวิจัย และข้อเสนอแนะ

1.5 ประโยชน์ที่จะได้รับจากการวิจัย

1. ได้ความรู้ของการหาเส้นทางส่งข้อมูลในเครือข่ายไร้สาย
2. ได้ศึกษาการจำลองของการหาเส้นทางส่งข้อมูลในเครือข่ายไร้สาย
3. ได้โปรโตคอล PMLS ซึ่งมีความคงทนของเส้นทาง และประสิทธิภาพการส่งข้อมูลที่ดี

บทที่ 2

การทบทวนวรรณกรรม

โปรโตคอลในการหาเส้นทางส่งข้อมูล จะมีกระบวนการทำงานหลัก ๆ สามกระบวนการด้วยกันคือ กระบวนการหาเส้นทาง กระบวนการบำรุงรักษาเส้นทาง และกระบวนการนำส่งข้อมูล กระบวนการหาเส้นทางในเครือข่าย MANET ที่ทำงานแบบ On-Demand โดยทั่ว ๆ ไปแล้วโหนดที่ต้องการส่งข้อมูลจะตรวจสอบตัวเองว่ามีเส้นทางไปยังเป้าหมายหรือไม่ หากไม่มีจะส่งการร้องขอเส้นทางไปยังโหนดต่าง ๆ ที่อยู่ติดกับตนว่ามีเส้นทางที่ไปยังเป้าหมายหรือไม่ หากมีก็จะส่งเส้นทางกลับมาให้กับโหนดต้นทางนั้น หากไม่มีก็จะส่งต่อการร้องขอเส้นทางนั้นให้กับโหนดถัดไปต่อกันไปเป็นทอด ๆ จนกว่าจะได้รับเส้นทางที่ต้องการกลับมา หลังจากโหนดต้นทางได้รับเส้นทางที่ต้องการแล้วมันก็จะเก็บข้อมูลของเส้นทางนั้นไว้เพื่อใช้ในการส่งข้อมูลต่อไป ซึ่งเส้นทางที่ได้รับมานั้นอาจจะเก็บไว้ในรูปแบบ Route Cache หรือ Route Table ก็ได้ ทั้งนี้ขึ้นอยู่กับการออกแบบ เช่น โปรโตคอล DSR จะเก็บเส้นทางไว้ในรูปแบบของ Route Cache ส่วน AODV จะเก็บไว้ในรูปแบบของ Route Table เป็นต้น ส่วนกระบวนการนำส่งข้อมูลก็จะใช้ IP Address ของโหนดปลายทางเป็นดรรชนีในการค้นหา หรือนำข้อมูลของเส้นทางที่ค้นหาพบนั้นผนวกไปเป็น Header ของข้อมูลเพื่อส่งต่อไปตามเส้นทางนั้น ๆ ส่วนกระบวนการบำรุงรักษาเส้นทางนั้นจะแตกต่างกันไปขึ้นอยู่กับลักษณะของการทำงาน หากเป็นประเภท On-Demand หรือ Active ก็จะไม่ต้องมีการปรับปรุงเส้นทางให้ทันสมัยอยู่เสมอต่างกับประเภท Proactive และเพื่อให้เข้าใจถึงแนวคิดของโปรโตคอล PMLS ในหัวข้างข้างล่างนี้จะกล่าวถึงหลักการการทำงานที่สำคัญ ๆ ของโปรโตคอลต่าง ๆ ที่เราได้นำมาใช้เป็นแนวทางในการออกแบบและพัฒนาโปรโตคอลตัวใหม่ของเราเท่านั้น ดังนี้

2.1 หลักการทำงานของโปรโตคอล PMP

(Supachote Lertvorratham and Pipat Hiranvanichakorn, 2007: 1-7) ได้นำเสนอโปรโตคอล PMP ซึ่งเป็นโปรโตคอลที่ทำงานในลักษณะ On-Demand Routing โดยนำหลักการ Source Routing มาจาก DSR แต่มีแนวคิดว่าเส้นทางที่ดีที่สุดนั้นไม่ใช่เส้นทางที่มีจำนวน Hop น้อยที่สุดเสมอไปทั้งนี้ต้องขึ้นอยู่กับความคงทนต่อการเปลี่ยนแปลงของสภาพเครือข่ายเมื่อมี

การเคลื่อนที่ออกห่างจากกันของโหนดแต่ละโหนด ตลอดจนประสิทธิภาพของการรับ-ส่งข้อมูลของแต่ละ Hop ในเส้นทางจากต้นทางถึงปลายทาง โดยโปรโตคอลนี้จะอาศัยหลักการของ Regression ในการคาดคะเนความแรงของสัญญาณของโหนดที่ติดกัน เพื่อพิจารณาว่าเส้นทางที่หาได้มีความคงทนมากน้อยเพียงใดในช่วงเวลาหนึ่ง นอกจากนั้นยังมีการใช้ Probe Message ในการหาค่าเฉลี่ยของการสูญหายของข้อมูลระหว่างโหนดที่ติดกัน เพื่อคำนวณหาประสิทธิภาพการส่งข้อมูลของเส้นทางส่งข้อมูล

2.1.1 โครงสร้างของ Route Table

Dest	Rank	Route

ภาพที่ 2.1 โครงสร้าง Route Table ของโปรโตคอล PMP

หมายเหตุ: ฟิลด์ต่าง ๆ ของ Route Table มีความหมายดังนี้

- **Dest** หมายถึง Address ของโหนดปลายทาง
- **Rank** หมายถึง ระดับความสำคัญของเส้นทางที่ไปยังปลายทางหนึ่ง ๆ
- **Route** หมายถึง รายการ Address ของโหนดต่าง ๆ ตั้งแต่โหนดต้นทางจนถึงโหนดปลายทาง

2.1.2 โครงสร้างของ Packet ต่าง ๆ

2.1.2.1 Route Request Packet (RREQ)

Seq#	reqID	Dest	Source Route Information

ภาพที่ 2.2 โครงสร้าง Route Request Packet (RREQ) ของโปรโตคอล PMP

หมายเหตุ: ฟิลด์ต่าง ๆ ของ RREQ มีความหมายดังนี้

- **Seq#** หมายถึง หมายเลขลำดับการส่ง Route Request ของโหนดต้นทางซึ่งจะเพิ่มค่าทีละหนึ่งเมื่อต้องส่ง RREQ ซ้ำให้โหนดที่อยู่ติดกับตนหลังจากรอคอยอยู่เป็นเวลาค่าหนึ่งแล้วยังไม่ได้รับ Route Reply กลับมา เพื่อระบุว่าเป็น RREQ ล่าสุด
- **reqID** หมายถึง ตัวระบุความแตกต่างของ RREQ เพื่อป้องกันไม่ให้โหนดต่าง ๆ รับ Packet นี้ซ้ำและใช้ในการส่ง Route Reply ต่อ RREQ นี้

- **Dest** หมายถึง Address ของโหนดปลายทาง

- **Source Route Information** หมายถึง ข้อมูลต่าง ๆ ของเส้นทางซึ่งประกอบไปด้วยค่า Packet Loss Ratio, Signal Strength ของ Link ที่มาถึงโหนดต่าง ๆ และ Address ของโหนดต่าง ๆ ที่ RREQ นี้ได้ท่องผ่านมา จะสังเกตได้ว่า Source Route Information จะมีขนาดยาวเพิ่มมากขึ้นเรื่อย ๆ จนกระทั่งถึงโหนดปลายทาง

2.1.2.2 Route Reply Packet (RREP)

reqID	receiverAddr	senderAddr	Rank	Route

ภาพที่ 2.3 โครงสร้าง Route Reply Packet (RREP) ของโปรโตคอล PMP

หมายเหตุ: ฟิลด์ต่าง ๆ ของ RREP มีความหมายดังนี้

- **reqID** หมายถึง ตัวระบุการตอบกลับต่อ Route Request เพื่อให้โหนดต้นทางสามารถทราบว่า Route Request ที่ตนได้ส่งไปนั้นได้รับการตอบกลับมาเรียบร้อยแล้ว

- **receiverAddr** หมายถึง Address ของโหนดผู้รับ RREP นี้

- **senderAddr** หมายถึง Address ของโหนดผู้ส่ง RREP นี้

- **Rank** หมายถึง ค่าแสดงระดับความสำคัญของเส้นทางนี้

- **Route** หมายถึง เส้นทางที่สมบูรณ์ทั้งหมดของเส้นทางนี้ซึ่งจะประกอบไปด้วยบรรดา Address ของโหนดต่าง ๆ ตั้งแต่ต้นทางจนถึงปลายทาง

2.1.2.3 Route Error Packet (RERR)

receiverAddr	brokenAddr

ภาพที่ 2.4 โครงสร้าง Route Error Packet (RERR) ของโปรโตคอล PMP

หมายเหตุ: ฟิลด์ต่าง ๆ ของ RERR มีความหมายดังนี้

- **receiverAddr** หมายถึง Address ของโหนดผู้รับ RERR นี้

- **brokenAddr** หมายถึง Address ของโหนดคู่ใด ๆ ของ Link ที่มีความผิดพลาดเกิดขึ้น ซึ่งจะประกอบไปด้วยโหนดที่เป็นผู้เริ่มรายงานความเสียหายที่เกิดขึ้นกับโหนดที่เสียหาย

2.1.2.4 Data Packet

Source Route	Data

ภาพที่ 2.5 โครงสร้าง Data Packet ของโปรโตคอล PMP

หมายเหตุ: ฟิลด์ต่าง ๆ ของ Data Packet มีความหมายดังนี้

- **Source Route** หมายถึง รายการ Address ของโหนดต่าง ๆ ตั้งแต่โหนดถัดไปถึงโหนดปลายทาง
- **Data** หมายถึง ข้อมูลที่ส่ง

2.1.3 กระบวนการหาเส้นทาง (Route Discovery)

เมื่อโหนดต้นทางต้องการส่งข้อมูลไปยังโหนดปลายทาง และไม่มีเส้นทางในการส่งข้อมูล มันจะสร้าง Route Request Packet (RREQ) ซึ่งมีโครงสร้างดังภาพที่ 2.2 โดยแนบ IP Address ของมันเป็น Source Route Information ส่งให้กับบรรดาโหนดที่อยู่ติดกับตน หากโหนดเหล่านั้นไม่ใช่โหนดปลายทาง พวกมันจะแทรก IP Address ของตัวมันเอง ค่า Signal Strength และ ค่า Packet Loss Ratio ของ Link ที่มาถึงพวกมัน เข้าไปใน Source Route Information ของ RREQ ที่มันได้รับ และส่ง RREQ นี้ให้โหนดซึ่งอยู่ติดกับมันต่อไป หากโหนดถัดไปเหล่านั้นไม่ใช่โหนดปลายทางอีก พวกมันก็จะทำอย่างเดียวกัน และส่ง RREQ นี้ต่อไปเป็นทอด ๆ จนกระทั่งถึงโหนดปลายทาง เมื่อโหนดปลายทางได้รับ RREQ เหล่านี้ มันจะนำเส้นทางต่าง ๆ ซึ่งอาจจะมีมากกว่าหนึ่งเส้นทางมาคัดเลือกไว้จำนวนหนึ่งโดยใช้ค่า Degree of Path Availability (DA) ซึ่งเป็นดรรชนีบ่งบอกถึงความคงทนของเส้นทางในการคัดเลือกเส้นทางที่มีค่า DA สูงกว่าวิกฤต (Threshold) และนำเส้นทางที่คัดเลือกเอาไว้นั้นมาจัดระดับความสำคัญโดยใช้ค่า Estimate Path Throughput Value (ETV) ซึ่งเป็นดรรชนีบ่งบอกถึงประสิทธิภาพการส่งข้อมูลของเส้นทางในการพิจารณาระดับความสำคัญ หลังจากนั้นมันจะนำเส้นทางแต่ละเส้นทางนี้ใส่ใน Route Reply Packet (RREP) ซึ่งมีโครงสร้างดังภาพที่ 2.3 เพื่อส่งให้กับโหนดระหว่างทางต่าง ๆ ตามลำดับจนถึงโหนดต้นทางตามเส้นทางนั้น ๆ โดยโหนดระหว่างทางและโหนดต้นทางของเส้นทางเหล่านี้จะเก็บเส้นทางของตนไว้ใน Route Table ซึ่งมีโครงสร้างดังภาพที่ 2.1 เพื่อใช้ในการส่งข้อมูลต่อไป

2.1.4 กระบวนการนำส่งข้อมูล (Data Forwarding)

โหนดต้นทางจะใช้ IP Address ของโหนดปลายทางเป็นดรรชนีค้นหาเส้นทางใน Route Table โดยใช้เส้นทางที่มีค่าระดับความสำคัญสูงที่สุดในบรรดาเส้นทางทั้งหมดที่ไปยังโหนดปลายทางเดียวกันเป็นเส้นทางหลักในการส่งข้อมูล ส่วน Header ของข้อมูลจะเป็น Source Route ซึ่งได้ระบุ IP Address ของโหนดระหว่างทางจนถึงโหนดปลายทางตามลำดับของเส้นทางนั้น เมื่อโหนดระหว่างทางได้รับข้อมูลนี้ มันจะส่งข้อมูลนี้ไปยังโหนดถัดไปตามลำดับตามที่ระบุไว้ใน Source Route จนกระทั่งถึงโหนดปลายทาง หากในระหว่างทางไม่สามารถส่งข้อมูลไปยังโหนดถัดไปตามที่ Source Route ระบุไว้ได้มันจะนำ IP Address ของโหนดปลายทางมาค้นหาเส้นทางอื่น ๆ ใน Route Table ของโหนดระหว่างทาง หากพบเส้นทางอื่นที่ไปยังโหนดปลายทางเดียวกันกับเส้นทางเดิมมันก็จะส่งข้อมูลนี้ไปตามเส้นทางใหม่ต่อไป หากในระหว่างทางไม่พบเส้นทางอื่นมันก็จะทิ้งข้อมูลนี้ไป

2.1.5 กระบวนการบำรุงรักษาเส้นทาง (Route Maintenance)

เส้นทางใน Route Table ที่โหนดแต่ละโหนดเก็บไว้จะถูกลบทิ้งเมื่อได้รับรายงานจากโหนดที่อยู่ติดกับตนเองว่าเส้นทางนี้ได้เกิดการเสียหายแล้ว อันเนื่องมาจากไม่สามารถส่งข้อมูลไปยังโหนดนั้นได้แล้ว หรือค่าความแรงของสัญญาณต่ำกว่าค่าที่กำหนดไว้ หลังจากเส้นทางหลักได้ถูกลบทิ้งไปเส้นทางสำรองที่มีใน Route Table ก็จะกลายเป็นเส้นทางหลักที่ใช้สำหรับการส่งข้อมูลต่อไป

ผู้นำเสนอโปรโตคอล PMP ได้แสดงให้เห็นว่าโปรโตคอลดังกล่าวสามารถหาเส้นทางและนำส่งข้อมูลที่มีประสิทธิภาพสูงกว่าโปรโตคอล DSR เมื่อเปรียบเทียบด้วย Average Number of Packet Route Discovery, Average Number of Route Discovery, Packet Delivery Ratio และ Normalized Routing Load รวมทั้งโปรโตคอลนี้ยังมีประสิทธิภาพสูงกว่าโปรโตคอล Split Multipath Routing (SMR) เมื่อเปรียบเทียบด้วย Packet Delivery Ratio, Throughput และ Number of Packets Dropped อีกด้วย

2.2 งานวิจัยและหลักการที่เกี่ยวข้องกับโปรโตคอล Label Switching

เทคนิคการทำ Label Switching นั้นเป็นที่รู้จักกันดีในเครือข่ายอินเทอร์เน็ตแบบมีสายซึ่งเป็นเทคนิคที่ทำให้การทำงานในการส่งข้อมูลทำได้อย่างรวดเร็ว และมีความยืดหยุ่นมากขึ้นโดยไม่ต้องปรับปรุงระบบของ Hardware ให้รองรับเทคนิคดังกล่าว แต่สำหรับในเครือข่าย MANET นั้น

ยังเป็นเทคนิคที่ต้องพัฒนาวิจัยกันอย่างกว้างขวางกันต่อไป เนื่องจากมีข้อจำกัดและธรรมชาติที่แตกต่างจากเครือข่ายแบบมีสายเป็นอย่างมาก ในระยะเวลาไม่นานที่ผ่านมานี้ได้มีการวิจัยต่าง ๆ ที่พยายามนำเทคนิคดังกล่าวไปผนวกรวมไว้กับหลักการการหาเส้นทางของโปรโตคอล DSR และ AODV ซึ่งผู้นำเสนองานวิจัยเหล่านั้นชี้ให้เห็นถึงประสิทธิภาพที่เพิ่มขึ้นจากโปรโตคอล DSR และ AODV เดิม ดังนั้นเราจึงได้นำหลักการและแนวคิดต่าง ๆ เหล่านั้นมาเป็นแนวทางในการพัฒนาโปรโตคอลของเราเพื่อผนวกเข้ากับประสิทธิภาพการหาเส้นทางของโปรโตคอล PMP โดยผลงานวิจัยที่เป็นแนวทางในการทำ Label Switching ของเรานั้นมีดังนี้

2.2.1 Multi-Protocol Label Switching (MPLS) (Rosen et al., 2001: 1-61)

ในเครือข่ายอินเทอร์เน็ตโดยทั่วไปจะใช้ IP Address ของโหนดปลายทางเป็นดรรชนีสำหรับการค้นหา Address ของโหนดถัดไปเพื่อใช้ในการส่งต่อข้อมูล (Destination-Base Forwarding) วิธีการแบบดั้งเดิมนี้ขนาดของ IP Address ที่จะนำไปใช้ในการค้นหา Address ของโหนดถัดไปนั้นจะมีความยาวมากทำให้สร้างภาระหนักต่อการคำนวณ แม้ว่าปัจจุบันจะมีการปรับปรุงประสิทธิภาพของ Hardware ให้รองรับเทคนิคดังกล่าวให้สามารถทำงานได้ดีมากขึ้นแล้วก็ตาม แต่ก็ยังมีวิธีการที่มีประสิทธิภาพเพื่อปรับปรุงการส่งต่อข้อมูลให้เร็วมากขึ้นได้โดยไม่ต้องไปยุ่งเกี่ยวกับ Hardware เลย ซึ่งเป็นที่รู้จักกันดีว่า Multi-Protocol Label Switching (MPLS)

หลักการของ Label-Switching ใน MPLS นั้น จะใช้ Label ที่มีขนาดคงที่ (Short Fixed-Length Label) เป็นดรรชนีในการหา Address ของโหนดถัดไป (Next Hop) ใน Route Table เนื่องจาก Label ที่ใช้นั้นมีขนาดที่สั้นกว่า IP Address จึงทำให้ภาระในการคำนวณนั้นน้อยลงมาก ส่งผลให้การทำงานมีความรวดเร็ว นอกจากนั้นวิธีการแบบเดิมยังไม่สามารถให้บริการการส่งข้อมูลไปยังเป้าหมายเดียวกันโดยใช้เส้นทางหลาย ๆ เส้นทางที่แตกต่างกันให้เหมาะสมกับประเภทและอัตราความเร็วในการส่งของชุดข้อมูลต่าง ๆ ได้เพื่อลดการแน่นขนัดของข้อมูล จึงทำให้ไม่มีความยืดหยุ่นในการทำงาน ซึ่งตรงกันข้ามกับวิธีการ Label-Switching ที่สามารถให้บริการสิ่งเหล่านี้ได้ ดังนั้น Label-Switching จึงเป็นวิธีการที่ดีกว่า Destination-Base Forwarding แบบเดิมอย่างเห็นได้ชัด

2.2.2 An On-Demand Routing Protocol in Ad Hoc Network Using Label Switching (ORAL) (Lv, Zhou, Wang and Liu, 2006: 95-106)

โปรโตคอล ORAL นั้นจะหาเส้นทางตามหลักการของโปรโตคอล DSR แต่หลังจากหาเส้นทางได้แล้วจะสร้าง Label ขึ้นให้เป็นดรรชนีประจำเส้นทางแต่ละเส้นทางเพื่อใช้ในการค้นหาข้อมูลของเส้นทางนั้น ๆ ในภายหลัง และเก็บข้อมูลของเส้นทางเหล่านั้นไว้ใน Route Table ของแต่

ละโหนด โดยภายใน Route Table ซึ่งแสดงในภาพที่ 2.6 นั้นแต่ละ Entry จะหมายถึงเส้นทางหนึ่งที่ไปยังโหนดปลายทางปลายทางหนึ่ง ๆ ซึ่งจะประกอบไปด้วย ส่วนของ Label และ ส่วนของ Route ในส่วนของ Label นั้นจะประกอบไปด้วย Outbound Label (oLabel), Address ของ Hop ถัดไป (nextHop) และ Address ปลายทาง (Dest), รายการ Address ของโหนดในด้าน Upstream ที่อยู่ติดกับโหนดปัจจุบันซึ่งอาจมีได้มากกว่า 1 Address หากบรรดาโหนด Upstream เหล่านี้ต่างก็มีเส้นทางจากโหนดปัจจุบันไปยังโหนดปลายทางเป็นเส้นทางเดียวกัน หรือมี Post Route เหมือนกัน ส่วนของ Route จะประกอบด้วยรายการ Address ของโหนดต่าง ๆ ที่อยู่ถัดจากโหนดปัจจุบันเรียงเป็นลำดับจนกระทั่งถึงโหนดปลายทาง (Post Route) และที่สำคัญคือ จะมี Label เป็นตัวชี้แต่ละ Entry ใน Route Table นี้

Label	oLabel	nextHop	Dest	Upstream	Post Route

ภาพที่ 2.6 โครงสร้าง Route Table ของโปรโตคอล ORAL

หมายเหตุ: ฟิลด์ต่าง ๆ ของ Route table ของโปรโตคอล ORAL มีความหมายดังนี้

- **Label** หมายถึง Index ของ Entry แต่ละเส้นทาง ซึ่งจะเป็นดรรชนีสำหรับการค้นหาเส้นทางต่าง ๆ ใน Route Table

- **oLabel** (Outbound Label) หมายถึง Label ที่ใช้เป็นดรรชนีสำหรับหาเส้นทางใน Route Table ของโหนดถัดไป

- **nextHop** หมายถึง Address ของโหนดถัดไป

- **Dest** หมายถึง Addrss ของโหนดปลายทาง

- **Upstream** หมายถึง รายการ Address ของโหนดในด้าน Upstream ที่อยู่ติดกับโหนดปัจจุบัน ซึ่งมี oLabel ใน Route Table เป็นตัวเดียวกับ Label ใน Route Table ของโหนดปัจจุบัน โดยรายการนี้อาจมีได้หลาย Address หากบรรดาโหนด Upstream เหล่านี้ต่างก็มีเส้นทางจากโหนดปัจจุบันไปยังโหนดปลายทางเป็นเส้นทางเดียวกัน หรือมี Post Route เหมือนกัน

- **Post Route** หมายถึง รายการ Address ของโหนดต่าง ๆ ในด้าน Downstream ซึ่งเรียงลำดับจากโหนดที่อยู่ติดกับโหนดนั้นจนถึงโหนดปลายทาง

RREQ

Src	Intermediate	Dest

RREP

Label	Route

RUPP

Label	Post Route	Flag

Data

oLabel	nextHop	Data

ภาพที่ 2.7 โครงสร้างหลัก ๆ ของ RREQ, RREP, RUPP และ Data Packet ของโปรโตคอล ORAL

หมายเหตุ: ฟิลด์ต่าง ๆ ของ RREQ, RREP, RUPP และ Data Packet ของโปรโตคอล ORAL มีความหมายดังนี้

- **Src** หมายถึง Address ของโหนดต้นทาง

- **Intermediate** หมายถึง รายการ Address ของโหนดระหว่างทางต่าง ๆ ที่ RREQ นี้ได้ท่องผ่านมาโดยจะมีขนาดยาวเพิ่มมากขึ้นเรื่อย ๆ จนกระทั่งถึงโหนดปลายทาง

- **Dest** หมายถึง Address ของโหนดปลายทาง

- **Label** หมายถึง ค่า Label ที่ถูกใช้เป็นดรรชนีในการค้นหาเส้นทางของโหนดที่ส่ง RREP หรือ RUPP นี้ ซึ่งจะเป็นค่าเดียวกับ oLabel ของเส้นทางเดียวกันที่จะถูกเพิ่ม หรือปรับปรุงใน Route Table ของโหนดผู้รับ RREP หรือ RUPP นี้ หากโหนดที่ส่ง RREP หรือ RUPP นี้เป็นโหนดปลายทางสำหรับเส้นทางนี้ มันจะมีค่าเป็น SIL (Self-Indication Label) ซึ่งจะมีค่าเดียวกันสำหรับโหนดปลายทางทุก ๆ โหนดในเครือข่าย

- **Route** หมายถึง เส้นทางที่สมบูรณ์ทั้งหมดของเส้นทางนี้ซึ่งจะประกอบไปด้วยบรรดา Address ของโหนดต่าง ๆ ตั้งแต่ต้นทางจนถึงปลายทาง

- **Post Route** หมายถึง รายการ Address ของโหนดต่าง ๆ ที่อยู่ถัดจากโหนดที่ส่งRUPP นี้ซึ่งอยู่ในด้าน Downstream เรียงเป็นลำดับจนกระทั่งถึงโหนดปลายทาง หาก Flag มีค่าเป็น Valid ค่าของ Post Route จะเป็นเส้นทางล่าสุด และหาก Flag มีค่าเป็น Invalid ค่าของ Post Route จะเป็นเส้นทางเดิมที่ถูกยกเลิก

- **Flag** หมายถึง ค่าที่ระบุว่าเส้นทางของ RUPP นี้ยังใช้งานได้ (Valid) หรือถูกยกเลิก (Invalid) หากมีค่า เป็น Valid โหนดผู้รับ RUPP นี้จะต้องปรับปรุง Post Route ตามที่ระบุ และหากมีค่าเป็น Invalid โหนดผู้รับ RUPP นี้ จะต้องลบเส้นทางนี้ทิ้งไป

- **oLabel** (Outbound Label) หมายถึง Label ที่ใช้เป็นดรรชนีสำหรับหาเส้นทางใน Route Table ของโหนดถัดไป

- **nextHop** หมายถึง Address ของโหนดถัดไป

- **Data** หมายถึง ข้อมูลที่ส่ง

กระบวนการทำงานของโปรโตคอลนี้มีรายละเอียดดังต่อไปนี้

2.2.2.1 กระบวนการหาเส้นทาง (Route Discovery) จะประกอบไปด้วยการทำงานต่าง ๆ ดังนี้

1) การร้องขอเส้นทาง (Route Request)

Route Request Packet (RREQ) ของโปรโตคอลนี้จะประกอบด้วย IP Address ของโหนดต้นทาง (Src), IP Address ของโหนดปลายทาง (Dest) และ IP Address ของโหนดระหว่างทาง (Intermediate) ซึ่งจะถูกแทรกเข้าไปหลังจาก Packet นี้ได้ท่องผ่านโหนดต่าง ๆ ก่อนจะถึงโหนดปลายทางที่ต้องการ

เมื่อโหนดใดโหนดหนึ่งต้องการจะส่งข้อมูลไปยังโหนดอื่น ๆ ในเครือข่าย หากไม่มีเส้นทางที่ไปยังโหนดปลายทางได้ มันจะจัดเก็บข้อมูลนี้ไว้ใน Buffer ชั่วคราวก่อนจะเริ่มกระบวนการ Route Discovery โดยส่ง RREQ ให้กับโหนดต่าง ๆ ที่อยู่ติดกันกับมันซึ่งสามารถรับ Packet นี้ได้ หลังจากนั้นมันจะรอคอย Route Reply Packet (RREP) ด้วยระยะเวลาค่าหนึ่งตามที่กำหนดไว้ โดยในระหว่างการรอคอยนี้จะไม่มีการส่ง RREQ ใด ๆ ไปยังโหนดปลายทางเดียวกันอีก เมื่อมันรอคอยจนถึงระยะเวลาที่กำหนดไว้แต่ยังไม่ได้รับ RREP กลับมา มันก็จะเริ่มส่ง RREQ ใหม่อีกครั้ง โดยเวลาการรอคอยจะถูกเพิ่มเป็นสองเท่าจากของเดิม มันจะทำเช่นนี้จนถึงขีดจำกัดค่าหนึ่ง หากยังไม่ได้รับ RREP กลับมา มันจะลบข้อมูลที่จะส่งไปยังโหนดปลายทางนี้ออกจาก Buffer ทั้งหมด เมื่อโหนดใด ๆ ได้รับ RREQ มันจะส่ง RREP กลับมาหากมันเป็นโหนดปลายทางของ RREQ นี้หรือมันมีเส้นทางที่ไปยังโหนดปลายทางนี้ หากไม่ใช่ทั้งสองกรณีนี้มันจะแทรก Address ของมันเข้าไปเป็นส่วนหนึ่งของ RREQ นี้ก่อนจะส่ง RREQ นี้ให้กับโหนดอื่น ๆ ที่อยู่ติดกันกับมันต่อไป ซึ่ง RREQ เดียวกันนี้จะต้องส่งออกจากตัวมันเพียงครั้งเดียวเท่านั้น และหาก RREQ นี้มีจำนวน Address ของโหนดต่าง ๆ เป็นจำนวนมากกว่าที่กำหนดไว้มันก็จะปฏิเสธ RREQ นี้ไป

2) การตอบกลับต่อการร้องขอเส้นทาง (Route Reply)

Route Reply Packet (RREP) จะประกอบไปด้วย Label ตัวหนึ่งและเส้นทางที่สมบูรณ์เส้นทางหนึ่ง หากโหนดปลายทางเป็นผู้เริ่มต้นส่ง RREP นี้ Label จะเป็น Label ที่มีการกำหนดเอาไว้ล่วงหน้าเพื่อเป็นตัวชี้ระบุถึงตัวมันเองซึ่งจะเรียกว่า SIL (Self-Indication Label) ส่วน

ของเส้นทางที่สมบูรณ์นั้นจะประกอบไปด้วยบรรดา Address ของโหนดต่าง ๆ ตั้งแต่โหนดต้นทางจนถึงโหนดปลายทาง หาก RREP นั้นไม่ได้เริ่มต้นจากโหนดปลายทางแต่เริ่มต้นจากโหนดระหว่างทางค่า Label จะเป็น Label ที่ชี้เส้นทางไปยังปลายทางที่อยู่ภายใน Route Table ของโหนดนั้น และส่วนของเส้นทางที่สมบูรณ์นั้นจะเป็นบรรดา Address ของโหนดต่าง ๆ ที่บรรจุอยู่ใน RREQ ผนวกด้วย Post Route ซึ่งโหนดที่อยู่ระหว่างทางนั้นได้ปะติดปะต่อเพื่อขจัดเส้นทางที่ทำให้เกิดการซ้ำวนไว้เรียบร้อยแล้ว (ไม่มี Address ของโหนดต่าง ๆ ซ้ำกัน) นอกจากนั้นยังมีการป้องกันไม่ให้เส้นทางมีจำนวน Hop มากเกินกว่าที่กำหนดด้วย

เมื่อ RREP ได้เดินทางมาถึงโหนดใด ๆ ที่อยู่ระหว่างทาง มันจะพิจารณาว่าใน Route Table ของมันนั้นมีเส้นทางที่ไปยังปลายทางเช่นเดียวกับ RREP นั้นหรือไม่ ในกรณีที่ไม่มีเส้นทางที่ไปยังโหนดปลายทางเดียวกันนี้อยู่เลย มันจะเพิ่มเส้นทางนี้ลงใน Route Table ของมัน แต่ถ้าหากเส้นทางเดิมนั้นใช้งานไม่ได้ (Invalid) หรือเส้นทางใหม่ที่ส่งมากับ RREP นั้นดีกว่าเส้นทางเดิมของมันโดยอาจจะใช้ความยาวของเส้นทาง (Path length) หรือคุณภาพของการให้บริการในการส่งข้อมูล (QoS) มาเปรียบเทียบก็ได้ มันจะปรับปรุง Route Table ของมันให้ใช้เส้นทางใหม่ตามที่ได้รับจาก RREP นี้ การเปลี่ยนไปใช้เส้นทางใหม่นั้นจะต้องเสียเวลาในการส่ง Route Update Packet (RUPP) ไปให้กับโหนดต่าง ๆ ในด้าน Upstream ทุกโหนดของตนและบรรดาโหนดเหล่านั้นก็จะส่ง RUPP ไปเป็นทอด ๆ จนกระทั่งถึงโหนดต้นทางด้วย หลังจากนั้นมันจะเปลี่ยน Label ของ RREP นี้ ให้เป็น Label ที่ชี้ Entry ของเส้นทางนี้ใน Route Table ของมัน ก่อนจะส่ง Packet นี้กลับไปยังโหนดต่าง ๆ ที่อยู่ในลำดับก่อนหน้ามันซึ่งระบุไว้ในเส้นทางที่สมบูรณ์ของ RREP ที่ได้รับมา โหนดต่าง ๆ ในระหว่างทางจะทำเช่นนี้และส่ง RREP ไปเป็นทอด ๆ จนกระทั่งถึงโหนดต้นทาง เมื่อโหนดต้นทางได้รับ RREP มันจะเพิ่มเส้นทางนี้ลงใน Route Table ของมัน และจะลบ Route Request ที่ไปยังปลายทางเดียวกับเส้นทางนี้ได้ พร้อมกับนำข้อมูลจาก Buffer ที่ต้องการส่ง ส่งไปตามเส้นทางใหม่นี้ต่อไป

2.2.2.2 กระบวนการนำส่งข้อมูล (Data Forwarding)

เมื่อโหนดต้นทางต้องการส่งข้อมูลมันจะใช้ Address ของโหนดปลายทางเป็นดรรชนีในการค้นหาเส้นทางที่เก็บไว้ใน Route Table หากมีเส้นทางที่ไปยังโหนดปลายทางนั้นซึ่งยังสามารถใช้งานได้อยู่ มันจะส่งข้อมูลนั้นไปยัง Hop ถัดไป ตามที่เส้นทางนั้นได้ระบุไว้ โดยนำ Outbound Label (oLabel) แนบไปกับ Data Packet นี้ด้วย ซึ่งจะเห็นได้ว่า Header ของ Data Packet ของโปรโตคอลนี้จะประกอบไปด้วย Address ของ Hop ถัดไป (nextHop) และ Outbound Label (oLabel) เท่านั้น ในกรณีที่ไม่สามารถหาเส้นทางไปยังโหนดปลายทางได้มันจะเก็บข้อมูลไว้ใน Buffer ชั่วคราวก่อนเริ่มกระบวนการ Route Discovery เพื่อหาเส้นทางต่อไป

เมื่อข้อมูลได้มาถึงโหนดถัดไปตามที่ระบุไว้ใน Data Packet มันจะใช้ Label จาก Data Packet นั้นเป็นดรรชนีสำหรับการหา Address ของโหนดถัดไป และ Outbound Label เพื่อนำมาเป็น Header ของ Data Packet เพื่อส่งให้กับโหนดถัดไปต่อไป เป็นทอด ๆ จนกระทั่งถึงโหนดปลายทาง

2.2.2.3 กระบวนการบำรุงรักษาเส้นทาง (Route Maintenance)

1) การทำ Local Repair

เมื่อการเชื่อมต่อของโหนดปัจจุบันกับโหนดถัดไปเสียหาย หรือไม่สามารถส่งข้อมูลไปยังโหนดถัดไปได้ โปรโตคอลนี้จะมีการแก้ไขเส้นทางที่เกิดการเสียหายนั้นด้วยการหมายเหตุไว้ว่าเส้นทางนี้ใช้งานไม่ได้ (Invalid) และจะเริ่มกระบวนการ Route Discovery เพื่อหาเส้นทางอื่นมาใช้แทน โดยในระหว่างการแก้ไขเส้นทางนี้มันจะเก็บข้อมูลที่จะส่งไปยังโหนดถัดไปของเส้นทางนี้ไว้ใน Buffer ก่อน หากการแก้ไขเส้นทางไม่ประสบผลสำเร็จ ข้อมูลใน Buffer เหล่านี้รวมทั้ง Entry ของเส้นทางนี้ใน Route Table ของมันก็จะถูกลบทิ้งไป หากการแก้ไขเส้นทางประสบผลสำเร็จ เส้นทางนี้จะถูกตั้งค่าสถานะให้เป็น Valid พร้อมที่จะใช้ในการส่งข้อมูลต่อไปได้

2) การทำ Route Update

เมื่อโหนดใด ๆ ได้ทำ Local Repair เพื่อแก้ไขเส้นทางที่เสียหายเสร็จเรียบร้อยแล้ว ไม่ว่าจะประสบผลสำเร็จหรือไม่ก็ตาม มันจะส่ง Route Update Packet (RUPP) ให้กับโหนดที่อยู่ติดกับตัวเองในด้าน Upstream และโหนดนี้ก็จะส่ง RUPP นี้ต่อไปเป็นทอด ๆ จนกระทั่งถึงโหนดต้นทาง เพื่อให้ทุกโหนดที่อยู่ในด้าน Upstream ทั้งหมดของเส้นทางนี้ทำการปรับปรุงเส้นทางให้ทันสมัยและสอดคล้องกัน โดย RUPP จะประกอบไปด้วย Post Route ของเส้นทางล่าสุด และ Label ที่เป็นตัวชี้ไปยัง Entry ของเส้นทางนี้ นอกจากนั้นยังมี Flag ตัวหนึ่งที่ใช้ในการระบุสถานะของเส้นทางนี้ โดยหากการทำ Local Repair แล้วไม่ประสบผลสำเร็จ Flag ตัวนี้ก็จะมีค่าเป็น Invalid และ Post Route ของ RUPP ก็จะเป็น Post Route ของเส้นทางเดิมที่เสียหาย แต่หากการทำ Local Repair ประสบผลสำเร็จ Flag ตัวนี้ก็จะมีค่าเป็น Valid และ Post Route ของ RUPP ก็จะเป็น Post Route ของเส้นทางใหม่ เมื่อโหนดต่าง ๆ ได้รับ RUPP มันจะเปรียบเทียบว่าเส้นทางใดใน Route Table ของตน ตรงกับที่ระบุไว้ใน RUPP หรือไม่ โดยจะพิจารณาดังนี้ หาก Entry ใดใน Route Table ของตน มี oLabel เป็นตัวเดียวกับ Label ของ RUPP , มี Hop ถัดไปเหมือนกับโหนดผู้ส่ง RUPP นี้ และมีโหนดปลายทางเหมือนกับที่ระบุไว้ใน RUPP แสดงว่า Entry นี้ต้องปรับปรุงให้มีสถานะตามที่ RUPP นั้นระบุ โดยหากสถานะ เป็น Valid ส่วนของ Post Route ของ Entry นี้จะต้องเปลี่ยนไปตามที่ RUPP ระบุ หากสถานะเป็น Invalid มันจะลบ Entry นี้ทิ้งไป

สิ่งที่สำคัญสำหรับการปรับปรุง Post Route ใน Route Table นั้น จะต้องไม่ให้มีรายการของโหนดตนเองรวมเข้าไปด้วย เพราะไม่เช่นนั้นแล้วจะทำให้มีเส้นทางที่ซ้ำวนเกิดขึ้นได้

ผู้นำเสนอโปรโตคอลนี้ได้ระบุไว้ว่า ขนาดของ Route Table ของโปรโตคอลนี้จะเป็นสัดส่วนโดยตรงกับจำนวนโหนดที่กำลังติดต่อสื่อสารกันในขณะนั้น เนื่องจากการเก็บเส้นทางไว้ในรูปแบบของ Post Route แทนที่จะเก็บเส้นทางที่สมบูรณ์ทั้งหมดตั้งแต่ต้นทางจนถึงปลายทาง และการใช้วิธีการ Label Switching นั้นทำให้ต้องการทรัพยากรน้อยลง ความเร็วในการรับส่งข้อมูลเพิ่มขึ้น และท้ายที่สุดจะลดช่วงเวลาการหน่วงของการส่งข้อมูลลง และทำให้มี Throughput ดีขึ้น โดยได้แสดงการเปรียบเทียบประสิทธิภาพการทำงานกับโปรโตคอล DSR และ AODV พบว่ามีปริมาณของ Control Packet ที่ถูกส่งออกมาในเครือข่าย (Control Packet Overhead) มากกว่า DSR แต่น้อยกว่า AODV ส่วน Packet Delivery Ratio นั้นสูงกว่าโปรโตคอลทั้งสอง และจำนวน Hop ที่ Data Packet ถูกส่งผ่านไปจนกระทั่งถึงโหนดปลายทางต่อจำนวน Data Packet ที่โหนดปลายทางรับได้สำเร็จนั้นมีค่าสูงกว่าโปรโตคอลทั้งสอง

2.2.3 โปรโตคอล Label Switching Multipath Routing (LSMR) (Wei, Wu and Yu, 2007: 546-551)

สำหรับโปรโตคอลนี้จะใช้วิธีการทำงานที่แตกต่างจากโปรโตคอล ORAL โดยจะสร้างและกระจาย Label สำหรับเส้นทางต่าง ๆ ไปพร้อม ๆ กับกระบวนการ Route Discovery เลย แต่สำหรับโปรโตคอล ORAL นั้นต้องได้เส้นทางที่สมบูรณ์มาก่อนที่จะสร้างและกระจาย Label ให้กับเส้นทางต่าง ๆ ซึ่งโปรโตคอลนี้จะใช้หลักการหาเส้นทางเหมือนกับโปรโตคอล AODV โดยจะเลือกเส้นทางที่ไปยังโหนดปลายทางเดียวกันที่มีจำนวน Hop น้อยที่สุดเพียงเส้นทางเดียวมาใช้ในการส่งข้อมูล ซึ่งจะเห็นได้ว่าเส้นทางที่ได้มาจากโปรโตคอลนี้ไม่ได้คำนึงถึงค่าความคงทนต่อความเสียหายและประสิทธิภาพในการส่งข้อมูลของเส้นทางนั้นเหมือนกับโปรโตคอล PMP แต่อย่างใด โปรโตคอล LSMR นั้นได้ต่อเติม Packet ต่าง ๆ ของโปรโตคอล AODV เพื่อให้สามารถผนวกวิธีการ Label Switching เข้าไปได้ด้วยการใช้ Label ห้อยติดไปกับ Path Request Packet (PREQ) และ Path Reply Packet (PREP) ดังภาพที่ 2.9 เพราะจะช่วยให้กระบวนการ Route Discovery นั้นใช้เวลาน้อยลงและลดภาระในการทำงานลงด้วย

โปรโตคอลนี้จะใช้ Source Neighbor's ID (SN) ซึ่งเป็นค่าระบุถึงโหนดถัดไปที่อยู่ติดกับโหนดต้นทางของเส้นทางนั้น ๆ เป็นตัวแยกความแตกต่างของเส้นทางที่ไปยังโหนดปลายทางเดียวกัน รวมทั้งสร้างและกระจาย Label ผูกติดไปกับ PREQ และ PREP ในกระบวนการหาเส้นทางเลย นอกจากนั้นยังมีการใช้ Table สอง Table คือ Incoming Data Matching Table (IDM) กับ

Incoming Label Matching Table (ILM) เก็บรักษาเส้นทาง โดย IDM Table นั้นจะติดตั้งไว้ที่โหนดต้นทางโดยจะใช้ Forwarding Equivalence Class (FEC) เป็นตัวกำหนดเส้นทางตามความเหมาะสมของประเภทและอัตราการส่งของข้อมูลที่ต้องการเพื่อลดการแน่นขนัดของข้อมูล ส่วน ILM Table นั้นจะติดตั้งไว้ที่โหนดระหว่างทางและโหนดปลายทาง เพื่อให้สามารถส่งข้อมูลไปยังโหนดถัดไปจนถึงโหนดปลายทางตามเส้นทางนั้นได้ โครงสร้างของทั้งสอง Table แสดงดังภาพที่ 2.8

IDM Table

FEC	Label Out	Next Hop MAC	Life Time

ILM Table

Label In	Label Out	Next Hop MAC	Life Time

ภาพที่ 2.8 โครงสร้าง IDM Table และ ILM Table ของโปรโตคอล LSMR

หมายเหตุ: ฟิลด์ต่าง ๆ ของ IDM Table และ ILM Table ของโปรโตคอล LSMR

มีความหมายดังนี้

- **FEC** (Forwarding Equivalence Class) หมายถึง ตัวระบุเส้นทางที่เหมาะสมตามประเภทหรืออัตราการส่งของข้อมูลชุดนั้น ๆ

- **Label Out** หมายถึง Label ที่ใช้เป็นดรรชนีสำหรับหาเส้นทางใน ILM Table ของโหนดถัดไป

- **Next Hop MAC** หมายถึง MAC Address ของโหนดถัดไป

- **Life Time** หมายถึง เวลาในหน่วย Milliseconds ที่เอาไว้พิจารณาว่าเส้นทางนั้นหมดอายุการใช้งานหรือไม่

- **Label In** หมายถึง Label ซึ่งเป็นดรรชนีสำหรับการค้นหา Label Out และ MAC Address ของโหนดถัดไป

PREQ

Label	SN	Src	Src_sq#	Broadcast_id	Dest	Dest_sq#	Hop Count

PREP

Label	Src	Dest	Dest_sq#	Hop Count	Life Time

Data

Label Out	Next Hop MAC	Data

ภาพที่ 2.9 โครงสร้าง PREQ, PREP และ Data packet ของโปรโตคอล LSMR

หมายเหตุ: ฟิลด์ต่าง ๆ ของ PREQ, PREP และ Data Packet ของโปรโตคอล LSMR มีความหมายดังนี้

- **Label** หมายถึง Label ที่ถูกใช้เป็นดรรชนีในการค้นหาเส้นทางของโหนดที่ส่งPREQ หรือ PREP นี้ ซึ่งจะเป็นค่าเดียวกับ **Label Out** ของเส้นทางเดียวกันที่จะถูกเพิ่มลงใน ILM Table ของโหนดผู้รับ PREQ หรือ PREP นี้

- **SN** (Source Neighbor's ID) หมายถึง ค่าที่ระบุถึงโหนดถัดไปที่อยู่ติดกับโหนดต้นทางของเส้นทางนั้น ๆ ซึ่งจะเป็นตัวแยกความแตกต่างของเส้นทางจากโหนดต้นทางที่ไปยังโหนดปลายทางเดียวกัน

- **Src** หมายถึง Address ของโหนดต้นทาง

- **Src_sq#** หมายถึง เลขลำดับที่โหนดต้นทางระบุ Reverse Path ล่าสุดโดยจะมีการเพิ่มค่าขึ้นทีละหนึ่งโดยโหนดต้นทาง

- **Broadcast_id** หมายถึง ตัวระบุความแตกต่างของ PREQ เพื่อป้องกันไม่ให้โหนดต่าง ๆ รับ PREQ นี้ซ้ำ

- **Dest** หมายถึง Address ของโหนดปลายทาง

- **Dest_sq#** สำหรับ PREQ หมายถึง เลขลำดับล่าสุดของโหนดปลายทางเพื่อให้โหนดระหว่างทางพิจารณาว่าต้องส่ง PREP กลับไปให้กับโหนดที่ส่ง PREQ นี้มาให้ตน หรือจะส่ง PREQ นี้ให้กับโหนดอื่น ๆ ต่อไป และสำหรับ PREP จะหมายถึง เลขลำดับที่โหนดปลายทางระบุ Positive Path ล่าสุด ซึ่งค่านี้จะเพิ่มค่าขึ้นทีละหนึ่งโดยโหนดปลายทาง

- **Hop Count** หมายถึง ตัวเลขที่ระบุจำนวน Hop ที่ PREQ หรือ PREP ได้ท่องผ่านตามเส้นทางนั้น ๆ

- **Life Time** หมายถึง เวลาหน่วยเป็น Millisecond ที่เอาไว้ให้โหนดที่รับ PREPพิจารณาว่าเส้นทางนั้นหมดอายุการใช้งานหรือไม่

- **Label Out** หมายถึง Label ที่ใช้เป็นดรรชนีสำหรับหาเส้นทางใน ILM Table ของโหนดถัดไป

- **Next Hop MAC** หมายถึง MAC Address ของโหนดถัดไป

- **Data** หมายถึง ข้อมูลที่ส่ง

กระบวนการทำงานของโปรโตคอลนี้ มีดังนี้

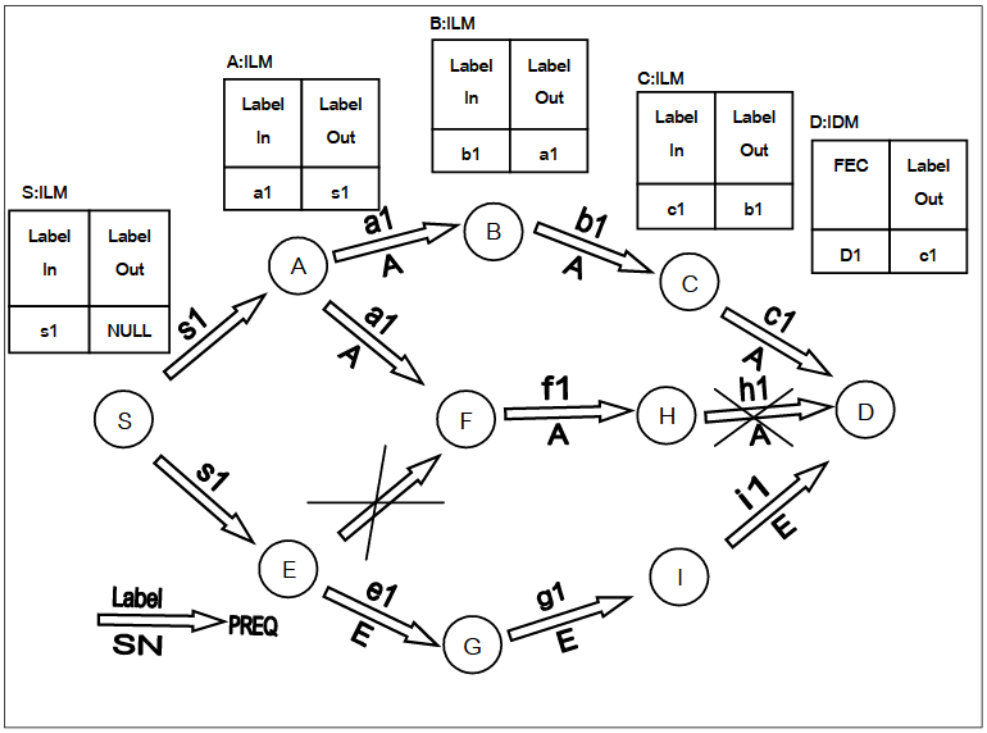
2.2.3.1 กระบวนการหาเส้นทาง (Route Discovery)

เมื่อโหนดต้นทางต้องการส่งข้อมูล และไม่มีเส้นทางสำหรับ FEC นั้นเพื่อส่งข้อมูล มันจะเริ่มกระบวนการหาเส้นทางโดยที่กระบวนการหาเส้นทางขาไปจากโหนดต้นทางไปยังโหนดปลายทางนั้นจะเป็นการสร้างเส้นทางขากลับจากโหนดปลายทางมายังโหนดต้นทาง (Reverse Path) ซึ่งเส้นทางที่ได้นี้จะนำมาใช้เมื่อโหนดปลายทางต้องการส่งข้อมูลกลับมายังโหนดต้นทาง ส่วนกระบวนการตอบกลับจากโหนดปลายทางกลับไปยังโหนดต้นทาง จะเป็นการสร้างเส้นทางขาไปจากโหนดต้นทางไปยังโหนดปลายทาง (Positive Path) ซึ่งเส้นทางที่ได้นี้จะนำมาใช้เมื่อโหนดต้นทางต้องการส่งข้อมูลไปยังโหนดปลายทาง ทั้งนี้เนื่องจากในเครือข่ายไร้สายนั้นสภาพสัญญาณและสภาวะแวดล้อมอื่น ๆ ในการส่งข้อมูลจะไม่เหมือนกันทั้งสองทิศทางดังนั้นเส้นทางขาไป (Positive Path) และขากลับ (Reverse Path) จึงไม่ใช่เส้นทางเดียวกันเสมอไป เพื่อให้เข้าใจการทำงานสำหรับกระบวนการนี้ของโปรโตคอล LSMR อย่างชัดเจนเราจะอธิบายด้วยการยกตัวอย่างดังภาพที่ 2.10 และภาพที่ 2.11 ดังต่อไปนี้

1. โหนดต้นทาง S จะส่ง PREQ ให้กับโหนดอื่น ๆ ที่อยู่ติดกับตนโดยแทรก “s1” เข้าไปเป็น Label ของ PREQ นี้ส่วน SN ของ PREQ จะเริ่มต้นด้วยค่า “NULL” ส่วนอื่น ๆ ของ PREQ ที่เหลือจะเหมือนกับ PREQ ของโปรโตคอล AODV สำหรับ ILM table ของโหนดต้นทางนั้น ฟิลด์ “Label In” จะมีค่าเป็น “s1” และ ฟิลด์ “Label Out” ถูกตั้งให้เป็น “NULL” เพราะโหนดต้นทางเป็นโหนดปลายทางของ Reverse Path นั่นเอง

2. โหนด A เมื่อได้รับ PREQ จากโหนด S มันจะเพิ่ม Label “s1” ซึ่งห้อยติดมากับ PREQ นั้นเข้าไปใน ฟิลด์ “Label Out” และสร้าง Label ขึ้นมาตัวหนึ่งในที่นี้ คือ “a1” เก็บไว้ใน ฟิลด์ “Label In” ของ ILM table พร้อมกับเปลี่ยน Label ของ PREQ เป็น “a1” และจะเพิ่ม ID ของมัน คือ “A” เข้าไปยัง ฟิลด์ SN ของ PREQ นี้ก่อนที่จะส่ง PREQ ให้กับโหนดอื่น ๆ ต่อไปซึ่งฟิลด์นี้จะไม่มีการเปลี่ยนแปลงใด ๆ ตลอดเส้นทาง โหนดระหว่างทางจะรับ PREQ ที่มาจากโหนดต้นทางเดียวกันเพียงครั้งเดียวเท่านั้น เช่น โหนดระหว่างทาง F ได้รับ PREQ จากโหนด E ซึ่งเป็น PREQ

ที่มาจากโหนดต้นทาง S เช่นเดียวกับที่ได้รับจากโหนด A มาแล้ว โหนด F ก็จะปฏิเสธ PREQ ที่มาจากโหนด E ไป เป็นต้น



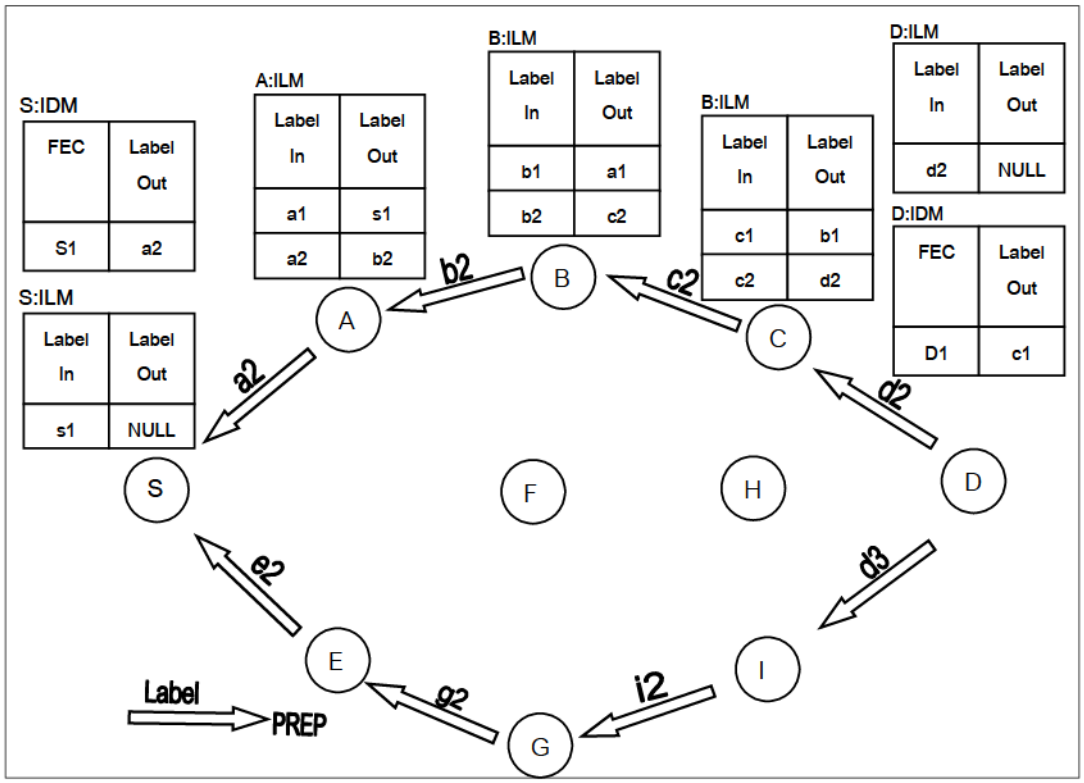
ภาพที่ 2.10 การส่ง Path Request Packet (PREQ) และการสร้าง Table ของโปรโตคอล LSMR

หมายเหตุ: เป็นตัวอย่างของการสร้าง Reverse Path สองเส้นทาง คือ D->C->B->A->S, D->I->G->E->S และ Table ที่เกี่ยวข้องของเส้นทาง D->C->B->A->S ค่าที่อยู่ข้างบนลูกศรคือ Label และค่าข้างใต้ลูกศรจะเป็น Source Neighbor's ID (SN) ของ PREQ

3. เมื่อโหนดปลายทาง ได้รับ PREQ มันจะตรวจสอบ Node ID ซึ่งอยู่ในฟิลด์ SN ของ PREQ ก่อนเป็นอันดับแรก หาก Node ID มีอยู่ในรายการ SN ที่ตนเคยรับมาก่อนแล้ว มันจะปฏิเสธ PREQ นี้ไป หาก Node ID ตัวนี้ไม่ได้อยู่ในรายการ SN ของตนมาก่อน มันจะแทรก Node ID ตัวนี้เข้าไปยังรายการ SN ที่มันเคยรับมาก่อน ยกตัวอย่างเช่นโหนดปลายทาง D ได้รับ PREQ ตัวหนึ่งในครั้งแรก จากโหนด C ซึ่ง SN เป็น A โหนด D จะเพิ่ม A เข้าไปยังรายการ SN ที่ตนเคยรับมาก่อน เนื่องจากตอนนี้นั้นรายการดังกล่าวของตนนั้นยังว่างอยู่ หลังจากนั้น Label "c1" จะถูกหยิบออกมาจาก PREQ และนำมันมาเก็บไว้ใน ฟิลด์ "Label Out" ของ IDM table ของตน แต่ถ้าหากโหนด D ได้รับ PREQ ตัวหนึ่งจาก H ด้วย SN "A" PREQ ตัวนี้จะถูกปฏิเสธทิ้งไปเนื่องจาก A อยู่ในรายการ

SN ที่โหนด D เคยรับมาก่อนหน้านี้แล้วโหนดปลายทางจะตอบรับ PREQ แต่ละตัวที่ได้ดำเนินการแล้วเสร็จ

และจะเริ่มสร้างเส้นทาง Positive Path ดังภาพที่ 2.11 โดยเราจะอธิบายให้เห็นการทำงานดังต่อไปนี้



ภาพที่ 2.11 การส่ง Path Reply Packet (PREP) และ การสร้าง Table ของโปรโตคอล LSMR

หมายเหตุ: เป็นตัวอย่างของการสร้าง Positive Path สองเส้นทางด้วยกันคือ S->A->B->C->D และ S->E->G->I->D และ Table ที่เกี่ยวข้องของเส้นทาง S->A->B->C->D ค่าที่อยู่ข้างบนลูกศรจะเป็น Label ที่มากับ PREP

1. โหนดปลายทางจะสร้าง Label ตัวหนึ่งและแทรกมันเข้าไปยัง ฟิลด์ “Label” ของ PREP และ ฟิลด์ “Label In” ของ ILM Table จะถูกตั้งค่าให้เป็น “d2” ฟิลด์ “Label Out” ของ ILM table จะถูกตั้งค่าให้เป็น “NULL”

2. โหนดระหว่างทาง จะแทรก Label ที่มากับ PREP เข้าไปใน ฟิลด์ “Label Out” ของ ILM Table รวมทั้งจะปรับปรุงฟิลด์ “Life Time” ของ Reverse path ที่สอดคล้องกับ Positive Path นี้ ใน ILM Table ของตนให้เป็นเวลาปัจจุบัน โหนดระหว่างทางจะสร้าง Label ขึ้นมาตัวหนึ่ง และ

แทรกมันเข้าไปในฟิลด์ “Label In” ของ ILM Table รวมทั้งฟิลด์ “Label” ของ PREP จะถูกเปลี่ยนแปลงให้เป็น Label ที่โหนดนี้สร้างขึ้นก่อนจะส่งต่อ PREP นี้ให้กับ Hop ถัดไป เมื่อโหนดต้นทางได้รับ PREP กลับมามันจะแทรก Label ที่มากับ PREP นี้เข้าไปในฟิลด์ “Label Out” ของ IDM Table และปรับปรุงฟิลด์ “Life Time” ของ Reverse Path ที่สอดคล้องกับ Positive Path นี้ใน ILM Table ของตนให้เป็นเวลาปัจจุบัน และแล้วเส้นทางทั้งสองทิศทางก็จะถูกสร้างขึ้นเสร็จสมบูรณ์

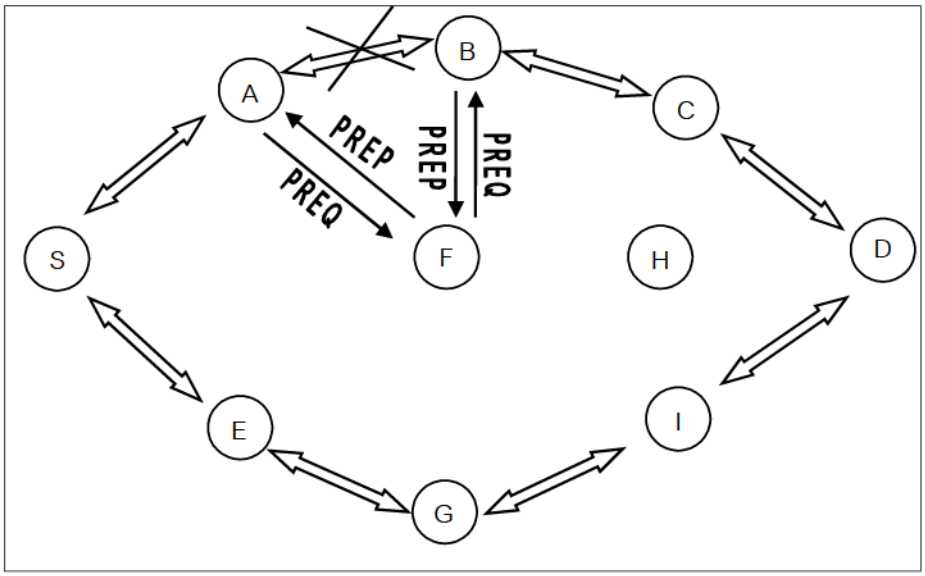
2.2.3.2 กระบวนการนำส่งข้อมูล (Data Forwarding)

Data Packet จะถูกจัดแยกกลุ่มเป็น FEC ซึ่งจะขึ้นอยู่กับ IP Header บนโหนดต้นทางของชุดข้อมูลนั้น ๆ FEC จะถูกใช้เป็นดรรชนีสำหรับหา Output Label และ MAC Address ของ Hop ถัดไป ใน IDM Table โดยจะนำค่าต่าง ๆ เหล่านี้ มาผนวกเป็น Header ของข้อมูลก่อนที่จะส่งออกไปให้กับโหนดถัดไป เมื่อโหนดระหว่างทาง ได้รับ Data Packet นี้มันก็จะนำ Output Label จาก Header ของ Data Packet นี้มาใช้เป็นดรรชนีในการหา Output Label และ MAC Address ของ Hop ถัดไปใน ILM Table เพื่อนำ Output Label ที่ได้นั้นมาปรับเปลี่ยนค่า Label ใน Packet นั้นก่อนที่จะถูกส่งออกไปยัง Hop ถัดไป เมื่อ Data Packet ใด ๆ ได้มาถึงปลายทางแล้ว ค่า Output Label นั้นจะมีค่าเป็น “NULL” และโหนดปลายทางนี้จะตัด Header ของ Data Packet นี้ออกไปกลายเป็น IP Packet ส่งต่อให้กับ Layer ที่อยู่สูงขึ้นไป สรุปได้ว่า การทำงานกับ IP Header ของ Packet นี้จะกระทำบนโหนดต้นทางเท่านั้น และการทำ Packet Switching ที่โหนดระหว่างทางนั้นจะใช้เวลาเพียงสั้น ๆ เนื่องจากใช้ Label ที่มีขนาดความยาวจำกัดเป็นดรรชนีของ ILM Table เพื่อใช้ในการหาค่าต่าง ๆ ที่ต้องการมาจัดทำเป็น Header ของ Packet เพื่อส่งต่อให้กับโหนดถัดไป ซึ่งการทำงานดังกล่าวนี้ใช้ Hardware ทำงานโดยตรงโดย ILM Table จะอยู่ที่ NIC และการค้นหา Mac Address ของโหนดถัดไปที่โหนดระหว่างทางต่าง ๆ เพื่อนำมาใช้ในการส่งต่อข้อมูลจะใช้ Label เป็นดรรชนีค้นหาใน ILM Table นี้ ซึ่งจะเห็นได้ว่าวิธีการทำ Label Switching ของโปรโตคอลนี้นั้นเป็นเทคโนโลยี Layer 2.5 โดย Packet ต่าง ๆ ไม่ต้องถ่ายโอนไปมาระหว่างอุปกรณ์เชื่อมต่อเครือข่าย (NIC) กับ CPU Buffer ของ Host นั้น ๆ กระบวนการส่งต่อข้อมูลทั้งหมดที่โหนดระหว่างทางจะอยู่บน NIC ของโหนดนั้นเท่านั้น จึงทำให้การส่งข้อมูลระหว่างโหนดต่าง ๆ มีความรวดเร็วและลดการหน่วงเวลาการส่งข้อมูลจากต้นทางถึงปลายทางลงได้มาก

2.2.3.3 กระบวนการบำรุงรักษาเส้นทาง (Route Maintenance)

การทำงานของโปรโตคอล LSMR นี้จะใช้ ฟิลด์ “Life Time” เป็นตัวกำหนดว่าเส้นทางต่าง ๆ นั้นยังคงใช้งานได้อยู่ ดังนั้นมันจึงต้องมีการปรับปรุงค่าดังกล่าวอยู่เสมอในขณะ

ทำงานไม่ว่าจะเป็นตอนส่งข้อมูลหรือตอนหาเส้นทางก็ตาม เพราะไม่เช่นนั้นแล้วเส้นทางที่มีอยู่ใน IDM Table และ ILM Table ก็จะถูกลบทิ้งไป



ภาพที่ 2.12 กระบวนการทำ Local Repair ของโปรโตคอล LSMR

เมื่อโหนดใด ๆ ไม่สามารถส่งข้อมูลไปยังโหนดถัดไปได้ หรือเส้นทางหมดเวลาการใช้งานทำให้เส้นทางนั้นเสียหายใช้งานไม่ได้ LSMR ก็จะแก้ไขเส้นทางเหล่านั้นด้วยวิธีการ Local Repair ดังตัวอย่างการทำงานตามภาพที่ 2.12 ซึ่งแสดงการทำ Local Repair เมื่อการเชื่อมโยงระหว่างโหนด A กับโหนด B ขาด โดยโหนด A จะเก็บ Packet ต่าง ๆ ที่ได้รับมาจากโหนด S ไว้ใน Buffer ของตนเป็นการชั่วคราว หลังจากนั้นจะส่ง PREQ ไปให้กับโหนดต่าง ๆ ที่อยู่ติดกับตน โดยระบุโหนดปลายทางให้เป็นโหนด B ต่อจากนั้นการดำเนินการต่าง ๆ ก็จะเป็นไปตามกระบวนการหาเส้นทาง ซึ่งในที่นี้โหนดที่อยู่ติดกับโหนด A คือโหนด F เมื่อโหนด F ได้รับ PREQ มันก็จะส่ง PREQ นี้ต่อไปให้กับโหนดที่อยู่ติดกับตน หากโหนดที่อยู่ติดกับตนนั้น เป็นโหนด B โหนด B ก็จะส่ง PREP กลับไปให้โหนด A หากการทำ Local Repair ล้มเหลว ทั้ง A และ B จะส่ง Route Error Packet (RERR) ไปแจ้งแก่โหนดในด้าน Upstream ต่าง ๆ ของเส้นทางที่เสียหายนั้นทั้งหมด จะสังเกตได้ว่า การทำ Local Repair ของโปรโตคอลนี้จะทำขึ้นที่โหนดระหว่างทางใด ๆ ซึ่งพบว่าไม่สามารถส่งข้อมูลให้กับโหนดถัดไปของตนได้ โดยแก้ไขเส้นทางเฉพาะส่วนที่เสียหาย คือจากโหนดระหว่างทางนั้นกับโหนดถัดไปที่เสียหายเท่านั้น ซึ่งผิดกับโปรโตคอล AODV ที่ต้องกระทำจากโหนดระหว่างทางนั้น ไปจนถึงโหนดปลายทางของเส้นทางที่เสียหาย ยกตัวอย่าง เช่น

จาก โหนด A ไปจนถึงโหนด D ดังนั้น เมื่อเปรียบเทียบกับ LSMR แล้ว AODV จะใช้เวลามากกว่า รวมทั้งจะส่ง PREQ ไปเพื่อแก้ไขเส้นทางที่เสียหายในปริมาณที่มากกว่าด้วย

ผู้นำเสนอโปรโตคอลนี้แสดงให้เห็นว่า ประสิทธิภาพการทำงานของโปรโตคอล LSMR นั้นดีกว่าโปรโตคอล AODV ไม่ว่าจะเป็น End-to-End Delay, Route Discovery Frequency, Packet Delivery Ratio หรือ Routing Overhead ก็ตาม

บทที่ 3

การทำงานของโปรโตคอล PMLS

จากผลงานวิจัยที่ได้กล่าวไว้ในบทที่ผ่านมาเราได้เลือกหลักการหาเส้นทางของ PMP มาใช้ในกระบวนการหาเส้นทางของโปรโตคอลที่เรานำเสนอ เนื่องจากเส้นทางที่ได้มานั้นมีความคงทนและประสิทธิภาพการส่งข้อมูลมากกว่าโปรโตคอล DSR ส่วนกระบวนการนำส่งข้อมูลนั้นเราได้นำหลักการของ Label Switching มาใช้แทนเนื่องจากกระบวนการนำส่งข้อมูลของ PMP นั้นยึดวิธีการเหมือนกับโปรโตคอล DSR ทำให้ Header ของ Data Packet มีขนาดยาวเนื่องจากเอา Address ของโหนดระหว่างทางจนถึงโหนดปลายทางของเส้นทางที่ Packet นั้นจะถูกส่งออกไปจากโหนดนั้น ๆ ผูกติดไปด้วยตลอดการทำงานของกระบวนการนำส่งข้อมูล ทำให้ประสิทธิภาพในการนำส่งข้อมูลลดต่ำลง ในโปรโตคอลนี้ยังได้นำเสนอการบำรุงรักษาเส้นทางด้วยวิธี Local Repair และ Route Discovery ที่โหนดระหว่างทางเพื่อเพิ่มประสิทธิภาพการทำงานอีกด้วย สำหรับรูปแบบขั้นตอนวิธีในการผนวก Label Switching เข้ากับ PMP นั้นเราใช้แนวคิดของ ORAL เนื่องจาก ORAL นั้นมีการสร้างและกระจาย Label ต่าง ๆ หลังจากได้เส้นทางที่สมบูรณ์เรียบร้อยแล้ว ทำให้มี Overhead ในกระบวนการหาเส้นทางน้อยกว่าวิธี LSMR อีกทั้งกระบวนการหาเส้นทางของ PMP นั้น ต้องให้โหนดปลายทางคัดสรรเส้นทางที่ดีที่สุดจำนวนหนึ่งโดยอาศัยข้อมูลของเส้นทางต่าง ๆ ตั้งแต่โหนดต้นทางจนถึงปลายทาง หากทำการสร้างและกระจาย Label ในขณะหาเส้นทางแล้ว จะต้องมี Label เป็นจำนวนมากที่กระจายอยู่ในโหนดต่าง ๆ แต่ไม่ได้นำมาใช้งานเพราะจะมีเส้นทางเพียงบางส่วนเท่านั้นที่จะถูกคัดสรรให้นำมาใช้งานได้โดยโหนดปลายทาง รายละเอียดของโปรโตคอลที่เรานำเสนอมีดังนี้

3.1 หลักการของ PMLS

ในโปรโตคอล PMLS นั้นจะมีตัววัดที่สำคัญสองตัวซึ่งนำมาใช้ในการเลือกเส้นทางที่ดีที่สุดสำหรับการส่งข้อมูลโดยจะเป็นเช่นเดียวกับ PMP คือ ความคงทน (Robustness) และความมีประสิทธิภาพ (Effectiveness) สำหรับการวัดค่าความคงทนของเส้นทางใด ๆ นั้น จะพิจารณาจากการเปลี่ยนแปลงของค่าความแรงของสัญญาณซึ่งเป็นผลมาจากการเคลื่อนที่ของโหนดต่าง ๆ ค่า

ดังกล่าวนี้สามารถนำมาใช้ในการคาดคะเนได้ว่าโหนดใดที่จะคงอยู่ต่อไปในอนาคตภายในเส้นทางหนึ่ง ๆ ได้ ส่วนการวัดด้วยค่า Effectiveness นั้นจะมุ่งไปที่การประมาณค่า Throughput ของเส้นทางจากต้นทางสู่ปลายทางของการสื่อสาร (End-to-End) ซึ่งเป็นความสามารถของเส้นทางใด ๆ ที่ข้อมูลได้ถูกส่งออกไปสำเร็จภายในช่วงเวลาที่แน่นอนเวลาหนึ่ง โดยอาศัย Packet Loss Ratio ซึ่งเป็นค่าความน่าจะเป็นที่ข้อมูล Packet หนึ่ง ๆ จะไม่ได้รับอย่างถูกต้องภายในช่วงเวลาที่แน่นอนเวลาหนึ่ง นั่นหมายความว่าหากเส้นทางใดก็ตามมี Packet Loss Ratio สูง แสดงว่าข้อมูลที่ไหลไปตามเส้นทางนั้นมีโอกาสที่จะเสียหายได้มากกว่าไหลไปตามเส้นทางที่มี Packet Loss Ratio ต่ำ หากวัดปริมาณข้อมูลในช่วงเวลาที่เท่ากันช่วงเวลาหนึ่งแล้ว โดยส่งข้อมูลจากต้นทางไปยังปลายทางในปริมาณที่เท่ากันบนเส้นทางที่แตกต่างกันสองเส้นทาง เส้นทางที่มี Packet Loss Ratio สูงย่อมมีปริมาณข้อมูลที่ส่งถึงปลายทางได้สำเร็จต่อหนึ่งหน่วยเวลา (Throughput) ต่ำกว่าเส้นทางที่มี Packet Loss Ratio ต่ำ สำหรับการหาค่าความแรงของสัญญาณ (Signal Strength) และอัตราการสูญหายของข้อมูล (Packet Loss Ratio) นั้นต้องอาศัยเทคนิคการส่ง Packet ทดสอบ (Probe) โดยเทคนิคนี้จะให้โหนดแต่ละโหนดส่ง Probe เป็นระยะ ๆ ด้วยเวลาที่แน่นอนค่าหนึ่งให้กับโหนดตัวอื่น ๆ ที่อยู่ติดกับตนเองซึ่งโหนดที่ได้รับจะนำ Probe ไปคำนวณหาค่า Signal Strength และ Packet Loss Ratio ต่อไป ตัววัดค่าที่ได้กล่าวมามีรายละเอียดดังนี้

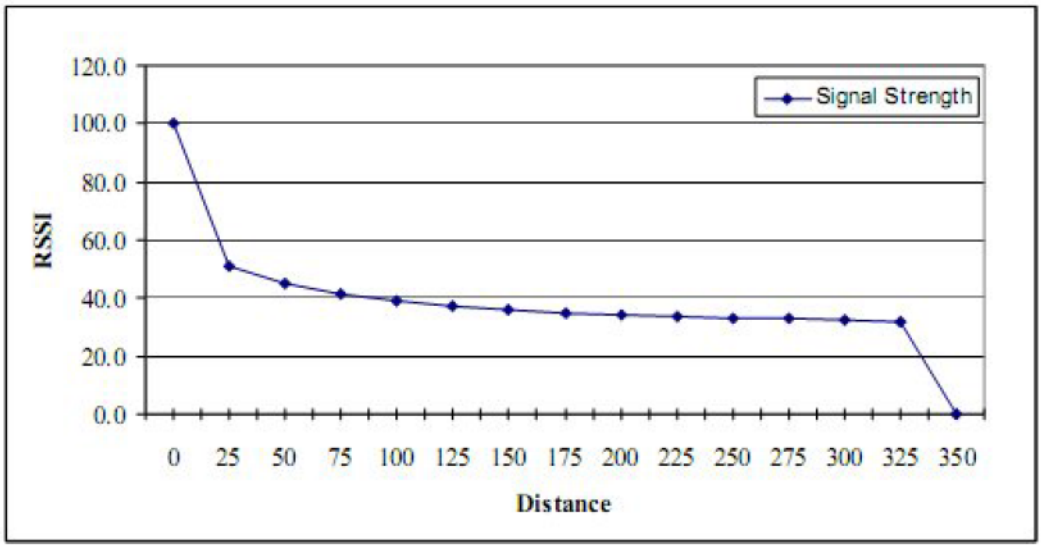
3.1.1 ตัววัดค่าความคงทนของเส้นทาง (Path Robustness Measurement)

เมื่อโหนดใด ๆ มีการเคลื่อนที่ มันมีแนวโน้มที่จะออกไปนอกขอบเขตการส่งสัญญาณวิทยุของบรรดาโหนดที่อยู่ติดกับมัน และเป็นเหตุให้การเชื่อมต่อระหว่างมันกับโหนดอื่น ๆ ที่อยู่ติดกับมันนั้นเสียหาย นอกจากนั้นหากมีโหนดตัวหนึ่งเคลื่อนตัวห่างออกไปจากมันความแรงของสัญญาณก็จะลดลงด้วยเช่นกัน จากพฤติกรรมดังกล่าว ในโปรโตคอล PMP สามารถประเมินค่าความเสียหายของเส้นทางใด ๆ ได้โดยใช้การคาดคะเน Signal Strength ที่จะเป็นไปในอนาคต ซึ่งการคาดคะเนดังกล่าวจะอาศัยการวัดค่า Signal Strength จากการทำงานใน Physical Layer ตามมาตรฐาน IEEE 802.11 โดย Signal Strength สามารถวัดได้ด้วย Receive Signal Strength Indicator (RSSI) ที่มีค่าระหว่าง 0 ถึง RSSI Max อย่างไรก็ตามยังไม่มีมาตรฐานกำหนดค่ามากสุดของ RSSI ไว้แต่ประการใด ค่านี้จึงแตกต่างกันตามบริษัทผู้ผลิต ดังนั้น โปรโตคอล PMP จึงใช้สมการความเป็นไปได้ของความแรงสัญญาณดังนี้

$$S = \frac{RSSI}{(RSSI\ Max + 1)} \qquad \text{..... (1)}$$

เมื่อ S = Signal Strength ซึ่งมีค่าตั้งแต่ 0 ถึง 1

ตัวอย่างเช่น ถ้า RSSI ที่ อุปกรณ์เชื่อมต่อเครือข่าย (Network Interface Card) บอกมีค่าเท่ากับ 16 และ ค่าของ RSSI Max = 31 แล้วค่าของ S ก็จะเท่ากับ 0.5 เป็นต้น

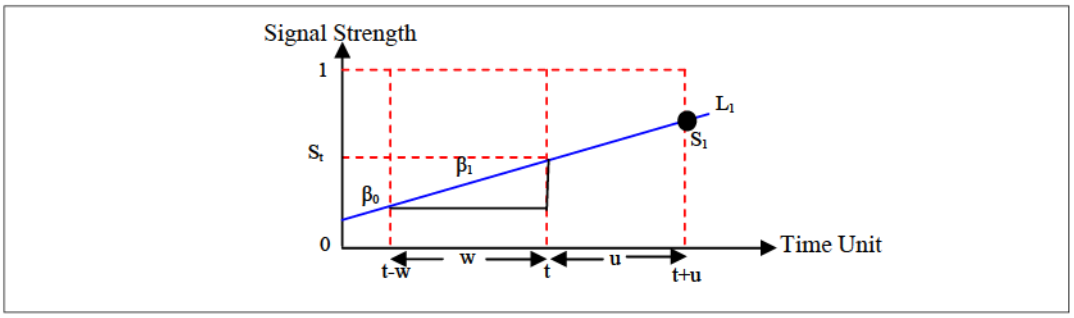


ภาพที่ 3.1 ความสัมพันธ์ระหว่าง RSSI และระยะทาง
แหล่งที่มา: Supachote Lertvorratham and Pipat Hiranvanichakorn, 2007: 2.

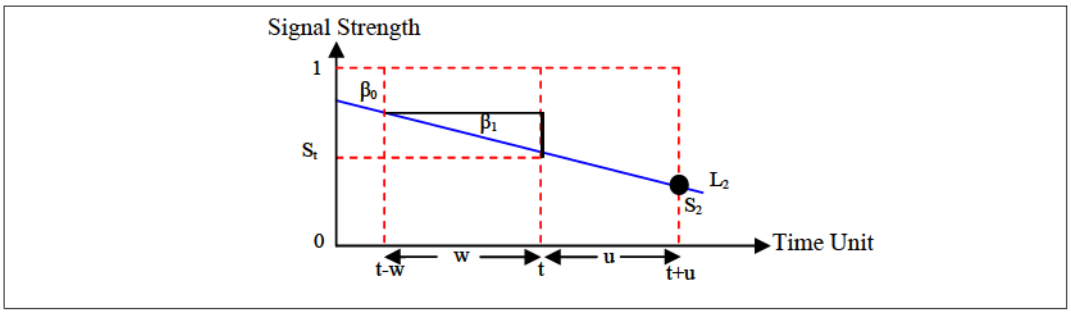
ผู้นำเสนอโปรโตคอล PMP ได้แสดงให้เห็นว่า ค่า RSSI และ ระยะห่างของโหนดคู่ที่อยู่ติดกันใด ๆ นั้น จะมีความสัมพันธ์ที่เกือบจะเป็นเส้นตรง ดังภาพที่ 3.1 ดังนั้นจึงสามารถใช้สมการเชิงเส้นตรงในการคาดคะเนค่า Signal Strength ได้

$$y = \beta_0 + \beta_1 x \qquad \text{..... (2)}$$

เมื่อ y คือ ตัวแปรตาม (Dependent Variable)
x คือ ตัวแปรต้น (Independent Variable)
β_0 คือ ค่าจุดตัดในแนวแกน y (Y-Intercept)
β_1 คือ ค่าความลาดชันของเส้นตรง(Slope of the Line)



ภาพที่ 3.2 การคาดคะเนความแรงของสัญญาณจากข้อมูลในอดีตในกรณีที่การเปลี่ยนแปลงความแรงของสัญญาณกำลังเพิ่มขึ้น



ภาพที่ 3.3 การคาดคะเนความแรงของสัญญาณจากข้อมูลในอดีตในกรณีที่การเปลี่ยนแปลงความแรงของสัญญาณกำลังลดลง

ในการคาดคะเนค่าความแรงสัญญาณ โหนดทุกตัวจะมีวินโดว์ W ที่มีขนาด n หน่วย สำหรับเก็บค่าความแรงสัญญาณที่ได้รับจากการ Probe ของโหนดที่อยู่ติดกับตนเองทุก ๆ เวลา i ตามภาพที่ 3.2 แสดงตัวอย่างความสัมพันธ์ของความแรงสัญญาณระหว่าง 2 โหนดเมื่อมีการเคลื่อนที่ หากการเปลี่ยนแปลงของเส้น L_1 เป็นเส้นแสดงการเปลี่ยนแปลงความแรงสัญญาณที่กำลังเพิ่มขึ้นแล้วความแรงสัญญาณที่คาดคะเนล่วงหน้า ณ ช่วงเวลา u หน่วยถัดมาหลังจากเวลา w หน่วยจะเป็น S_1 ในทางตรงข้ามหากการเปลี่ยนแปลงความแรงสัญญาณกำลังลดลงดังแสดงด้วยเส้น L_2 ตามภาพที่ 3.3 ความแรงสัญญาณที่คาดคะเนล่วงหน้าก็จะลดลงเป็น S_2 จากทั้งสองกรณีที่กล่าวมานี้เมื่อ β_0 เป็นค่าความแรงสัญญาณ ณ เวลา t-w แล้วจะสามารถคำนวณ β_1 ได้ ซึ่ง β_1 นั้นเป็นค่าสัมประสิทธิ์ความลาดชันตามสมการเชิงเส้นที่หาได้จากค่าความแรงสัญญาณในเวลาที่ผ่านมาในช่วงเวลา t-w ถึง เวลา t โดยใช้สมการดังนี้

$$\beta_1 = \frac{\mathrm{cov}(TU,S)}{V_{TU}^2} \quad \ldots\ldots\ldots\ldots\ldots (3)$$

$$V_{TU}^2 = \frac{\sum\left(TU_i - \overline{TU}\right)^2}{n-1} \quad \ldots\ldots\ldots\ldots\ldots (4)$$

$$\mathrm{cov}(TU, S) = \frac{\sum\left(TU_i - \overline{TU}\right)\left(S_i - \bar{S}\right)}{n-1} \quad \ldots\ldots\ldots\ldots\ldots (5)$$

β_1 = ค่าสัมประสิทธิ์ความลาดชัน (Slope Coefficient) ของสมการเชิงเส้น

cov(TU, S) = สัมประสิทธิ์ของเวลาและความแรงสัญญาณ (Coefficient of Time Unit and Signal Strength)

V_{TU}^2 = ค่าความแปรปรวนของเวลา (Variance of Time) จากเวลา t-w ถึงเวลา t

TU_i = เวลา i จากเวลา t-w ถึงเวลา t

S_i = ความแรงสัญญาณ (Signal Strength) ณ เวลาใด ๆ i จากเวลา t-w ถึงเวลา t

$\bar{S}$ = $\frac{\sum S_i}{n}$ = ความแรงสัญญาณโดยเฉลี่ย (Average Signal Strength) จากเวลา t-w ถึงเวลา t

$\overline{TU}$ = $\frac{\sum TU_i}{n}$ = เวลาเฉลี่ย (Average Time) จากเวลา t-w ถึงเวลา t

n = ขนาดของวินโดว์ (Number of Window Size) ที่เก็บค่าความแรงสัญญาณซึ่งจะได้รับจาก Probe ของโหนดที่อยู่ติดกับตนตลอดช่วงเวลาของการ Probe

กำหนดให้ช่วงเวลา u เป็นเวลาที่โหนดตัวหนึ่ง ๆ ยังคงรักษาเส้นทางนั้นไว้ให้มีความสามารถที่จะนำมาใช้งานได้อยู่ ดังนั้นโหนดตัวหนึ่ง ๆ จะคาดคะเนค่าความแรงสัญญาณจากโหนดอื่นที่อยู่ติดกับมัน ณ เวลาปัจจุบัน t ตามสมการ

$$S_{(t+u)} = \min(1, \beta_1 u + S_t) \quad \ldots\ldots\ldots\ldots\ldots (6)$$

เมื่อ $S_{(t+u)}$ = ความแรงสัญญาณของโหนดที่อยู่ติดกับโหนดนั้น ณ ช่วงเวลา u หน่วยโดยจะมีค่าตั้งแต่ 0 ถึง 1

S_t = ความแรงสัญญาณ ณ เวลาปัจจุบัน t

β_1 = สัมประสิทธิ์ความลาดชันที่ได้มาจากพฤติกรรมช่วงสั้น ๆ ในลักษณะเชิงเส้น

การสูญหายของสัญญาณการเชื่อมต่อระหว่างโหนดคู่ใดคู่หนึ่ง (Link) เพียงคู่เดียวของเส้นทางใด ๆ เป็นสาเหตุหลักที่ทำให้เส้นทางนั้นเสียหายได้ จึงต้องพิจารณาถึงค่าความแรงสัญญาณต่ำสุดของโหนดแต่ละคู่ทุก ๆ คู่ของเส้นทางนั้นเป็นสำคัญ เพื่อใช้ในการกำหนดว่าเส้นทางนั้นมีโอกาสที่จะเสียหายได้มากหรือน้อยเพียงใด โดยมีการกำหนดค่า Degree of Path Availability (DA) ซึ่งหากค่า DA นี้มีค่าน้อยกว่าค่าวิกฤตแล้วจะกำหนดให้เส้นทางนั้นเสียหาย

$$DA = \min(S_1, S_2, S_3, \ldots, S_i, \ldots, S_n) \quad \ldots\ldots\ldots\ldots\ldots\ldots (7)$$

เมื่อ DA คือ Degree of Path Availability

S_i คือ ความแรงสัญญาณของ Link ที่ i ใน เส้นทางนั้น ๆ

3.1.2 ตัววัดค่าความมีประสิทธิภาพของเส้นทาง (Path Efficiency Measurement)

ในการคำนวณตัววัดค่าความมีประสิทธิภาพของเส้นทาง มีการใช้อัตราการสูญหายของข้อมูล (Packet Loss Ratio) ซึ่งคือ สัดส่วนระหว่างจำนวนของ Packet ที่สูญหายในระหว่างการส่งข้อมูลต่อจำนวน Packet ทั้งหมดที่โหนดหนึ่ง ๆ ควรจะได้รับ โดยค่า Packet Loss Ratio นี้ ในทุก ๆ เวลา t โหนดต่าง ๆ จะนำ Probe ที่ใช้ในการคาดคะเน Signal Strength ซึ่งเก็บไว้ในวินโดว์ที่มีขนาด n มาคำนวณ Packet Loss Ratio ตามสมการ

$$L = \frac{(E - P)}{E} \quad \ldots\ldots\ldots\ldots\ldots\ldots (8)$$

โดย L = Packet Loss Ratio

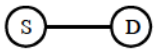
P = จำนวน Probe ที่ได้รับระหว่างการเปิดวินโดว์ที่มีขนาด n

E = จำนวน Probe ที่ควรจะได้รับแล้วเสร็จ

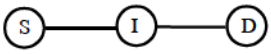
ค่าของ L ที่ได้นั้นจะนำไปคำนวณ Throughput ของเส้นทางต่อไป

ค่าประสิทธิผลในการส่งข้อมูลของเส้นทาง (Estimated Path Throughput Value) คือ จำนวน Packet ทั้งหมดที่ถูกส่งโดยต้นทางและรับโดยปลายทางอย่างสมบูรณ์ภายในช่วงระยะเวลาหนึ่ง ซึ่งขึ้นอยู่กับค่าการสูญหายของ Packet และจำนวน Hop ของเส้นทางนั้น ๆ ดังเช่นเมื่อข้อมูลที่ส่งไปนั้นเกิดการสูญหายโหนดที่ทำหน้าที่ส่งข้อมูลต้องส่งข้อมูลนั้นซ้ำไปอีกครั้งหนึ่งจึงทำให้ประสิทธิผลในการส่งข้อมูลลดลง 50% ดังนั้นหากพิจารณาเส้นทางการส่งข้อมูลเมื่อไม่มีโหนดระหว่างทาง ดังภาพที่ 3.4 ถ้าไม่มีการสูญหายของข้อมูลเลยในระหว่างการส่งข้อมูลโหนดต้นทางก็จะส่งข้อมูลออกจากตัวมันเองเพียง 1 ครั้งเท่านั้น นั่นหมายความว่าข้อมูลจำนวน 1 Packet ถึงโหนด

ปลายทางได้โดยมีการส่งข้อมูลเพียงจำนวน 1 Packet เท่านั้น ดังนั้นประสิทธิผลในการส่งข้อมูลของเส้นทางนี้ก็จะเป็น 100%



ภาพที่ 3.4 เส้นทางที่ไม่มีโหนดระหว่างทาง



ภาพที่ 3.5 เส้นทางที่มีโหนดระหว่างทางจำนวน 1 โหนด

แต่หากพิจารณาเส้นทางการส่งข้อมูลเมื่อมีโหนดระหว่างทางจำนวนหนึ่งโหนด ดังแสดงภาพที่ 3.5 หากไม่มีการสูญหายของข้อมูลในระหว่างการส่งข้อมูลเลย โหนดต้นทางและโหนดระหว่างทางก็จะส่งข้อมูลออกจากตัวมันเองตัวละ 1 ครั้งหรือตัวละ 1 Packet รวมกันเป็นจำนวน 2 Packet เพื่อที่จะให้ข้อมูล 1 Packet ที่ต้องการส่งไปยังปลายทางได้สำเร็จ ดังนั้นประสิทธิผลในการส่งข้อมูลของเส้นทางนี้ก็จะเป็น 50% จากที่ได้กล่าวมาจะเห็นได้ว่าประสิทธิผลในการส่งข้อมูลของเส้นทางจะขึ้นอยู่กับจำนวนโหนดระหว่างทางของเส้นทางนั้นด้วย และเมื่อพิจารณาในสถานการณ์ที่มีอัตราการสูญหายของข้อมูล (Packet Loss Ratio) ดังภาพที่ 3.5 ในกรณีที่โหนดต้นทางและโหนดระหว่างทางจะต้องส่งข้อมูลออกจากตัวมันเองตัวละสองครั้ง หรือตัวละ 2 Packet รวมกันเป็นจำนวน 4 Packet เพื่อให้ข้อมูลจำนวน 1 Packet ที่ต้องการส่งถึงโหนดปลายทางได้โดยสมบูรณ์ ค่าประสิทธิผลในการส่งข้อมูลของเส้นทางนี้จะเป็น $\frac{1}{4}$ หรือ 0.25

ค่าประสิทธิผลในการส่งข้อมูลของเส้นทาง (ETV) คำนวณโดยใช้การประเมินจากจำนวน Probe Packet ที่ถูกส่งในเส้นทางนั้น ๆ ภายในระยะเวลาที่กำหนด ซึ่งค่าของ ETV นั้นจะมีค่าอยู่ระหว่าง 0 และ 1 เมื่อมาพิจารณาภาพที่ 3.4 ซึ่งแสดงการสื่อสารของโหนดสองโหนด คือ โหนด S และ โหนด D นั้น หากการสื่อสารของโหนดทั้งสองสมบูรณ์ กล่าวคือไม่มีการสูญหายของข้อมูลเลย และได้ส่งข้อมูลจำนวน 1 Packet แล้วการคำนวณหาค่า ETV จะเป็นไปตามสมการ

$$\text{ETV} = \frac{1}{\text{Total Number of Flooding Packet Along the Path}} \quad \ldots\ldots\ldots\ldots\ldots\ldots (9)$$

จากสมการจะทำให้ค่าสูงสุดของ ETV มีค่าเป็น 1 เนื่องจากมีเพียงข้อมูลจำนวน 1 packet เท่านั้นที่จะถูกส่งออกไปในเส้นทางนั้น ๆ แต่หากมีโหนดระหว่างทางเพิ่มขึ้นมาในเส้นทางดังแสดงในภาพที่ 3.5 จะต้องมีข้อมูลอย่างน้อยที่สุด จำนวน 2 Packet ที่จะกระจายอยู่ภายในเส้นทางนั้น เพื่อให้การส่งข้อมูลจำนวน 1 Packet ไปยังปลายทางที่ต้องการได้ หากไม่มีการสูญหายของข้อมูลเลย ETV ก็จะมีค่าเป็น 0.5 นั่นเอง ในสถานการณ์ที่มี Packet สูญหาย 50% ด้วยการสื่อสารจำนวนสอง Hop เส้นทางนั้นอาจจะต้องมีข้อมูลถึง 4 Packet กระจายอยู่ภายในเครือข่ายต่อการส่งข้อมูล 1 Packet เพื่อให้ Packet นั้นถึงปลายทางได้ ดังนั้นค่า ETV เท่ากับ $\frac{1}{4}$ นั่นเอง

เนื่องจาก Packet Loss Ratio ได้ถูกนำมาใช้ในการประเมินจำนวนข้อมูลที่ส่งตลอดเส้นทางนั้น และหากกำหนดค่า Throughput มากสุดให้เป็น 1 แล้ว ETV ก็จะเป็นดังสมการ

$$ETV = \frac{1}{\sum_{i=1}^{n} \frac{1}{1-L_i}} \qquad \text{..... (10)}$$

เมื่อ L_i คือ Packet Loss Ratio ของ Link i

n คือ จำนวนของ Link ทั้งหมดในเส้นทางนั้น ๆ

ในโปรโตคอลนี้จะใช้ค่า ETV ในการพิจารณาคัดเลือกเส้นทางต่าง ๆ ตามระดับความสำคัญของเส้นทาง (Rank) โดยให้เส้นทางที่มีค่า ETV สูงกว่าเป็นเส้นทางที่มีระดับความสำคัญสูงกว่า ซึ่งเส้นทางที่มีค่าระดับความสำคัญสูงที่สุดจะเป็นเส้นทางหลัก ส่วนเส้นทางที่มีค่าระดับความสำคัญต่ำกว่าก็จะเป็นเส้นทางสำรองเพื่อใช้ในการทำงานต่อไป

3.2 โครงสร้างของ Route Table

ภายใน Route Table แต่ละ Entry คือ เส้นทางที่ไปยังโหนดปลายทางโดยดรรชนีของแต่ละ Entry ถูกใช้เป็นค่า Label สำหรับกระบวนการ Label Switching และแต่ละ Entry ประกอบไปด้วยส่วนของ Label และส่วนของ Path

ในส่วนของ Label จะประกอบไปด้วย

- outLb หมายถึง Label ที่จะถูกใช้เป็นดรรชนีสำหรับหาเส้นทางใน Route Table ของโหนดถัดไป
- nextHop หมายถึง Address ของโหนดถัดไป

ในส่วนของ Path จะประกอบไปด้วย

- Dest หมายถึง Address ของโหนดปลายทาง
- Rank หมายถึง ระดับความสำคัญของเส้นทางที่ไปยังปลายทางหนึ่ง ๆ
- Upstream หมายถึง Address ของโหนดด้าน Upstream ที่อยู่ติดกับโหนดนั้น
- Post Route หมายถึง รายการ Address ของโหนดต่าง ๆ ในด้าน Downstream ซึ่งเรียงลำดับจากโหนดที่อยู่ติดกับโหนดนั้นจนถึงโหนดปลายทาง

Index	outLb	nextHop	Dest	Rank	Upstream	Post Route

ภาพที่ 3.6 โครงสร้าง Route Table ของโปรโตคอล PMLS

3.3 โครงสร้างของ Packet ต่าง ๆ

3.3.1 Route Request Packet

Seq#	reqID	Dest	Source Route Information

ภาพที่ 3.7 โครงสร้าง Route Request Packet (RREQ) ของโปรโตคอล PMLS

หมายเหตุ: ฟิลด์ต่าง ๆ ของ RREQ มีความหมายดังนี้

- **Seq#** หมายถึง หมายเลขลำดับการส่ง Route Request ของโหนดต้นทางซึ่งจะเพิ่มค่าขึ้นทีละหนึ่งเมื่อต้องส่ง RREQ ซ้ำให้โหนดที่อยู่ติดกับตนหลังจากรอคอยอยู่เป็นเวลาค่าหนึ่งแล้วยังไม่ได้รับ Route Reply กลับมา เพื่อระบุว่าเป็น RREQ ล่าสุด
- **reqID** หมายถึง ตัวระบุความแตกต่างของ RREQ เพื่อป้องกันไม่ให้โหนดต่าง ๆ รับ Packet นี้ซ้ำและใช้ในการส่ง Route Reply ต่อ RREQ นี้
- **Dest** หมายถึง Address ของโหนดปลายทาง
- **Source Route Information** หมายถึง ข้อมูลต่าง ๆ ของเส้นทางซึ่งประกอบไปด้วยค่า Packet Loss Ratio, Signal Strength และ Address ของโหนดต่าง ๆ ที่ Packet นี้ได้ท่องผ่านมา จะสังเกตได้ว่า Source Route Information จะมีขนาดยาวเพิ่มมากขึ้นเรื่อย ๆ จนกระทั่งถึงโหนดปลายทาง

Packet นี้จะถูกใช้ในหัวข้อ 3.4.1

3.3.2 Route Reply Packet

reqID	outLb	receiverAddr	senderAddr	Rank	Route

ภาพที่ 3.8 โครงสร้าง Route Reply Packet (RREP) ของโปรโตคอล PMLS

หมายเหตุ: ฟิลด์ต่าง ๆ ของ RREP มีความหมายดังนี้

- **reqID** หมายถึง ตัวระบุการตอบกลับต่อ Route Request เพื่อให้โหนดต้นทางสามารถทราบว่า Route Request ที่ตนได้ส่งไปนั้นได้รับการตอบกลับมาเรียบร้อยแล้ว

- **outLb** (Outgoing Label) หมายถึง ค่า Label ที่ถูกใช้เป็นดรรชนีในการค้นหาเส้นทางของโหนดที่ส่ง RREP นี้ ซึ่งจะเป็นค่าเดียวกับ outLb ของเส้นทางเดียวกันที่ จะถูกเพิ่มลงใน Route table ของโหนดผู้รับ RREP นี้ (receiverAddr) หากโหนดที่ส่ง RREP นี้เป็นโหนดปลายทางสำหรับเส้นทางนี้ มันจะมีค่าเป็น SIL (Self-Identification Label)ซึ่งจะมีค่าเดียวกันสำหรับโหนดปลายทางทุก ๆ โหนดในเครือข่าย

- **receiverAddr** หมายถึง Address ของโหนดผู้รับ RREP นี้

- **senderAddr** หมายถึง Address ของโหนดผู้ส่ง RREP นี้

- **Rank** หมายถึง ค่าแสดงระดับความสำคัญของเส้นทางนี้

- **Route** หมายถึง เส้นทางที่สมบูรณ์ทั้งหมดของเส้นทางนี้ซึ่งจะประกอบไปด้วยบรรดา Address ของโหนดต่าง ๆ ตั้งแต่ต้นทางจนถึงปลายทาง

Packet นี้จะถูกใช้ในหัวข้อ 3.4.2

3.3.3 Data Packet

Seq#	outLb	nextHop	Data

ภาพที่ 3.9 โครงสร้าง Data Packet ของโปรโตคอล PMLS

หมายเหตุ: ฟิลด์ต่าง ๆ ของ Data Packet มีความหมายดังนี้

- **Seq#** (Sequence Number) หมายถึง หมายเลขลำดับการส่ง Data Packet ของโหนดต่าง ๆ ซึ่งจะเพิ่มค่าขึ้นทีละหนึ่งเมื่อต้องส่ง Data Packet ซ้ำให้โหนดที่อยู่ติดกับตนหลังจากรอคอยอยู่เป็นเวลาค่าหนึ่งแล้วยังไม่ได้รับ ACK กลับมา

- **outLb** (Outgoing Label) หมายถึง Label ที่เป็นดรรชนีในการค้นหาเส้นทางของโหนดที่รับ Data Packet นี้ ซึ่งจะเป็นค่าเดียวกับ outLb ของเส้นทางเดียวกันที่จะได้จาก Route Table ของโหนดที่ส่ง Data Packet นี้ หากโหนดที่รับ Data Packet นี้เป็นโหนดปลายทางสำหรับเส้นทางนี้มันจะมีค่าเป็น SIL เสมอ

- **nextHop** หมายถึง Address ของโหนดผู้รับ Data Packet นี้ ซึ่งจะเป็นค่าเดียวกับnextHop ของเส้นทางเดียวกันที่ได้จาก Route Table ของโหนดที่ส่ง Data Packet นี้

Packet นี้จะถูกใช้ในหัวข้อ 3.5

3.3.4 Route Error Packet

receiverAddr	brokenAddr

ภาพที่ 3.10 โครงสร้าง Route Error Packet (RERR) ของโปรโตคอล PMLS

หมายเหตุ: ฟิลด์ต่าง ๆ ของ RERR มีความหมายดังนี้

- **receiverAddr** หมายถึง Address ของโหนดที่อยู่ติดกับโหนดที่ส่ง RERR นี้ซึ่งอยู่ในด้าน Upstream

- **brokenAddr** หมายถึง Address ของโหนดคู่ใด ๆ ของ Link ที่มีความผิดพลาดเกิดขึ้นซึ่งจะประกอบไปด้วย Address ของโหนดที่เริ่มรายงานความเสียหายที่เกิดขึ้นกับโหนดที่เสียหาย

Packet นี้จะถูกใช้ในหัวข้อ 3.6

3.3.5 Local Repair Request Packet

lreqID	Dest	senderAddr

ภาพที่ 3.11 โครงสร้าง Local Repair Request Packet (LREQ) ของโปรโตคอล PMLS

หมายเหตุ: ฟิลด์ต่าง ๆ ของ LREQ มีความหมายดังนี้

- **lreqID** หมายถึง ตัวระบุลำดับการส่ง Request เพื่อป้องกันไม่ให้โหนดต่าง ๆ รับ LREQ นี้ซ้ำ

- **Dest** หมายถึง Address ของโหนดปลายทางของเส้นทางที่ต้องการ

- **senderAddr** หมายถึง Address ของโหนดที่ส่ง LREQ นี้

Packet นี้จะถูกใช้ในหัวข้อ 3.6

3.3.6 Local Repair Reply Packet

lreqID	outLb	receiverAddr	senderAddr	Rankl	Post Route

ภาพที่ 3.12 โครงสร้าง Local Repair Reply Packet (LREP) ของโปรโตคอล PMLS

หมายเหตุ: ฟิลด์ต่าง ๆ ของ LREP มีความหมายดังนี้

- **lreqID** หมายถึง ตัวระบุการตอบกลับต่อ Request เพื่อให้โหนดที่ส่ง Request สามารถทราบว่า Request ที่ตนได้ส่งไปนั้นได้รับการตอบกลับมาเรียบร้อยแล้ว

- **outLb** (Outgoing Label) หมายถึง Label ที่เป็นดรรชนีในการค้นหาเส้นทางนี้ของโหนดที่ส่ง LREP นี้ ซึ่งจะเป็นค่าเดียวกับ outLb ของเส้นทางเดียวกันที่จะถูกเพิ่มลงใน Route Table ของโหนดที่รับ LREP นี้ หากโหนดที่ส่ง LREP นี้เป็นโหนดปลายทางสำหรับเส้นทางนี้ มันจะมีค่าเป็น SIL (Self-Identification Label) เสมอ

- **receiverAddr** หมายถึง Address ของโหนดผู้รับ LREP นี้

- **senderAddr** หมายถึง Address ของโหนดผู้ส่ง LREP นี้

- **Rankl** หมายถึง ค่าระดับความสำคัญของเส้นทางนี้ ซึ่งจะถูกกำหนดให้มีค่าน้อยกว่าค่าระดับความสำคัญของทุกเส้นทางที่ได้จาก RREP ของกระบวนการ Route Discovery

- **Post Route** หมายถึง เส้นทางบางส่วนของเส้นทางนี้ซึ่งจะประกอบไปด้วยบรรดา Address ของโหนดต่าง ๆ ตั้งแต่โหนดถัดไปจนถึงปลายทาง

Packet นี้จะถูกใช้ในหัวข้อ 3.6

3.4 กระบวนการหาเส้นทาง (Route Discovery)

กระบวนการ Route Discovery จะถูกใช้เมื่อโหนดใด ๆ ต้องการจะติดต่อสื่อสารกับโหนดอื่น แต่ไม่มีเส้นทางให้ใช้งานได้

3.4.1 การส่ง Route Request (RREQ)

หากไม่มีเส้นทางไปยังปลายทางโหนดต้นทางจะแนบ Address ของตนเองเข้าไปใน Route Request Packet (RREQ) ก่อนที่จะแพร่ RREQ นั้น ไปในเครือข่าย และจะรอคอยอยู่ชั่วขณะหนึ่งเพื่อรับ Route Reply Packet (RREP) จากปลายทาง หากรอจนถึงเวลาหนึ่งแล้วไม่มี RREP ตอบกลับมาจากปลายทาง โหนดต้นทางนั้นก็จะส่ง RREQ ซ้ำไปเรื่อย ๆ จนถึงขีดจำกัดค่าหนึ่ง โหนดต้นทางก็จะหยุดการส่ง RREQ แม้ว่าจะไม่ได้รับ RREP จากปลายทางก็ตาม

เมื่อ RREQ จากต้นทางมาถึงโหนดใด ๆ ที่อยู่ภายในเครือข่ายโหนดต่าง ๆ เหล่านั้นก็จะตรวจสอบว่าตนเป็นโหนดปลายทางหรือไม่หากตนไม่ใช่โหนดปลายทางให้ดำเนินการตามขั้นตอนต่าง ๆ ดังนี้

1. หากหมายเลขลำดับ (Seq#) ของ RREQ เป็นลำดับใหม่ แสดงว่ามีการเริ่มส่ง RREQ เดียวกันจากโหนดต้นทางใหม่อีกครั้งหนึ่ง เนื่องจากโหนดต้นทางรอคอยจนถึงเวลาหนึ่งแล้วยังไม่ได้รับ RREP กลับมา ดังนั้นให้รับ RREQ นี้และเก็บมันไว้ใน Cache ของตนแล้วทำตามขั้นตอนต่อไป เพราะหากไม่ทำเช่นนี้แล้วข้อมูลของเส้นทางเก่าที่เก็บอยู่ใน Cache ซึ่งเกิดจากกระบวนการ Route Discovery ในครั้งก่อนซึ่งโหนดต้นทางไม่สามารถรับ RREP ได้แล้วจะถูกนำมาใช้ในการทำงานต่อไปเหมือนเดิม

2. ตรวจสอบว่า RREQ ที่เข้ามานั้นเป็น Packet ที่ซ้ำกับของเดิมหรือไม่ โดยพิจารณาจาก reqID, Dest และ Soure Route Information หากซ้ำให้ทิ้ง RREQ นี้ไปแล้วทำตามขั้นตอนต่อไป

3. ตรวจสอบว่า Source Route Information ของ RREQ นั้นซึ่งมีฟิลด์ของ Address ของโหนดต่าง ๆ ที่ RREQ นี้ได้ท่องผ่านมา มี Address ของตนอยู่หรือไม่ เพื่อป้องกันการเกิดการซ้ำวนของเส้นทาง (Route Loop) หากมีอยู่แล้วให้ทิ้ง RREQ นี้ไป หากไม่มีให้ทำตามขั้นตอนต่อไป

4. ตรวจสอบค่า Signal Strength ของ Link ที่ผ่านมายังโหนดนี้ (Link ที่ส่งข้อมูลมาให้มันหรือ Link ที่เชื่อมโยงระหว่างโหนดที่ส่ง RREQ กับตัวมัน) ว่าต่ำกว่าค่าวิกฤตหรือไม่ หากต่ำกว่าให้ทิ้ง RREQ นี้ไป ในกรณีที่ไม่ต่ำกว่าให้ทำตามขั้นตอนต่อไป

5. ตรวจสอบว่า Source Route Information นั้นมีเส้นทางบางส่วนซ้อนทับ (Overlapped Path) กับเส้นทางที่ผ่านมายังโหนดนี้หรือไม่ หากไม่มีให้เก็บ Source Route Information นี้ไว้ใน Cache หากมี Overlapped Path แม้เพียงบางส่วนก็ตามให้ตรวจสอบ ETV ของเส้นทางนี้ว่าสูงกว่าเส้นทางก่อนหน้านี้หรือไม่ หากมีค่าต่ำกว่าให้ทิ้ง RREQ นี้ไป หากสูงกว่าให้นำ Source Route Information นี้มาเก็บทับแทนเส้นทางก่อนหน้านี้ใน Cache และรับ RREQ นี้ พร้อมกับแนบ Address ของมันและ ค่า Signal Strength กับ ค่า Packet Loss Ratio ของ Link ที่ผ่านมายังโหนดนี้

เข้าไป ใน Source Route Information หลังจากนั้นโหนดนี้จะส่ง RREQ นั้นให้กับโหนดที่อยู่ติดกับมันต่อไป และ RREQ จะถูกส่งต่อไปเรื่อย ๆ จนถึงโหนดปลายทาง

เมื่อโหนดปลายทางรับ RREQ แรกแล้วโหนดปลายทางจะรอคอยอยู่เป็นเวลาค่าหนึ่งเพื่อเก็บรวบรวมเส้นทางอื่นจาก RREQ อื่น ๆ อีก และจะนำเส้นทางเหล่านั้นมาพิจารณาเส้นทางที่สามารถนำมาใช้งานได้โดยดำเนินการดังนี้

1. คำนวณค่า DA และ ETV ของแต่ละเส้นทาง และตัดเส้นทางที่มีค่า DA ต่ำกว่าค่าวิกฤตทิ้งไป

2. เส้นทางต่าง ๆ ที่มีส่วนของเส้นทางซ้อนทับกัน (Overlapped Path) จะเปรียบเทียบค่า ETV ของมัน และเลือกเส้นทางที่มี ETV มากที่สุด ส่วนที่เหลือให้ตัดทิ้งไป

3. จัดระดับความสำคัญ (Rank) ของเส้นทางที่เลือกไว้ด้วยค่า ETV โดยเส้นทางใดที่มีค่า ETV สูงสุดก็จะมีค่าระดับความสำคัญสูงสุด

อนึ่งเพื่อป้องกันปัญหาการพบเส้นทางจำนวนมากเกินควร โหนดปลายทางจึงต้องจำกัดจำนวนของเส้นทางที่จะนำมาใช้ได้ และหยุดการเลือกเส้นทางเมื่อจำนวนเส้นทางที่รับมานั้นมีค่าถึงขีดสุดตามที่กำหนด

3.4.2 การส่ง Route Reply (RREP)

หลังจากที่โหนดปลายทางได้คัดสรรเส้นทางจาก RREQ เสร็จสิ้นแล้ว ข้อมูลของเส้นทางแต่ละเส้นทางที่ผ่านเกณฑ์การพิจารณาจะถูกบรรจุไว้ใน Route Reply Packet (RREP) เส้นทางละ 1 Packet และส่งกลับไปยังโหนดต้นทาง หากมีเส้นทางมากกว่าหนึ่งเส้นทางโหนดปลายทางจะกำหนดค่าระดับความสำคัญ (Rank) เป็น Sequence Number ของ RREP เหล่านั้นด้วย และเนื่องจากในการส่งข้อมูลนั้นใช้หลักการ Label Switching ซึ่งแต่ละโหนดจะมี Route Table โดยแต่ละ Entry ของ Route Table นั้นคือเส้นทางที่ไปยังโหนดปลายทางโดยที่ค่า Label ถูกใช้เป็นดรรชนีของแต่ละ Entry ดังนั้นจึงใช้ RREP เพื่อช่วยสร้าง Route Table ของแต่ละโหนดโดยโหนดปลายทางจะจัดเตรียม Label ที่บ่งบอกถึงตนเอง (Self-Identification Label) ไว้ล่วงหน้าแนบเป็นค่า outLb ไปกับ RREP ด้วย และเมื่อโหนดระหว่างทางได้รับ RREP มันจะนำข้อมูลของเส้นทางและค่า outLb ที่แนบมากับ RREP นั้นมาเก็บไว้ภายใน Route Table ของตน พร้อมกับนำ Label ซึ่งเป็นดรรชนีที่ชี้ Entry ของเส้นทางนี้ของ Route Table ของตนมาใส่เป็นค่าแทน outLb ของเดิมใน RREP นั้นแล้วส่งกลับไปยังโหนดที่อยู่ติดกับตนเองในด้าน Upstream ต่อไป ทำเช่นนี้ไปเป็นทอด ๆ จนกระทั่ง RREP นั้นไปถึงโหนดต้นทาง โหนดต้นทางก็จะนำข้อมูลเส้นทางนั้นมาเก็บไว้ใน Route Table ของตนเพื่อใช้ในการส่งข้อมูลต่อไป

3.5 กระบวนการนำส่งข้อมูล (Data Forwarding)

เมื่อโหนดใด ๆ ต้องการส่งข้อมูลไปยังปลายทางมันจะหาเส้นทางที่ดีที่สุดซึ่งมีค่าระดับความสำคัญสูงสุดใน Route Table โดยใช้ Address ของปลายทางเป็นดรรชนีในการค้นหาหากมีเส้นทางที่ต้องการ มันจะนำ outLb และ nextHop จาก Entry นั้น มาเติมเป็นส่วนหัว (Header) ของข้อมูลที่ต้องการส่ง แล้วแพร่ Data Packet นี้ไปยังโหนดที่อยู่ติดกันกับมัน เมื่อโหนดที่อยู่ติดกันกับมันได้รับ Data Packet มันจะทำตามขั้นตอนต่าง ๆ ดังนี้

1. ตรวจสอบว่าค่า nextHop ของ Packet นี้เป็น Address ของตนหรือไม่ หากไม่ใช่ให้ทิ้ง Packet นี้ไปหากใช่จะส่ง ACK ให้กับโหนดที่ส่ง Packet นี้มา แล้วตรวจสอบว่าเคยรับ Packet นี้มาก่อนหรือไม่ โดยพิจารณาจากค่า Seq# หากเคยรับมาก่อนให้ทิ้ง Packet นี้ไป หากยังไม่เคยรับมาก่อนให้ทำตามขั้นตอนต่อไป

2. ให้ตรวจสอบ outLb ที่มากับ Packet นี้ หากเป็น Self-Identification Label (SIL) แสดงว่าตัวมันเป็นโหนดปลายทาง ให้ส่งข้อมูลนี้แก่ Layer ถัดไป หาก outLb ที่มากับ Packet นี้ ไม่ใช่ SIL ให้ทำตามขั้นตอนต่อไป

3. นำ outLb ที่มากับ Packet นี้เป็นดรรชนีในการค้นหาส่วนของ Label ใน Route Table ของตนเพื่อนำ outLb และ nextHop ใน Route Table มาเป็นส่วนหัวของข้อมูลนี้เพื่อส่งต่อให้กับโหนดถัดไป

เมื่อโหนดระหว่างทางส่งข้อมูลให้กับโหนดถัดไปซึ่งอยู่ติดกันกับมันไม่ได้รับการตอบรับกลับมาในเวลาที่กำหนด มันจะส่งข้อมูลนี้ซ้ำไปใหม่จนกระทั่งถึงขีดจำกัดค่าหนึ่ง หากยังไม่มีการตอบรับกลับมาอีก มันจะเริ่มทำตามขั้นตอนต่าง ๆ ดังนี้

1. ทำการ Invalid เส้นทางที่ใช้ในการส่งข้อมูลนั้นใน Route Table ของตนเอง พร้อมทั้งสร้าง Route Error Packet (RERR) รายงานความเสียหายของ Link ที่เชื่อมโยงระหว่างตนเองกับโหนดที่ตนส่งข้อมูลนั้นไปให้บรรดาโหนดต่าง ๆ ที่อยู่ติดกันกับมัน แล้วทำตามขั้นตอนต่อไป

2. ค้นหาเส้นทางใหม่ภายใน Route Table ของตนโดยใช้ Address ของปลายทาง (Dest) เป็นดรรชนีในการค้นหา หากเจอเส้นทางใหม่ให้นำส่วนของ Label และ nextHop ของเส้นทางใหม่นั้นมาเป็นส่วนหัวของข้อมูลส่งให้แก่โหนดถัดไปต่อไป หากค้นหาแล้วไม่เจอเส้นทางใด ๆ เลย ให้เริ่มกระบวนการ Route Maintenance ต่อไป

3.6 กระบวนการบำรุงรักษาเส้นทาง (Route Maintenance)

แต่ละโหนดจะมีการตรวจจับความเสียหายของเส้นทาง 3 กรณีดังนี้

1. กรณีที่ส่งข้อมูลไปยังโหนดถัดไปแต่ไม่ได้รับ ACK กลับมาในเวลาที่กำหนดจนกระทั่งพยายามส่งข้อมูลซ้ำไปเป็นจำนวนครั้งเกินขีดจำกัดที่ได้กำหนดไว้

2. กรณีที่ค่า Signal Strength ของโหนดถัดไปต่ำกว่าค่าวิกฤตที่กำหนดไว้

3. กรณีที่มี Data Packet ส่งมาให้ตัวมันแต่เมื่อค้นหาใน Route Table แล้วไม่สามารถหาเส้นทางเพื่อส่งต่อไปยังโหนดถัดไปได้

เมื่อตรวจจับความเสียหายได้ โหนดที่ตรวจจับได้จะทำการยกเลิกเส้นทางที่เสียหายใน Route table ของตนเองทั้งหมด หลังจากนั้นจะสร้าง Route Error Packet (RERR) ส่งให้กับโหนดที่อยู่ติดกับตนในด้าน Upstream ที่เกี่ยวข้องกับความเสียหายนั้น เพื่อให้โหนดเหล่านั้นทำการยกเลิกเส้นทางที่เสียหายใน Route Table ของตนต่อไป หลังจากนั้นบรรดาโหนดเหล่านี้ก็จะสร้าง Route Error Packet (RERR) เพื่อส่งให้กับโหนดที่อยู่ติดกับตนในด้าน Upstream ที่เกี่ยวข้องกับความเสียหายนั้นต่อไปเป็นทอด ๆ จนกระทั่งถึงโหนดต้นทาง เพื่อยกเลิกเส้นทางที่เสียหายต่อไป

การแก้ไขเพื่อหาเส้นทางส่งข้อมูลต่อไปนั้นมีสามวิธีการด้วยกันคือ

1. Backup Path
2. Local Repair
3. Route Discovery ที่โหนดระหว่างทาง

การแก้ไขเส้นทางด้วยวิธี Backup Path นั้นเมื่อเส้นทางหลักใช้งานไม่ได้โหนดนั้นก็จะเปลี่ยนไปใช้เส้นทางสำรองแทนโดยใช้ Address ของโหนดปลายทางเป็นดรรชนีค้นหาเส้นทางสำรองที่มีอยู่ใน Route Table แต่หากพบว่าไม่สามารถหาเส้นทางสำรองมาทดแทนเส้นทางหลักเพื่อส่งข้อมูลได้ โหนดนั้นจะใช้ทั้งวิธี Local Repair และ Route Discovery โดยในขณะที่ทำ Local Repair เพื่อนำ Data Packet จาก Buffer ของมันส่งไปยังโหนดถัดไปนั้น โหนดนั้นก็จะส่ง RREQ ไปยังโหนดปลายทางเพื่อหาเส้นทางใหม่จากโหนดนั้นไปยังปลายทางสำหรับส่ง Data Packet ถัด ๆ มาที่กำลังจะผ่านเข้ามายังโหนดนั้นด้วย ซึ่ง Data Packet เหล่านั้นได้ถูกส่งออกจากโหนดต้นทางมาแล้วก่อนที่จะทราบว่าเส้นทางที่ใช้ส่งข้อมูลนั้นเสียหาย เพราะการทำ Local Repair นั้นจะเป็นการหาเส้นทางมาใช้แบบเร่งด่วน โดยจะร้องขอเส้นทางจากโหนดที่อยู่ติดกับโหนดที่ต้องการแก้ไขเส้นทางเท่านั้น ส่วนการทำ Route Discovery ที่โหนดระหว่างทางนั้นจะทำงานเหมือนกับกระบวนการหาเส้นทางทุกประการยกเว้นจะเริ่มทำที่โหนดระหว่างทางเลยซึ่งต้องใช้เวลา และเสีย Overhead มากกว่าการทำ Local Repair แต่จะได้เส้นทางที่มีความคงทนต่อความเสียหายและ

ทันสมัยกว่า เพราะได้รับเส้นทางจากโหนดปลายทางโดยตรงจึงเหมาะสมสำหรับการนำมาใช้งานแทนที่เส้นทางที่ได้จากการทำ Local Repair ในเวลาต่อไป

ในการแก้ไขด้วยวิธีการ Local Repair นั้นโหนดที่ต้องการแก้ไขเส้นทางที่เสียหายจะแพร่ Local Repair Request (LREQ) ไปให้โหนดที่อยู่ติดกันกับมัน และหยุดรอรับ Local Repair Reply (LREP) เป็นระยะเวลาหนึ่ง เมื่อโหนดที่อยู่ติดกันกับมันได้รับ LREQ จะไม่มีการส่ง LREQ ต่อไปเป็นทอด ๆ ให้ถึงปลายทางเหมือนกับการทำ Route Discovery โดยมันจะดำเนินตามขั้นตอนดังนี้

1. หากพบว่าตัวมันเองเป็นปลายทางสำหรับ LREQ มันจะส่ง Address ของตนเองพร้อมกับแนบ SIL เป็นค่า outLb ไปกับ LREP เพื่อส่งให้กับโหนดที่ส่ง LREQ นี้มา หากไม่ใช่จะทำตามขั้นตอนต่อไป

2. ค้นหาใน Route Table ของตนเองว่ามีเส้นทางที่ไปยังปลายทางตามที่ระบุไว้ใน LREQ หรือไม่ หากมี มันจะเลือกเส้นทางที่ดีที่สุดซึ่งจะต้องไม่มี Address ของโหนดที่ส่ง LREQ นี้อยู่ในเส้นทางนั้นและมีจำนวน Hop ที่ไม่เกินครึ่งหนึ่งของจำนวน Hop ที่มากที่สุดของเส้นทางที่ยอมให้มีได้ในเครือข่าย เพื่อป้องกันไม่ให้เกิด Route Loop และมีจำนวน Hop ในการส่งข้อมูลมากเกินไป หลังจากนั้นจะทำตามขั้นตอนต่อไป หากค้นหาแล้วไม่เจอเส้นทางใดเลย ให้ทิ้ง LREQ นั้นไป

3. นำ Post Route และ Label ซึ่งเป็นดรรชนีชี้ Entry ของเส้นทางนี้ ส่งเป็น LREP กลับไปให้กับโหนดที่ส่ง LREQ นี้

เมื่อโหนดผู้ส่งได้รับ LREP กลับมาจากโหนดต่าง ๆ ที่อยู่ติดกันกับมัน มันจะตรวจสอบว่าเคยได้รับ LREP กลับมาแล้วหรือยัง หากยัง หรือเคยได้รับมาแล้ว แต่เส้นทางที่มีอยู่นั้นมีจำนวน Hop มากกว่า ให้เก็บเส้นทางจาก LREP นี้ไว้ใน Route Table หากไม่เป็นดังที่กล่าวมาให้ทิ้ง LREP นี้ไป เส้นทางจากการทำ Local Repair ซึ่งเก็บไว้ใน Route Table จะมีการกำหนดค่าระดับความสำคัญ (Rank) ให้มีค่าน้อยกว่าเส้นทางที่ได้จากการทำ Route Discovery ที่โหนดที่ต้องการแก้ไขเส้นทางได้ทำควบคู่ไปกับ Local Repair เนื่องจากเส้นทางที่ได้จากการทำ Local Repair นั้นเป็นเส้นทางที่มีความคงทนและทันสมัยน้อยกว่า และค่าระดับความสำคัญของเส้นทางที่ได้จากการทำ Local Repair นี้จะใช้จำนวน Hop มาจัดระดับความสำคัญของเส้นทางโดยเส้นทางที่มีจำนวน Hop มากจะมีระดับความสำคัญต่ำกว่าเส้นทางที่มีจำนวน Hop น้อย เพราะไม่ได้ใช้ค่า DA และ ETV เลือกเส้นทางเหมือนกับการทำ Route Discovery

การใช้จำนวน Hop แต่เพียงอย่างเดียวมากำหนดค่าระดับความสำคัญเช่นนี้ จะขัดแย้งกับหลักการของโปรโตคอล PMP ซึ่งต้องนำค่า DA และ ETV มากำหนดเส้นทางที่ดีที่สุด แต่เส้นทางในกระบวนการ Local Repair ไม่สามารถนำข้อมูลดังกล่าวมากำหนดได้ เพราะเส้นทางจาก Route Table ของโหนดที่ส่งเส้นทางนี้มาให้นั้นเป็นเส้นทางที่มีปลายทางเดียวกันแต่ไม่ได้มาจาก

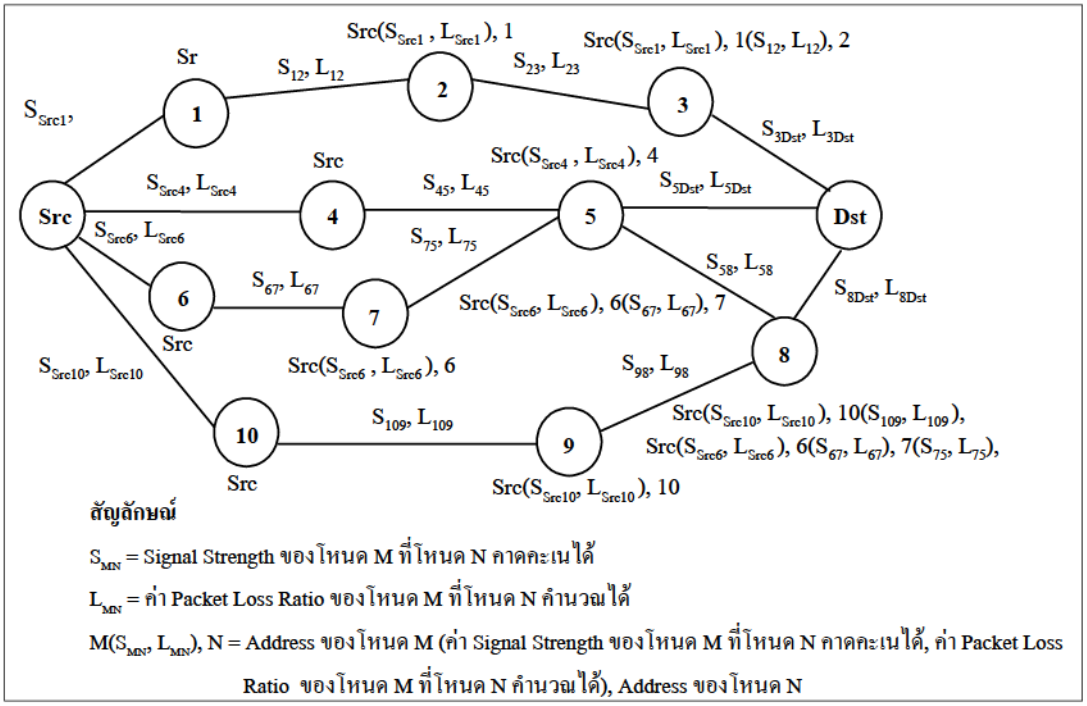
กระบวนการหาเส้นทางจากโหนดต้นทางเดียวกัน ดังที่ได้กล่าวไว้แล้วว่า ค่า DA และ ETV นั้นต้องนำค่า Signal Strength และค่า Packet Loss Ratio ของโหนดทุกโหนดตั้งแต่โหนดต้นทางถึงโหนดปลายทางของเส้นทางนั้น ๆ มาพิจารณา การนำเส้นทางต่าง ๆ ของโหนดอื่นซึ่งโหนดตนเองไม่ได้มีส่วนร่วมในกระบวนการหาเส้นทางตั้งแต่ต้นนั้น ย่อมไม่สามารถนำหลักการทั้งหมดของโปรโตคอล PMP มากำหนดเส้นทางที่ดีที่สุดได้ แต่หากนำหลักการของ PMP มาพิจารณาโดยตัดปริมาณต่าง ๆ ที่ไม่สามารถทราบค่าได้ทิ้งไป ปริมาณที่เหลือซึ่งมีอิทธิพลต่อการกำหนดว่าเส้นทางใดดีที่สุดก็จะเหลือเพียงจำนวน Hop ของเส้นทางเท่านั้น ด้วยเหตุผลดังที่ได้กล่าวมาในกระบวนการ Local Repair ของเรานั้นการจัดระดับความสำคัญของเส้นทางที่ได้จากกระบวนการนี้จึงไม่เป็นไปตามหลักการของโปรโตคอล PMP อย่างเคร่งครัด อย่างไรก็ตามเส้นทางที่ได้จากการทำ Local Repair นั้นจะถูกนำมาใช้งานในระยะเวลาสั้น ๆ เท่านั้น เมื่อการทำ Route Discovery สำเร็จเส้นทางเหล่านี้ก็จะถูกแทนที่ด้วยเส้นทางใหม่ที่เป็นไปตามหลักการของโปรโตคอล PMP เช่นเดิม จะเห็นได้ว่าการแก้ไขเส้นทางเพื่อนำมาใช้ในระยะสั้น ๆ แบบเร่งด่วนของเรานั้นจะไม่ยึดหลักการอย่างเคร่งครัด แต่การแก้ไขเส้นทางเพื่อนำมาใช้งานในระยะยาวเราจะยึดหลักการอย่างเคร่งครัดทั้งนี้เพื่อให้ระบบที่มีการเปลี่ยนแปลงได้ตลอดเวลาอย่าง MANET นั้นทำงานได้อย่างมีประสิทธิภาพนั่นเอง

3.7 ตัวอย่างกระบวนการทำงาน (Example)

เพื่อแสดงรายละเอียดการทำงานของโปรโตคอลที่เรานำเสนอให้ชัดเจนยิ่งขึ้น เราจะยกตัวอย่างเครือข่ายเครือข่ายหนึ่งซึ่งมีรูปร่างของเครือข่ายดังภาพที่ 3.13 เพื่ออธิบายกระบวนการทำงานของโปรโตคอล PMLS

เมื่อ Src ต้องการจะส่งข้อมูลไปยัง Dst แต่ไม่มีเส้นทางใด ๆ เลยที่สามารถไปยัง Dst ได้ จึงเริ่มกระบวนการหาเส้นทาง (Route Discovery) โดยการส่ง RREQ ไปยังโหนดที่อยู่ติดกับตน ในที่นี้ คือ โหนด 1, 4, 6 และ 10 เมื่อโหนดทั้งสี่ ได้รับ RREQ มันก็จะแนบค่า Signal Strength และค่า Packet Loss Ratio ของ Link ก่อนหน้ามันพร้อมกับ Address ของตนไปกับ RREQ นั้นส่งต่อไปเป็นทอด ๆ จนถึงปลายทาง ในที่นี้คือ Dst เช่น เมื่อโหนด 1 ได้รับ RREQ จาก Src จะแนบค่า Signal Strength และค่า Packet Loss Ratio ที่ส่งออกมาจาก Src ถึงตน ในที่นี้คือ (S_{src1}, L_{src1}) พร้อมกับ Address ของตนไปกับ RREQ ส่งต่อไปให้กับโหนดที่อยู่ติดกับตน เมื่อโหนด 2 ได้รับ RREQ จากโหนด 1 ก็จะแนบค่า Signal Strength และค่า Packet Loss Ratio ที่ส่งออกมาจากโหนด 1 มาถึงตน ในที่นี้คือ (S_{12}, L_{12}) เพื่อส่ง RREQ ให้กับโหนดถัดไป เมื่อโหนด 3 ได้รับ RREQ นี้ก็จะทำ

เช่นเดียวกัน โดยแนบค่า Signal Strength และ Packet Loss Ratio ของโหนด 2 ที่ส่งมาถึงตนก่อนจะส่ง RREQ นี้ไปยังโหนดถัดไป ซึ่งโหนดถัดไปในที่นี้ คือ Dst ซึ่งเป็นปลายทางสำหรับ RREQ นี้ Dst จะนำค่า Signal Strength และ Packet Loss Ratio ของโหนด 3 ที่ส่งมาถึงตนมาใช้ในการหาค่า DA และ ETV เพื่อใช้ในการคัดสรรเส้นทางต่อไป ในช่วงเวลาเดียวกันนี้โหนดระหว่างทางอื่น ๆ ก็จะดำเนินการตามกระบวนการเช่นเดียวกับเส้นทางที่ 1) Src>>1>>2>>3>>Dst



ภาพที่ 3.13 แสดง Source Route Information, Signal Strength และ Packet Loss Ratio ที่ได้รับในแต่ละ Hop

เราจะมาพิจารณาเส้นทางที่ผ่านมาถึงโหนด 8 เรียงตามลำดับก่อนและหลังซึ่งมีจำนวนทั้งสิ้นสองเส้นทางด้วยกันคือ 3) Src>>6>>7>>5>>8 และ 5) Src>>4>>5>>8 ตามลำดับ เมื่อโหนด 8 ได้รับข้อมูลของเส้นทาง 3) Src>>6>>7>>5>>8 มันจะเก็บข้อมูลของเส้นทางนี้ไว้ใน Cache ของตนก่อนจะส่งเส้นทางนี้ให้กับโหนดถัดไปต่อไป ในเวลาต่อมาเมื่อมีเส้นทางที่ 5) Src>>4>>5>>8 มาถึงตน มันจะพบว่าเส้นทางที่รับมาใหม่นั้นมีส่วน Overlap กับเส้นทาง 3) Src>>6>>7>>5>>8 มันจึงนำข้อมูลของเส้นทาง 3) Src>>6>>7>>5>>8 มาคำนวณหาค่า ETV หากมีค่าสูงกว่าค่า ETV ของเส้นทาง 5) Src>>4>>5>>8 มันจะปฏิเสธเส้นทางนี้ไปหากค่า ETV ของเส้นทางนี้สูงกว่าเส้นทาง 3) Src>>6>>7>>5>>8 มันจะตอบรับและส่งเส้นทางนี้ให้กับโหนดถัดไปต่อไป แต่ในที่นี้

เส้นทาง 3) Src>>6>>7>>5>>8 มีค่า ETV สูงกว่าเส้นทาง 5) Src>>4>>5>>8 โหนด 8 จึงปฏิเสธเส้นทางที่ 5) Src>>4>>5>>8 ไป

โหนด Dst นั้นจะต้องมีการจำกัดจำนวนเส้นทางที่จะสามารถรับไว้ได้สูงสุดด้วย และหากเส้นทางใด มีค่า DA ต่ำกว่าค่าที่กำหนดก็ให้ทิ้งเส้นทางนั้นไป ในที่นี้หลังจาก Dst รอรับ RREQ จากเส้นทางต่าง ๆ เป็นระยะเวลาหนึ่งแล้ว จะมีเส้นทางมาถึงตนและค่า DA ของเส้นทางเหล่านั้นไม่ต่ำกว่าค่าที่กำหนดไว้ จำนวนทั้งสิ้น 4 เส้นทางคือ

1) Src>>1>>2>>3>>Dst
2) Src>>4>>5>>Dst
3) Src>>6>>7>>5>>8>>Dst
4) Src>>10>>9>>8>>Dst

จะสังเกตว่า เส้นทางที่ 3) กับ 4) จะมี Overlapped Path ในกรณีเช่นนี้ Dst จะนำเส้นทาง Overlapped Path มาเปรียบเทียบค่า ETV ในที่นี้ ค่า ETV ของเส้นทางที่ 3) นั้นมีค่า ETV มากกว่าเส้นทางที่ 4) เส้นทางที่ 4) จึงถูกทิ้งไป ดังนั้นเส้นทางที่ Dst ยอมรับทั้งหมดคือ เส้นทางที่ 1), 2) และ3) หลังจากนั้นก็เอาเส้นทางทั้งสามมาจัด Rank ตามค่า ETV หลังจากจัด Rank จะได้เส้นทางตามระดับความสำคัญจากมากไปน้อยได้ดังนี้

1) Src>>1>>2>>3>>Dst
2) Src>>4>>5>>Dst
3) Src>>6>>7>>5>>8>>Dst

หลังจากนั้น Dst จะแนบ Self-Identification Label (SIL) ไปเป็น outLb เข้ากับเส้นทางแต่ละเส้นทางเพื่อส่ง RREP กลับไปยัง โหนดถัดไปตามเส้นทางนั้น ๆ เมื่อ โหนด ระหว่างทางได้รับ RREP ของเส้นทางของตน มันจะเพิ่ม Entry ซึ่งมี outLb จาก RREP ที่ได้รับลงใน Route Table ของตนพร้อมกับนำ Label ซึ่งเป็นดรรชนีที่ชี้ที่ Entry นี้ไปแทนที่ outLb เดิม ใน RREP ก่อนจะส่ง RREP นี้ต่อไปยังโหนดถัดไปจนกระทั่งถึงโหนดต้นทาง เช่นเส้นทาง 1) Src>>1>>2>>3>>Dst Dst จะแนบ SIL ไปกับ RREP เพื่อส่งไปยัง โหนด 3 เมื่อ โหนด 3 ได้รับ RREP มันจะเพิ่ม Entry ของเส้นทางนี้ ลงไปใน Route Table ของตนโดยมีฟิลด์ต่าง ๆ ตามตารางที่ 3.1

หลังจากนั้นมันจะนำค่า 12 ซึ่งเป็น Index ที่ชี้ Entry นี้ แนบเป็น outLb ของ RREP เพื่อส่งให้กับ โหนด 2 เมื่อ โหนด 2 ได้รับ RREP นี้ มันจะเพิ่ม Entry ของเส้นทางนี้ ลงไปใน Route table ของตน โดยมี outLb เป็น 12 พร้อมกับนำค่า 23 ซึ่งเป็น Index ของ Entry นี้ แนบเป็น outLb ของ RREP เพื่อส่งให้กับ โหนด 1 เมื่อ โหนด 1 ได้รับ RREP นี้ มันก็จะเพิ่ม Entry ของเส้นทางนี้ลงใน Route Table ของตนโดยมี outLb เป็นค่า 23 พร้อมกับนำค่า 10 ซึ่งเป็น Index ของ Entry นี้แนบเป็น

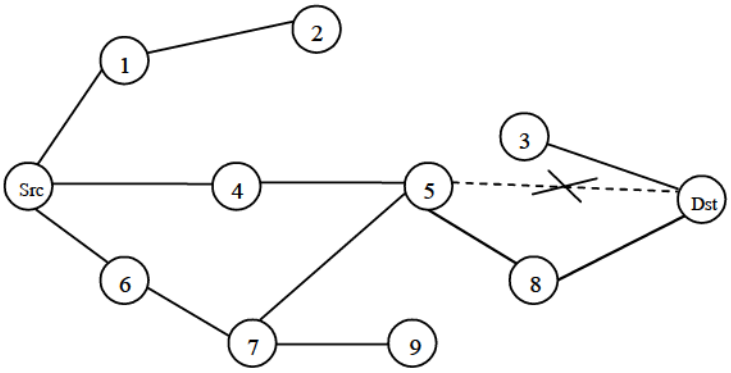
outLb ของ RREP เพื่อส่งให้กับ Src (ต้นทาง) เมื่อ Src ได้รับ RREP นี้ มันจะเพิ่ม Entry ของเส้นทางนี้ ลงใน Route Table ของตน เป็นอันสิ้นสุด กระบวนการ Route Discovery ของเส้นทางนี้ ส่วนเส้นทางอื่น ๆ ก็จะเป็นเช่นเดียวกันโดย Route Table ของโหนดต่าง ๆ เป็นไปตามตารางที่ 1 มาถึงตอนนี้ Src ก็จะมีเส้นทางในการส่งข้อมูลไปยัง Dst จำนวน 3 เส้นทาง โดยมีเส้นทางที่ 1) เป็นเส้นทางหลัก มีเส้นทาง ที่ 2) และ 3) เป็นเส้นทางสำรองตามลำดับ

ตารางที่ 3.1 แสดง Route Table บางส่วนของโหนด ต่าง ๆ

Path No.	Time	Node	Index	outLb	nextHop	Dest	Rank	Upstream	Post Route
1	t0	3	12	SIL	Dst	Dst	0	2	Dst
	t1	2	23	12	3	Dst	0	1	3,Dst
	t2	1	10	23	2	Dst	0	Src	2,3,Dst
	t3	Src	55	10	1	Dst	0	Null	1,2,3,Dst
2	t0	5	14	SIL	Dst	Dst	1	4	Dst
	t1	4	34	14	5	Dst	1	Src	5,Dst
	t2	Src	11	34	4	Dst	1	Null	4,5,Dst
3	t0	8	30	SIL	Dst	Dst	2	5	Dst
	t1	5	28	30	8	Dst	2	7	8,Dst
	t2	7	31	28	5	Dst	2	6	5,8,Dst
	t3	6	22	31	7	Dst	2	Src	7,5,8,Dst
	t4	Src	44	22	6	Dst	2	Null	6,7,5,8,Dst

ในการส่งข้อมูล Src จะค้นหาเส้นทางเพื่อส่งข้อมูลโดยใช้ Address ของปลายทางเป็นดรรชนีสำหรับการค้นหาใน Route Table ของตน และจะเลือกเส้นทางที่ดีที่สุดเป็นเส้นทางหลักในการส่งข้อมูล (Rank = 0) ในที่นี้ คือเส้นทาง ที่ 1) เมื่อ Src หาเส้นทางหลักในการส่งข้อมูลพบ มันจะนำส่วนของ Label ซึ่งได้แก่ outLb และ nextHop มาเป็นส่วนหัวของข้อมูล เพื่อแพร่ไปให้ โหนดถัดไปตามเส้นทางนี้ ในที่นี้ คือ 10 และ 1 ตามลำดับ เมื่อโหนดถัดไป (โหนด 1) ได้รับ Data Packet นี้ มันจะตรวจสอบว่า nextHop คือมันหรือไม่หากไม่ใช่ก็จะไม่สนใจ Packet นี้ แต่ในที่นี้ nextHop คือ โหนด 1 มันจึงตอบรับ Packet นี้ โดยส่ง ACK กลับไปให้โหนดผู้ส่ง Packet นี้ (Src) หลังจากที่

มันรับ Packet นี้มามันจะตรวจสอบต่อไปว่ามันเคยรับ Packet นี้มาก่อนหรือไม่หากเคยรับมาแล้วก็จะทิ้ง Packet นี้ไป หากยังไม่เคยรับมาก่อนมันจะนำ outLb ซึ่งเป็นส่วนหัวของข้อมูลนี้มาใช้เป็นดรรชนีในการค้นหาเส้นทางใน Route Table ของตน นำค่า outLb และ nextHop จาก Entry ตามดรรชนีนี้มาเป็นส่วนหัวของข้อมูลเพื่อส่งให้กับโหนดถัดไปต่อไป ในที่นี้ outLb และ nextHop คือ 23 และ 2 ตามลำดับ เมื่อโหนดถัดไป (โหนด 2) ได้รับ Data Packet จากโหนด 1 ก็จะทำเช่นเดียวกัน เมื่อ Data Packet มาถึงโหนด 3 มันจะใช้ outLb จาก Data Packet มาค้นหาใน Route Table ซึ่งในที่นี้คือ 12 เป็นดรรชนีในการค้นหา ซึ่งจะได้ outLb และ nextHop คือ SIL และ Dst ตามลำดับ ซึ่งแสดงว่าโหนดถัดไปเป็นปลายทางสำหรับข้อมูลนี้ เมื่อโหนดถัดไปได้รับ Data Packet นี้ มันก็จะนำข้อมูลนี้ส่งต่อให้ Layer ถัดไป เป็นอันสิ้นสุดกระบวนการนำส่งข้อมูล (Data Forwarding)

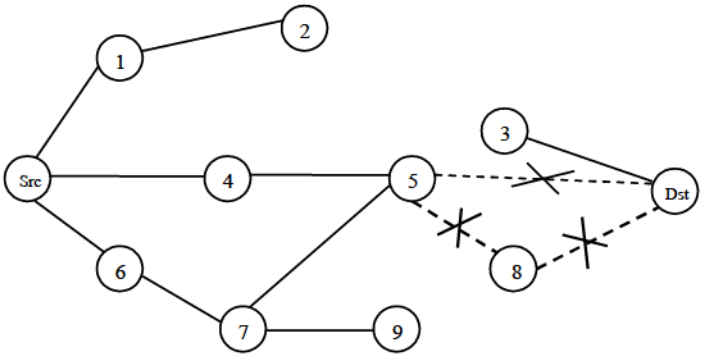


ภาพที่ 3.14 แสดงรูปร่างเครือข่ายก่อนโหนด 8 เสียหาย

เมื่อเวลาผ่านไปโหนด 3 ได้เคลื่อนที่ออกจากรัศมีส่งสัญญาณของโหนด 2 ดังแสดงภาพที่ 3.14 ซึ่งโหนด 2 จะตรวจพบความเสียหายดังกล่าว หลังจากได้รับค่าความแรงของสัญญาณของโหนด 3 น้อยกว่าค่าวิกฤตโหนด 2 จะทำการยกเลิก (Invalid) เส้นทางทุกเส้นทางที่ผ่านโหนด 3 พร้อมกับส่ง RERR ให้กับโหนดที่เกี่ยวข้องทั้งหมดรับรู้ถึงความเสียหายดังกล่าวและให้บรรดาโหนดต่าง ๆ เหล่านั้นทำการยกเลิกเส้นทางที่เกี่ยวข้องทั้งหมดต่อไปเป็นทอด ๆ จนกระทั่งถึงโหนดต้นทาง ซึ่งในที่นี้เส้นทางหลัก Src>>1>>2>>3>>Dst ก็เป็นเส้นทางหนึ่งที่เสียหาย ดังนั้นโหนด 1 และ Src ก็จะยกเลิกเส้นทางหลักที่ไปยัง Dst หาก Src ต้องการส่งข้อมูลไปยัง Dst ก็จะใช้เส้นทางสำรอง คือเส้นทางที่ 2) Src>>4>>5>>Dst เป็นเส้นทางหลักแทน

ในเวลาต่อมาโหนด 5 ได้รับข้อมูลจากโหนด 4 แต่ได้พยายามส่งข้อมูลไปยังโหนดถัดไป (โหนด Dst) แต่ไม่ได้รับ ACK กลับมาแม้ว่าจะพยายามส่งข้อมูลนั้นซ้ำไปจนกระทั่งถึงขีดจำกัด

ตามที่กำหนดไว้แล้วก็ตามซึ่งในที่นี้จะกำหนดไว้สามครั้ง แต่ก็ยังไม่ได้รับ ACK กลับมา มันจึงทำการยกเลิกเส้นทางนี้พร้อมกับส่ง RERR ให้กับโหนดที่เกี่ยวข้องทั้งหมดเพื่อทำการยกเลิกเส้นทางนี้ หลังจากนั้นมันจะนำ Address ของโหนดปลายทาง ในที่นี้คือ Dst ไปหาเส้นทางอื่นใน Route Table ซึ่งพบว่ามีเส้นทางที่ไปยัง Dst ได้อีกเส้นทางหนึ่งคือ เส้นทางที่ 3) Src>>6>>7>>5>>8>>Dst มันจึงนำ outLb และ nextHop (30 และ 8 ตามลำดับ) ของเส้นทางนี้มาเป็นส่วนหัวของข้อมูลเพื่อส่งไปยังโหนดถัดไป (โหนด 8)



ภาพที่ 3.15 แสดงรูปร่างเครือข่ายหลังจากโหนด 8 เสียหาย

ในเวลาต่อมาโหนด 8 เสียหายทำให้เส้นทางที่ 3) Src>>6>>7>>5>>8>>Dst นั้นใช้งานไม่ได้ ดังแสดงภาพที่ 3.15 เมื่อโหนด 5 ได้รับข้อมูลจากโหนด 7 และต้องการส่งข้อมูลไปยังปลายทาง Dst แต่ในขณะนี้เส้นทางต่าง ๆ ทั้งหมดของโหนด 5 นั้นเสียหายจนหมดแล้ว มันจึงเริ่มกระบวนการบำรุงรักษาเส้นทาง (Route Maintenance) โดยเก็บข้อมูลดังกล่าวไว้ใน Buffer และแพร่ LREQ พร้อมกับ RREQ ให้กับโหนดที่อยู่ติดกันกับมัน ซึ่งโหนดที่ได้รับคือโหนด 3 ซึ่งเคลื่อนที่ออกจากรัศมีสัญญาณของโหนด 2 มาอยู่ระหว่างโหนด 5 กับ Dst

เมื่อโหนด 3 ได้รับ LREQ จาก โหนด 5 มันจะใช้ Address ปลายทางจาก LREQ เป็นดรรชนีค้นหาเส้นทางที่ไปยังปลายทางนั้นใน Route Table เมื่อโหนด 3 พบเส้นทางที่ไปยังปลายทางมันจะสร้าง LREP ซึ่งแนบ Address ของมันเอง, Post Route และ Index ที่ชี้ที่ Entry ของเส้นทางนี้ ส่งกลับไปยังโหนด 5 ซึ่ง Post Route และ Index ของ Entry นี้คือ Dst และ 12 ตามลำดับ เมื่อโหนด 5 ได้รับ LREP มันจะนำเส้นทางดังกล่าวมาเพิ่มลงใน Route Table ดังตารางที่ 3.2 และตารางที่ 3.3

ตารางที่ 3.2 แสดง Route Table ของโหนด 3 หลังจากการทำ Local Repair เสร็จสิ้น

Index	outLb	nextHop	Dest	Rank	Upstream	Post Route
12	SIL	Dst	Dst	0	2	Dst

ตารางที่ 3.3 แสดง Route Table ของโหนด 5 หลังจากการทำ Local Repair เสร็จสิ้น

Index	outLb	nextHop	Dest	Rank	Upstream	Post Route
15	12	3	Dst	4*	Null	3,Dst

ซึ่งจะสังเกตได้ว่าค่า Rank ของเส้นทางดังกล่าวมีค่าเป็น 4 เนื่องจากในที่นี้ได้กำหนดจำนวนเส้นทางมากที่สุดที่จะนำมาใช้งานได้จำนวน 3 เส้นทาง ดังนั้น ค่า 0, 1 และ 2 จะใช้สำหรับกำหนดระดับความสำคัญ (Rank) ของเส้นทาง ที่ได้จาก Route Discovery ส่วนค่าที่มากกว่า 2 จะใช้สำหรับกำหนด Rank ของเส้นทางที่ได้จาก Local Repair และเส้นทางนี้ซึ่งได้จาก Local Repair นั้นมีจำนวน 2 Hop ดังนั้น ค่า Rank จึงถูกกำหนดให้มีค่าเป็น 4

หลังจากนั้นโหนด 5 ก็จะนำข้อมูลที่เก็บไว้ใน Buffer มาหาเส้นทางโดยใช้ Address ปลายทางเป็นดรรชนีในการค้นหาซึ่งจะได้เส้นทางที่ได้จากการทำ Local Repair เพื่อนำข้อมูลนั้นส่งไปยังปลายทางได้ ในขณะเดียวกันหลังจากที่โหนด 3 ได้รับ RREQ มันจะแนบค่าความแรงของสัญญาณและค่าอัตราการสูญหายของข้อมูลของโหนด 5 ไปกับ RREQ เช่นเดียวกับกระบวนการ Route Discovery ตามปกติ เมื่อ Dst ได้รับ RREQ ของเส้นทางดังกล่าวมันจะรออยู่ชั่วระยะเวลาหนึ่งก่อนคัดเลือกเส้นทางและส่ง RREP กลับไปให้กับโหนด 5 โดยผ่านทางโหนด 3 ทั้งโหนด 3 และ โหนด 5 ก็จะเพิ่ม Entry ของเส้นทางลงไปใน Route Table ของตน ซึ่งในที่นี้จะมีเพียงเส้นทางเดียวคือ 5>>3>>Dst ดังแสดงตารางที่ 3.4 และตารางที่ 3.5 ซึ่งโหนด 5 จะใช้เส้นทางนี้เป็นเส้นทางหลักแทนเส้นทางที่ได้จากการทำ Local Repair

ตารางที่ 3.4 แสดง Route Table ของโหนด 3 หลังจากการทำ Route Maintenance เสร็จสิ้น

Index	outLb	nextHop	Dest	Rank	Upstream	Post Route
12	SIL	Dst	Dst	0	2	Dst
13	SIL	Dst	Dst	0	5	Dst

ตารางที่ 3.5 แสดง Route Table ของโหนด 5 หลังจากการทำ Route Maintenance เสร็จสิ้น

Index	outLb	nextHop	Dest	Rank	Upstream	Post Route
15	12	3	Dst	4	Null	3,Dst
16	13	3	Dst	0	Null	3,Dst

บทที่ 4

การจำลองและการเปรียบเทียบประสิทธิภาพของการทำงาน

4.1 โปรแกรมจำลองการทำงาน (Simulation)

เนื่องจากในปัจจุบันนั้นการจำลองการทำงานของเครือข่ายไร้สายเคลื่อนที่ได้มีความจำเป็นต่อการวิจัยและพัฒนาการทำงานของเครือข่ายดังกล่าวเป็นอย่างมาก จึงมีผู้คิดค้นพัฒนาเครื่องมือช่วยในการพัฒนาที่สร้างความสะดวกรวดเร็วมีความยืดหยุ่น และประสิทธิภาพเพิ่มมากขึ้น เครื่องมือที่เราเลือกใช้สำหรับการพัฒนาโปรแกรมจำลองการทำงานของโปรโตคอลที่เรานำเสนอนี้ คือ JIST-SWANS คิดค้นและพัฒนาโดย Rimon Barr เป็นเครื่องมือที่ทำงานบนภาษา Java

หลักการการทำงานที่สำคัญของ (Barr, 2004a: 1-34) JIST คือเวลาในการทำงานเราสามารถกำหนดได้เองโดยเรียกใช้ JIST API ตามที่ได้จัดเตรียมไว้ให้ (Simulation Time) ซึ่งเวลาในการจำลองดังกล่าวจะเดินไปตามโปรแกรมที่เรากำหนดไว้ ส่วน (Barr, 2004b: 1-15) SWANS (Scalable Wireless Ad Hoc Network Simulator) จะสร้างอยู่บน JIST Platform โดยจัดเตรียมเครื่องมือรวมทั้งตัวอย่าง Source Code ของการจำลองการทำงานของเครือข่าย Ad Hoc ไว้อย่างครบครันโดยแบ่งเป็นชั้น ๆ ตามการทำงานและสามารถนำมาปรับแต่งให้เหมาะสมกับการทำงานของเราได้เป็นอย่างดี

ในงานวิจัยของโปรโตคอล PMP ได้ทำการทดลองเปรียบเทียบประสิทธิภาพการทำงานระหว่างโปรโตคอล PMP และโปรโตคอล DSR แล้วปรากฏว่าโปรโตคอล PMP ได้ผลดีกว่า ส่วนโปรโตคอล AODV นั้นจะใช้จำนวน Hop ที่น้อยที่สุดในการเลือกเส้นทางที่ดีที่สุด ซึ่งงานวิจัยของโปรโตคอล PMP ได้แสดงให้เห็นแล้วว่าการหาเส้นทางที่ใช้ทั้งประสิทธิภาพในการส่งข้อมูลและความคงทนต่อความเสียหายของเส้นทางตามหลักการของตนนั้น ได้เส้นทางที่ดีกว่าการใช้จำนวน Hop ของเส้นทางแต่เพียงอย่างเดียว และเนื่องจากโปรโตคอล ORAL และ LSMR นั้นใช้หลักการหาเส้นทางจากโปรโตคอล DSR และ AODV ตามลำดับ ดังนั้นในงานวิจัยนี้เราจึงเปรียบเทียบประสิทธิภาพการทำงานเฉพาะกับโปรโตคอล PMP เท่านั้น

การกำหนดสภาวะแวดล้อมการทำงานของโปรโตคอล PMP และโปรโตคอลของเรา ต้องทำอยู่ภายใต้สภาวะแวดล้อมเดียวกัน เราจึงต้องกำหนดค่าคงที่ต่าง ๆ ของโปรแกรมดังกล่าวให้เป็น

ค่าเดียวกัน ค่าบางค่าไม่สามารถกำหนดได้โดยตรงภายในโปรแกรมได้ เช่น ค่ารัศมีการส่งสัญญาณ (Transmission Range) เราจึงเขียนโปรแกรมเพื่อกำหนดค่าดังกล่าวให้เป็นค่าโดย ประมาณตามที่เราต้องการ โดยกำหนดค่าสภาวะแวดล้อมต่าง ๆ ให้คงที่ และสร้างโหนด 2 โหนดให้มีระยะห่างตาม Transmission Range ที่เราต้องการ หลังจากนั้นให้โหนดตัวหนึ่งส่งข้อมูลไปให้โหนดอีกตัวหนึ่ง แล้วลดค่าความแรงในการส่งสัญญาณ (Transmission Strength) ของโหนดทั้งสองไปเรื่อย ๆ จนกระทั่งโหนดทั้งสองไม่สามารถรับ-ส่งข้อมูลได้ หลังจากนั้นเราก็จะนำค่า Transmission Strength และค่าคงที่อื่น ๆ ในโปรแกรมนี้มากำหนดในโปรแกรมจำลองการทำงานของเรา

4.2 สภาวะแวดล้อมของการจำลองการทำงาน

เพื่อให้การเปรียบเทียบประสิทธิภาพการทำงานของโปรโตคอล PMP และโปรโตคอลที่เรานำเสนอเป็นไปด้วยความถูกต้อง เราจึงกำหนดสภาวะแวดล้อมการทำงานของโปรโตคอลทั้งสองให้เหมือนกันดังนี้

1. กำหนดให้เครือข่ายประกอบด้วยโหนดที่สามารถเคลื่อนที่ได้แบบสุ่มภายในพื้นที่การจำลองขนาด 1000 x 1000 เมตร

2. แต่ละโหนดส่งสัญญาณวิทยุด้วยรัศมีการส่งสัญญาณ (Transmission Range) โดยประมาณ 250 เมตร ด้วยขนาด Bandwidth 2 Mbps

3. การเคลื่อนที่ของแต่ละโหนดจะ (Hyytiä, Koskinen, Lassila, Penttinen and Virtamo, 2005: 1-55) เคลื่อนที่แบบสุ่มทั้งทิศทางและความเร็ว (Random Waypoint) ด้วยความเร็ว 0 – 10 เมตร/ วินาที

4. รายละเอียดการทำงานของโปรโตคอลในชั้น MAC เป็นไปตาม (IEEE, 2003: 85-87) IEEE 802.11b Distribution Coordinate Function (DCF)

5. กำหนดให้มีโหนดต้นทางส่งข้อมูลและโหนดปลายทางรับข้อมูลจำนวน 10 คู่ มีการส่งข้อมูลด้วยอัตราการส่งข้อมูลคงที่ (CBR) ในอัตรา 10 Packets/วินาที ขนาดของข้อมูลใน Packet (Pay Load) มีจำนวน 512 Bytes

6. การจำลองจะทำงานเป็นเวลาทั้งสิ้น 300 วินาที โดยจะหยุดทุก ๆ 30 วินาที เพื่อวัดประสิทธิภาพของการทำงาน

7. ทุกโหนดมีการแพร่ Probe Packet ไปทุก ๆ 1 วินาที โดยในช่วงเวลาเริ่มต้นการทำงาน 30 วินาทีแรกจะมีการแพร่ Probe Packet ไปจำนวนหนึ่งจนกระทั่งเครือข่ายมีเสถียรภาพกล่าวคือทุกโหนดในเครือข่ายได้รับ Probe Packet อย่างทั่วถึงจึงจะเริ่มส่งข้อมูล

4.3 ตัววัดเพื่อใช้ในการเปรียบเทียบประสิทธิภาพการทำงาน

การเปรียบเทียบประสิทธิภาพการทำงานของโปรโตคอล PMP และโปรโตคอลที่เรานำเสนอจะใช้ตัววัดต่าง ๆ ดังนี้

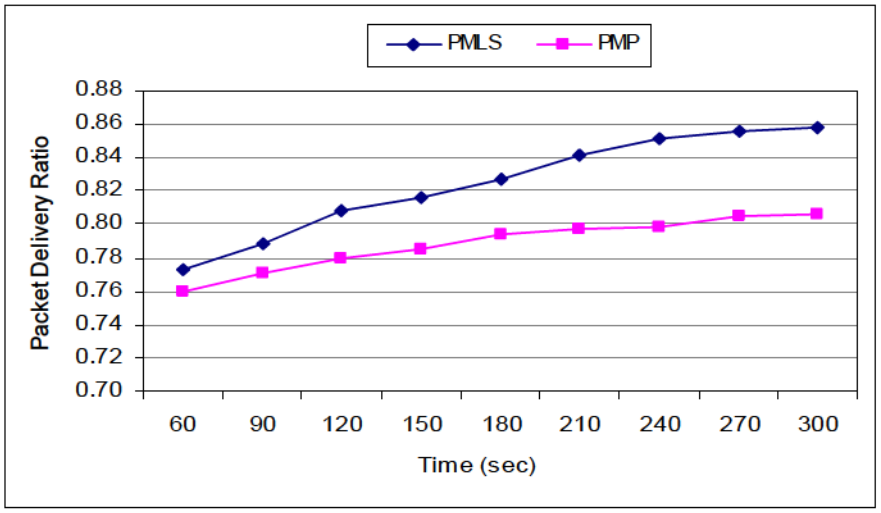
1. สัดส่วนของปริมาณข้อมูลที่ปลายทางรับได้สำเร็จต่อปริมาณข้อมูลที่ส่งจากต้นทางในช่วงเวลา หนึ่ง ๆ (Packet Delivery Ratio)

2. ปริมาณข้อมูลที่ต้องทิ้งไปในระหว่างการส่งข้อมูลในช่วงเวลาหนึ่ง ๆ (Number of Packets Dropped)

3. สัดส่วนของปริมาณข้อมูลที่ปลายทางรับได้สำเร็จต่อปริมาณข้อมูลที่ทุกโหนดในเครือข่ายส่งออกมาในช่วงเวลาหนึ่ง ๆ (Network Throughput)

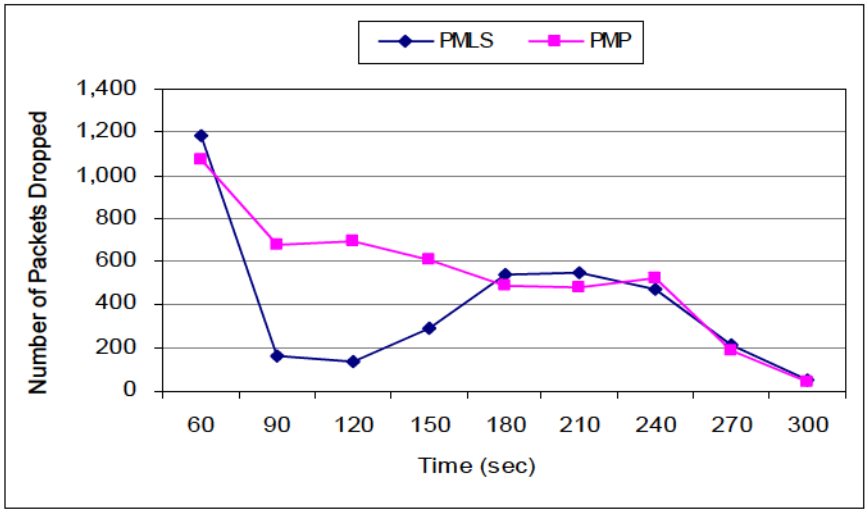
4.4 ผลการทดสอบและเปรียบเทียบประสิทธิภาพการทำงาน

หลังจากที่เราได้ประมวลผลโปรแกรมการจำลองการทำงานของ PMP และ PMLS ของงานวิจัยนี้ โดยทั้งสองโปรโตคอลนั้นในการทดสอบใช้เวลาในการจำลองครั้งละ 300 วินาที จำนวน 100 ครั้ง นำผลลัพธ์มาหาค่าเฉลี่ยเพื่อวัดค่าประสิทธิภาพ สำหรับผลการทดสอบได้แสดงไว้ดังภาพที่ 4.1, ภาพที่ 4.2 และ ภาพที่ 4.3



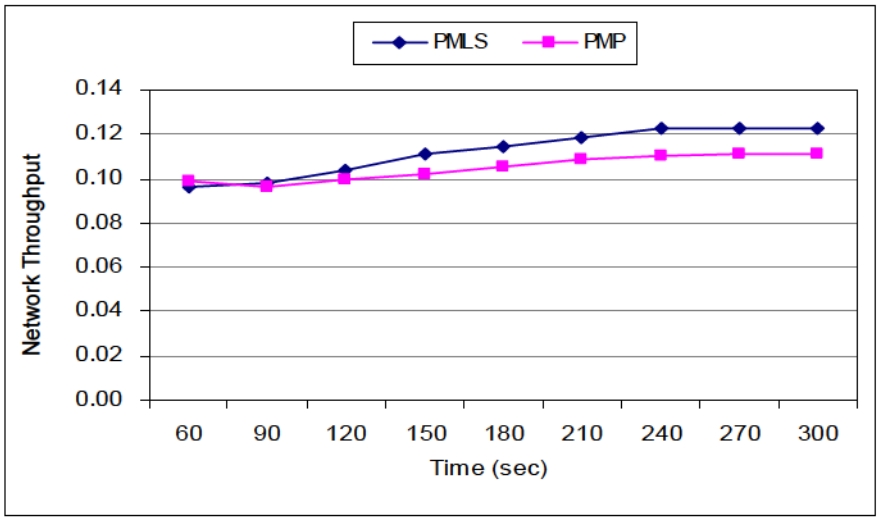
ภาพที่ 4.1 Packet Delivery Ratio ของ PMLS และ PMP

จากภาพที่ 4.1 จะเห็นได้ว่า ค่า Packet Delivery Ratio ของ PMLS จะสูงกว่า PMP โดยค่าสูงสุดของ Packet Delivery Ratio ของ PMLS จะสูงกว่า PMP ประมาณ 6% เนื่องจาก PMLS นั้นมีกระบวนการนำส่งข้อมูล และกระบวนการบำรุงรักษาเส้นทางที่ดีกว่า PMP เพราะ Data Packet มีขนาดสั้นและความรวดเร็วในการส่งผ่านข้อมูล (Data Forwarding)โดยใช้วิธีการ Label Switching อีกทั้งมีการแก้ไขเส้นทางด้วยวิธีการ Local Repair และด้วยกระบวนการ Route Discovery ที่โหนดระหว่างทางควบคู่กันไป การทำ Local Repair นั้นจะทำให้ข้อมูลถูกส่งไปยังเป้าหมายได้รวดเร็ว ซึ่งเป็นการแก้ไขเส้นทางในช่วงระยะเวลาสั้น ๆ ให้ได้เส้นทางมาใช้งานก่อนอย่างรวดเร็ว ส่วนการทำ Route Discovery ที่โหนดระหว่างทางนั้นจะได้เส้นทางที่มีความคงทนและเป็นเส้นทางที่ทันสมัยกว่าเส้นทางที่ได้จากการทำ Local Repair แต่จะต้องใช้เวลาและภาระการคำนวณมากกว่า การแก้ไขเส้นทางด้วยการใช้วิธีการทั้งสองวิธีควบคู่กันไปจะเป็นการส่งเสริมให้มี Packet Delivery Ratio สูงขึ้น นอกจากนั้น Packet Delivery Ratio ของโปรโตคอลทั้งสองเมื่อเวลาผ่านไปนานขึ้นค่าจะมีแนวโน้มสูงขึ้น



ภาพที่ 4.2 Number of Packets Dropped ของ PMLS และ PMP

จากภาพที่ 4.2 ผลการทดสอบ Number of Packets Dropped ของ PMLS นั้นต่ำกว่า PMP โดยเฉลี่ย 25% ทั้งนี้เนื่องจาก PMLS นั้นมีขนาด Data Packet ที่สั้นกว่า และมีการแก้ไขเส้นทางด้วยวิธีการ Local Repair และ Route Discovery ที่โหนดระหว่างทางหลังจากพบว่าไม่มีเส้นทางไปยังปลายทาง แทนที่จะ Drop ข้อมูลทิ้งไป

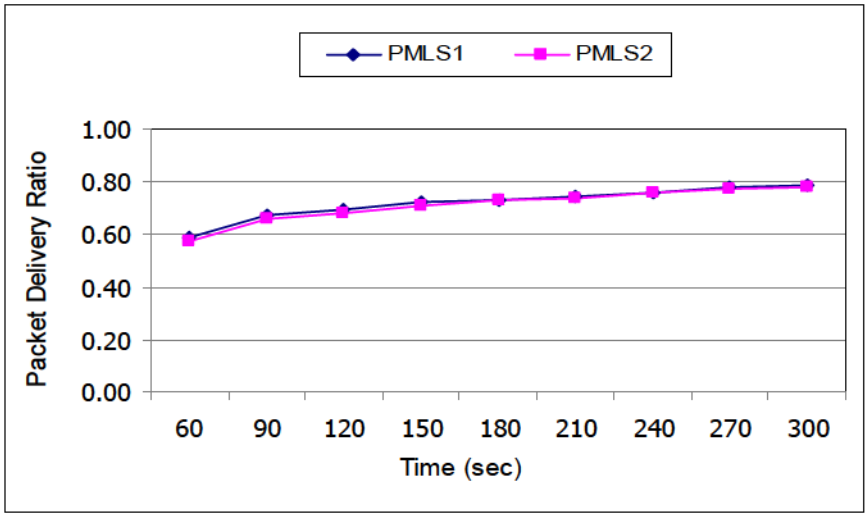


ภาพที่ 4.3 Network Throughput ของ PMLS และ PMP

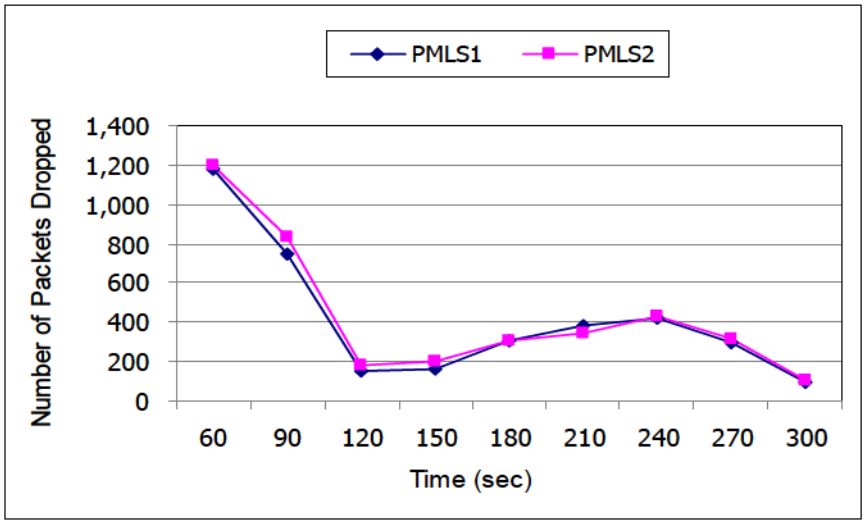
จากภาพที่ 4.3 สิ่งที่มีอิทธิพลต่อค่า Network Throughput ของ PMP และ PMLS คือความหนาแน่นของเครือข่าย จำนวน Hop ทั้งหมดที่ข้อมูลผ่านจากต้นทางจนถึงปลายทาง และประสิทธิภาพของการเชื่อมต่อสื่อสารของโหนดที่อยู่ติดกันในแต่ละคู่ (Link) จนถึงปลายทาง เนื่องจากปัจจัยดังกล่าวค่าของ Network Throughput จะมีค่าเกือบจะเท่ากัน แต่จากการทดสอบพบว่า PMLS นั้นมีค่า Network Throughput สูงกว่า PMP โดยเฉลี่ย 7% เนื่องจาก Data Packet ของ PMLS นั้นมีขนาดสั้นกว่า รวมทั้งการแก้ไขเส้นทางโดยวิธี Local Repair นั้นเราได้จำกัดจำนวน Hop เอาไว้ไม่ให้เกินครึ่งหนึ่งของจำนวน Hop ที่มากที่สุดของเส้นทางที่ยอมให้มีได้ในเครือข่าย

เพื่อตรวจสอบประสิทธิภาพการทำงานของโปรโตคอล PMLS ที่มีการตรวจจับความเสียหายของเส้นทางในกรณีที่ค่า Signal Strength ของโหนดถัดไปต่ำกว่าค่าวิกฤตที่กำหนดไว้ (PMLS1) กับโปรโตคอล PMLS ที่ไม่มีการตรวจจับความเสียหายของเส้นทางในกรณีดังกล่าว (PMLS2) ได้ผลแสดงดังภาพที่ 4.4, ภาพที่ 4.5, ภาพที่ 4.6 และภาพที่ 4.7 ซึ่งจะเห็นได้ว่าการตรวจจับความเสียหายของเส้นทางในกรณีดังกล่าวทำให้ประสิทธิภาพในการทำงานของ PMLS นั้นดีกว่า โดย Packet Delivery Ratio นั้นสูงกว่า 1% Number of Packets Dropped นั้นต่ำกว่า 5% ส่วน Network Throughput นั้นสูงกว่า 5% ทั้งนี้เนื่องจากหากมีการตรวจจับความเสียหายของเส้นทางในกรณีที่ค่า Signal Strength ของโหนดถัดไปต่ำกว่าค่าวิกฤตที่กำหนดไว้โหนดต่าง ๆ ในเส้นทางที่ตรวจจับความเสียหายได้นั้นจะสามารถนำเส้นทางอื่น ๆ มาใช้แทนเส้นทางนั้นได้ก่อนล่วงหน้าโดยไม่ต้องส่งข้อมูลไปยังโหนดถัดไปซ้ำ ๆ จนกระทั่งถึงขีดจำกัดค่าหนึ่งจึงจะรับรู้ได้ว่าเส้นทางนั้นเสียหายทำให้ปริมาณข้อมูลที่ส่งออกไปจากโหนดต่าง ๆ ลดลงส่งผลให้ประสิทธิภาพการ

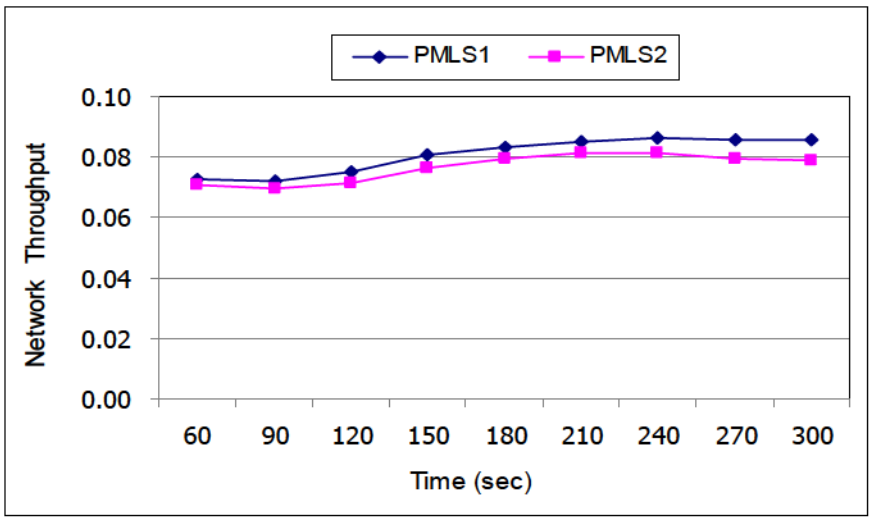
ทำงานนั้นดีขึ้น นอกจากนั้นสัดส่วนของจำนวน Probe Packet, RREQ, RREP, RERR, LREQ และ LREP ทั้งหมดที่ทุกโหนดส่งออกมาต่อจำนวน Data Packet ที่ปลายทางรับได้สำเร็จ (Routing Overhead) สูงกว่าประมาณ 0.2% ซึ่งน้อยมากจนแทบไม่เห็นความแตกต่าง



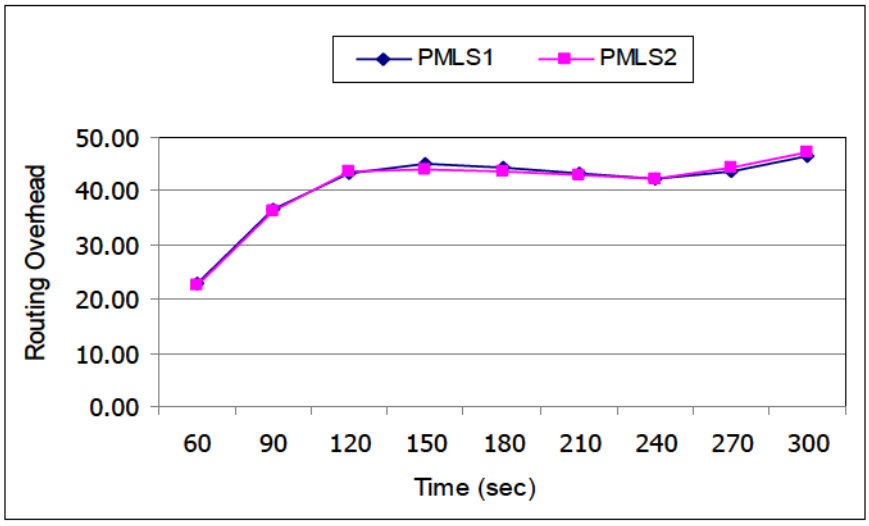
ภาพที่ 4.4 Packet Delivery Ratio ของ PMLS1 และ PMLS2



ภาพที่ 4.5 Number of Packets Dropped ของ PMLS1 และ PMLS2



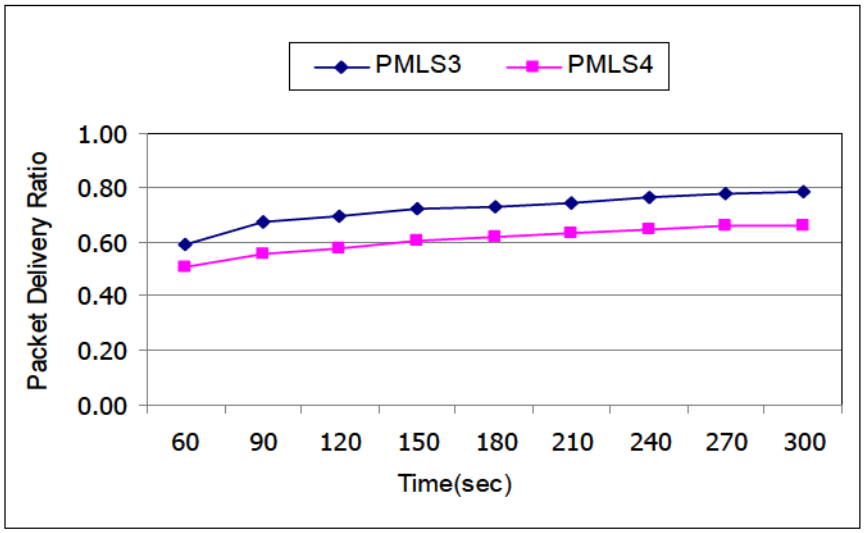
ภาพที่ 4.6 Network Throughput ของ PMLS1 และ PMLS2



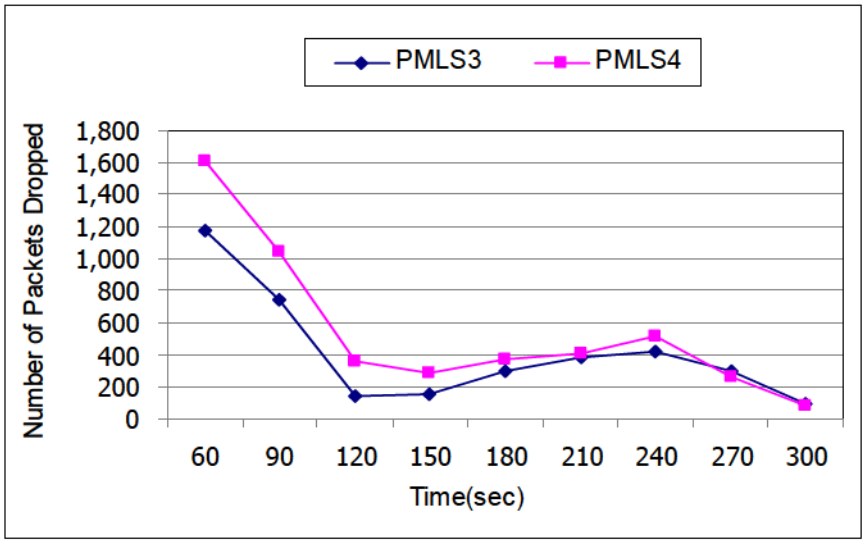
ภาพที่ 4.7 Routing Overhead ของ PMLS1 และ PMLS2

เพื่อเปรียบเทียบประสิทธิภาพการทำงานของโปรโตคอล PMLS ที่ใช้ Label เป็นดรรชนีสำหรับการค้นหาเส้นทางส่งข้อมูล (PMLS3) กับโปรโตคอล PMLS ที่ใช้ IP Address ของโหนดปลายทางเป็นดรรชนีสำหรับการค้นหาเส้นทางส่งข้อมูล (PMLS4) ได้ผลแสดงดังภาพที่ 4.8, ภาพที่ 4.9 และภาพที่ 4.10 ซึ่งจะเห็นได้ว่าประสิทธิภาพการทำงานของโปรโตคอล PMLS โดยใช้ Label เป็นดรรชนีในการค้นหาเส้นทางส่งข้อมูลนั้นดีกว่า โดย Packet Delivery Ratio นั้นสูงกว่า 19% ส่วน Number of Packets Dropped นั้นต่ำกว่า 25% และ Network Throughput นั้นสูงกว่า 2% ทั้งนี้

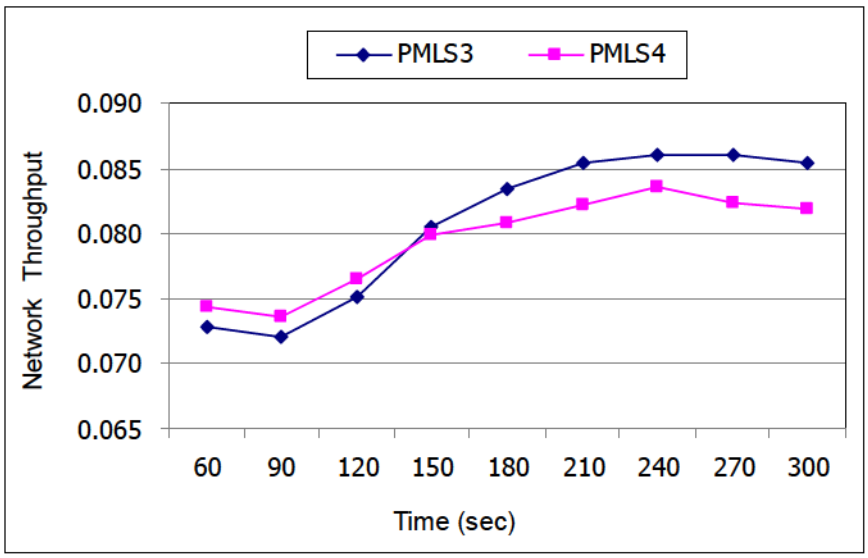
เนื่องจากการใช้ Label ซึ่งมีขนาดสั้นกว่า IP Address เป็นดรรชนีสำหรับการค้นหาเส้นทางใน Route Table นั้นสามารถลดภาระการคำนวณได้อย่างมากทำให้การทำงานมีความรวดเร็วลดเวลาการหน่วงในการส่งต่อข้อมูลของโหนดระหว่างทางลงส่งผลให้ประสิทธิภาพการทำงานนั้นดีขึ้น



ภาพที่ 4.8 Packet Delivery Ratio ของ PMLS3 และ PMLS4



ภาพที่ 4.9 Number of Packets Dropped ของ PMLS3 และ PMLS4



ภาพที่ 4.10 Network Throughput ของ PMLS3 และ PMLS4

บทที่ 5

สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

งานวิจัยนี้ได้นำเสนอโปรโตคอลหาเส้นทางส่งข้อมูลในเครือข่ายไร้สาย ซึ่งเรียกว่า Predicted Multipath Label Switching Protocol (PMLS) โดยใช้หลักการหาเส้นทางของ PMP เพราะเส้นทางที่ได้มีความคงทนและประสิทธิภาพในการส่งข้อมูลของเส้นทางสูง และใช้หลักการ Label Switching ในการส่งข้อมูล เพราะทำให้ Header ของข้อมูลมีขนาดสั้นลงทั้งยังสนับสนุนการจัดการคุณภาพบริการในการส่งข้อมูลได้อีกด้วย ในงานวิจัยยังได้เสนอการบำรุงรักษาเส้นทางด้วยวิธีการ Local Repair และ Route Discovery ที่โหนดระหว่างทางเพื่อแก้ไขเส้นทางที่เสียหาย และในการวัดประสิทธิภาพของโปรโตคอลที่นำเสนอโดยการเปรียบเทียบกับ PMP นั้นผลการทดลองได้แสดงว่าทั้ง Packet Delivery Ratio, Number of Packets Dropped และ Network Throughput นั้น PMLS มีผลการทำงานที่ดีกว่า PMP

ในกระบวนการนำส่งข้อมูลของเราจะแตกต่างจากโปรโตคอล PMP เป็นอย่างมากเนื่องจากโปรโตคอล PMP นั้นจะใช้ Header ของข้อมูลที่มีขนาดยาวกว่า รวมทั้งการส่งต่อข้อมูลจะใช้ Source Route ที่มากับ Data Packet โดยตรงทำให้ได้เส้นทางที่ไม่ทันสมัยเท่ากับโปรโตคอลของเรา Data Packet จะต้องถูกส่งไปให้โหนดถัดไปจนกระทั่งไม่สามารถส่งไปได้จึงจะทราบว่า nextHop นั้นใช้งานไม่ได้แล้ว แต่สำหรับโปรโตคอลของเราจะใช้ Route Table ซึ่งมีการ Update ข้อมูลของเส้นทางทันทีเมื่อพบว่า nextHop นั้นเสียหายอันเนื่องมาจากสภาพความแรงของสัญญาณของโหนดถัดไปนั้นต่ำกว่าค่า DA ที่กำหนดไว้ Data Packet จึงสามารถเปลี่ยนไปใช้เส้นทางอื่นได้เร็วกว่าไม่ต้องเสียเวลาส่งไปยังโหนดถัดไปจนกว่าจะส่งไปไม่ได้แล้วจึงเปลี่ยนไปใช้เส้นทางอื่น จะเห็นได้ว่าการใช้วิธีการ Label Switching ในการส่งต่อข้อมูลนั้นทำให้ได้ Data Packet ที่มีขนาดสั้นกว่าและการสับเปลี่ยนไปใช้เส้นทางใหม่ทำได้รวดเร็วและมีความยืดหยุ่นกว่าด้วย

สำหรับโปรโตคอล LSMR นั้นการส่งต่อข้อมูลแม้จะใช้วิธีการ Label Switching เหมือนกับของเราก็ตาม แต่มันจะทำงานใน NIC โดยใช้ MAC Address ของโหนดถัดไป จาก Route Table ที่เก็บไว้ที่ NIC แทนที่จะเป็น IP Address จาก Route Table ที่อยู่ในหน่วยความจำของ CPU

เหมือนกับของเรา หรือกล่าวได้ว่าโปรโตคอล LSMR นั้นต้องมีหน่วยความจำและโครงสร้างพิเศษภายใน NIC เพื่อรองรับการทำงานส่วนของเรานั้นไม่ต้องมี จะสังเกตได้ว่าการส่งต่อข้อมูลของโปรโตคอล LSMR จะให้แต่ละโหนดส่งข้อมูลไปยังโหนดถัดไปให้ได้เท่านั้น ใน Route Table จึงไม่มีเส้นทางที่ไปยังปลายทางเก็บเอาไว้จะมีแต่เฉพาะ Label กับ MAC Address ของโหนดถัดไปและ Life Time เท่านั้น หากไม่สามารถส่งข้อมูลไปยังโหนดถัดไปได้จะมีการแก้ไขเส้นทางโดยการทำ Local Repair ให้ได้เส้นทางใหม่เพื่อไปยังโหนดถัดไปนั้นให้ได้เท่านั้น นอกจากนั้นการส่งข้อมูลในแต่ละ Hop จะต้องเสียเวลาในการปรับปรุง Life Time ของ Route Table ด้วยเพราะไม่เช่นนั้นแล้วเส้นทางที่มีอยู่ใน Route Table จะถูกลบทิ้งไปแม้ว่าจะยังคงใช้งานได้อยู่ก็ตาม

กระบวนการนำส่งข้อมูลของเราจะใกล้เคียงกับโปรโตคอล ORAL มากที่สุดเนื่องจากใช้แนวคิดและโครงสร้างในการเก็บเส้นทางที่คล้าย ๆ กัน ต่างกันเฉพาะในโปรโตคอล ORAL นั้นโหนดระหว่างทางจะไม่มีการใช้ IP Address ของโหนดปลายทางมาหาเส้นทางอื่นใน Route Table เพื่อใช้ในการส่งข้อมูลเมื่อไม่สามารถส่งข้อมูลนั้นไปยังโหนดถัดไปได้

นอกจากนั้นในกระบวนการบำรุงรักษาเส้นทางของเรานั้นจะมีความแตกต่างกับโปรโตคอล ORAL และ LSMR เนื่องจาก ORAL และ LSMR นั้นจะมีการแจ้งความผิดพลาดเฉพาะกรณีที่ไม่สามารถส่งข้อมูลไปยังโหนดถัดไปได้ และหาเส้นทางใน Route Table ของตนเพื่อส่งต่อไปยังโหนดถัดไปไม่ได้เท่านั้น แต่จะไม่มีการแจ้งความผิดพลาดของ Link ใด ๆ ในเส้นทางนั้นหากค่าความแรงของสัญญาณของโหนดถัดไปต่ำเกินกว่าค่าที่กำหนด ส่วนโปรโตคอล PMP นั้น จะไม่มีการแจ้งความผิดพลาดในกรณีหาเส้นทางใน Route Table ของตนเพื่อส่งต่อไปยังโหนดถัดไปไม่ได้ เนื่องจากในกระบวนการนำส่งข้อมูลนั้นโปรโตคอล PMP ไม่ได้ใช้ Route Table เพื่อส่งต่อข้อมูลให้กับโหนดถัดไป แต่ใช้ Source Route ที่ผูกติดมากับ Data Packet ดังนั้นโหนดต่าง ๆ จะทราบว่าเส้นทางใด ๆ เสียหายก็ต่อเมื่อโหนดใดโหนดหนึ่งไม่สามารถส่งข้อมูลไปยังโหนดถัดไปได้เท่านั้น ทำให้เส้นทางต่าง ๆ ใน Route Table ของโหนดต่าง ๆ ไม่ทันสมัยเท่ากับโปรโตคอลของเรา ความแตกต่างที่เห็นได้ชัดอีกประการหนึ่งก็คือ โปรโตคอล PMP นั้นจะไม่มีกระบวนการแก้ไขเส้นทางด้วยวิธีการ Local Repair และ Route Discovery ที่โหนดระหว่างทางมีแต่การนำเส้นทางสำรองมาใช้แทนเส้นทางหลักเมื่อเส้นทางเสียหายเท่านั้น ส่วนโปรโตคอล ORAL และ LSMR นั้นจะไม่มีการแก้ไขเส้นทางโดยใช้เส้นทางสำรอง และสำหรับการทำ Local Repair ของ LSMR นั้นจะแก้ไขเส้นทางเฉพาะส่วนที่จะไปยังโหนดถัดไปเท่านั้น ไม่ได้เป็นเส้นทางใหม่ที่ไปยังโหนดปลายทางทั้งเส้นทาง การทำเช่นนี้อาจจะเสีย Overhead น้อย แต่อาจจะได้เส้นทางที่มีจำนวน Hop เพิ่มมากขึ้นกว่าเดิม ซึ่งหมายความว่าโอกาสที่ข้อมูลจะสูญหายไปในระหว่างทางก่อนจะถึงปลายทางก็จะเพิ่มมากขึ้นด้วย และหากมีการทำ Local Repair ซ้ำอีกบนเส้นทางดังกล่าวก็จะยิ่งทำ

ให้ประสิทธิภาพในการส่งข้อมูลลดลงไปอีกเนื่องจากเส้นทางของโปรโตคอล LSMR นั้นไม่ได้ใช้หลักการของโปรโตคอล PMP จึงทำให้เส้นทางนั้นมีความคงทนต่อความเสียหายน้อย แต่สำหรับการทำ Local Repair ของเรานั้นสามารถนำเส้นทางที่ดีที่สุดจาก Route Table ของบรรดาโหนดต่าง ๆ ที่อยู่ติดกันซึ่งเป็นไปตามหลักการของโปรโตคอล PMP มาเลือกเส้นทางที่มีจำนวน Hop ที่น้อยที่สุดเพื่อใช้เป็นเส้นทางใหม่ในการส่งข้อมูลได้ เส้นทางที่ได้จึงมีความคงทนต่อความเสียหายมากกว่า รวมทั้งเสีย Overhead น้อยเนื่องจากร้องขอเส้นทางจากโหนดที่ติดอยู่กับตนเท่านั้น การทำ Local Repair ของเราจึงมีประสิทธิภาพดีกว่าการทำ Local Repair ของโปรโตคอล LSMR สำหรับการทำ Local Repair ของเราดูจะใกล้เคียงกับโปรโตคอล ORAL มากที่สุดแต่โปรโตคอล ORAL นั้นจะหาเส้นทางใหม่โดยจะส่งการร้องขอต่อไปเป็นทอด ๆ จนกว่าจะได้รับเส้นทางที่สามารถไปยังปลายทางที่ต้องการได้โดยจะไม่มีการพิจารณาว่าเป็นเส้นทางที่ดีที่สุดหรือมีจำนวน Hop น้อยที่สุดหรือไม่ เส้นทางที่ได้อาจจะได้จากโหนดปลายทางโดยตรงหรือโหนดระหว่างทางใด ๆ ซึ่งมีเส้นทางที่ไปยังปลายทางที่ต้องการเก็บอยู่ใน Route Table ก็ได้ ส่วนการทำ Local Repair ของเรานั้นจะจำกัดการส่งการร้องขอไว้เฉพาะโหนดที่อยู่ติดกับโหนดที่ทำ Local Repair เท่านั้น ทำให้มี Overhead น้อยกว่า และที่สำคัญหากการแก้ไขเส้นทางสำเร็จเส้นทางที่ได้จะเป็นเส้นทางที่มีความคงทนกว่าโปรโตคอล ORAL แต่หากแก้ไขเส้นทางด้วยวิธีการ Local Repair ไม่สำเร็จก็จะรอรับเส้นทางจากการทำ Route Discovery ที่โหนดระหว่างทางกลับมาจากปลายทางโดยตรงซึ่งเส้นทางที่ได้มาจากกระบวนการนี้จะดีกว่าและทันสมัยมากกว่าโปรโตคอล ORAL ยิ่งขึ้นไปอีก

5.2 ข้อเสนอแนะ

โดยการนำ Label Switching มาผนวกรวมไว้กับ PMP เราพบว่าโปรโตคอลดังกล่าวสามารถนำมาใช้ในการกระจายข้อมูลส่งไปตามเส้นทางหลาย ๆ เส้นทางพร้อม ๆ กันเพื่อลดความแน่นขนัดของข้อมูลได้ ดังนั้นในการพัฒนางานวิจัยต่อไปนั้นควรมุ่งเน้นไปในเรื่องการส่งข้อมูลที่ต้องการการจัดการคุณภาพบริการการส่งข้อมูล (Qos) เช่น ข้อมูลเสียง และข้อมูลภาพเคลื่อนไหว ตลอดจนศึกษาถึงความมั่นคงของเครือข่าย และยังสามารถปรับแต่งโปรโตคอลดังกล่าวให้สามารถนำไปใช้งานได้จริงตามสภาวะแวดล้อมต่าง ๆ โดยทดสอบตามสภาวะแวดล้อมที่หลากหลายยิ่งขึ้นเพื่อให้ได้ประสิทธิภาพที่ดีตรงตามสภาวะการใช้งานจริง นอกจากนั้นสามารถทำให้ประสิทธิภาพการรับส่งข้อมูลดีขึ้นได้โดยใช้ MAC Address แทนค่า nextHop โดยการรับส่งข้อมูลให้ทำงานโดยใช้ NIC แทนการทำงานในระดับชั้นเครือข่ายดังเช่นการส่งข้อมูลของโปรโตคอล LSMR

บรรณานุกรม

Barr, Rimon. 2004a. **JiST-Java in Simulation Time User Guide.** Pp.1-34. Retrieve October 18, 2008 from http://jist.ece.cornell.edu/docs/040319-jist-user.pdf

Barr, Rimon. 2004b. **SWANS-Scalable Wireless Ad hoc Network Simulator User Guide**. Pp. 1-15. Retrieve October 18, 2008 from http://jist.ece.cornell.edu/docs/040319-swans-user.pdf

Clausen, Thomas and Jacquet, Philippe, eds. 2003. Optimized Link State Routing Protocol (OLSR). **Network Working Group, Request for Comments: 3626.** Pp. 1-75. Retrieve February 18, 2008 from http://www.ietf.org/rfc/rfc3626.txt

Haas, Zygmunt J. 1997. A New Routing Protocol for the Reconfigurable Wireless Networks. In **Proceeding of IEEE 6th International Conference on Universal Personal Communications 97**. San Diego, CA: IEEE Computer Society. Pp. 562–566.

Hyytiä, Esa; Koskinen, Henri; Lassila, Pasi; Penttinen, Aleksi and Virtamo, Jorma. 2005. Random Waypoint Model in Wireless Network. **Networks and Algorithms: complexity in Physics and Computer Science**. Pp. 1-55. Retrieve February 18, 2008 from http://mathstat.helsinki.fi/mathphys/EVERGROW/virtamo.pdf

IEEE. 2003. **ANSI/IEEE Std 802.11, 1999 Edition (R2003)--Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications**. Pp. i-513. Retrieve October 8, 2008 from http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1389197&isnumber=30234

Joa-Ng, Mario and Lu, I-Tai. 1999. A Peer-to-Peer Zone-Based Two-Level Link State Routing for Mobile Ad Hoc Networks**. IEEE Journal on Selected Areas In Communication.** 1(8): 1415-1425.

Johnson, David B.; Hu, Yih-Chun and Maltz, David A. 2007. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPV4. **Network Working Group, Request for Comments: 4728.** Pp. 1-107. Retrieve March 13, 2008 from http://www.ietf.org/rfc/rfc4728.txt

Lee, Sung and Gerla, Mario. 2001. Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks. In **Proceedings of IEEE ICC 2001**. Communications. Helsinki: IEEE Computer Society. Pp. 3201-3205.

Lv, Shaohe; Zhou, Xingming; Wang, Xiaodong and Liu, Chi. 2006. An On-Demand Routing Protocol in Ad Hoc Network Using Label Switching. In **Mobile Ad-hoc and Sensor Networks**. J. Cao et al., eds. Lecture Notes in Computer Science. Berlin: Springer. Pp. 95-106.

Marina, Mahesh K. and Das, Samor R. 2001. On-demand Multipath Distance Vector Routing in Ad Hoc Networks. In **Proceedings of the Ninth International Conference on Network Protocols.** Washington, DC: IEEE Computer Society. Pp. 14-23.

Ogier, Richard G.; Templin Fred L. and Lewis, Mark G. 2004. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). **Network Working Group, Request for Comments: 3684.** Pp. 1-46. Retrieve March 20, 2009 from http://www.faqs.org/ftp/rfc/pdf/rfc3684.txt.pdf

Perkins, Charles E.; Belding-Royer, Elizabeth M. and Das, Samir R. 2003. Ad hoc On-Demand Distance Vector (AODV) Routing. **Network Working Group, Request for Comments: 3561.** Pp. 1-37. Retrieve December 28, 2008 from http://www.ietf.org/rfc/rfc3561.txt

Perkins, Charles E. and Bhagwat, Pravin. 1994. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In **SIGCOMM '94: Proceedings of the Conference on Communications Architectures, Protocols and Applications**. Pp. 234-244.

Ramasubramanian,Venugopalan; Hass, Zygmunt J. and Sirer, Emin Gün. 2003. SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks. In **Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing**. Annapolis: ACM. Pp. 303 – 314.

Rosen, Eric C.; Viswanathan, Arun and Callon, Rose. 2001. Multiprotocol Label Switching Architecture. **Network Working Group, Request for Comments: 3031.** Pp. 1-61. Retrieve March 14, 2008 from http://www.ietf.org/rfc/rfc3031.txt

Supachote Lertvorratham and Pipat Hiranvanichakorn. 2007. Path Selection Strategies for Multipath Ad Hoc Network. In **Communication Systems and Networks (AsiaCSN-2007).** M., Hamza, ed. Phuket: ACTA Press. Pp. 1-7.

Wei, Rong; Wu, Muqing and Yu, Tianhang. 2007. LSMR:A Label Switching Multipath Routing Protocol for Ad Hoc Networks. In **Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing - Volume 02**. Feng,Wenying and Gao, Feng, eds. Qingdao: IEEE Computer Society. Pp 546-551.

ประวัติผู้เขียน

ชื่อ ชื่อสกุล	นาวาอากาศตรี วิบูลย์ คูบูรณ์
ประวัติการศึกษา	วิศวกรรมศาสตรบัณฑิต (วิศวกรรมโยธา) โรงเรียนนายเรืออากาศ ปีที่สำเร็จการศึกษา พ.ศ. 2541
ประสบการณ์การทำงาน	พ.ศ. 2541 - 2552 แผนกวิศวกรรมโยธา กองออกแบบและก่อสร้าง กองวิทยาการ กรมช่างโยธาทหารอากาศ กองทัพอากาศ