โพรโทคอลการค้นหาเส้นทางในเครือข่ายเฉพาะกิจเพื่อหลีกเลี่ยงการขาดการเชื่อมต่อของเส้นทางบนพื้นฐานของโพรโทคอล DSR

Ad-Hoc Routing Protocol Avoiding Route Breaks Based on DSR

คำนำ

เครือข่ายการสื่อสารข้อมูลในยุคปัจจุบันนี้ ไม่ได้มีการใช้งานแต่เพียงเครือข่ายพื้นฐานเท่านั้น ยังมีเครือข่ายการสื่อสารไร้สายที่นับวันยิ่งทวีความสำคัญมากยิ่งขึ้น เห็นได้จากการที่มีการผลิตอุปกรณ์ โดยเฉพาะอุปกรณ์ที่ใช้ในการสื่อสารที่มีการพัฒนาให้ดีขึ้นมาก เพื่อตอบสนองและสนับสนุนความต้องการของผู้ใช้งานให้มากที่สุด อย่างเช่น เครื่องโน้ตบุ๊กคอมพิวเตอร์ (notebook) เครื่องช่วยงานส่วนบุคคลแบบดิจิทัล (PDA: personal digital assistant) พาล์มท็อปคอมพิวเตอร์ (palm) และโทรศัพท์เคลื่อนที่ (mobile phone) เป็นต้น

ปัจจุบันการใช้งานเครือข่ายไร้สายแบ่งการใช้งานเป็น 2 ลักษณะ โดยเครือข่ายในลักษณะแรกนั้นเป็นระบบเครือข่ายที่ต้องมีสถานีฐานในการให้บริการในรัศมีที่จำกัด (infrastructure network) อาทิ ระบบโทรศัพท์เคลื่อนที่และระบบ WLAN (wireless local area network) เป็นต้น ส่วนเครือข่ายไร้สายอีกลักษณะนั้นเป็นระบบเครือข่ายเฉพาะกิจ (ad hoc network) (Chlamtac *et al*., 2003) ซึ่งอุปกรณ์เคลื่อนที่ (mobile unit) ทำหน้าที่คล้ายกับสถานีฐานด้วย โดยเครือข่ายลักษณะนี้ไม่จำเป็นต้องติดตั้งสถานีฐานแต่มีข้อจำกัดในเรื่องของการควบคุมและการเชื่อมต่อสื่อสารของอุปกรณ์เคลื่อนที่ในเครือข่าย เนื่องจากอุปกรณ์เหล่านั้นมีการเคลื่อนที่อยู่ตลอดเวลา ดังนั้นการค้นหาเส้นทางในเครือข่ายเฉพาะกิจเพื่อให้อุปกรณ์รับและส่งสามารถทำการติดต่อกันได้จึงเป็นสิ่งจำเป็นอย่างยิ่ง

เครือข่ายเฉพาะกิจนั้นเป็นเครือข่ายไร้สายที่มีความเป็นอิสระ โดยลักษณะการเชื่อมต่อมีรูปแบบการเชื่อมต่อเครือข่ายที่มีการเปลี่ยนแปลงแบบพลวัต (dynamic) ซึ่งโพรโทคอลการค้นหาเส้นทาง (routing protocols) ที่ใช้นั้นจำเป็นต้องมีประสิทธิภาพและสามารถสนับสนุนความต้องการของเครือข่ายในลักษณะนี้ได้

โพรโทคอลการค้นหาเส้นทางที่มีใช้ในเครือข่ายเฉพาะกิจแบ่งได้ 2 ประเภท (Royer and Toh, 1999) คือ แบบ Proactive และแบบ Reactive

โพรโทคอลแบบ Proactive (Royer and Toh, 1999; Perkins, 2001) นี้มีลักษณะคือแต่ละอุปกรณ์เคลื่อนที่หรือแต่ละโหนดในเครือข่ายจำเป็นต้องเก็บข้อมูลเส้นทางไปยังทุกโหนดไว้ในตารางเส้นทาง (routing table) และข้อมูลในตารางเส้นทางจะมีการเปลี่ยนแปลงเป็นรายคาบเวลา (period) ซึ่งในการติดต่อสื่อสารกันนั้นโหนดเริ่มต้นสามารถหาเส้นทางการติดต่อสื่อสารไปยังโหนดในเครือข่ายที่ต้องการติดต่อได้ทันทีโดยดูจากข้อมูลในตารางเส้นทางที่ได้เก็บไว้

โพรโทคอลแบบนี้มีข้อเสียเปรียบในกรณีเครือข่ายมีขนาดที่ใหญ่ขึ้น แต่ละโหนดจำเป็นต้องเก็บข้อมูลเส้นทางไปยังทุกโหนดที่อยู่ภายในเครือข่าย ซึ่งขนาดของข้อมูลเส้นทางในตารางเส้นทางของแต่ละโหนดมีขนาดแปรผันตามจำนวนของโหนดภายในเครือข่าย ด้วยเหตุนี้เมื่อเครือข่ายขนาดที่ใหญ่ขึ้นจึงมีจำนวนโหนดเพิ่มขึ้น ทำให้มีการสิ้นเปลืองหน่วยความจำเป็นจำนวนมากที่นำไปใช้เพื่อการจัดเก็บข้อมูลตารางเส้นทาง นอกจากนี้เมื่อโทโปโลยี (topology) ภายในเครือข่ายเกิดการเปลี่ยนแปลง โหนดภายในเครือข่ายจึงต้องเปลี่ยนแปลงข้อมูลในตารางเส้นทางของแต่ละโหนดตามลักษณะการเปลี่ยนแปลงโทโปโลยีของเครือข่าย ส่งผลถึงการใช้ทรัพยากรของเครือข่ายในปริมาณที่เพิ่มขึ้นเพื่อใช้ในการเปลี่ยนแปลงข้อมูลในตารางเส้นทางดังกล่าว

โพรโทคอลแบบ Reactive (Royer and Toh, 1999; Perkins, 2001) นี้มีลักษณะคือ เมื่อโหนดเริ่มต้นต้องการติดต่อสื่อสารกับโหนดในเครือข่าย โหนดเริ่มต้นจะทำการส่งแพคเกตร้องขอสำหรับใช้ในการค้นหาเส้นทางโดยกระจายแพคเกต (broadcast) ไปยังโหนดที่อยู่ข้างเคียง และโหนดที่ได้รับแพคเกตนี้หากไม่ใช่โหนดปลายทาง โหนดจะส่งแพคเกตนี้กระจายออกไปอีกครั้ง (rebroadcast) จนกระทั่งถึงโหนดปลายทาง เมื่อโหนดปลายทางได้รับแพคเกตร้องขอนี้ โหนดปลายทางจะตอบกลับไปยังโหนดเริ่มต้นด้วยแพคเกตตอบกลับที่จะทำการส่งผ่านโหนดตัวกลางและส่งแพคเกตต่อไปจนกระทั่งโหนดเริ่มต้นได้รับแพคเกตตอบกลับ โหนดเริ่มต้นจึงจะใช้เส้นทางจากแพคเกตตอบกลับนี้ในการติดต่อสื่อสารกับโหนดปลายทาง

โพรโทคอลแบบนี้มีข้อได้เปรียบคือโหนดภายในเครือข่ายไม่จำเป็นต้องเก็บข้อมูลเส้นทางไว้ในหน่วยความจำของแต่ละโหนดตลอดเวลา โดยจะเก็บข้อมูลเส้นทางไว้เฉพาะเมื่อมีการติดต่อสื่อสารกัน ซึ่งข้อมูลเส้นทางจะได้มาก็ต่อเมื่อโหนดเริ่มต้นมีการร้องขอเส้นทางเท่านั้น ทำให้ลด

การสิ้นเปลืองทรัพยากรของเครือข่ายเพื่อใช้ในการดูแลข้อมูลเส้นทางให้สามารถพร้อมใช้งานได้ตลอดเวลา นอกจากนี้กรณีที่บางโหนดไม่มีความจำเป็นต้องใช้ข้อมูลเส้นทางที่เส้นทางมีการเปลี่ยนแปลงไป โหนดเหล่านั้นไม่จำเป็นต้องสูญเสียเวลาและทรัพยากรในการเปลี่ยนแปลงข้อมูลเส้นทางต่างๆ ตามการเปลี่ยนแปลงที่เกิดขึ้น แต่มีข้อเสียเปรียบในเรื่องของเวลาที่ใช้ไปเพื่อให้ได้เส้นทางการติดต่อสื่อสารทำให้การเริ่มต้นส่งแพคเกตข้อมูลกระทำได้ช้า เนื่องจากในขณะที่โหนดเริ่มต้นต้องการใช้เส้นทางเพื่อติดต่อสื่อสารกับโหนดปลายทาง โหนดเริ่มต้นต้องทำการค้นหาเส้นทางก่อนเพราะเส้นทางไม่ได้มีการจัดเตรียมไว้ให้สามารถใช้ได้ทันที แต่ในขณะที่โพรโทคอลแบบ Proactive นั้นสามารถได้เส้นทางการติดต่อสื่อสารทันทีจากข้อมูลในตารางเส้นทางที่แต่ละโหนดเก็บไว้

ทั้งโพรโทคอลแบบ Proactive และ Reactive อาจมีปัญหาเกิดขึ้นกับเส้นทางการติดต่อสื่อสารในขณะที่แต่ละโหนดทำการรับส่งแพคเกตกัน เช่น เส้นทางขาดการติดต่อกัน อาจเพราะเครือข่ายมีความต้องการใช้เส้นทางใดเส้นทางหนึ่งมากจนเกิดปัญหาบนเส้นทางนั้น ส่งผลให้เส้นทางดังกล่าวไม่สามารถให้การสนับสนุนการติดต่อสื่อสารต่อไปได้ หรือเกิดจากปัญหาทางเทคนิคของตัวอุปกรณ์สื่อสาร เช่น แหล่งจ่ายพลังงานหมดลงทำให้โหนดดังกล่าวขาดการติดต่อเป็นต้น หรือมีสาเหตุมาจากคุณลักษณะของเครือข่าย เช่น ช่องความถี่ทางการสื่อสารที่มีอยู่อย่างจำกัด ซึ่งโหนดในเครือข่ายมีความต้องการใช้งานช่องความถี่ในปริมาณที่สูงทำให้ไม่เพียงพอต่อการใช้งาน เป็นต้น และจากการที่โหนดในเครือข่ายมีการเคลื่อนที่ ทำให้โทโปโลยีของเครือข่ายมีการเปลี่ยนแปลงส่งผลให้เส้นทางการติดต่อสื่อสารที่ใช้งานอยู่มีการเปลี่ยนแปลงของเส้นทาง ซึ่งโหนดอาจขาดการติดต่อสื่อสารกันก่อนที่จะส่งแพคเกตออกไปได้ ซึ่งปัญหาเหล่านี้ทำให้เกิดการติดต่อสื่อสารที่ไม่ลื่นไหลก่อให้เกิดการสูญหายแพคเกตที่มีการรับส่งไปมาระหว่างกัน

จากปัญหาการติดต่อสื่อสารที่ไม่ลื่นไหลดังกล่าวโพรโทคอล AODV-BA (ad hoc on-demand distance vector with break avoidance) (Tauchi *et al*., 2005) ได้นำเสนอวิธีหลีกเลี่ยงการใช้เส้นทางที่อาจขาดการติดต่อ โดยนำพารามิเตอร์ (parameter) 4 พารามิเตอร์คือ ความแรงของสัญญาณวิทยุที่รับได้โดยโหนดที่เป็นเครื่องรับสัญญาณ (receiver) การเหลื่อมกันของเส้นทางที่เกิดขึ้นจากโหนดตัวกลางที่รับภาระจากหลายเส้นทางให้ทำหน้าที่ส่งต่อแพคเกต พลังงานแบตเตอรี่ที่เหลือของแต่ละโหนด และความหนาแน่นของโหนดที่มีอยู่ภายในเครือข่ายมาทำการปรับใช้กับโพรโทคอลการค้นหาเส้นทางแบบตามความต้องการบนพื้นฐานของโพรโทคอล AODV (ad hoc on-demand distance vector) (Perkins *et al*., 2002) เพื่อตรวจสอบอันตรายที่อาจเกิดขึ้นกับเส้นทาง

การส่งแพคเกตข้อมูล โดยทำการค้นหาเส้นทางใหม่มาทดแทนเส้นทางเดิม ก่อนที่เส้นทางเดิมจะไม่สามารถใช้งานได้

งานวิจัยนี้จึงได้นำพารามิเตอร์ดังกล่าวมาปรับใช้กับโพรโทคอล DSR (dynamic source routing) (Johnson *et al*., 2003) ซึ่งเป็นโพรโทคอลการค้นหาเส้นทางแบบตามความต้องการ เช่นเดียวกับโพรโทคอล AODV แต่โพรโทคอล DSR มีการสูญหายของแพคเกตในเครือข่ายที่มากกว่าโพรโทคอล AODV (Das *et al*., 2000) อีกทั้งลักษณะการจัดเก็บข้อมูลเส้นทางของทั้งสองโพรโทคอลนั้นแตกต่างกัน โดยโพรโทคอล DSR จัดเก็บข้อมูลเส้นทางในแคชเส้นทาง (route cache) และเส้นทางที่ใช้เพื่อการส่งต่อแพคเกตข้อมูลโหนดเริ่มต้นจะกำหนดเส้นทางซึ่งเรียกเส้นทางดังกล่าวว่า source route และทำการดูแลเส้นทางการเชื่อมต่อโดยอาศัยข้อมูลจากแคชเส้นทางเป็นหลัก ในขณะที่โพรโทคอล AODV ทำการจัดเก็บข้อมูลเส้นทางลงในตารางเส้นทาง (routing table) เมื่อต้องการส่งต่อแพคเกตข้อมูลจะทำการส่งต่อไปยังโหนดถัดไป (next node) ที่ระบุไว้จากตารางเส้นทาง และเมื่อต้องการดูแลรักษาเส้นทางการเชื่อมต่อจะอาศัยโหนดเพื่อนบ้าน (neighbor node) ช่วยทำหน้าที่ในการเปลี่ยนแปลงข้อมูลในตารางเส้นทาง จากความแตกต่างของโพรโทคอลที่กล่าวมาทั้งสอง ดังนั้นในงานวิจัยนี้จึงได้นำเสนอการปรับใช้พารามิเตอร์ดังกล่าวกับโพรโทคอล DSR ทั้งนี้เพื่อตรวจสอบอันตรายที่อาจเกิดขึ้นกับเส้นทางการติดต่อสื่อสาร ให้สามารถหาเส้นทางการเชื่อมต่อเส้นทางใหม่ในแคชเส้นทางมาใช้แทนเส้นทางเดิม ก่อนที่เส้นทางเดิมจะไม่สามารถทำการติดต่อสื่อสาร ทั้งนี้เพื่อให้สามารถลดการสูญหายแพคเกตที่ทำการรับส่งในเครือข่ายอันเนื่องมาจากเส้นทางการรับส่งแพคเกตข้อมูลขาดการเชื่อมต่อสื่อสารกัน

<u>วัตถุประสงค์</u>

1. เพื่อลดการสูญหายแพคเกตที่รับส่งในเครือข่ายขณะทำการติดต่อสื่อสารกันของโพรโทคอล DSR ในเครือข่ายเฉพาะกิจ

2. เพื่อศึกษาหาวิธีการเพิ่มสมรรถนะของโพรโทคอล DSR ในเครือข่ายเฉพาะกิจ

ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถลดจำนวนแพคเกตที่ต้องสูญหายไปในเครือข่ายเฉพาะกิจระหว่างที่โหนดทำการติดต่อสื่อสารกัน โดยใช้โพรโทคอล DSR ในการค้นหาเส้นทาง

2. สามารถนำไปใช้เป็นแนวทางในงานวิจัยและพัฒนา ตลอดจนนำไปประยุกต์ใช้ในการติดต่อสื่อสารไร้สายได้อย่างมีประสิทธิภาพ

3. ผู้วิจัยมีความเข้าใจในระบบการสื่อสารไร้สาย มีประสบการณ์เกี่ยวกับการทำวิจัยและเข้าใจถึงวิธีการพัฒนาและวิธีการวัดประสิทธิภาพการทำงานของโพรโทคอลการค้นหาเส้นทางในเครือข่ายไร้สายเฉพาะกิจ

ขอบเขตของงานวิจัย

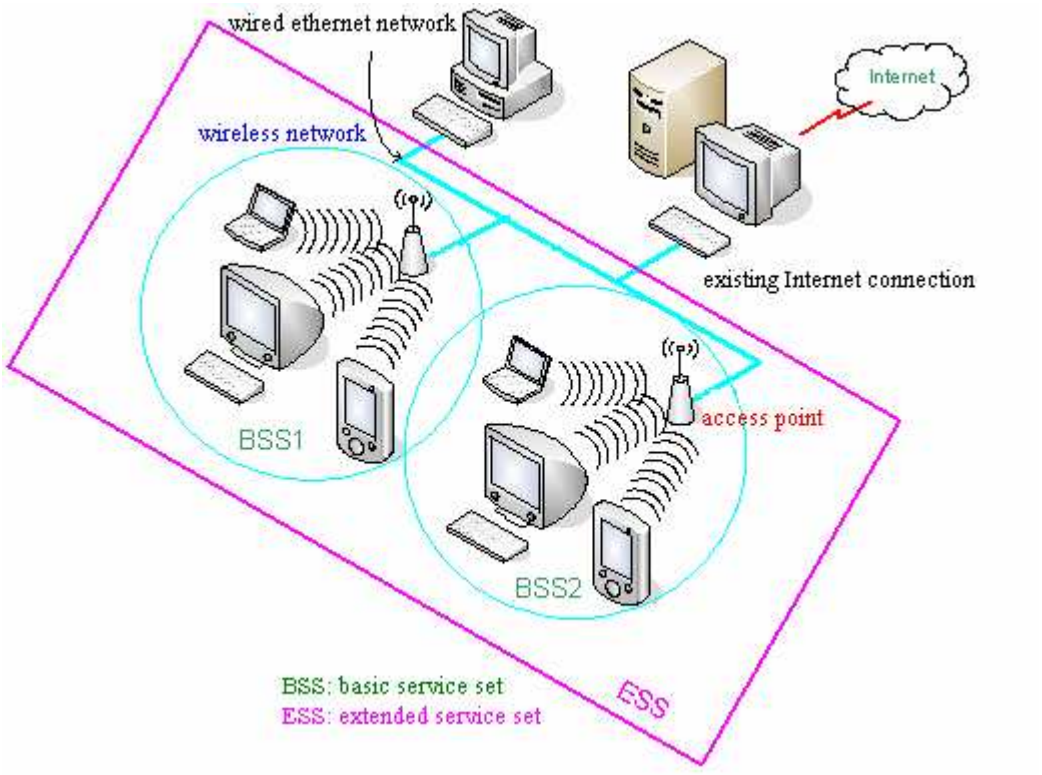
1. เปรียบเทียบจำนวนการสูญหายแพคเกตข้อมูลในขณะที่โหนดทำการติดต่อสื่อสารกันในเครือข่ายเฉพาะกิจระหว่างโพรโทคอล DSR (dynamic source routing) และ DSR-BA (dynamic source routing with break avoidance)

2. สร้างแบบจำลองเครือข่ายและทดสอบการทำงานในเครือข่ายเฉพาะกิจโดยใช้โปรแกรมจำลองการทำงาน GlomoSim เปรียบเทียบโพรโทคอล DSR และโพรโทคอล DSR-BA

การตรวจเอกสาร

ความรู้พื้นฐานเกี่ยวกับเครือข่ายเฉพาะกิจ

ความรู้พื้นฐานเกี่ยวกับเครือข่ายเฉพาะกิจ (Chlamtac *et al*., 2003; Royer and Toh, 1999; Perkins, 2001) นั้น จำเป็นอย่างยิ่งที่จะต้องทราบถึงลักษณะโครงสร้างของเครือข่ายไร้สาย โดยเครือข่ายไร้สายรูปแบบที่มีใช้งานกันอยู่นั้นประกอบด้วยอุปกรณ์ไร้สายหรืออุปกรณ์เคลื่อนที่ที่มีลักษณะการเชื่อมต่อสื่อสารกันโดยอาศัยจุดเชื่อมต่อการเข้าถึงเครือข่าย (access point) ซึ่งมีรูปแบบการเชื่อมต่อกันดังภาพที่ 1 ระบบการเชื่อมต่อสื่อสารที่ต้องมีสถานีฐานในการให้บริการในรัศมีที่จำกัด (infrastructure network)



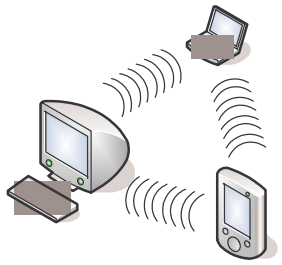
ภาพที่ 1 ระบบการเชื่อมต่อสื่อสารที่ต้องมีสถานีฐานในการให้บริการในรัศมีที่จำกัด

ภาพที่ 1 BSS (basic service set) (ศิวรักษ์, 2546) หมายถึงบริเวณของเครือข่าย IEEE 802.11 WLAN ที่มีสถานีแม่ข่าย (access point) 1 สถานี ซึ่งสถานีผู้ใช้ภายในขอบเขตของ BSS นี้ทุกสถานีจะต้องสื่อสารข้อมูลผ่านสถานีแม่ข่ายเท่านั้น ตัวอย่างของสถานีผู้ใช้ภายใน BSS1 และ

BSS2 เช่น เครื่องคอมพิวเตอร์ส่วนบุคคล เครื่องช่วยงานส่วนบุคคลแบบดิจิทัล พาล์มท็อป คอมพิวเตอร์ โทรศัพท์เคลื่อนที่ เครื่องโน้ตบุ๊กคอมพิวเตอร์ เป็นต้น

ESS (extended service set) (ศิวรักษ์, 2546) หมายถึงบริเวณของเครือข่าย IEEE 802.11 WLAN ที่ประกอบด้วย BSS มากกว่าหนึ่งที่เชื่อมต่อเข้าด้วยกัน โดยสถานีผู้ใช้จาก BSS1 สามารถเคลื่อนย้ายจาก BSS1 ไปอยู่ใน BSS2 ได้โดย BSS เหล่านี้จะทำการติดต่อสื่อสารกัน (roaming) เพื่อทำการโอนย้ายการให้บริการสำหรับสถานีผู้ใช้

สถานีแม่ข่ายของแต่ละ BSS จะเชื่อมต่อกันเป็นเครือข่ายผ่านทางสายสัญญาณ (wire ethernet network) โดยเมื่อสถานีผู้ใช้บริการเครือข่ายภายใน BSS ต้องการรับส่งข้อมูลกับสถานีผู้ใช้จากเครือข่ายอื่น สถานีผู้ใช้นี้สามารถรับส่งข้อมูลได้โดยตรงกับสถานีแม่ข่ายที่ให้บริการแก่สถานีผู้ใช้ภายใน BSS จากนั้นสถานีแม่ข่ายของ BSS จะทำหน้าที่ส่งต่อ (forward) ข้อมูลที่ได้รับจากสถานีผู้ใช้ไปยังจุดหมายปลายทางหรือส่งต่อข้อมูลที่ได้รับจากเครือข่ายอื่นมายังสถานีผู้ใช้



ภาพที่ 2 เครือข่ายเฉพาะกิจที่มีอุปกรณ์ทุกตัวเชื่อมต่อถึงกันได้โดยตรง

สำหรับเครือข่ายเฉพาะกิจมีลักษณะการเชื่อมต่อของเครือข่ายแสดงดังภาพที่ 2 เครือข่ายเฉพาะกิจที่มีอุปกรณ์ทุกตัวเชื่อมต่อถึงกันได้โดยตรง ซึ่งการเชื่อมต่อระหว่างสถานีผู้ใช้ภายในเครือข่ายนั้น สามารถติดต่อสารสื่อกันได้โดยเชื่อมต่อถึงกันโดยตรง ดังนั้นเมื่อสถานีผู้ใช้มีความต้องการรับส่งข้อมูลกันสามารถกระทำได้โดยไม่ต้องส่งผ่านสถานีแม่ข่าย

ลักษณะของเครือข่ายเฉพาะกิจนั้น สถานีผู้ใช้ภายในเครือข่ายจะมีรูปแบบการเชื่อมต่อสื่อสารที่มีการเปลี่ยนแปลงตลอดเวลา ซึ่งเครือข่ายสามารถจัดการทุกอย่างได้เองและเป็นไปอย่างอัตโนมัติ โดยการเชื่อมต่อกันของสถานีผู้ใช้นั้นเป็นแบบไร้สาย (wireless) และไม่จำเป็นต้องใช้โครงสร้างพื้นฐาน (infrastructureless) อีกทั้งเครือข่ายไม่จำเป็นต้องมีศูนย์กลางควบคุมการทำงาน

(non-centralized administration) โดยเครือข่ายสามารถทำงานได้โดยลำพัง และสามารถเชื่อมโยงเข้าสู่เครือข่ายอินเทอร์เน็ต (Internet) ขนาดใหญ่ได้ นอกจากนี้ ลักษณะการเคลื่อนที่ของอุปกรณ์เคลื่อนที่นั้นเป็นไปอย่างอิสระโดยมีการเคลื่อนที่แบบสุ่ม (randomly) ในเครือข่าย ด้วยลักษณะการเคลื่อนที่แบบนี้จึงส่งผลให้โทโปโลยีเปลี่ยนแปลงได้อย่างรวดเร็วและไม่สามารถทำนายการเคลื่อนที่ของอุปกรณ์เคลื่อนที่ได้ ในการติดต่อสื่อสารระหว่างอุปกรณ์เคลื่อนที่นั้น เส้นทางที่ใช้อาจเป็นแบบหลายฮอพ (multiple hops) ซึ่งจำเป็นต้องอาศัยโหนดตัวกลาง (intermediate nodes) ในการส่งต่อข้อมูลแบบฮอพต่อฮอพ (hop-by-hop) ดังนั้นเครือข่ายลักษณะนี้จึงต้องมีความยืดหยุ่นและสามารถใช้งานได้สะดวกอีกด้วย โดยได้มีการนำไปประยุกต์ใช้งานในหลายๆ ด้านทั้งทางด้านการทหาร การช่วยเหลืออย่างฉุกเฉิน การใช้ในห้องเรียนและการแลกเปลี่ยนข้อมูลในการประชุม รวมถึงการใช้งานด้านอุปกรณ์ตรวจจับต่างๆ เป็นต้น

เครือข่ายเฉพาะกิจไร้สายได้รับการสืบทอดปัญหามาจากการสื่อสารไร้สายและเครือข่ายไร้สายที่มีอยู่เดิม โดยปัญหาดังกล่าว ได้แก่ ปัญหาของโหนดตัวกลางที่ไม่สมบูรณ์ โหนดตัวกลางอาจอยู่นอกขอบเขตการรับแพคเกตทำให้ไม่สามารถรับแพคเกตได้ ปัญหาตัวกลางไร้สายมีความน่าเชื่อถือน้อยกว่าตัวกลางใช้สาย ปัญหาช่องสัญญาณไม่สามารถป้องกันสัญญาณจากภายนอกได้ และช่องสัญญาณมีการเปลี่ยนแปลงตามเวลา ปัญหาเรื่องคุณสมบัติของการแพร่กระจายคลื่นที่ไม่สมมาตร เป็นต้น ซึ่งปัญหาต่างๆ ดังกล่าวยังคงต้องแก้ปัญหาต่อไปและพิจารณาเลือกใช้งานเครือข่ายในรูปแบบต่างๆ ตามความเหมาะสม

<u>โพรโทคอลการค้นหาเส้นทางที่ใช้กับเครือข่ายเฉพาะกิจ</u>

โพรโทคอลการค้นหาเส้นทางเครือข่ายที่ใช้สายไม่เหมาะที่จะนำมาใช้กับเครือข่ายการเคลื่อนที่เฉพาะกิจ (MANETs: mobile ad hoc networks) เนื่องจากเครือข่ายเฉพาะกิจเป็นเครือข่ายที่โหนดมีการเคลื่อนที่นั่นเอง

ในงานวิจัยนี้นำเสนอเพียงโพรโทคอลการค้นหาเส้นทางที่ใช้กับเครือข่ายเฉพาะกิจ (Royer and Toh, 1999; Chlamtac *et al*., 2003; Johnson, 2003) ที่เกี่ยวข้องกับงานวิจัย ได้แก่

1. <u>โพรโทคอลแบบ Pro-active</u>

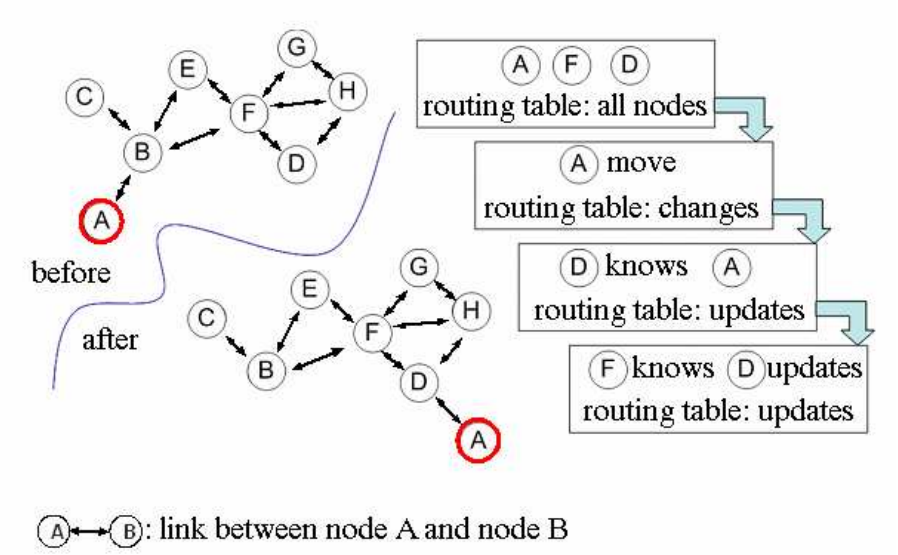
มีวิธีการค้นหาเส้นทางโดยทำการกระจายแพคเกตค้นหาเส้นทางไปยังทุกโหนดที่อยู่ใกล้เคียง จากนั้นแต่ละโหนดทำการเก็บข้อมูลเส้นทางไว้ในตารางเส้นทาง ซึ่งข้อมูลในตารางเส้นทางมีไว้เพื่อใช้ในการค้นหาเส้นทางระหว่างโหนดเริ่มต้นไปยังโหนดปลายทาง โดยข้อมูลในตารางเส้นทางต้องปรับปรุงข้อมูลในตารางเป็นระยะๆ เมื่อโหนดเริ่มต้นต้องการเส้นทางการเชื่อมต่อไปยังโหนดปลายทาง โหนดเริ่มต้นจะทราบเส้นทางการเชื่อมไปยังโหนดปลายทางได้ทันที และสามารถใช้เส้นทางนี้ในการติดต่อสื่อสารระหว่างโหนดเริ่มต้นและปลายทางโดยข้อมูลที่ใช้ในการเปลี่ยนแปลงตารางเส้นทางของแต่ละโหนดนั้น ได้มาจากข้อมูลของโหนดเพื่อนบ้านที่มีการเปลี่ยนแปลง

1.1 โพรโทคอล DSDV (destination-sequenced distance vector routing) (Perkins and Bhagwat, 1994) เป็นโพรโทคอลสำหรับการค้นหาเส้นทางที่มีรากฐานมาจากแนวความคิดของโพรโทคอล Bellman-Ford (Cheng *et al*., 1989) โดยพัฒนาวิธีการจัดเก็บเส้นทางที่ได้ลงในตารางเส้นทางเพื่อไม่ให้มีการวนซ้ำวงรอบเดิมและให้แต่ละโหนดมีการเก็บข้อมูลตารางเส้นทางไว้ด้วย โพรโทคอลนี้จะทำการค้นหาเส้นทางตามระยะทางโดยเก็บข้อมูลระยะทางไว้ในตารางเส้นทาง ซึ่งข้อมูลในตารางเส้นทางได้มาจากการกระจายแพคเกตการค้นหาไปทั้งเครือข่าย และเก็บจำนวนฮอพที่ไปแต่ละโหนดปลายทาง โดยแต่ละเส้นทางกำหนดเลขลำดับไว้ด้วยโหนดปลายทาง ทั้งนี้ใช้เพื่อหลีกเลี่ยงการเกิดเส้นทางที่วนซ้ำวงรอบของโพรโทคอล DSDV ซึ่งให้การรับประกันในเรื่องของการวนซ้ำวงรอบและเลือกใช้เส้นทางที่สั้นที่สุดเพื่อการเชื่อมต่อสื่อสาร

ในการกระจายแพคเกตให้ทั่วทุกโหนดเพื่อทำการปรับเปลี่ยนข้อมูลในตารางเส้นทางนั้น ทำให้เกิดความคับคั่งของแพคเกตในเครือข่าย ซึ่งมีวิธีในการลดความคับคั่ง 2 วิธีคือ การส่งเส้นทางที่มีการปรับเปลี่ยนของทุกๆ โหนดหรือการส่งเส้นทางที่มีการปรับเปลี่ยนเฉพาะของโหนดที่มีการเปลี่ยนแปลงเท่านั้น จากการที่เส้นทางมีการปรับเปลี่ยนบ่อยครั้งส่งผลทำให้ข้อมูลเส้นทางในตารางมีการเปลี่ยนแปลงได้โดยง่าย และการกระทำเช่นนี้จะไม่สนใจด้วยว่าโหนดเริ่มต้นต้องการเส้นทางที่ปรับเปลี่ยนใหม่นั้นหรือไม่

ภาพที่ 3 แสดงถึงการเปลี่ยนแปลงข้อมูลเส้นทางของโพรโทคอล DSDV ในเครือข่ายเฉพาะกิจ โดยในขณะเริ่มต้นโหนด A, B, C, D, E, F, G และ H วางตัวอยู่ในตำแหน่งต่างๆ ภายใน

เครือข่ายดังรูปก่อนการเปลี่ยนแปลง (before) ซึ่งทุกโหนดจะมีข้อมูลเส้นทางไปยังโหนดต่างๆ อยู่ในตารางเส้นทางของตนเอง และรูปหลังการเปลี่ยนแปลง (after) ซึ่งโหนดในเครือข่ายจะต้องทำการปรับเปลี่ยนข้อมูลในตารางเส้นทางของตนเองตามการเปลี่ยนแปลงที่เกิดขึ้น โดยได้แสดงตัวอย่างตารางเส้นทางของโหนด F และโหนด D ดังในภาพที่ 4



ภาพที่ 3 การทำงานของโพรโทคอล DSDV ในเครือข่ายเฉพาะกิจ

routing table of node D before node A moves

destination	next hop	metric	sequence number
A	F	3	S516_A
B	F	2	S228_B
C	F	3	S764_C
D	D	0	S820_D
E	F	2	S502_E
F	F	1	S204_F
G	F	2	S238_G
H	H	1	S148_H

routing table of node F before node A moves

destination	next hop	metric	sequence number
A	B	2	S406_A
B	B	1	S238_B
C	E	2	S764_C
D	D	1	S820_D
E	E	1	S502_E
F	F	0	S204_F
G	G	1	S238_G
H	H	1	S160_H

routing table of node D after node A moves

destination	next hop	metric	sequence number
A	A	1	S525_A
B	F	2	S228_B
C	F	3	S764_C
D	D	0	S820_D
E	F	2	S502_E
F	F	1	S204_F
G	F	2	S238_G
H	H	1	S148_H

routing table of node F after node A moves

destination	next hop	metric	sequence number
A	D	2	S516_A
B	B	1	S238_B
C	E	2	S764_C
D	D	1	S820_D
E	E	1	S502_E
F	F	0	S204_F
G	G	1	S238_G
H	H	1	S160_H

ภาพที่ 4 ตัวอย่างตารางเส้นทางของโหนด D และโหนด F

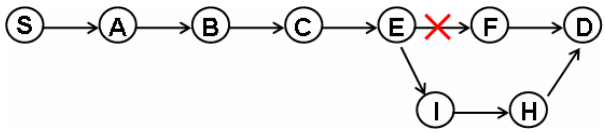
ภาพที่ 4 สดมภ์ (column) ของ destination และ next hop แสดงโหนดเคลื่อนที่ในเครือข่ายโดย A หมายถึงโหนด A สดมภ์ของ metric แสดงด้วยสัญลักษณ์หมายเลขหมายถึงจำนวนฮอพไปถึงโหนดปลายทางนั้นๆ และสดมภ์ของ sequence number แสดงสัญลักษณ์ด้วยเลขลำดับที่กำหนดขึ้นจากโหนดปลายทาง

จากภาพที่ 3 และภาพที่ 4 เมื่อโหนด A เคลื่อนที่จากตำแหน่งเดิมไปอยู่ใกล้กับโหนด D ทำให้ข้อมูลเส้นทางของโหนด A เปลี่ยนแปลงไป ดังนั้นโหนดในเครือข่ายจึงต้องทำการปรับเปลี่ยนข้อมูลเส้นทางไปยังโหนดต่างๆ ในตารางเส้นทางของตนเอง โดยที่โหนด D รับทราบตำแหน่งของโหนด A ได้เคลื่อนที่มาใกล้ โหนด D จึงทำการปรับเปลี่ยนข้อมูลเส้นทางโดยจากเดิมสามารถติดต่อกับโหนด A ได้ด้วยระยะ 3 ฮอพโดยมี next node คือโหนด F เปลี่ยนไปเป็นโหนด A ด้วยระยะ 1 ฮอพ โหนด F อยู่ใกล้กับโหนด D และรับทราบการเปลี่ยนแปลงจากโหนด D ดังนั้นโหนด F จึงต้องทำการปรับเปลี่ยนข้อมูลเส้นทางในตารางเส้นทางของตนเองด้วยโดยจากเดิมสามารถติดต่อกับโหนด A ได้ด้วยระยะ 2 ฮอพ มี next node คือโหนด B เปลี่ยนไปเป็นโหนด D ด้วยระยะ 2 ฮอพ ซึ่งตัวอย่างตารางเส้นทางที่มีการปรับเปลี่ยนข้อมูลได้แสดงไว้ดังภาพที่ 4 ในบริเวณกรอบหนาได้แสดงให้เห็นถึงการเปลี่ยนแปลงข้อมูลในตารางเส้นทางของโหนด D และโหนด F ทั้งก่อนและหลังจากการที่โหนด A เคลื่อนที่ไปจากตำแหน่งเดิม จากการที่โหนด A ได้เคลื่อนที่ไปจากตำแหน่งเดิมนั้นจะส่งผลให้โหนดอื่นๆ ในเครือข่ายต้องทำการปรับเปลี่ยนข้อมูลเส้นทางในตารางเส้นทางของตนเองด้วย ซึ่งถ้า next node ยังเป็นโหนดเดิม โหนดที่ขาดการติดต่อจะต้องรอการปรับเปลี่ยนข้อมูลเส้นทางในตารางเส้นทางก่อน จากนั้นจึงใช้ next node ในเส้นทางที่สามารถเชื่อมต่อไปยังโหนดปลายทางได้

ถ้าหากในขณะที่มีโหนดใดๆ ทำการติดต่อสื่อสารกันอยู่ เมื่อบางโหนดในเครือข่ายมีการเคลื่อนที่ไปจากตำแหน่งเดิมทำให้เส้นทางขาดการติดต่อ โหนดที่ขาดการติดต่อนั้นจะทำการค้นหา next node เพื่อทำการเชื่อมต่อโดยดูจากตารางเส้นทางของตนเอง

ภาพที่ 5 แสดงถึงเหตุการณ์ที่โหนด S ได้ทำการติดต่อสื่อสารกับโหนด D โดยใช้เส้นทาง (D: S, A, B, C, E, F, D) แต่เมื่อสื่อสารได้ระยะหนึ่งโหนด E ขาดการเชื่อมต่อกับโหนด F โหนด E จะหาโหนดเชื่อมต่อเส้นทางโหนดใหม่โดยดูจากตารางเส้นทางของตนเอง จากเดิมโหนด E สามารถติดต่อกับโหนด D ได้ด้วยระยะ 2 ฮอพ มี next node คือโหนด F แต่เส้นทางนี้ขาดการเชื่อมต่อเกิดการสูญหายแพคเกต ดังนั้นโหนด E จะต้องรอให้มีการปรับเปลี่ยนข้อมูลในตาราง

เส้นทางก่อน ซึ่งได้ next node คือโหนด I โหนด E จึงใช้โหนดใหม่นี้ในการส่งแพคเกตด้วยระยะ 3 ฮอพ โดยค่าเลขลำดับที่เปลี่ยนไปนั้นใช้เพื่อป้องกันการวนซ้ำของเส้นทางที่ใช้ในการส่งแพคเกตในเครือข่าย



routing table of E: before update

destination	next hop	metric	sequence number
D	F	2	S516_D

routing table of E: after update

destination	next hop	metric	sequence number
D	I	3	S520_D

ภาพที่ 5 ตารางเส้นทางของโหนด E เมื่อโหนด S ทำการติดต่อสื่อสารกับโหนด D จากนั้นโหนด E ขาดการเชื่อมต่อกับโหนด F

ข้อได้เปรียบของโพรโทคอล DSDV คือเมื่อโหนดใดๆ ต้องการเส้นทางสามารถทราบเส้นทางไปใช้งานได้ทันทีจากตารางเส้นทางที่แต่ละโหนดมีการเก็บบันทึกไว้ เหมาะกับเครือข่ายที่มีขนาดเล็กทำให้สามารถส่งข้อมูลได้รวดเร็ว มีการรับประกันในเรื่องของเส้นทางการวนซ้ำไม่ให้เกิดเหตุการณ์การวนซ้ำของเส้นทางการส่งแพคเกตและทราบเส้นทางที่สั้นที่สุดด้วย แต่ก็ยังมีข้อเสียเปรียบคือจากการที่ทุกโหนดในเครือข่ายต้องทำการเก็บข้อมูลเส้นทางไว้ที่ตารางเส้นทางของตัวเอง ทำให้สิ้นเปลืองพื้นที่ในหน่วยความจำซึ่งแปรผันตามจำนวนโหนดในเครือข่ายและมีข้อมูลเส้นทางที่เก็บบันทึกไว้ในหลายๆ เส้นทางที่โหนดไม่ได้นำมาใช้งาน นอกจากนี้ในกรณีที่ถ้าเครือข่ายไม่ค่อยมีการเปลี่ยนแปลงมากนัก นั่นหมายถึงว่าโหนดบางโหนดไม่จำเป็นต้องมีการเปลี่ยนแปลงข้อมูลเส้นทางในตารางเส้นทางของโหนด แต่เนื่องจากโพรโทคอลนี้จะต้องมีการปรับเปลี่ยนข้อมูลเส้นทางเป็นระยะหรือเป็นรายคาบเวลา ทำให้เกิดการสิ้นเปลืองทรัพยากรของเครือข่ายไปโดยไม่จำเป็น

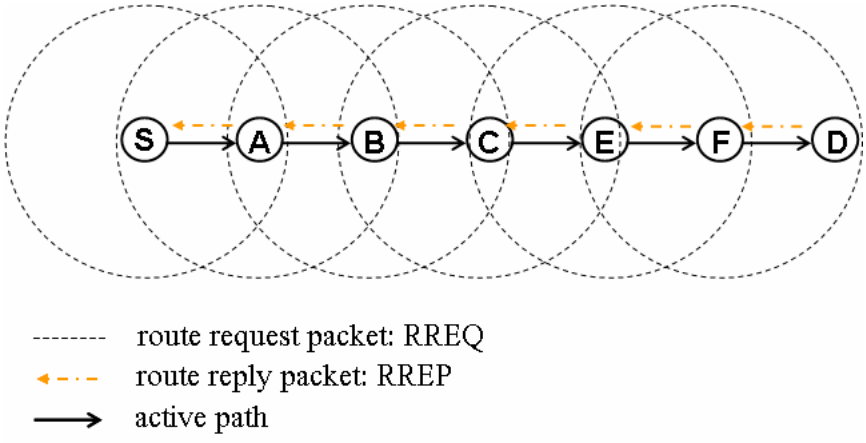
2. <u>โพรโทคอลแบบ Reactive</u>

ลักษณะของโพรโทคอลแบบนี้ แต่ละโหนดจะไม่ทราบข้อมูลเส้นทางการเชื่อมต่อกับโหนดปลายทางล่วงหน้า ก่อนที่จะมีความต้องการใช้เส้นทางนั้น โดยโหนดที่ต้องการติดต่อสื่อสารจะทำการค้นหาเส้นทางก็ต่อเมื่อโหนดนั้นๆ มีการร้องขอ ทำให้โหนดไม่จำเป็นต้องเก็บข้อมูลเส้นทางไว้ตลอดเวลา โดยข้อมูลเส้นทางจะมีการค้นหาก็ต่อเมื่อผู้ส่งหรือโหนดเริ่มต้น (source node) ต้องการใช้เส้นทางเชื่อมต่อหรือเมื่อมีการร้องขอเส้นทางเชื่อมต่อจากโหนดเริ่มต้นไปโหนดปลายทาง (destination node) เท่านั้นซึ่งโพรโทคอลนี้ใช้แพคเกต 2 ชุดในการค้นหาเส้นทางคือ แพคเกตร้องขอเส้นทาง (route request packet: RREQ) เป็นแพคเกตที่โหนดเริ่มต้นใช้ในการค้นหาเส้นทางในเครือข่าย และแพคเกตตอบกลับเส้นทาง (route reply packet: RREP) เป็นแพคเกตที่ใช้ตอบกลับมาจากโหนดปลายทาง และมีแพคเกตแจ้งความผิดพลาดของเส้นทาง (route error packet: RERR) ที่ใช้สำหรับการดูแลรักษา (maintainance) เส้นทางเมื่อเส้นทางขาดการเชื่อมต่อ

2.1 โพรโทคอล AODV (ad hoc on-demand distance vector routing) (Perkins *et al*., 2002) มีการปรับปรุงวิธีการมาจากโพรโทคอล DSDV โดยโพรโทคอล AODV ทำการพัฒนาโพรโทคอล DSDV ให้สามารถลดจำนวนของการแพร่กระจายแพคเกตต่างๆ ที่ใช้ไปสำหรับการดูแลรักษาเส้นทางโดยการนำเสนอวิธีการค้นหาเส้นทางและการเก็บตารางข้อมูลเส้นทางก็ต่อเมื่อโหนดเริ่มต้นต้องการใช้เส้นทางเพื่อการเชื่อมต่อหรือเมื่อมีการร้องขอเกิดขึ้นเท่านั้น ซึ่งจะแตกต่างจากโพรโทคอล DSDV ที่แต่ละโหนดมีข้อมูลเส้นทางที่พร้อมใช้งานเก็บอยู่ในตารางข้อมูลเส้นทางอยู่ก่อนแล้ว และทั้งสองโพรโทคอลยังมีความเหมือนกันในเรื่องของการรับประกันเส้นทางไม่ให้เกิดเส้นทางการวนซ้ำในขณะที่โหนดทำการติดต่อสื่อสารกัน

โพรโทคอล AODV มีขั้นตอนการทำงานคือโหนดเริ่มต้นจะส่ง RREQ ไปยังโหนดปลายทาง เมื่อโหนดใดๆ ที่ไม่ใช่โหนดปลายทางได้รับแพคเกตร้องขอนี้จะทำการส่งต่อแพคเกตออกไปจนกระทั่งถึงโหนดปลายทาง เมื่อโหนดปลายทางได้รับแพคเกตร้องขอนี้ โหนดจะทำการตอบกลับเส้นทางด้วยแพคเกตการตอบสนองเส้นทางย้อนกลับไปยังโหนดเริ่มต้นผ่านโหนดตัวกลาง (intermediate node) ต่างๆ จนแพคเกตนี้ส่งถึงโหนดเริ่มต้น และโหนดจะใช้เส้นทางนี้ในการติดต่อสื่อสารกัน จนกระทั่งเมื่อโหนดใดๆ ขาดการติดต่อสื่อสารกันทำให้แพคเกตข้อมูลไม่สามารถส่งต่อไปได้ โหนดที่พบปัญหาจะทำการดูแลรักษาเส้นทางด้วยการค้นหาเส้นทางใหม่เมื่อ

เหตุการณ์เกิดใกล้กับโหนดปลายทางและถ้าเหตุการณ์เกิดใกล้กับโหนดเริ่มต้น โหนดที่พบปัญหาจะส่ง RERR กลับไปยังโหนดเริ่มต้นเพื่อให้โหนดเริ่มต้นค้นหาเส้นทางใหม่อีกครั้ง

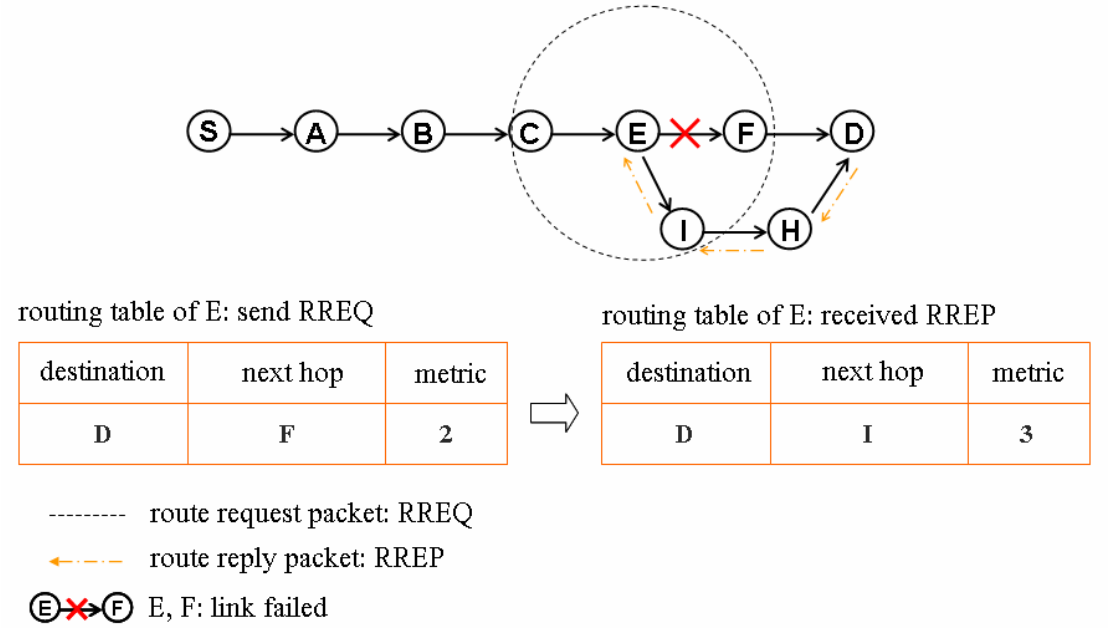


ภาพที่ 6 โพรโทคอล AODV เมื่อค้นหาเส้นทางจากโหนด S ไปยังโหนด D

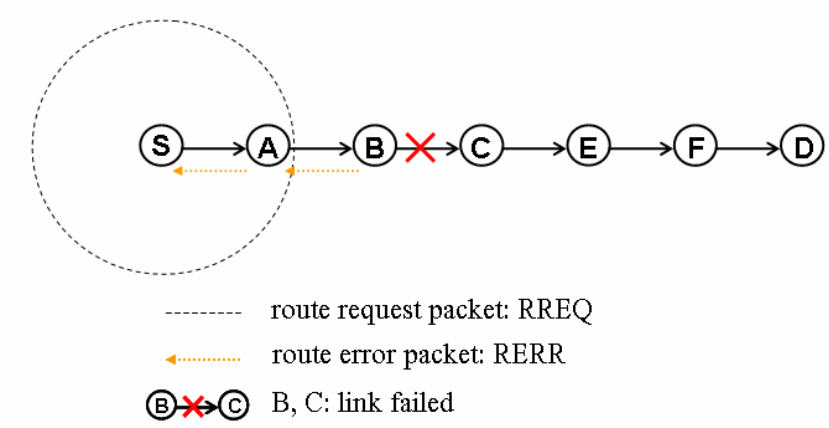
ภาพที่ 6 แสดงถึงการเริ่มต้นค้นหาเส้นทางจากโหนด S ไปยังโหนด D ในที่นี้โหนดเริ่มต้นคือโหนด S ทำการแพร่กระจาย RREQ ออกไปโดยโหนด S มีรัศมีการส่ง RREQ ไปยังโหนด A หลังจากโหนด A ได้รับ RREQ จากโหนด S โหนด A ตรวจเช็คว่า RREQ นั้นมีโหนดปลายทางคือโหนด A หรือไม่ ในที่นี้โหนดปลายทางคือโหนด D เมื่อโหนด A ตรวจสอบแล้วพบว่าโหนด A ไม่ใช่โหนดปลายทางของ RREQ นี้ โหนด A ทำการส่ง RREQ นี้ต่อไปยังโหนดต่างๆ ที่อยู่ในรัศมีการส่งของโหนด A นั่นคือโหนด B ขณะเดียวกันโหนด A จะสร้างเส้นทางย้อนกลับ (reverse path) ไปยังโหนดที่ส่ง RREQ มายังโหนด A เมื่อโหนดต่างๆ ในเครือข่ายได้รับ RREQ จะทำเช่นเดียวกันกับโหนด A จน RREQ ส่งไปถึงยังโหนดปลายทาง ในที่นี้ RREQ ถูกส่งผ่านโหนดต่างๆ คือ A, B, C, E และ F

เมื่อ RREQ มาถึงโหนด D ซึ่งเป็นโหนดปลายทาง โหนด D จะส่ง RREP ไปให้โหนด S ย้อนกลับตามเส้นทางที่ RREQ ถูกส่งผ่านมาแล้วนั้นนั่นคือผ่านไปยังโหนด F E C B และ A ซึ่งโหนดตัวกลางทั้งห้าโหนดนี้จะเป็นโหนดที่อยู่ในเส้นทางการส่งของ RREP เมื่อโหนดได้รับ RREP โหนดเหล่านี้จะทำการตรวจสอบโหนดปลายทางของ RREP เป็นโหนดตนเองหรือไม่ เมื่อตรวจพบว่าไม่ใช่ โหนดนั้นๆ จะสร้างเส้นทางไปยังโหนดที่ส่ง RREP (forward path) มาให้และทำการส่งต่อ RREP ไปจนกระทั่งถึงโหนด S ในขณะที่โหนดอื่นๆ จะไม่กระทำการใดๆ กับ RREP นี้

เมื่อโหนด S ได้รับ RREP แล้ว โหนด S จะทราบว่าตนเองสามารถติดต่อสื่อสารกับโหนด D ได้ในระยะ 5 ฮอพโดยผ่าน next node คือโหนด A



(ก)



(ข)

ภาพที่ 7 โพรโทคอล AODV เมื่อเส้นทางเกิดปัญหาขาดการเชื่อมต่อ

(ก) ระหว่างโหนด E และโหนด F

(ข) ระหว่างโหนด B และโหนด C

หลังจากที่โหนด S ได้เส้นทางการติดต่อสื่อสารไปยังโหนด D แล้ว และทำการส่งแพคเกตข้อมูลจำนวนหนึ่งด้วยเส้นทางนี้ หลังจากนั้นเส้นทางดังกล่าวมีบางโหนดได้ขาดการ

เชื่อมต่อกันจึงไม่สามารถส่งแพคเกตต่อไปได้ทำให้เกิดการสูญหายแพคเกตขึ้น ดังตัวอย่างในภาพที่ 7 ซึ่งแสดงถึงเหตุการณ์ของปัญหาที่อาจเกิดขึ้นได้ เช่น เมื่อโหนดตัวกลางหรือโหนดปลายทางมีการเคลื่อนที่ไปจากตำแหน่งเดิมหรือเส้นทางที่ใช้ในการส่งแพคเกตระหว่างที่โหนดยังคงติดต่อสื่อสารกันอยู่นั้นเกิดปัญหาขึ้น ไม่สามารถใช้งานได้ตามปกติดังภาพที่ 7(ก) แสดงปัญหาที่เกิดขึ้นระหว่างโหนด E และโหนด F ซึ่งอยู่ใกล้กับโหนดปลายทาง ในที่นี้โหนด E จะทำการซ่อมแซมเส้นทาง ณ จุดนั้นๆ (local repair) นั่นคือ โหนด E จะทำหน้าที่ในการค้นหาเส้นทางไปยังโหนดปลายทางทดแทนเส้นทางเดิมที่เกิดปัญหา โดยการแพร่กระจาย RREQ ไปยังโหนดเพื่อนบ้านผ่านโหนดตัวกลางอื่นๆจนกระทั่งถึงโหนด D ในที่นี้ได้เส้นทางผ่านโหนด I และ H ซึ่งเมื่อโหนด D ได้รับ RREQ ที่สร้างจากโหนด E โหนด D จะตอบกลับเส้นทางด้วย RREP ผ่านทางโหนด H และ I เมื่อโหนด E ได้รับ RREP นี้แล้ว โหนด E จะทราบว่าสามารถไปยังโหนด D ด้วยเส้นทางใหม่ได้ในระยะ 2 ฮอพโดยผ่าน next node คือโหนด I และเส้นทางนี้จะยังคงใช้ต่อไปจนกระทั่งเสร็จสิ้นการสื่อสารหรือจนกว่าจะพบปัญหาอีก

ภาพที่ 7(ข) แสดงเส้นทางที่มีปัญหาเกิดขึ้นที่โหนด B และโหนด C ซึ่งอยู่ใกล้กับโหนดเริ่มต้น ในกรณีนี้โหนด B จะไม่ทำ local repair แต่จะส่ง RERR กลับไปยังโหนด S ย้อนกลับตามเส้นทางการส่งแพคเกตข้อมูล โดยมีโหนด A อยู่ในเส้นทางการส่ง RERR นี้ เมื่อโหนด A ได้รับ RERR โหนด A จะทราบว่าโหนด B และโหนด C ได้ขาดการเชื่อมกันแล้ว โหนดจะทำการปรับเปลี่ยนข้อมูลเส้นทางในตารางเส้นทางของตนเอง จากนั้นจึงทำการตรวจสอบว่าเป็นโหนดปลายทางของ RERR นี้หรือไม่ ถ้าตรวจพบว่าไม่ใช่ โหนด A จะส่ง RERR ต่อไปจนถึงโหนด S เมื่อโหนด S ได้รับ RERR โหนด S จะทราบว่าโหนด B และโหนด C ได้ขาดการเชื่อมกันแล้วและทำการปรับเปลี่ยนข้อมูลเส้นทางในตารางเส้นทางของตนเอง จากนั้นจึงแพร่กระจาย RREQ เพื่อค้นหาเส้นทางไปยังโหนด D อีกครั้ง แล้วใช้เส้นทางใหม่ที่จะได้มาทดแทนเส้นทางเดิม

ในกระบวนการค้นหาเส้นทางของโพรโทคอล AODV นั้น หลังจากที่โหนดเริ่มต้นกระจาย RREQ ออกไปแล้ว โหนดต่างๆ ที่อยู่ในเครือข่าย เมื่อได้รับ RREQ นี้ จะสามารถตอบกลับ RREQ ได้ก็ต่อเมื่อเป็นโหนดปลายทางเท่านั้นและส่ง RREP ย้อนกลับตามเส้นทางของ RREQ ซึ่งกระบวนการนี้โหนดเริ่มต้นจะได้เส้นทางการตอบกลับเป็นแบบสมมาตร (symmatric link) เพื่อใช้ในการส่งแพคเกตข้อมูล

ถ้าเกิดเหตุการณ์ที่ RREQ สูญหายไประหว่างทางนั่นคือโหนดเริ่มต้นไม่ได้รับ RREP ในช่วงเวลาที่กำหนด โหนดเริ่มต้นจะต้องทำการแพร่กระจาย RREQ อีกครั้งซึ่งในครั้งนี้โหนดจะเพิ่มระยะเวลาในการรอ RREP ที่ตอบกลับจากโหนดปลายทางให้นานขึ้น โดยในการแพร่กระจาย RREQ แต่ละครั้งนั้น มีการกำหนดขอบเขตของการแพร่กระจายไว้เรียกว่า TTL (time to live) เมื่อโหนดใดได้รับ RREQ ที่ TTL มีค่าถึงขอบเขตของการแพร่กระจายนี้ให้หยุดการแพร่กระจาย RREQ ต่อไปด้วยกระบวนการนี้ทำให้สามารถควบคุม RREQ ที่อยู่ในเครือข่ายได้

ลักษณะการจัดเก็บข้อมูลเส้นทางที่ใช้ในการเชื่อมต่อไปยังโหนดต่างๆ ภายในเครือข่ายของโพรโทคอล AODV นั้น จะจัดเก็บข้อมูลไว้ในตารางเส้นทางของแต่ละโหนด โดยในตารางเส้นทางจะมีข้อมูลคือ โหนดปลายทาง จำนวนฮอพที่ใช้เพื่อติดต่อกับโหนดปลายทาง และ next node ที่ใช้เพื่อส่งต่อแพคเกตข้อมูล ซึ่งถ้าโหนดเริ่มต้นมีการเคลื่อนที่ไปจากตำแหน่งเดิม โหนดเริ่มต้นสามารถเริ่มค้นหาเส้นทางใหม่อีกครั้งได้เรียกว่า reinitiate route discovery และเส้นทางใหม่นี้จะนำไปเปลี่ยนแปลงในตารางเส้นทางของแต่ละโหนด

ข้อได้เปรียบของโพรโทคอล AODV คือ สามารถใช้แบนด์วิธได้อย่างมีประสิทธิภาพ ไม่ต้องมีการปรับปรุงข้อมูลเส้นทางในตารางเส้นทางบ่อยๆ เพราะจะปรับปรุงเฉพาะช่วงที่ใช้งานและมีการเปลี่ยนแปลงของโหนดที่ใช้สื่อสารกันอยู่เท่านั้น ซึ่งข้อมูลเส้นทางจะไม่ได้เก็บข้อมูลเส้นทางเอาไว้ตลอดเวลาทำให้ปริมาณของแพคเกตที่ใช้ในการปรับปรุงข้อมูลภายในเครือข่ายน้อยกว่าโพรโทคอล DSDV นอกจากนี้โพรโทคอล AODV สามารถตอบสนองต่อการเปลี่ยนแปลงของโทโปโลยีได้ดี และยังสามารถขยายขนาดของเครือข่ายได้เพิ่มขึ้น อีกทั้งยังคงคุณสมบัติการให้การรับประกันในเรื่องของเส้นทางการวนซ้ำเช่นเดียวกับโพรโทคอล DSDV และสามารถควบคุมจำนวนของการแพร่กระจายของแพคเกตควบคุมทั้งหมดที่ถูกส่งต่อไปในทั้งเครือข่ายได้ด้วยค่า TTL แต่อย่างไรก็ตามโพรโทคอล AODV ยังคงมีข้อจำกัดคือ ถ้าเครือข่ายมีขนาดใหญ่ขึ้นมากๆ จะทำให้โพรโทคอลเสียเวลาในการสร้างเส้นทางการติดต่อสื่อสารเพิ่มขึ้น ใช้เวลาในการค้นหาเส้นทางนานมากขึ้น ทำให้ในบางครั้งอาจไม่พบเส้นทางการติดต่อสื่อสารไปยังโหนดที่ต้องการได้ เพราะการที่ RREQ มีค่า TTL กำกับไว้นั้นเมื่อโหนดปลายทางอยู่ห่างไกลมากๆ อาจทำให้ RREQ หมดอายุไปก่อน จึงทำให้ค้นหาโหนดปลายทางไม่เจอได้ นอกจากนี้ยังคงมีการสูญหายแพคเกตข้อมูลที่สูงกว่าโพรโทคอล DSDV ทั้งนี้เพราะการที่โหนดมีการเปลี่ยนแปลงบ่อยๆ ทำให้โหนดขาดการเชื่อมต่อบ่อยครั้ง แพคเกตข้อมูลส่งไปไม่ได้จึงมีการสูญหายเพิ่มขึ้น ขณะเดียวกันโหนดต้อง

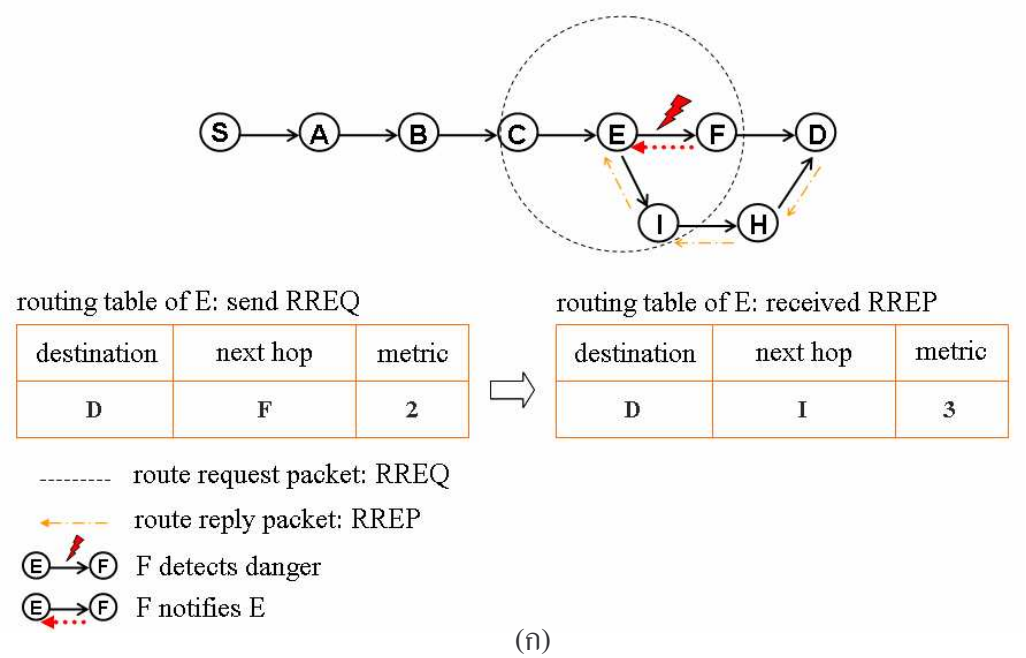
เสียเวลาหาเส้นทางใหม่มาทดแทนเพื่อใช้ในการส่งต่อแพคเกต ทำให้ช่วงเวลาที่เสียไปนั้นไม่สามารถส่งแพคเกตข้อมูลไปได้ ส่งผลถึงแพคเกตข้อมูลมีการหน่วงของเวลาเพิ่มขึ้นด้วย

2.2 โพรโทคอล AODV-BA (AODV with break avoidance) (Tauchi *et al*., 2005) ได้ทำการปรับปรุงโพรโทคอล AODV จากเดิมที่โหนดรอให้เกิดการขาดการเชื่อมไปก่อนแล้วจึงหาเส้นทางใหม่มาใช้นั้น ทำให้มีปัญหาการสูญหายแพคเกตเกิดขึ้นจำนวนมาก งานวิจัยนี้จึงได้นำเสนอวิธีการลดการสูญหายแพคเกตที่เกิดขึ้นดังกล่าวด้วยการหาเส้นทางใหม่มาใช้ก่อนที่เส้นทางการติดต่อสื่อสารเดิมจะขาดการติดต่อ ซึ่งอาศัยค่าพารามิเตอร์ 4 พารามิเตอร์มาใช้ตรวจสอบอันตรายที่อาจจะเกิดขึ้นกับเส้นทางการส่งแพคเกตข้อมูล โดยมีพารามิเตอร์แรกคือ ความแรงของสัญญาณวิทยุที่รับได้โดยโหนดที่เป็นเครื่องรับสัญญาณเมื่อระดับสัญญาณวิทยุที่รับได้มีกำลังอ่อนลงจนถึงค่าขีดเริ่มเปลี่ยน (threshold) ซึ่งเป็นช่วงที่โหนดอาจจะขาดการเชื่อมต่อในภายหลังได้ พารามิเตอร์ที่สองคือ การเหลื่อมกันของเส้นทางที่เกิดขึ้นจากโหนดตัวกลางที่รับภาระจากหลายเส้นทางให้ทำหน้าที่ส่งต่อแพคเกต ซึ่งอาจทำให้โหนดนั้นๆ รับภาระสูงเกินกำลังของโหนดจะรับไว้ได้ พารามิเตอร์ที่สามคือ พลังงานแบตเตอรี่ที่เหลือของแต่ละโหนด ซึ่งเมื่อพลังงานแบตเตอรี่ลดลงจนถึงค่าขีดเริ่มเปลี่ยน นั่นหมายถึงว่าพลังงานของโหนดอาจจะใช้ได้อีกไม่นานนัก และพารามิเตอร์สุดท้ายคือ ความหนาแน่นของโหนดที่มีอยู่ภายในเครือข่าย หากมีความหนาแน่นของโหนดเพื่อนบ้านสูงจะส่งผลถึงทรัพยากรเครือข่ายที่จะต้องใช้ร่วมกันมีสูงขึ้นด้วย

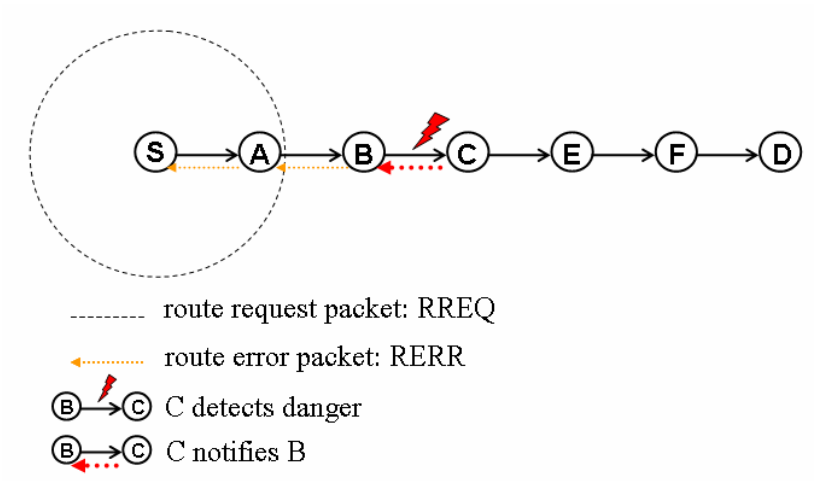
โพรโทคอล AODV-BA ได้นำพารามิเตอร์ทั้งสี่นี้มาใช้สำหรับการตรวจสอบเส้นทางการติดต่อสื่อสารที่คาดว่าอาจเกิดอันตรายขึ้นได้ โดยกำหนดให้โหนดลำดับต่อไป (downstream) ทำการตรวจสอบพารามิเตอร์ทั้งสี่แล้วแจ้งกลับไปยังโหนดลำดับก่อนหน้า (upstream) ถึงอันตรายที่อาจเกิดขึ้นกับเส้นทางการติดต่อระหว่างสองโหนดนี้ก่อนเส้นทางที่ใช้งานนี้จะขาดการเชื่อมต่อ ซึ่งโพรโทคอล AODV-BA จะต้องทำหน้าที่หาเส้นทางใหม่มาทดแทนทั้งนี้เพื่อหลีกเลี่ยงการใช้เส้นทางเดิมที่อาจจะขาดการเชื่อมต่อได้

โพรโทคอล AODV-BA มีขั้นตอนในการค้นหาเส้นทางเช่นเดียวกับโพรโทคอล AODV แต่ขั้นตอนในการส่งแพคเกตข้อมูลและการดูแลรักษาจะแตกต่างออกไปคือ หลังจากโหนดเริ่มต้นได้เส้นทางการติอต่อสื่อสารไปยังโหนดปลายทางแล้ว และได้ทำการส่งแพคเกตข้อมูลจำนวนหนึ่งด้วยเส้นทางนี้ หลังจากนั้นเส้นทางดังกล่าวมีบางโหนดที่ตรวจพบอันตรายที่อาจจะเกิดขึ้นได้ระหว่างโหนดที่เชื่อมต่อกัน นั่นคือโหนดที่เป็นโหนด downstream ในที่นี้หมายถึง

โหนดที่อยู่ลำดับต่อไปในเส้นทางที่ใช้ส่งแพคเกตข้อมูลไปยังโหนดปลายทางโดยนับจากการเชื่อมต่อระหว่างโหนดที่เกิดปัญหา ซึ่งโหนดจะตรวจพบอันตรายและแจ้งกลับไปยังโหนดที่เป็นโหนด upstream ในที่นี้หมายถึงโหนดที่อยู่ลำดับก่อนหน้าในเส้นทางที่ใช้ส่งแพคเกตข้อมูลไปยังโหนดปลายทางโดยนับจากการเชื่อมต่อระหว่างโหนดที่เกิดปัญหา หลังจากโหนด upstream ได้รับการแจ้งกลับ โหนดจะทำกระบวนการดูแลรักษาเส้นทางต่อไปดังตัวอย่างในภาพที่ 8 เมื่อเส้นทางเกิดปัญหาขาดการเชื่อมต่อขึ้น



(ก)



(ข)

ภาพที่ 8 โพรโทคอล AODV-BA เมื่อเส้นทางเกิดปัญหาขาดการเชื่อมต่อ

(ก) ระหว่างโหนด E และโหนด F

(ข) ระหว่างโหนด B และโหนด C

ภาพที่ 8 (ก) เมื่อปัญหาที่เกิดขึ้นระหว่างโหนด E และโหนด F ซึ่งอยู่ใกล้กับโหนดปลายทางในที่นี้โหนด E เป็นโหนด upstream และโหนด F เป็นโหนด downstream โหนด F ตรวจพบถึงอันตรายที่อาจเกิดขึ้นได้จึงแจ้งไปยังโหนด E หลังจากที่โหนด E ได้รับการแจ้งเตือนจากโหนด F แล้ว โหนด E จะทราบว่าตนเองมีหน้าที่ในการค้นหาเส้นทางการติดต่อสื่อสารเส้นทางใหม่ก่อนที่เส้นทางเดิมจะขาดการติดต่อ โดยโหนด E จะระบุข้อมูลเส้นทางในตารางเส้นทางของตนเองที่ฮอพลำดับต่อไปเป็นโหนด F ให้มีสถานะเป็น weak state (ws) ซึ่งถือว่ายังคงเป็นโหนดในเส้นทางที่สามารถใช้งานได้ จากนั้นโหนด E จึงทำ local repair โดยการแพร่กระจาย RREQ ไปยังโหนดเพื่อนบ้านเพื่อหาเส้นทางไปยังโหนดปลายทางคือโหนด D เมื่อโหนด D ได้รับ RREQ นี้แล้วจะตอบกลับ RREP มายังโหนด E ย้อนกลับตามเส้นทางของ RREQ ในที่นี้คือผ่านทางโหนด H และ I เมื่อโหนด E ได้รับ RREP แล้ว จะทำการเปลี่ยนแปลงข้อมูลเส้นทางในตารางเส้นทางของตนเองทันที ซึ่งเมื่อมีแพคเกตข้อมูลใหม่เข้ามาโหนด E จะส่งแพคเกตข้อมูลใหม่นี้ ไปยัง next node คือโหนด I แทนโหนด F เดิมที่อาจจะขาดการเชื่อมต่อได้ ด้วยกระบวนการนี้โหนด E จะสามารถหลีกเลี่ยงการใช้เส้นทางเดิมที่อาจจะขาดการเชื่อมต่อได้

ภาพที่ 8 (ข) เมื่อปัญหาเกิดขึ้นที่โหนด B และโหนด C ซึ่งอยู่ใกล้กับโหนดเริ่มต้นในที่นี้โหนด B เป็นโหนด upstream และโหนด C เป็นโหนด downstream โหนด C ตรวจพบอันตรายที่อาจเกิดขึ้นจึงแจ้งไปยังโหนด B หลังจากที่โหนด B ได้รับการแจ้งเตือน ในกรณีนี้โหนด B จะไม่ทำ local repair แต่จะส่ง RERR ย้อนกลับไปยังโหนดเริ่มต้นพร้อมกับบอกสถานะของเส้นทางระหว่างโหนด B และโหนด C ด้วย W (weak) ไว้ใน RERR ซึ่งสถานะของเส้นทางนี้ยังใช้งานได้อยู่ แต่จะไม่รับ RREQ ที่มีโหนดปลายทางเป็นโหนด C ในระหว่างที่ RERR ถูกส่งออกไปนั้น โหนดในเส้นทางของ RERR เมื่อได้รับแพคเกตแล้วจะทำการเปลี่ยนแปลงข้อมูลเส้นทางในตารางเส้นทางที่มี next node เป็นโหนด B ให้เป็น ws (weak state) นั่นคือเมื่อโหนดนั้นๆ ได้รับ RREQ ที่มีโหนดปลายทางเป็นโหนด C และมี next node ในตารางเส้นทางเป็นโหนด B โหนดเหล่านั้นจะละทิ้ง RREQ นี้ จากนั้นจึงตรวจว่าเป็นโหนดปลายทางของ RERR นี้หรือไม่ ถ้าตรวจพบว่าไม่ใช่โหนดจะส่งต่อ RERR ไปจนถึงโหนด S เมื่อโหนด S ได้รับ RERR จะทำการปรับข้อมูลเส้นทางในตารางเส้นทางที่มี next node เป็นโหนด B ให้เป็น ws เช่นกัน จากนั้นโหนด S จะแพร่กระจาย RREQ เพื่อค้นหาเส้นทางไปยังโหนด D เมื่อโหนด S ได้รับเส้นทางใหม่แล้วจึงทำการเปลี่ยนแปลงข้อมูลเส้นทางใหม่นี้ในตารางเส้นทางและใช้เส้นทางนี้ในการส่งแพคเกตต่อไป

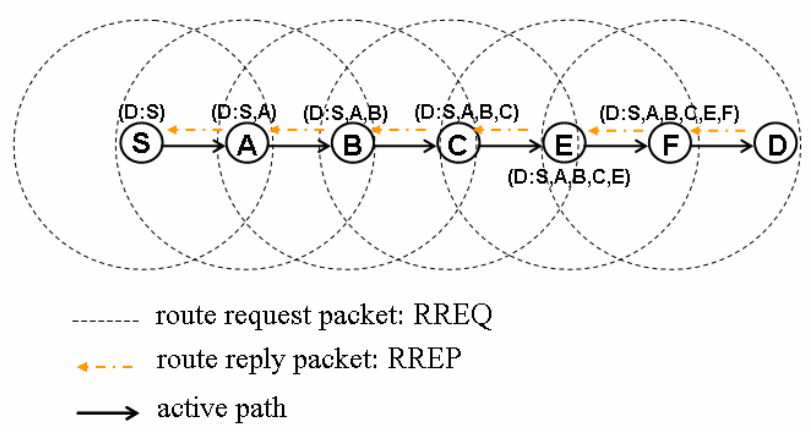
ข้อได้เปรียบของโพรโทคอล AODV-BA คือ สามารถลดจำนวนการสูญหายแพคเกตข้อมูลได้มากกว่าโพรโทคอล AODV ทั้งนี้เพราะเมื่อโหนดตรวจพบอันตรายที่อาจจะเกิดขึ้นกับเส้นทาง แล้วทำการหาเส้นทางใหม่มาใช้ได้ทันก่อนที่เส้นทางเดิมจะขาดการเชื่อมต่อไปนั่นเอง ซึ่งกระบวนการนี้เสมือนการหลีกเลี่ยงการใช้เส้นทางที่คาดว่าจะเสี่ยงอันตรายได้ ในขณะที่แพคเกตควบคุมที่ใช้ไปเพิ่มขึ้นไม่มากโดยแพคเกตที่เพิ่มขึ้นมาก็เพราะการทำ RREQ ที่เพิ่มขึ้นหลังจากที่ตรวจพบอันตรายบ่อยครั้งนั่นเอง อีกทั้งค่าการหน่วงเวลาของแพคเกตข้อมูลก็ลดลงด้วยเนื่องจากเวลาที่ใช้ในการค้นหาเส้นทางลดลงเพราะการขาดการเชื่อมต่อที่ลดลงนั่นเอง แต่ยังมีการสูญหายแพคเกตข้อมูลจำนวนมากในกรณีที่โหนดมีการเคลื่อนที่เร็ว ซึ่งทำให้โหนดไม่สามารถหาเส้นทางใหม่ได้ทันใช้ อีกทั้งเมื่อมีปริมาณการจราจรของข้อมูล (data traffic) ที่สูงขึ้น โพรโทคอลนี้มีแนวโน้มของจำนวนการขาดการเชื่อมต่อที่น่าจะมากกว่าโพรโทคอล AODV ทั้งนี้เพราะจะมีการตรวจพบอันตรายที่มีจำนวนมากยิ่งกว่าเดิม ส่งผลทำให้มีการแพร่กระจาย RREQ เพิ่มมากขึ้นด้วย ซึ่งจะทำให้มีการแย่งช่องสัญญาณกันเพื่อใช้ส่งแพคเกตต่างๆ นั่นคือความสิ้นเปลืองต่างๆ ที่อาจจะเกิดขึ้นได้ ดังนั้นโพรโทคอล AODV-BA นี้จึงเหมาะกับการใช้งานในเครือข่ายที่มีปริมาณการจราจรของข้อมูลจำนวนน้อยๆ การเคลื่อนที่ของโหนดไม่เร็วนัก มีปริมาณโหนดในเครือข่ายขนาดปานกลาง (20-200 โหนด) ซึ่งโดยภาพรวมโพรโทคอล AODV-BA ทำให้โพรโทคอล AODV มีประสิทธิภาพที่ดีขึ้นได้

2.3 โพรโทคอล DSR (dynamic source routing) (Johnson, 2003) มีความคล้ายคลึงกับโพรโทคอล AODV เนื่องจากทั้งสองโพรโทคอลนี้เป็นโพรโทคอลแบบ Reactive เช่นเดียวกัน ซึ่งในการค้นหาเส้นทางและการดูแลรักษาเส้นทางของทั้งสองโพรโทคอลจะใช้แพคเกตที่เหมือนกัน คือ RREQ, RREP และ RERR แต่กระบวนการทำงานจะแตกต่างกัน อีกทั้งลักษณะการจัดเก็บข้อมูลเส้นทางก็ต่างกันด้วย โดยโพรโทคอล DSR จะเก็บข้อมูลเส้นทางทั้งเส้นทางไว้ในแคชเส้นทาง ซึ่งเมื่อโหนดเริ่มต้นทำการส่งแพคเกตข้อมูล โหนดจะใช้ข้อมูลเส้นทางที่มีอยู่ โดยนำข้อมูลเส้นทางทั้งเส้นทาง (source route) ใส่ไว้ที่ส่วนหัวของแพคเกต (header) และส่งแพคเกตข้อมูลตามเส้นทางที่ระบุนี้ ในการเปลี่ยนแปลงข้อมูลเส้นทางนั้นโพรโทคอลจะอาศัยโหมดการดักฟัง (promiscuous mode) หรือได้รับ RREP จากนั้นจึงนำข้อมูลเส้นทางที่ได้จากส่วนหัวของ RREP นี้ นำไปเก็บไว้ในแคชเส้นทางของตนเองและโหนดจะทำการลบเส้นทางที่ขาดการเชื่อมต่อที่อยู่ในแคชเส้นทางก็ต่อเมื่อโหนดได้รับหรือดักฟัง RERR นั้นๆ ในการส่งแพคเกตข้อมูลถ้าหากเส้นทางการส่งแพคเกตข้อมูลที่ใช้งานอยู่ขาดการเชื่อมต่อ โพรโทคอลจะใช้ข้อมูลเส้นทางที่อยู่ในแคชเส้นทาง เพื่อนำส่งแพคเกตข้อมูลไปยังโหนดปลายทาง (salvage) และแจ้งความผิดพลาดที่เกิด

ขึ้นกับเส้นทางกลับไปยังโหนดเริ่มต้นเพื่อลบเส้นทางเดิมและทำการหาเส้นทางใหม่ในแคชเส้นทางของโหนดเริ่มต้น ในขณะที่โพรโทคอล AODV จะเก็บข้อมูลเฉพาะ next node ไว้ในรูปของตารางเส้นทาง เมื่อโหนดเริ่มต้นจะทำการส่งแพคเกตข้อมูลด้วยข้อมูลเส้นทางที่มีอยู่ โหนดจะระบุเพียง next node ที่จะต้องทำการส่งไปให้เท่านั้นโดยระบุไว้ที่ส่วนหัวของแพคเกต ซึ่งแต่ละโหนดจะทราบเฉพาะ next node ที่ตนเองจะต้องส่งไปให้ และการที่จะทำการเปลี่ยนแปลงข้อมูลเส้นทางในตารางเส้นทางนั้น โหนดจะอาศัย Hello packet ที่มีการส่งออกไปเป็นระยะๆ ทั้งนี้เพื่อตรวจสอบการคงอยู่ของโหนดเพื่อนบ้านและโหนดจะทำการลบเส้นทางที่ขาดการเชื่อมต่อออกจากตารางเส้นทางก็ต่อเมื่อโหนดได้รับ RERR และในการส่งแพคเกตข้อมูลนั้น ถ้าในขณะที่โหนดทำการส่งแพคเกตข้อมูลด้วยเส้นทางที่ใช้งานอยู่มีบางโหนดเกิดขาดการเชื่อมต่อขึ้น โพรโทคอล AODV จะแบ่งกระบวนการดูแลรักษาเส้นทางเป็นสองกรณีคือกรณีที่มีปัญหาเกิดใกล้กับโหนดปลายทาง โหนดจะทำ local repair และกรณีที่มีปัญหาเกิดใกล้กับโหนดเริ่มต้น โหนดจะแจ้งความผิดพลาดที่เกิดขึ้นกับเส้นทางกลับไปยังโหนดเริ่มต้นและโหนดเริ่มต้นค้นหาเส้นทางใหม่

โพรโทคอล DSR มีขั้นตอนการทำงานดังนี้คือ โหนดเริ่มต้นจะส่ง RREQ ไปยังโหนดเพื่อนบ้าน เมื่อโหนดเพื่อนบ้านได้รับแพคเกตจะตรวจสอบโหนดปลายทางของแพคเกตนี้ใช่โหนดตนเองหรือไม่ ถ้าตรวจพบว่าไม่ใช่ โหนดจะทำการเพิ่มรหัสที่ระบุว่าเป็นโหนดตนเองไว้ที่ส่วนหัวของแพคเกต จากนั้นจึงส่งแพคเกตนี้ต่อไปจนกระทั่งถึงโหนดปลายทาง จากนั้นโหนดปลายทางจะตอบกลับทุก RREQ ตามเส้นทางที่ระบุในส่วนหัวของแต่ละ RREQ ด้วย RREP เมื่อโหนดเริ่มต้นได้รับ RREP แล้ว โหนดจะเลือกใช้เส้นทางจาก RREP ที่มีจำนวนฮอพน้อยที่สุดเพื่อนำไปใช้ในการส่งแพคเกตข้อมูลโดยโหนดเริ่มต้นจะพิจารณาจากเส้นทางที่ระบุในส่วนหัวของ RREP นั้นๆ ส่วนเส้นทางอื่นๆ ที่ได้จาก RREP จะถูกนำไปเก็บไว้ในแคชเส้นทาง

หลังจากที่โหนดเริ่มต้นได้เส้นทางในการส่งแพคเกตข้อมูลแล้ว และได้ส่งแพคเกตข้อมูลไปแล้วจำนวนหนึ่ง เส้นทางดังกล่าวนั้นมีบางโหนดขาดการเชื่อมต่อทำให้โหนดไม่สามารถส่งต่อแพคเกตข้อมูลไปได้ โหนดจะทำการลบข้อมูลเส้นทางที่มี next node เป็นโหนดที่ขาดการเชื่อมต่อไปแล้วนี้ออกจากแคชเส้นทาง และหาเส้นทางใหม่ในแคชเส้นทางเพื่อใช้นำพาแพคเกตข้อมูลไปให้ถึงยังโหนดปลายทางและในขณะเดียวกันโหนดจะส่ง RERR กลับไปยังโหนดเริ่มต้นด้วย เมื่อโหนดเริ่มต้นได้รับ RERR แล้วจะนำข้อมูลเส้นทางที่มีเส้นทางการเชื่อมต่อด้วยโหนดทั้งสองที่ระบุมาในแพคเกตลบออกจากแคชเส้นทาง แล้วหาเส้นทางใหม่จากแคชเส้นทางของตนเองเพื่อนำไปใช้ในการส่งแพคเกตข้อมูลลำดับต่อไป



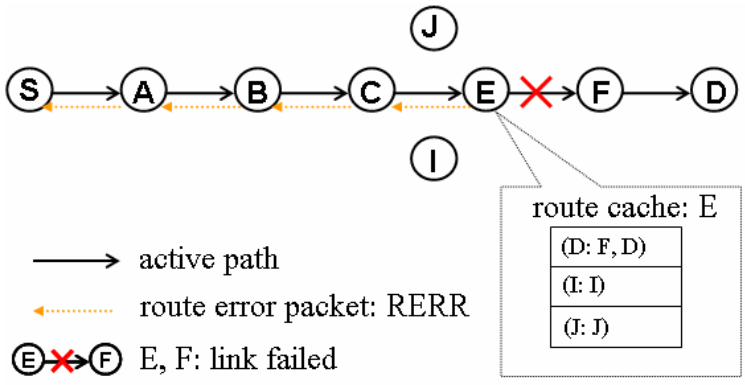
ภาพที่ 9 โพรโทคอล DSR เมื่อค้นหาเส้นทางจากโหนด S ไปยังโหนด D

ภาพที่ 9 แสดงถึงการค้นหาเส้นทางของโพรโทคอล DSR โดยมีโหนดเริ่มต้นคือโหนด S ต้องการติดต่อกับโหนดปลายทางคือโหนด D โพรโทคอลจะเริ่มต้นจากโหนด S ใส่รหัสของโหนดไว้ที่ส่วนหัวของ RREQ และทำการแพร่กระจาย RREQ นี้ไปยังโหนดเพื่อนบ้านที่อยู่ในรัศมีการส่งของโหนดในที่นี้ได้แก่โหนด A ดังนั้นเมื่อโหนด A ได้รับ RREQ โหนดจะตรวจสอบว่าโหนดปลายทางเป็นโหนดตนเองหรือไม่ หรือมีเส้นทางใดในแคชเส้นทางของตนไปยังโหนดปลายทางได้ ถ้าตรวจพบว่าตนเองไม่ใช่โหนดปลายทางและไม่มีเส้นทางใดๆ ในแคชเส้นทาง โหนด A จะใส่รหัสของโหนดเพิ่มลงไปในส่วนหัวของ RREQ ดังนั้นจะได้เส้นทางคือ (D: S, A) จากนั้นจึงแพร่กระจาย RREQ นี้ต่อไปจนกระทั่งถึงโหนด D

เมื่อโหนด D ได้รับ RREQ แล้วโหนดจะส่ง RREP กับไปยังโหนด S ในทุกเส้นทางที่ระบุไว้ในส่วนหัวของแต่ละ RREQ ที่ได้รับเข้ามาและจะยังไม่เก็บเส้นทางเหล่านี้ไว้ในแคชเส้นทางของตนเอง ในที่นี้ RREQ มีลำดับเส้นทางคือ (D: S, A, B, C, E, F, D) และ RREP มีลำดับเส้นทางคือ (S: D, F, E, C, B, A, S) โดยโหนด D หรือโหนดใดๆ จะทำการเก็บข้อมูลเส้นทางไว้ในแคชเส้นทางเมื่อโหนดเหล่านั้นได้รับหรือดักฟัง RREP เท่านั้น

หลังจากที่โหนด S ได้รับ RREP แล้วโหนดจะเก็บข้อมูลเส้นทางจากทุกๆ RREP นี้ไว้ในแคชเส้นทางของตนเอง เมื่อโหนดจะส่งแพคเกตข้อมูลใดๆ โหนดจะเลือกใช้เส้นทางที่สั้นที่สุดที่มีอยู่ในแคชเส้นทางนี้เป็นเส้นทางที่จะนำไปใช้ในการส่งแพคเกตข้อมูล (packet route) โดยมีลำดับเส้นทางการส่งแพคเกตข้อมูลเริ่มตั้งแต่โหนดเริ่มต้นผ่านโหนดตัวกลางต่างๆ จนกระทั่งถึงโหนด

ปลายทางซึ่งเส้นทางดังกล่าวจะเป็นเส้นทางที่สมบูรณ์ และจะถูกระบุไว้ที่ส่วนหัวของทุกแพคเกตข้อมูลที่ส่งออกไปจากโหนด S ในที่นี้เส้นทางที่ใช้คือ (D: S, A, B, C, E, F, D)



ภาพที่ 10 โพรโทคอล DSR เมื่อเส้นทางเกิดปัญหาขาดการเชื่อมต่อระหว่างโหนด E และโหนด F โดยที่โหนด E ไม่มีเส้นทางใหม่ในแคชเส้นทาง

ถ้าในขณะที่เส้นทางที่ใช้ส่งแพคเกตข้อมูลอยู่นั้น มีบางโหนดเกิดปัญหาไม่สามารถส่งแพคเกตข้อมูลนี้ออกไปได้ ดังตัวอย่างในภาพที่ 10 เมื่อเกิดปัญหาเส้นทางขาดการเชื่อมต่อที่เกิดขึ้นระหว่างโหนด E และโหนด F ในที่นี้โหนด E จะทำหน้าที่ดูแลรักษาเส้นทางโดยการลบเส้นทางที่มี next node เป็นโหนด F ออกจากแคชเส้นทาง และหาเส้นทางใหม่ไปยังโหนด D แต่โหนด E ไม่มีเส้นทางใหม่ในแคชเส้นทางเพื่อใช้นำพาแพคเกตข้อมูลนี้ไปให้ถึงโหนด D โหนด E จะทำการส่ง RERR แจ้งกลับไปยังโหนด S ซึ่งการส่งในครั้งนี้ทำให้เกิดการสูญหายแพคเกตข้อมูล

เมื่อโหนด S ได้รับ RERR นี้ โหนดจะลบข้อมูลเส้นทางที่มีการเชื่อมต่อระหว่างโหนด E และโหนด F ในทุกๆ เส้นทางออกจากแคชเส้นทางของตนเองและตรวจสอบแคชเส้นทางเพื่อหาเส้นทางใหม่นำไปใช้ในการส่งแพคเกตข้อมูลลำดับถัดไปที่สามารถไปถึงโหนด D ได้ ถ้าหากมีข้อมูลเส้นทางใหม่ในแคชเส้นทางโหนด S จะใช้เส้นทางใหม่ระบุในส่วนหัวของแพคเกตข้อมูลลำดับถัดไป แต่ถ้าไม่มีเส้นทางใหม่โหนด S จะค้นหาเส้นทางใหม่อีกครั้งโดยแพร่กระจาย RREQ ออกไปและรอรับ RREP จากโหนด D หลังจากนั้นโหนด S จึงเริ่มนำเส้นทางใหม่นี้มาใช้แทน

ในขณะที่โหนดขาดการเชื่อมต่อนั้น โพรโทคอล DSR มีปัญหาที่สำคัญปัญหาหนึ่งคือ เส้นทางในแคชเส้นทางที่โหนดจะนำมาใช้เพื่อนำพาแพคเกตข้อมูลไปให้ถึงยังโหนดปลายทางนั้น อาจจะเป็นเส้นทางเก่าที่ไม่สามารถใช้งานได้แล้ว (stale route) อันเนื่องมาจากเส้นทางที่มีอยู่

อาจจะเก็บไว้นานซึ่งบางโหนดในเส้นทางอาจขาดการเชื่อมต่อกันไปแล้ว หรือในขณะที่โหนดอื่นได้รับหรือดักฟัง RERR แต่มีบางโหนดได้เคลื่อนที่ออกไปแล้ว จึงไม่ได้รับหรือดักฟัง RERR ดังกล่าวจึงทำให้ข้อมูลในแคชเส้นทางไม่ได้ถูกลบทิ้งออกไป ซึ่งข้อมูลเส้นทางนี้จะส่งผลถึงการที่โหนดจะเลือกเส้นทางมาใช้ จึงเป็นเส้นทางที่ใช้ไม่ได้ทำให้โหนดขาดการเชื่อมต่อ และมีการสูญหายแพคเกตเกิดขึ้นได้ จากการที่โพรโทคอล DSR มีโหมดการดักฟังทำให้โหนดสามารถกระทำการต่างๆ ได้หลายอย่างอาทิ การตอบกลับเส้นทางด้วยเส้นทางในแคชเส้นทางของตนเองที่สามารถไปยังโหนดปลายทางได้ด้วยเส้นทางที่สั้นกว่าเส้นทางจาก RREQ ที่ได้รับ หรือจากการดักฟังเส้นทางการส่งแพคเกตข้อมูล ซึ่งทั้งสองกรณีนี้โหนดที่ทำการตอบกลับเส้นทางอาจจะส่งเส้นทางที่ใช้งานไม่ได้แล้วนี้กลับไม่ให้กับโหนดเริ่มต้นได้ เมื่อโหนดเริ่มต้นใช้เส้นทางนี้ในการส่งแพคเกตข้อมูลอาจทำให้เกิดการสูญหายแพคเกตข้อมูลขึ้นได้เช่นกัน

ภาพที่ 11 เมื่อเส้นทางเกิดปัญหาขาดการเชื่อมต่อระหว่างโหนด E และโหนด F โดยในภาพที่ 11 (ก) เมื่อโหนดตัวกลางมีการเคลื่อนที่ไปจากตำแหน่งเดิมหรือเส้นทางที่ใช้ในการส่งแพคเกตระหว่างที่โหนดยังคงติดต่อสื่อสารกันอยู่นั้นเกิดปัญหาขึ้น เช่น เมื่อเหตุการณ์เกิดขึ้นระหว่างโหนด E และโหนด F ในที่นี้โหนด E ขาดการติดต่อกับโหนด F โหนด E จึงลบข้อมูลเส้นทางที่มี next node เป็นโหนด F ออกจากแคชเส้นทางของตนเอง

จากนั้นในภาพที่ 11 (ข) โหนด E ส่ง RERR กลับไปยังโหนดเริ่มต้น และตรวจสอบข้อมูลหาเส้นทางใหม่ในแคชเส้นทางของตนเองมีเส้นทางใดสามารถไปถึงยังโหนด D ได้บ้าง ในที่นี้มีเส้นทางใหม่คือ (D: H, D) ซึ่งเป็นเส้นทางที่สั้นที่สุดที่โหนด E มีข้อมูลอยู่แต่เส้นทางดังกล่าวใช้ไม่ได้แล้ว อาจเป็นเพราะโหนด H ได้เคลื่อนที่ออกจากโหนด E ทำให้ไม่สามารถติดต่อถึงกันได้ ดังนั้นเมื่อโหนด E ใช้เส้นทางนี้จะทำให้มีการสูญหายแพตเกตเกิดขึ้น เมื่อโหนด E ส่งแพคเกตข้อมูลไปไม่ถึงโหนด H โหนด E จึงลบเส้นทางที่มี next node เป็นโหนด H ออกจากแคชเส้นทางของตนเอง และขณะเดียวกันหลังจากที่โหนด E ส่ง RERR ออกไปจนถึงโหนด S แล้ว โหนด S จะลบข้อมูลเส้นทางที่มีการเชื่อมต่อระหว่างโหนด E และโหนด F ที่ระบุมากับ RERR ออกจากแคชเส้นทาง จากนั้นโหนด S หาเส้นทางใหม่ในแคชเส้นทางไปถึงยังโหนด D ดังในภาพที่ 11 (ค)

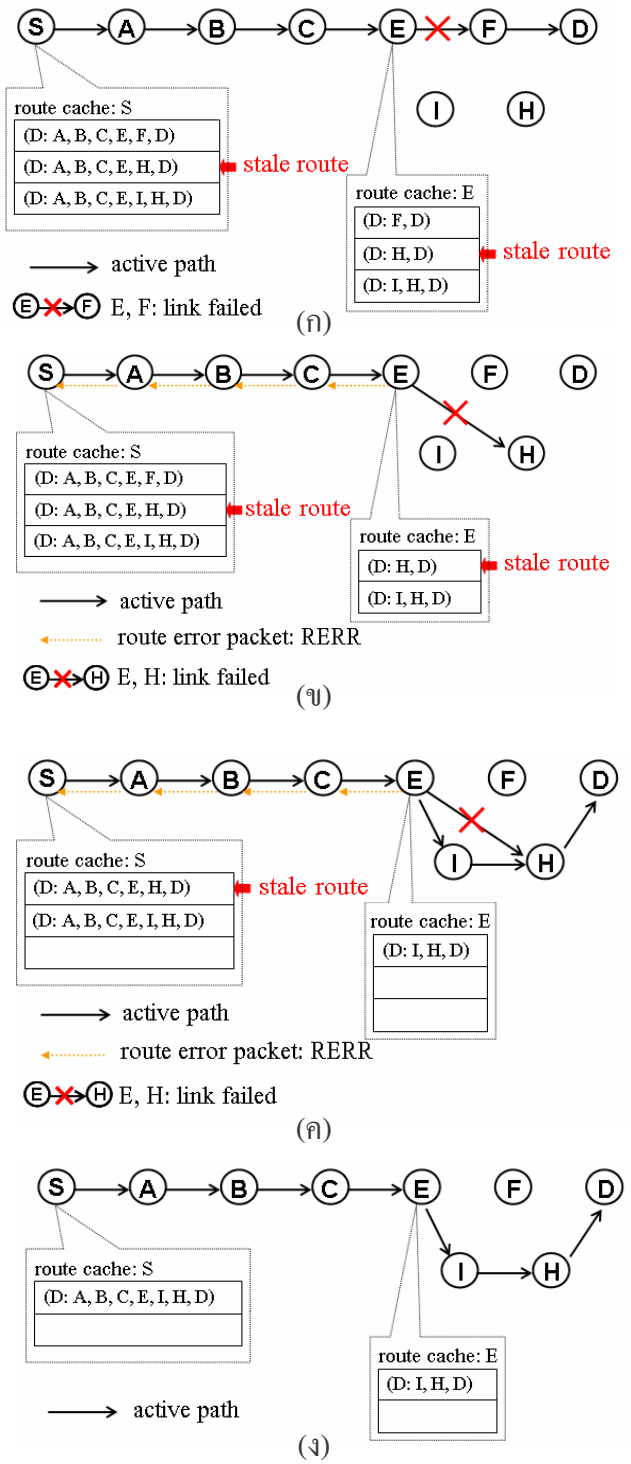
ภาพที่ 11 (ค) ในที่นี้ได้เส้นทางใหม่คือ (D: A, B, C, E, H, D) ซึ่งเส้นทางนี้โหนด E และโหนด H ได้ขาดการเชื่อมต่อกันไปแล้ว ดังนั้นเมื่อแพคเกตข้อมูลไปถึงโหนด E จึงส่งต่อไปโหนด H ไม่ได้ โหนด E จึงลบเส้นทางที่มี next node เป็นโหนด H ออกจากแคชเส้นทาง หลังจาก

นั้นโหนด E จะหาเส้นทางใหม่ไปโหนด D ในที่นี้ได้เส้นทางคือ (D: I, H, D) เป็นเส้นทางที่ยังคงใช้งานได้ทำให้แพคเกตข้อมูลนี้ไม่สูญหายไป ในขณะเดียวกันโหนด E ได้ส่ง RERR ไปถึงยังโหนด S โหนด S จึงทำการลบข้อมูลเส้นทางที่มีการเชื่อมต่อระหว่างโหนด E และโหนด H ออกจากแคชเส้นทาง จากนั้นค้นหาเส้นทางใหม่ในแคชเส้นทางที่ไปถึงยังโหนด D ดังในภาพที่ 11 (ง)

ภาพที่ 11 (ง) ในที่นี้โหนด Sได้เส้นทางใหม่คือ (D: A, B, C, E, I, H, D) เป็นเส้นทางที่โหนดยังคงสามารถใช้งานได้ จากตัวอย่างข้างต้นนี้โหนด E ใช้เส้นทางไปยังโหนด H ซึ่งไม่สามารถติดต่อกันได้โดยตรงแม้จะเป็นเส้นทางที่สั้นที่สุดก็ตาม ยังคงทำให้เกิดการสูญหายแพคเกตได้ และโหนด S ก็เช่นกันยังคงมีเส้นทางที่ไม่สามารถใช้งานได้เก็บอยู่ในแคชเส้นทาง ซึ่งหลังจากใช้เส้นทางดังกล่าวแล้วทำให้เกิดการขาดการเชื่อมต่อแต่ยังไม่เกิดการสูญหายแพคเกตขึ้น เนื่องจากโหนด E มีเส้นทางที่ยังคงใช้งานได้นำพาแพคเกตข้อมูลส่งไปถึงยังโหนด D ได้นั่นเอง

ขณะที่เส้นทางขาดการเชื่อมต่ออยู่นั้น หลังจากที่โหนดเริ่มต้นได้รับ RERR หากโหนดเริ่มต้นไม่มีเส้นทางใหม่ไปยังโหนดปลายทาง โหนดเริ่มต้นจะทำการค้นหาเส้นทางใหม่ด้วยการแพร่กระจาย RREQ ออกไปและรอรับเส้นทางใหม่จาก RREP ที่ส่งกลับมาอาจจะมาจากโหนดตัวกลางที่มีเส้นทางไปถึงโหนดปลายทางได้หรือจากโหนดปลายทาง

โพรโทคอล DSR ในกระบวนการค้นหาเส้นทางนั้น หากโหนดรับ RREQ มาแล้วแต่ยังไม่ได้ทำการแพร่กระจาย RREQ ชุดนี้ออกไป และในขณะเดียวกัน โหนดได้รับ RREQ ที่มีเลขลำดับ (sequence number) หมายเลขเดียวกันที่มาจากโหนดอื่นๆ โหนดดังกล่าวนี้จะนำ RREQ ใหม่มาตรวจสอบเส้นทาง หาก RREQ ใหม่มีเส้นทางที่สั้นกว่า โหนดจะเลือกแพร่กระจาย RREQ ใหม่นี้ออกไปในเครือข่าย และในระหว่างที่ RREQ ถูกส่งผ่านไปยังแต่ละโหนดในเครือข่าย จะมีข้อมูลเส้นทางที่ผ่านไปยังแต่ละโหนดที่ระบุด้วยรหัสของโหนดอยู่ในส่วนหัวของแพคเกตนี้ด้วย ซึ่งถ้ามีจำนวนฮอพที่ผ่านมาจากแต่ละโหนดจำนวนมาก ขนาดส่วนหัวของ RREQ ก็จะมากขึ้นโดยขนาดที่เพิ่มขึ้นนั้นจะแปรผันตามจำนวนของโหนดที่ผ่าน และขนาดส่วนหัวของ RREQ จะยังคงเพิ่มขึ้นไปเรื่อยๆ จนกว่าจะถึงโหนดปลายทาง



ภาพที่ 11 โพรโทคอล DSR เส้นทางเกิดปัญหาขาดการเชื่อมต่อระหว่างโหนด E และโหนด F

(ก) เมื่อโหนด E ขาดการเชื่อมต่อกับโหนด F

(ข) เมื่อโหนด E ใช้ stale route

(ค) เมื่อโหนด Sใช้เส้นทางส่งข้อมูลที่เป็น stale route

(ง) เมื่อใช้เส้นทางการเชื่อมต่อเส้นทางใหม่

เมื่อโหนดปลายทางได้รับ RREQ จะตอบกลับไปในทุกเส้นทางที่ RREQ มาถึง ซึ่งถ้า intermediate nodes ไม่สามารถส่งต่อ RREP ไปได้ จะส่ง RERR กลับไปยังโหนดปลายทางเพื่อทำการลบเส้นทางออกจากแคชเส้นทางและในขณะเดียวกันจะหาเส้นทางใหม่ในแคชเส้นทางเพื่อใช้นำพา RREP นี้ไปยังโหนดเริ่มต้น แต่ถ้า RREP ยังคงไปไม่ถึงโหนดเริ่มต้นจะทำให้เกิดการสูญหายขึ้น และถ้า RERR ที่ถูกส่งกลับไปไม่ถึงโหนดปลายทาง ดังนั้น intermediate nodes ที่พบปัญหาจะหาเส้นทางในแคชเส้นทางเพื่อนำพา RERR นี้ส่งไปให้ถึงโหนดปลายทาง ถ้า RERR นี้กลับไปไม่ถึงจะทำให้เกิดการสูญหายขึ้น ขณะเดียวกัน intermediate nodes ดังกล่าวหาเส้นทางใหม่ในแคชเส้นทางเพื่อใช้นำพา RREPที่ส่ง RREP ไปไม่ได้ซึ่งเป็นโหนดที่ส่งแพคเกตแจ้งความผิดพลาดไปยังโหนดปลายทางให้รับทราบ และถ้าทั้งแพคเกตแจ้งความผิดพลาดไปยังโหนดปลายทางที่โหนดพยายามนำพาอยู่นั้นและแพคเกตแจ้งความผิดพลาดที่ส่งไปยังโหนดที่ส่ง RREP ไปไม่ได้ซึ่งเป็นโหนดที่ส่งแพคเกตแจ้งความผิดพลาดไปยังโหนดปลายทางยังคงส่งไปไม่ได้ ทั้งสองแพคเกตนี้ก็จะเกิดการสูญหาย

ข้อมูลเส้นทางที่แต่ละโหนดเก็บไว้นั้น โหนดเก็บเส้นทางเป็นแบบทั้งเส้นทาง (path) ซึ่งต่างจากการเก็บแบบการเชื่อมโยง (link) ทำให้โหนดไม่สามารถเรียนรู้เส้นทางไปยังโหนดอื่นๆ แต่โหนดจะทราบว่าสามารถไปยังโหนดปลายทางได้ด้วยเส้นทางที่สมบูรณ์เส้นทางใด อีกทั้งยังง่ายต่อการตรวจสอบการวนลูปของเส้นทางด้วย

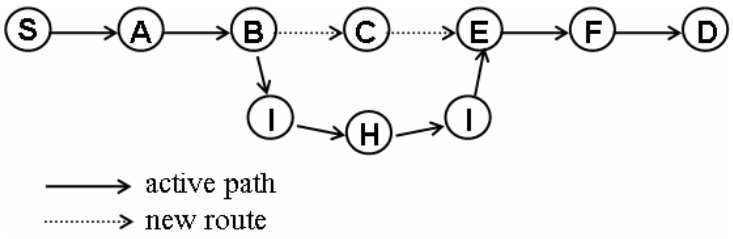
ข้อได้เปรียบของโพรโทคอล DSR คือ ไม่จำเป็นต้องส่งแพคเกคควบคุมออกไปในเครือข่ายบ่อยๆ เพื่อปรับปรุงข้อมูลเส้นทางเนื่องจากข้อมูลเส้นทางไม่ได้เก็บไว้ในรูปของตารางเส้นทางที่เก็บเฉพาะ next node แต่เก็บไว้ในรูปของแคชเส้นทางโดยเก็บเส้นทางที่สมบูรณ์ทั้งเส้นทาง ซึ่งข้อมูลเส้นทางที่เก็บไว้นั้น จะเก็บไว้เฉพาะช่วงเวลาที่โหนดติดต่อสื่อสารกัน อีกทั้งข้อมูลเส้นทางของโพรโทคอล DSR นั้น ได้มาจากการดักฟังข้อมูลเส้นทางจากโหนดเพื่อนบ้าน ซึ่งช่วยให้ลดจำนวนของแพคเกตควบคุมและลดการใช้ทรัพยากรเครือข่ายที่ต้องใช้ไปเพื่อปรับปรุงข้อมูลเส้นทาง มีการรับประกันในเรื่องของเส้นทางการวนซ้ำเนื่องจากโหนดสามารถตรวจสอบเส้นทางได้จากส่วนหัวของแพคเกต อีกทั้งยังสามารถหยุดการแพร่กระจายของแพคเกตควบคุมทั้งหมดที่ส่งต่อกันอยู่ในเครือข่ายได้ด้วยค่า TTL เช่นเดียวกับโพรโทคอล AODV นอกจากนี้ในขณะที่โหนดดูแลรักษาเส้นทางเมื่อเกิดขาดการเชื่อมต่อกันโหนดจะทำได้ไวเนื่องจากมีเส้นทางรองรับหลายเส้นทางในแคชเส้นทางของแต่ละโหนด ซึ่งจะต่างกับโพรโทคอล AODV ที่จะต้องแพร่กระจาย RREQ และรอรับการตอบกลับเส้นทางซึ่งจะใช้เวลาที่นานกว่าในการดูแลรักษา

เส้นทาง แต่จะเกิดขึ้นได้ในกรณีที่โพรโทคอล DSR จะต้องมีข้อมูลเส้นทางที่สามารถใช้งานได้มากกว่าเส้นทางที่ใช้งานไม่ได้ในแคชเส้นทางของแต่ละโหนด โพรโทคอลนี้จึงจะได้เปรียบอย่างชัดเจน

ในส่วนข้อเสียเปรียบที่เกิดขึ้นได้อันเนื่องมาจากมีการสูญหายของแพคเกตในขณะที่มีการติดต่อสื่อสารกันมากกว่าโพรโทคอล AODV ซึ่งเป็นโพรโทคอลตามความต้องการเช่นเดียวกัน (Das et al., 2000) ที่เป็นเช่นนี้อาจเป็นได้ว่าบางข้อมูลเส้นทางในแคชเส้นทางนั้นอาจไม่สามารถใช้งานได้เมื่อเกิดปัญหาเส้นทางขาดการติดต่อขึ้นมาจริงๆ ที่กล่าวว่าไม่สามารถใช้งานได้จริงๆ นั้นในที่นี้คือ ไม่สามารถบอกได้ว่า ณ ขณะนั้นข้อมูลเส้นทางที่เก็บไว้ในแคชเส้นทางยังคงใช้งานได้หรือไม่ อาจเป็นเพราะข้อมูลเส้นทางนั้นเป็นข้อมูลที่เก็บไว้นานแล้วก่อนที่โหนดในเครือข่ายจะมีการเคลื่อนที่ไปจากตำแหน่งเดิม ดังนั้นเมื่อนำเส้นทางดังกล่าวมาใช้งานจึงเกิดเหตุการณ์เส้นทางขาดการเชื่อมต่อขึ้นมาได้ และอาจทำให้มีการสูญหายแพคเกตเกิดขึ้น นอกจากนี้การที่โหนดมีการเคลื่อนที่ กระบวนการแจ้งความผิดพลาดของเส้นทางอาจทำได้ไม่สมบูรณ์นัก ทำให้มีบางโหนดที่เคลื่อนที่ออกไปก่อนที่จะได้รับหรือดักฟัง RERR ยังคงมีเส้นทางที่อาจจะต้องกำจัดออกไปคงอยู่ในแคชเส้นทาง ซึ่งอาจจะส่งผลถึงการตอบกลับเส้นทางด้วยเส้นทางที่ใช้ไม่ได้แต่เป็นเส้นทางที่สั้นที่สุดที่โหนดมีข้อมูลเส้นทางอยู่ ดังนั้นปัญหาหลักของโพรโทคอล DSR จึงเป็นปัญหาของเส้นทางที่เก็บอยู่ในแคชเส้นทางนั่นเอง นอกจากนี้ยังมีข้อเสียเปรียบอื่นๆ อีกคือจากการที่โหนดเก็บข้อมูลเส้นทางไว้ที่ส่วนหัวของแพคเกตตลอดเวลาเวลาที่มีการส่งแพคเกตกันนั้น ทำให้แพคเกตที่ส่งไปนั้นมีขนาดใหญ่โดยแปรผันตามจำนวนของโหนดที่ต้องใช้ในแต่ละเส้นทาง ดังนั้นเมื่อมีการขยายขนาดของเครือข่ายส่วนหัวของแพคเกตจะมีขนาดใหญ่เพิ่มขึ้นไปด้วย และข้อเสียเปรียบอีกประการคือเวลาที่ใช้ในการค้นหาเส้นทางจะนานกว่าโพรโทคอล DSDV เพราะจะมีเส้นทางก็ต่อเมื่อต้องการใช้เท่านั้น ซึ่งโพรโทคอล DSDV มีเส้นทางไว้รอให้ใช้อยู่ก่อนแล้ว อีกทั้งขณะดูแลรักษาเส้นทางถ้าหากข้อมูลเส้นทางในแคชเส้นทางเป็นเส้นทางที่ใช้ไม่ได้จะทำให้แพคเกตสูญเสียไปได้

2.4 DSR performance improvement จากการที่โพรโทคอล DSR มีการสูญหายแพคเกตจำนวนมากจึงได้มีผู้วิจัยหาวิธีการที่จะปรับปรุงประสิทธิภาพของโพรโทคอล DSR โดย Wu *et al*., 1999 ได้นำเสนอวิธีการใช้ route optimal เพื่อให้ได้เส้นทางการส่งแพคเกตข้อมูลที่เหมาะสมที่สุด ซึ่งเป็นการจัดการเกี่ยวกับการใช้เส้นทางของโพรโทคอล DSR ให้มีประสิทธิภาพ มีวิธีการดังภาพที่ 12 แสดงการใช้เส้นทางการส่งแพคเกตข้อมูลที่เหมาะสมที่สุดของโพรโทคอล DSR ด้วยวิธีการ

ของ Wu *et al*., 1999 ซึ่งจากเดิมที่มีการใช้เส้นทางในการส่งแพคเกต (D: S, A, B, I, H, K, E, F, D) เมื่อโหนดเพื่อนบ้าน (โหนด C) ได้ดักฟังเส้นทางที่ใช้ส่งแพคเกตข้อมูลเส้นทางนี้แล้วพบว่าในแคชเส้นทางของตนเองมีข้อมูลเส้นทางที่สั้นกว่าในการไปถึงโหนดปลายทาง (โหนด D) จึงส่ง RREP ไปยังโหนดเริ่มต้น (โหนด S) ให้ทำการเปลี่ยนเส้นทาง โดยในที่นี้โหนด C ดักฟังเส้นทางจากโหนด B คือ (D: S, A, B) และมีเส้นทางคือ (D: C, E, F, D) และ จากนั้นโหนด C จะส่ง RREP ไปยังโหนด S ด้วยเส้นทาง (D: S, A, B, C, E, F, D)



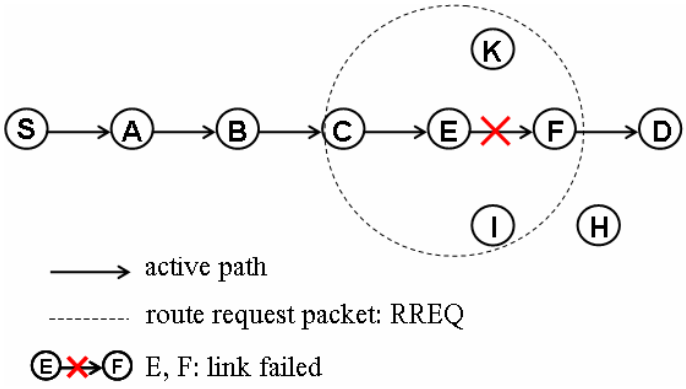
ภาพที่ 12 เส้นทางการส่งแพคเกตข้อมูลที่เหมาะสมที่สุดด้วยวิธีของ Wu *et al*.

วิธีการนี้จะมีประสิทธิภาพมากขึ้นถ้าหากข้อมูลเส้นทางที่โหนด C มีอยู่เป็นเส้นทางที่ยังคงใช้งานได้เพราะจะทำให้แพคเกตข้อมูลมีค่าการหน่วงของแพคเกตข้อมูลลดลงเนื่องจากได้เส้นทางที่สั้นขึ้น แต่จะไม่เป็นผลดีเลยถ้าเส้นทางที่โหนด C ส่งไปเป็นเส้นทางที่ใช้ไม่ได้ จะทำให้เกิดการสูญหายแพคเกตมากขึ้นเมื่อเทียบกับเส้นทางเดิมที่เส้นทางอาจจะไม่ขาดการเชื่อมต่อเลย แต่อาจจะมีค่าที่การหน่วงของแพคเกตข้อมูลที่มากกว่าบ้าง แต่การที่ได้เส้นทางที่สั้นในการส่งแพคเกตข้อมูลเป็นเรื่องที่ดีนั่นคือเส้นทางมีความเสี่ยงในการขาดการเชื่อมต่อที่น้อยกว่าเส้นทางที่ยาวขึ้น

วิธีการต่อมา Sengul, 2003 ได้นำเสนอวิธีการค้นหาเส้นทางใหม่มาทดแทนเส้นทางเดิมที่ใช้ในการส่งแพคเกตข้อมูลหลังจากที่โหนดได้ขาดการเชื่อมต่อไปแล้วโดยการทำ local route recovery ซึ่งจะคล้ายกับการทำ local repair ของโพรโทคอล AODV ดังภาพที่ 13 แสดงการค้นหาเส้นทางใหม่เมื่อเส้นทางเดิมขาดการเชื่อมต่อด้วยวิธีการของ Sengul ซึ่งจากเดิมที่มีการใช้เส้นทางในการส่งแพคเกตข้อมูล (D: S, A, B, C, E, F, D) เมื่อโหนด E ขาดการเชื่อมต่อกับโหนด F โหนด E จะลบเส้นทางเก่านี้ และตรวจสอบข้อมูลเส้นทางในแคชเส้นทางไปปยังโหนด D ก่อน ถ้าหากไม่มีเส้นทางใหม่ในแคชเส้นทาง โหนด E จะยังไม่ส่ง RERR ไปยังโหนดเริ่มต้น แต่จะทำ local route recovery โดยการแพร่กระจาย RREQ เพื่อค้นหาเส้นทางไปยังโหนดปลายทาง เมื่อโหนด E ได้รับ RREP กลับมาจากโหนดปลายทาง โหนด E จะใช้เส้นทางใหม่นี้แทนเส้นทางเดิม แต่ถ้าหากโหนด

E ไม่สามารถหาเส้นทางใหม่ได้ โหนด E จึงจะส่ง RERR ไปยังโหนดเริ่มต้น เพื่อให้โหนดเริ่มต้นทำการค้นหาเส้นทางใหม่ต่อไป

วิธีการนี้อาศัยข้อมูลในแคชเส้นทางเป็นพื้นฐานในการหาเส้นทางใหม่ ถ้าโหนด E มีเส้นทางใหม่ในแคชเส้นทางและเป็นเส้นทางที่ใช้ไม่ได้ วิธีการนี้ก็ยังคงมีปัญหาเช่นเดิม แต่ถ้าในกรณีที่โหนด E ไม่มีเส้นทางไปโหนดปลายทาง โหนดจะเสียเวลาในการค้นหาเส้นทางใหม่เพื่อนำพาแพคเกตข้อมูลไปให้ถึงโหนด D ซึ่งแพคเกตจะไม่สูญหายแต่จะมีค่าหน่วงเวลาของแพคเกตเกิดขึ้นเล็กน้อย การที่โหนด E ไม่ส่ง RERR ไปให้โหนด S ทราบทันที อาจจะไม่เป็นผลดีนักเพราะจะทำให้โหนดเริ่มต้นใช้เส้นทางเดิมตลอด เนื่องจากโพรโทคอล DSR เป็นโพรโทคอลที่เป็นแบบ source route ดังนั้นเมื่อแพคเกตข้อมูลที่ส่งจากโหนด S ส่งมาถึงยังโหนด E โหนด E ยังคงส่งไปไม่ถึงโหนด F ทำให้เกิดการขาดการเชื่อมต่อ แต่โหนด E อาจนำพาแพคเกตข้อมูลนี้ไปยังโหนด D ได้ เนื่องจากได้เส้นทางใหม่ที่เพิ่งค้นหามา ซึ่งการกระทำนี้เป็นการพึงพิงการทำ salvage packet เพราะการที่แพคเกตข้อมูลมาถึงโหนด E จะต้องมีการขาดการเชื่อมต่อก่อน โหนดจึงจะใช้เส้นทางได้ ซึ่งถ้าเส้นทางที่มีอยู่ใช้ไม่ได้แพคเกตก็จะสูญหาย ดังนั้นควรจะแจ้งให้โหนด S ทราบถึงเส้นทางที่ได้ขาดการเชื่อมต่อนี้ เพื่อหลีกเลี่ยงการใช้เส้นทางดังกล่าว แต่นั่นจะต้องหมายความว่าข้อมูลเส้นทางในแคชเส้นทางของโหนด S จะต้องใช้งานได้ด้วย เพราะถ้าใช้งานไม่ได้ การทำ salvage packet ต่อไปคงจะดีกว่าแล้วผลักภาระให้กับโหนด E ดูแลรักษาเส้นทางต่อไป



ภาพที่ 13 การค้นหาเส้นทางใหม่เมื่อเส้นทางเดิมขาดการเชื่อมต่อด้วยวิธีการของ Sengul

นอกจากงานวิจัยด้านการปรับปรุงประสิทธิภาพที่เกี่ยวข้องกับเส้นทางที่ใช้ในการส่งแพคเกตข้อมูลของโพรโทคอล DSR แล้ว ได้มีงานวิจัยที่ปรับปรุงประสิทธิภาพของแคชเส้นทางโดย Marina and Das, 2001 ได้นำเสนอวิธีปรับปรุงข้อมูลเส้นทางในแคชเส้นทางให้เป็นข้อมูลที่

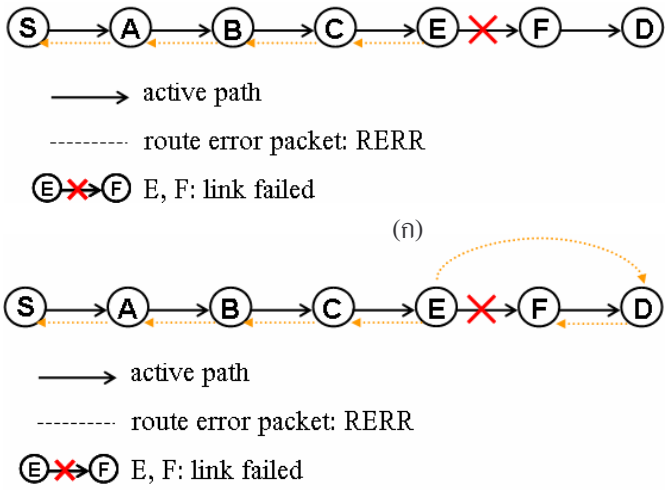
สามารถใช้งานได้ให้มากที่สุด ด้วย 3 วิธีการคือวิธี wider error notification นั่นคือการแจ้งความผิดพลาดของเส้นทางให้กับโหนดต่างๆ ในเครือข่ายได้รับทราบด้วยการแพร่กระจาย RERR เพื่อให้โหนดต่างๆ ที่ได้รับ RERR ทำการปรับปรุงข้อมูลในแคชเส้นทาง วิธี timer-based route expiry วิธีการนี้ทำได้ด้วยการกำหนดอายุของเส้นทางที่อยู่ในแคชเส้นทางเพื่อกำจัดเส้นทางเก่าๆ ที่ไม่ได้ใช้งานนำออกจากแคชเส้นทาง และวิธี negative caches จะเก็บข้อมูลเส้นทางการเชื่อมต่อที่ขาดการเชื่อมต่อไปแล้วมาใช้เปรียบเทียบกับแพคเกตต่างๆ ที่รับเข้ามาเพื่อกำจัดแพคเกตนั้นทิ้ง และไม่บันทึกเส้นทางที่มีการใช้เส้นทางที่เป็นเส้นทางที่ได้ขาดการเชื่อมต่อลงในแคชเส้นทาง

จะเห็นว่า นักวิจัยกลุ่มนี้มองที่ตัวปัญหาหลักของโพรโทคอล DSR เนื่องจากเชื่อว่าการจัดการกับแคชเส้นทางโดยตรงน่าจะเป็นการดีที่สุด ทั้งนี้ก็เพื่อให้แคชเส้นทางที่โหนดเก็บไว้นั้น มีเส้นทางที่ยังคงใช้งานได้ให้มากที่สุด แต่ว่าโพรโทคอล DSR จะเลือกใช้เส้นทางที่สั้นที่สุดเพียงเส้นทางเดียวในแต่ละเส้นทางการส่งแพคเกตข้อมูลที่เก็บอยู่ในแคชเส้นทาง ซึ่งวิธีการจัดการกับแคชเส้นทางนี้ จะได้ประโยชน์มากในกรณีที่โหนดตัวกลางขาดการเชื่อมต่อบ่อยๆ โดยมองในแง่ของการจัดการข้อมูลในแคชเส้นทาง โดยโหนดในเครือข่ายก็จะลบเส้นทางได้บ่อยๆ และคาดว่าข้อมูลในแคชเส้นทางนี้น่าจะยังคงใช้งานได้เยอะขึ้น อีกประการหนึ่งก็เพื่อกำจัดแพคเกตอื่นๆ ที่คาดว่าจะมีเส้นทางที่ใช้ไม่ได้ แต่การขาดการเชื่อมต่อบ่อยๆ ใช่ว่าจะเป็นสิ่งที่ดี ซึ่งแพคเกตข้อมูลมีความเสี่ยงที่จะสูญหายได้มากขึ้นด้วย วิธีการนี้จะต้องเพิ่มหน่วยความจำไว้สำหรับเก็บข้อมูลเส้นทางที่ได้ขาดการเชื่อมต่อไปแล้วด้วย

นอกจากนี้ยังมีนักวิจัยได้นำเสนอวิธีการในการกำจัดเส้นทางเก่าๆ ในแคชเส้นทางไว้ด้วย โดย Yu and Kedem, 2005 ได้นำเสนอวิธีการปรับปรุงแคชเส้นทางด้วยการแจ้งความผิดพลาดของเส้นทางที่เกิดขึ้นแก่โหนดต่างๆ ที่มีเส้นทางที่ผิดพลาดนั้นอยู่ในแคชเส้นทาง เพื่อให้โหนดเหล่านั้นลบเส้นทางที่ผิดพลาดออกจากแคชเส้นทางทั้งนี้เพื่อให้แคชเส้นทางมีข้อมูลเส้นทางที่เป็นเส้นทางเก่าที่ใช้งานไม่ได้ให้น้อยที่สุดโดยแบ่งเป็นสองกรณีคือกรณีที่เส้นทางขาดการเชื่อมต่อขณะที่โหนดยังไม่ได้ใช้งานเส้นทางดังกล่าวหลังจากที่โหนดเริ่มต้นได้รับ RREP แล้ว และอีกกรณีคือเส้นทางขาดการเชื่อมต่อขณะที่โหนดได้เคยใช้เส้นทางนั้นไปแล้วซึ่งแสดงวิธีการได้ดังภาพที่ 14

ภาพที่ 14 (ก) หลังจากที่โหนด S ได้รับ RREP จากโหนด D โหนดต่างๆ ที่ได้รับและดักฟัง RREP นี้จะมีเส้นทางที่โหนด S อาจจะใช้ในการส่งแพคเกตคือ (D: S, A, B, C, E, F, D) เมื่อโหนด S ส่งแพคเกตข้อมูลลำดับแรกด้วยเส้นทางนี้ จนกระทั่งถึงโหนด E แล้วโหนดเกิดขาดการ

เชื่อมต่อกับโหนด F ก่อนที่จะได้ใช้เส้นทางนี้ในการส่งแพคเกตข้อมูล โหนด E จะส่ง RERR ระหว่างโหนด E และโหนด F ย้อนกลับไปถึงโหนด S เพื่อให้โหนด C, B, A และ S ลบข้อมูลเส้นทางในแคชเส้นทาง



ภาพที่ 14 วิธีการปรับปรุงแคชเส้นทางของโพรโทคอล DSR ด้วยวิธีการของ Yu and Kedem
(ก) เมื่อเส้นทางยังไม่เคยใช้ส่งแพคเกตข้อมูลมาก่อน
(ข) เมื่อเส้นทางได้เคยใช้ส่งแพคเกตข้อมูลมาแล้ว

ภาพที่ 14 (ข) หลังจากที่โหนด S ใช้งานเส้นทาง (D: S, A, B, C, E, F, D) เพื่อส่งแพคเกตข้อมูล และใช้งานเส้นทางนี้ไปได้ระยะหนึ่ง โหนด E เกิดขาดการเชื่อมต่อกับโหนด F โหนด E จะส่ง RERR ระหว่างโหนด E และโหนด F ย้อนกลับไปถึงโหนด S เพื่อให้โหนด C, B, A และ S ลบข้อมูลเส้นทางในแคชเส้นทาง และขณะเดียวกันโหนด E จะหาเส้นทางอื่นๆ เพื่อส่ง RERR ไปยังโหนด D และให้โหนด D หาเส้นทางเพื่อส่ง RERR ให้กับโหนด F ด้วย ทั้งนี้ก็เพื่อให้โหนดที่มีเส้นทางในการส่งแพคเกตด้วยเส้นทาง (D: S, A, B, C, E, F, D) ทำการกำจัดเส้นทางดังกล่าวออกจากแคชเส้นทางของตนเอง

ด้วยวิธีการนี้จะเห็นว่า แม้โหนด F จะขาดการเชื่อมต่อกับโหนด E ไม่ว่าโหนด E หรือโหนด F จะเคลื่อนที่ออกไปก็ตาม โหนด F จะยังมีโอกาสได้ลบเส้นทางเก่านี้ออกไป โดยอาศัยโหนด D ส่ง RERR มาให้ ซึ่งนั่นหมายถึงว่าโหนด E จะต้องมีเส้นทางที่ไปถึงโหนด D และโหนด D ยังต้องมีเส้นทางไปถึงโหนด F ด้วยจึงจะสำเร็จได้ แต่ถ้าหากโหนด E หรือโหนด D ไม่มีเส้นทางรองรับการกระทำนี้ก็ไม่สำเร็จผล โหนด F ก็ยังคงมีเส้นทางที่ใช้ไม่ได้อยู่แคชเส้นทางเช่นเดิม

อุปกรณ์และวิธีการ

อุปกรณ์

1. ฮาร์ดแวร์ระบบ

1.1 เครื่องคอมพิวเตอร์ Notebook ซีพียู Mobile Pentium 1.4 กิกะเฮิร์ต หน่วยความจำขนาด 1 กิกะเฮิร์ต ฮาร์ดดิสก์ 20 กิกะไบต์

1.2 เครื่องคอมพิวเตอร์แม่ข่าย Nilapat และ Sukreep

1.3 เครื่องคอมพิวเตอร์ตั้งโต๊ะ หน่วยประมวลผลความเร็ว 3 กิกะเฮิร์ต หน่วยความจำขนาด 512 เมกะไบต์ หน่วยเก็บข้อมูลความจุ 40 กิกะไบต์

2. ซอฟต์แวร์ระบบ

2.1 โปรแกรมจำลองการทำงานเครือข่ายไร้สาย GloMoSim (Terwilliger and Meyer, 2001) บนระบบปฏิบัติการลีนุกซ์

2.2 ระบบปฏิบัติการ Fedora Core4

2.3 ภาษาเขียนโปรแกรมซี

2.4 ตัวคอมไพเลอร์ GNU C

2.5 VIM editor

2.6 BASH shell script

วิธีการ

1. โพรโทคอล DSR-BA (DSR with break avoidance)

จากปัญหาของโพรโทคอล DSR ที่เกิดขึ้นและวิธีการแก้ปัญหาในแบบต่างๆ ของกลุ่มนักวิจัยที่ได้นำเสนอมาเพื่อแก้ปัญหาของโพรโทคอลนี้ทั้งในส่วนของการจัดการเส้นทางการเชื่อมต่อหรือการจัดการข้อมูลเส้นทางในแคชเส้นทาง ซึ่งวิธีการทั้งหลายเหล่านั้นจะกระทำการหลังจากที่โหนดได้ขาดการเชื่อมต่อไปแล้ว ดังนั้นในงานวิจัยนี้จึงได้นำเสนอโพรโทคอล DSR-BA ซึ่งทำการปรับปรุงโพรโทคอล DSR ในส่วนของการหาเส้นทางใหม่มารองรับเพื่อใช้ทดแทนเส้นทางเดิมก่อนที่เส้นทางการติดต่อสื่อสารเดิมจะขาดการเชื่อมต่อ โดยการนำพารามิเตอร์การตรวจสอบอันตรายที่อาจจะเกิดขึ้นได้กับเส้นทางการเชื่อมต่อของโพรโทคอล AODV-BA มาปรับใช้กับโพรโทคอล DSR-BA ซึ่งในที่นี้ได้นำค่าความแรงของสัญญาณวิทยุที่รับได้โดยโหนดที่เป็นเครื่องรับสัญญาณมาปรับใช้เพื่อที่จะตรวจสอบเส้นทางการติดต่อสื่อสารที่คาดว่าอาจจะขาดการเชื่อมต่อของเส้นทางขึ้นได้ โดยให้โหนด downstream ทำการตรวจสอบพารามิเตอร์นี้ เมื่อตรวจพบปัญหาแล้วจะแจ้งไปยังโหนด upstream ถึงอันตรายที่อาจเกิดขึ้นกับเส้นทางการติดต่อระหว่างสองโหนดนี้ ซึ่งก่อนที่เส้นทางจะขาดการเชื่อมต่อนั้นโพรโทคอล DSR-BA จะทำหน้าที่หาเส้นทางใหม่มารองรับเพื่อใช้งานแทนเส้นทางเดิม

โพรโทคอล DSR-BA มีขั้นตอนการทำงานคือ ในส่วนของกระบวนการค้นหาเส้นทางนั้นจะเหมือนกับโพรโทคอล DSR และหลังจากที่โหนดเริ่มต้นได้เส้นทางและทำการส่งแพคเกตข้อมูลแล้ว มีบางโหนดตรวจพบว่าเส้นทางอาจจะขาดการเชื่อมต่อ จากนั้นโหนดที่ตรวจพบนี้จะทำการเลือกโหนดให้ทำหน้าที่ดูแลรักษาเส้นทางก่อนที่เส้นทางที่ใช้งานอยู่เดิมจะขาดการเชื่อมต่อหลังจากเลือกโหนดได้แล้ว ถ้าโหนดที่ถูกเลือกเป็นโหนด upstream โหนดจะทำหน้าที่ในการตรวจสอบข้อมูลเส้นทางในแคชเส้นทางของตนเองไปยังโหนดปลายทางเพื่อหาเส้นทางรองรับการใช้งานมาไว้ในแคชเส้นทางก่อนที่โหนดในเส้นทางจะขาดการเชื่อมต่อและส่ง RERR กลับไปยังโหนดเริ่มต้น แต่ถ้าโหนดที่ถูกเลือกเป็นโหนด downstream โหนดจะทำหน้าที่ในการตรวจสอบข้อมูลเส้นทางในแคชเส้นทางของตนเองไปยังโหนดเริ่มต้นโดยเส้นทางที่ได้นี้จะเป็นตัวเลือกหนึ่งเพื่อให้โหนดเริ่มต้นเลือกใช้เพื่อส่งแพคเกตข้อมูลลำดับต่อไป

1.1 กระบวนการในการตรวจสอบเส้นทางที่คาดว่าอาจจะขาดการเชื่อมต่อ มีดังนี้คือ กำหนดให้โหนด downstream ทำหน้าที่ในการตรวจสอบกำลังงานที่รับได้ (received power) จากกำลังงงานที่โหนด upstream ทำการส่งแพคเกตมาให้ หากกำลังงานที่รับได้ต่ำกว่าค่าขีดเริ่มเปลี่ยนที่กำหนดและเป็นแพคเกตข้อมูล จากนั้นโหนดจึงจะทำกระบวนการเลือกโหนดให้ทำหน้าที่ดูแลรักษาเส้นทาง

1.2 กระบวนการเลือกโหนดให้ทำหน้าที่ดูแลรักษาเส้นทาง คือ เมื่อการเชื่อมต่อเส้นทางระหว่างโหนด upstream และ downstream เกิดปัญหาอาจจะขาดการเชื่อมต่อ ถ้าจำนวนฮอพที่นับจากโหนด downstream ไปยังโหนดโหนดเริ่มต้นมีจำนวนที่น้อยกว่าจำนวนฮอพที่นับจากโหนด upstream ไปยังโหนดปลายทาง ในที่นี้โหนดที่ถูกเลือกคือโหนด downstream แต่ถ้าจำนวนฮอพที่นับจากโหนด uptream ไปยังโหนดปลายทางมีจำนวนที่น้อยกว่าจำนวนฮอพที่นับจากโหนด downstream ไปยังโหนดเริ่มต้น ในที่นี้โหนดที่ถูกเลือกคือโหนด upstream

1.3 ขั้นตอนในการดูแลรักษาเส้นทาง หลังจากเลือกโหนดแล้วโหนดที่ถูกเลือกจะมีขั้นตอนในการดูแลรักษาเส้นทางคือ

1.3.1 กรณีโหนดที่ถูกเลือกเป็นโหนด upstream ในที่นี้โหนด downstream จะทำการส่งแพคเกตแจ้งความผิดพลาดของเส้นทางที่ควรหลีกเลี่ยง (route error packet with breaking avoidance: RERRBA) ไปยังโหนดเริ่มต้นพร้อมทั้งส่งแพคเกตไปแจ้งให้กับโหนด upstream ทราบว่าได้ถูกเลือกให้ทำหน้าที่ดูแลรักษาเส้นทาง เมื่อโหนด upstream รับทราบจึงทำการส่งแพคเกตตรวจสอบความพร้อมใช้งานของเส้นทาง (route refresh packet: RREF) ไปตรวจสอบเส้นทางที่ไปยังโหนดปลายทาง ซึ่งโหนด upstream จะกระทำการส่ง RREF เพียงครั้งเดียว จากนั้นเมื่อโหนดปลายทางได้รับแพคเกตนี้แล้ว จะตอบกลับย้อนมาตามเส้นทางที่ระบุในแพคแกตตรวจสอบความพร้อมใช้งานของเส้นทางนี้ ด้วยแพคเกตตอบกลับความพร้อมใช้งานของเส้นทาง (route refresh reply packet: RFRP) จนกระทั่งถึงโหนด upstream จากนั้นโหนด upstream จะลบเส้นทางที่มี next node เป็นโหนด downstream และเป็นเส้นทางไปโหนดปลายทาง ทั้งนี้เพื่อให้เส้นทางอื่นๆ ที่ใช้การเชื่อมต่อด้วยโหนดคู่นี้ยังคงใช้งานได้ ถ้าการตรวจสอบเส้นทางที่อาจจะขาดการเชื่อมต่อแล้วโหนดดังกล่าวเกิดขาดการเชื่อมต่อขึ้นจริง โหนด upstream จะใช้เส้นทางที่โหนดได้ทำการตรวจสอบไปแล้วนี้ในการนำพาแพคเกตข้อมูลไปยังโหนดปลายทาง

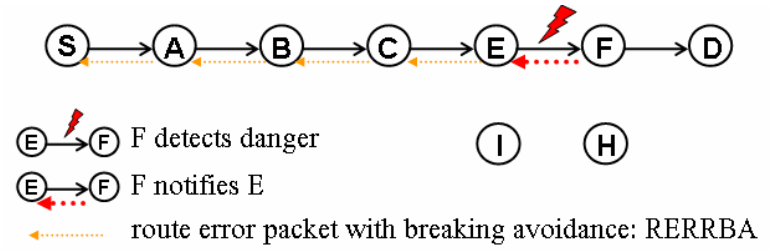
ถ้าขณะที่โหนด upstream ไม่สามารถส่ง RREF ไปยัง next node ในเส้นทางใหม่ได้ แสดงว่าข้อมูลเส้นทางในแคชเส้นทางที่ถูกเตรียมไว้ใช้เมื่อโหนดขาดการเชื่อมต่อนั้นจะเป็นเส้นทางที่ใช้ไม่ได้ โหนด upstream จะลบเส้นทางใหม่นี้ออกจากแคชเส้นทาง ซึ่งเมื่อโหนดขาดการเชื่อมต่อเส้นทางนี้จะไม่ถูกเรียกใช้งาน แต่ยังไม่แน่ว่าเส้นทางลำดับถัดไปจะยังใช้งานได้หรือไม่

ถ้าขณะที่โหนด upstream สามารถส่ง RREF ไปยัง next node ในเส้นทางใหม่ได้ แต่ส่งไปยังโหนดต่อๆ ไปที่ระบุไว้ในเส้นทางไม่ได้ หรือไม่ได้รับ RFRP หรือ RFRP ส่งไปยังโหนดต่อๆ ไปไม่ได้เช่นกัน เมื่อเป็นเช่นนี้โหนด upstream จะยังคงไม่ทราบว่าเส้นทางดังกล่าวนั้นเป็นเส้นทางที่ใช้งานไม่ได้แล้ว ซึ่งขณะเดียวกันโหนดที่ส่งต่อไปไม่ได้นั้นจะลบเส้นทางที่มี next node เป็นโหนดที่ส่งไปไม่ได้นี้ออกจากแคชเส้นทาง ซึ่งถ้าการตรวจสอบเส้นทางที่อาจจะขาดการเชื่อมต่อแล้วคู่โหนดดังกล่าวเกิดขาดการเชื่อมต่อขึ้นจริง โหนด upstream จะยังคงใช้เส้นทางนี้และอาจจะเกิดการสูญหายแพคเกตข้อมูล แต่ถ้าการตรวจสอบเส้นทางที่อาจจะขาดการเชื่อมต่อแล้วคู่โหนดดังกล่าวไม่ขาดการเชื่อมต่อ โหนดก็ยังคงสามารถใช้งานเส้นทางได้ตามปกติ

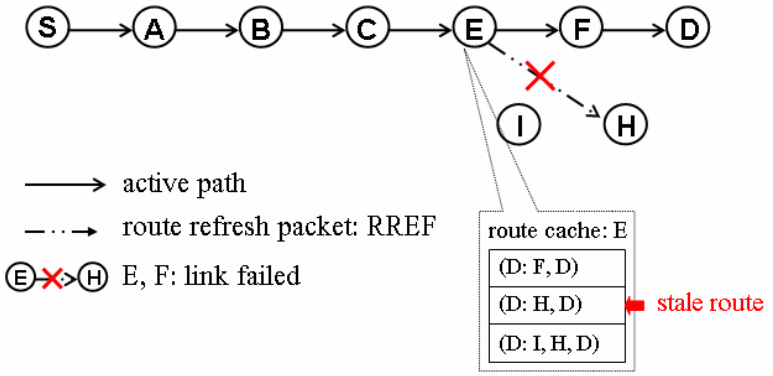
เมื่อ RERRBA ไปถึงยังโหนดเริ่มต้น โหนดจะลบทุกเส้นทางที่มีการเชื่อมต่อที่ระบุไว้นี้ จากนั้นโหนดจะเลือกเส้นทางใหม่จากแคชเส้นทางของตนเองโดยหลีกเลี่ยงการใช้เส้นทางเดิมที่อาจจะขาดการเชื่อมต่อนี้ ซึ่งเส้นทางใหม่นี้อาจจะเป็นเส้นทางเก่าที่ไม่สามารถใช้งานได้ ถึงแม้จะได้เส้นทางใหม่ด้วยการแพร่กระจายแพคเกตร้องขอ โหนดเริ่มต้นก็อาจได้รับเส้นทางใหม่ที่เป็นเส้นทางที่ไม่อาจใช้งานได้จากการตอบกลับเส้นทางด้วยโหนดกลางทางที่มีเส้นทางเก่าในแคชเส้นทางที่อาจขาดการเชื่อมต่อไปแล้ว

จากการที่โหนด upstream ยังคงไม่ทราบว่าเส้นทางดังกล่าวนั้นใช้งานไม่ได้แล้ว อาจแก้ไขได้ด้วยการแจ้งความผิดพลาดของเส้นทางที่เกิดขึ้นกลับมายังโหนด upstream หรือการกำหนดเวลารอคอย RFRP แก่โหนด upstream ทั้งนี้ก็เพื่อให้โหนด upstream และโหนดอื่นๆ ที่ได้รับหรือดักฟังอยู่ทำการลบเส้นทางที่ใช้ไม่ได้นี้ออกจากแคชเส้นทาง

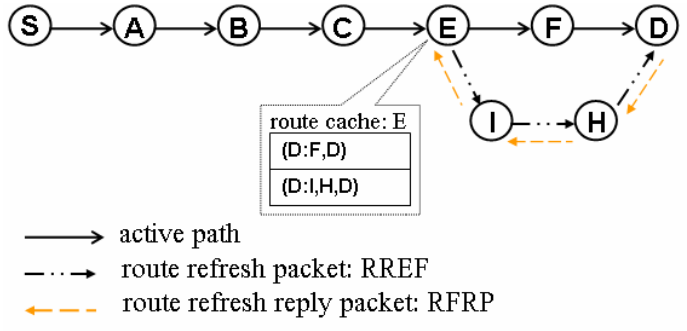
ถ้าหากมีการตรวจพบปัญหาเส้นทางอาจขาดการเชื่อมต่อเกิดขึ้นใกล้กับโหนดปลายทางซึ่งโหนด upstream ถูกเลือกให้ทำหน้าที่ตรวจสอบเส้นทางดังภาพที่ 15



(ก)



(ข)



(ค)

ภาพที่ 15 โพรโทคอล DSR-BA เมื่อโหนด F ตรวจพบปัญหาที่อาจเกิดขึ้นกับเส้นทาง
(ก) เมื่อปัญหาอาจเกิดขึ้นระหว่างโหนด E และโหนด F
(ข) เมื่อโหนด E มีเส้นทางลำดับถัดไปเป็น stale route
(ค) เมื่อโหนด E มีเส้นทางลำดับถัดไปที่พร้อมใช้งาน

ภาพที่ 15 (ก) เมื่อมีปัญหาเกิดขึ้นระหว่างโหนด E และโหนด F ซึ่งอยู่ใกล้กับโหนดปลายทาง โหนด F เป็นโหนด downstream ตรวจพบถึงอันตรายที่อาจเกิดขึ้นได้จึงส่ง RERRBA ไปยังโหนดเริ่มต้นและส่งข้อความแจ้งไปยังโหนด upstream คือโหนด E หลังจากที่โหนด E ได้รับการแจ้งเตือนจากโหนด downstream โหนด E จะทราบว่าตนเองมีหน้าที่ในการตรวจสอบเส้นทางการติดต่อสื่อสารเส้นทางใหม่ก่อนที่เส้นทางเดิมจะขาดการเชื่อมต่อ จากนั้น

โหนด E จะส่ง RREF เพื่อตรวจสอบเส้นทางไปยังโหนดปลายทางคือโหนด D ในขณะที่เส้นทางเดิมคือ (D: S, A, B, C, E, F, D) จะยังคงสามารถใช้งานได้ตามปกติ

ภาพที่ 15 (ข) หลังจากที่โหนด E รับทราบหน้าที่แล้ว โหนด E จะส่ง RREF ไปตรวจสอบเส้นทางที่ไปยังโหนด D ในเส้นทาง (D: E, H, D) ซึ่งเป็นเส้นทางที่สั้นที่สุดที่มีอยู่ในแคชเส้นทางของตนเองแต่เส้นทางนี้เป็นเส้นทางที่ใช้งานไม่ได้แล้วเนื่องจากโหนด H ขาดการเชื่อมต่อกับโหนด E ดังนั้นโหนด E ไม่สามารถส่ง RREF ให้กับโหนด H ได้ โหนด E จึงลบเส้นทางทุกเส้นทางที่มี next node เป็นโหนด H ออกจากแคชเส้นทางของตนเอง โดยโหนด E จะกระทำการตรวจสอบเส้นทางในแคชเส้นทางเพียงเส้นทางเดียวเท่านั้น

ถ้าหากโหนด E ขาดการเชื่อมต่อกับโหนด F จริงๆ โหนด E จะทำ salvage ด้วยเส้นทางใหม่คือ (D: E, I, H, D) แทนการใช้เส้นทาง (D: E, H, D) ส่งผลให้ไม่สูญหายแพคเกตข้อมูล แต่ถ้าหากโหนด E ไม่ขาดการเชื่อมต่อกับโหนด F จริงๆ โหนด E ก็จะยังคงส่งแพคเกตข้อมูลได้ตามปกติ

ภาพที่ 15 (ค) หลังจากที่โหนด E รับทราบหน้าที่แล้ว โหนด E จะส่ง RREF ไปตรวจสอบเส้นทางที่ไปยังโหนด D ในเส้นทาง (D: E, I, H, D) ซึ่งเป็นเส้นทางที่สั้นที่สุดที่มีอยู่ในแคชเส้นทางของตนเองและเส้นทางนี้เป็นเส้นทางที่ยังคงใช้งานได้ เมื่อ RREF ส่งไปถึงยังโหนด D โหนด D จะส่ง RFRP ย้อนกลับมายังโหนด E จากนั้นโหนด E เมื่อได้รับ RFRP โหนดจะลบเส้นทางที่มี next node เป็นโหนด F และเป็นเส้นทางไปยังโหนด D ออกจากแคชเส้นทางของตนเอง

ถ้าหากโหนด E ขาดการเชื่อมต่อกับโหนด F จริงๆ โหนด E จะทำ salvage ด้วยเส้นทางใหม่คือ (D: E, I, H, D) ซึ่งโหนดได้ตรวจสอบเส้นทางไปแล้ว ดังนั้นแพคเกตข้อมูลอาจจะไม่ต้องสูญหายไป แต่ยังไม่แน่ว่าขณะที่ตรวจสอบเส้นทางอยู่นั้น เส้นทางยังคงใช้งานได้ แต่เมื่อต้องการใช้งานจริงๆ โหนดอาจกำลังจะขาดการเชื่อมต่อไปก็ได้ แต่ถ้าหากโหนด E ไม่ขาดการเชื่อมต่อกับโหนด F จริงๆ โหนด E ก็จะยังคงส่งแพคเกตข้อมูลได้ตามปกติ

1.3.2 กรณีโหนดที่ถูกเลือกเป็นโหนด downstream ในที่นี้โหนด downstream จะทำการส่ง RERRBA ไปยังโหนดเริ่มต้นพร้อมทั้งส่ง RREF ไปตรวจสอบเส้นทางที่ไปยังโหนดเริ่มต้นเพียงครั้งเดียว จากนั้นเมื่อโหนดเริ่มต้นได้รับแพคเกตนี้แล้ว โหนดจะลบเส้นทางที่มีการเชื่อมต่อ

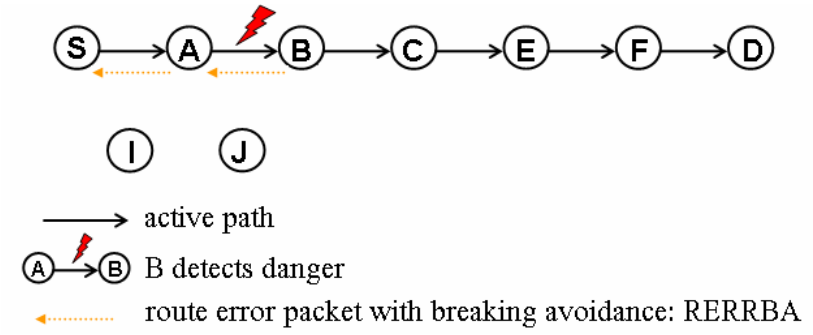
ระหว่างคู่โหนดที่อาจจะขาดการเชื่อมต่อและเป็นเส้นทางไปยังโหนดปลายทางออกจากแคชเส้นทาง ทั้งนี้เพื่อให้เส้นทางอื่นๆ ยังคงใช้งานได้อยู่ และโหนดเริ่มต้นจะนำเส้นทางที่ระบุในแพคแกตนี้ไปเก็บไว้ในแคชเส้นทางของตนเอง จากนั้นเมื่อโหนดเริ่มต้นจะส่งแพคเกตข้อมูลลำดับต่อไป โหนดจะเลือกเส้นทางใหม่ในแคชเส้นทาง ซึ่งอาจจะเป็นเส้นทางที่โหนดได้รับมาจาก RREF หรืออาจจะเป็นเส้นทางอื่นๆ ก็ได้ เนื่องจากโพรโทคอล DSR จะเลือกเส้นทางที่สั้นที่สุดที่มีอยู่มาใช้งาน

ถ้าการตรวจสอบเส้นทางที่อาจจะขาดการเชื่อมต่อแล้วคู่โหนดดังกล่าวเกิดขาดการเชื่อมต่อขึ้นจริง โหนด upstream จะลบเส้นทางที่ขาดการเชื่อมต่อนี้และหาเส้นทางใหม่จากแคชเส้นทางของตนเอง เพื่อนำพาแพคเกตข้อมูลไปยังโหนดปลายทาง แต่ถ้าเส้นทางใช้ไม่ได้จะเกิดการสูญหายแพคเกตข้อมูล ในขณะที่แพคเกตข้อมูลลำดับต่อมาอาจจะไม่ได้ใช้เส้นทางที่ต้องผ่านโหนด upstream หรืออาจจะใช้เส้นทางอื่นๆ แทนเส้นทางเดิมนี้ก็ไม่ได้หมายความว่าเส้นทางใหม่จะยังคงใช้งานได้ ซึ่งอาจจะมีบางโหนดขาดการเชื่อมต่อไปแล้วก็ได้

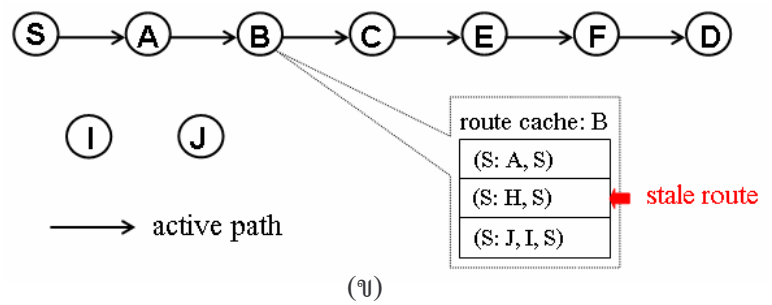
ถ้าโหนด upstream ส่ง RREF ไปไม่ถึงโหนดเริ่มต้น โหนดที่ส่งแพคเกตไปไม่ถึงนั้นจะลบเส้นทางที่ใช้ไม่ได้นี้ออกจากแคชเส้นทาง เมื่อได้ทำตรวจสอบเส้นทางที่อาจจะขาดการเชื่อมต่อแล้วโหนดดังกล่าวเกิดขาดการเชื่อมต่อขึ้นจริง การทำงานจะก็ยังคงเป็นไปตามปกติของโพรโทคอล DSR

ถ้าหากมีการตรวจพบปัญหาเส้นทางอาจขาดการเชื่อมต่อเกิดขึ้นใกล้กับโหนดเริ่มต้นซึ่งโหนด downstream ถูกเลือกให้ทำหน้าที่ตรวจสอบเส้นทางดังภาพที่ 16 โดยภาพที่ 16 (ก) คือ เมื่อปัญหาเกิดขึ้นระหว่างโหนด A และโหนด B ซึ่งอยู่ใกล้กับโหนดเริ่มต้น โหนด B เป็นโหนด downstream ตรวจพบถึงอันตรายที่อาจเกิดขึ้นได้กับโหนด upstream (โหนด A) โหนด B จึงส่ง RERRBA ไปยังโหนดเริ่มต้นพร้อมทั้งส่ง RREF เพื่อตรวจสอบเส้นทางไปยังโหนด S ในขณะที่เส้นทางเดิมคือ (D: S, A, B, C, E, F, D) จะยังคงสามารถใช้งานได้ตามปกติ ในกรณีนี้โหนด B มีเส้นทางที่ไปยังโหนด S แต่เส้นทางใช้งานไม่ได้ เมื่อโหนด B ส่ง RREF ด้วยเส้นทาง (S: B, H, S) โหนด S จะยังไม่มีการเปลี่ยนแปลงใดๆ แต่โหนด B จะลบเส้นทางที่มี next node เป็นโหนด H ออกจากแคชเส้นทาง

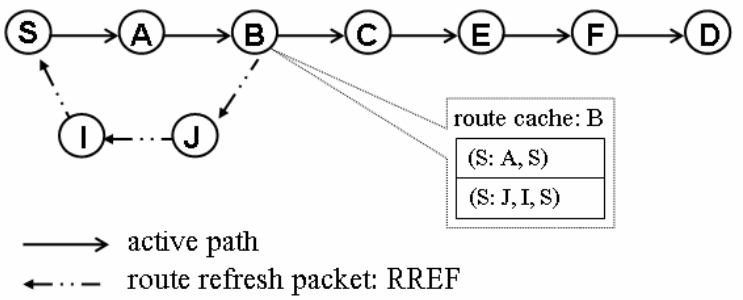
ถ้าหากโหนด A ขาดการเชื่อมต่อกับโหนด B จริงๆ โหนด A จะทำ salvage ด้วยเส้นทางใหม่ที่มีในแคชเส้นทางของตนเอง ซึ่งถ้าเส้นทางใหม่ใช้ได้ก็จะไม่เกิดแพคเกตสูญหาย แต่ถ้าเส้นทางใหม่ใช้ไม่ได้แพคเกตก็จะสูญหาย ในที่นี้โหนด S จะยังไม่รับทราบถึงอันตรายที่อาจจะเกิดขึ้นได้ โดยโหนด S จะเปลี่ยนเส้นทางใหม่ก็ต่อเมื่อได้รับ RERRBA หรือได้รับ RERR เมื่อโหนด A ไม่สามารถส่งแพคเกตข้อมูลไปถึงโหนด B ได้ แต่ถ้าหากโหนด A ไม่ขาดการเชื่อมต่อกับโหนด B จริงๆ โหนด A ก็จะยังคงส่งแพคเกตข้อมูลได้ตามปกติ



(ก)



(ข)



(ค)

ภาพที่ 16 โพรโทคอล DSR-BA เมื่อโหนด B ตรวจพบปัญหาที่อาจเกิดขึ้นกับเส้นทาง

(ก) เมื่อปัญหาอาจเกิดขึ้นระหว่างโหนด A และโหนด B

(ข) เมื่อโหนด B มีเส้นทางลำดับถัดไปเป็น stale route

(ค) เมื่อโหนด B มีเส้นทางลำดับถัดไปที่พร้อมใช้งาน

ภาพที่ 16 (ข) หลังจากที่โหนด B รับทราบหน้าที่แล้ว โหนด B จะส่ง RREF ไปตรวจสอบเส้นทางที่ไปยังโหนด S ในเส้นทาง (S: B, J, I, S) ซึ่งเป็นเส้นทางที่สั้นที่สุดที่มีอยู่ในแคชเส้นทางของตนเองและเส้นทางนี้เป็นเส้นทางที่ยังคงใช้งานได้ เมื่อ RREF ส่งไปถึงยังโหนด S แล้วโหนด S จะลบเส้นทางที่มีคู่โหนด A และ B และเป็นเส้นทางไปยังโหนด D ออกจากแคชเส้นทางของตนเอง จากนั้นจึงนำเส้นทางที่ได้จาก RREF ใส่ในแคชเส้นทางของตนเอง เมื่อโหนด S จะส่งแพคเกตข้อมูลลำดับต่อไป โหนดจะเลือกเส้นทางในแคชเส้นทาง ซึ่งเส้นทางที่เลือกนั้นอาจจะเป็นเส้นทางจาก RREF นี้หรือเป็นเส้นทางอื่นๆ ก็ได้

ถ้าหากโหนด A ขาดการเชื่อมต่อกับโหนด B จริงๆ โหนด A จะลบเส้นทางนี้ออกจากแคชเส้นทางและทำ salvage ด้วยเส้นทางใหม่จากแคชเส้นทาง ถ้าเส้นทางดังกล่าวใช้งานได้ แพคเกตข้อมูลอาจจะไม่ต้องสูญหายไป แต่ถ้าใช้ไม่ได้แพคเกตก็จะสูญหาย เมื่อโหนด A ส่ง RERR ไปยังโหนด S โหนด S จะลบทุกเส้นทางที่มีการเชื่อมต่อระหว่างโหนด A และโหนด B ออกจากแคชเส้นทาง แต่ถ้าหากโหนด A ไม่ขาดการเชื่อมต่อกับโหนด B จริงๆ โหนด A ก็จะยังคงส่งแพคเกตข้อมูลได้ตามปกติ

2. <u>การวางแผนการทดลอง</u>

2.1 กำหนดค่า RADIO-BA-THRESHOLD เท่ากับ -80.5 dBm ซึ่งจากเดิม RADIO-RX-THRESHOLD เท่ากับ -81.0 dBm ดังภาพที่ 17

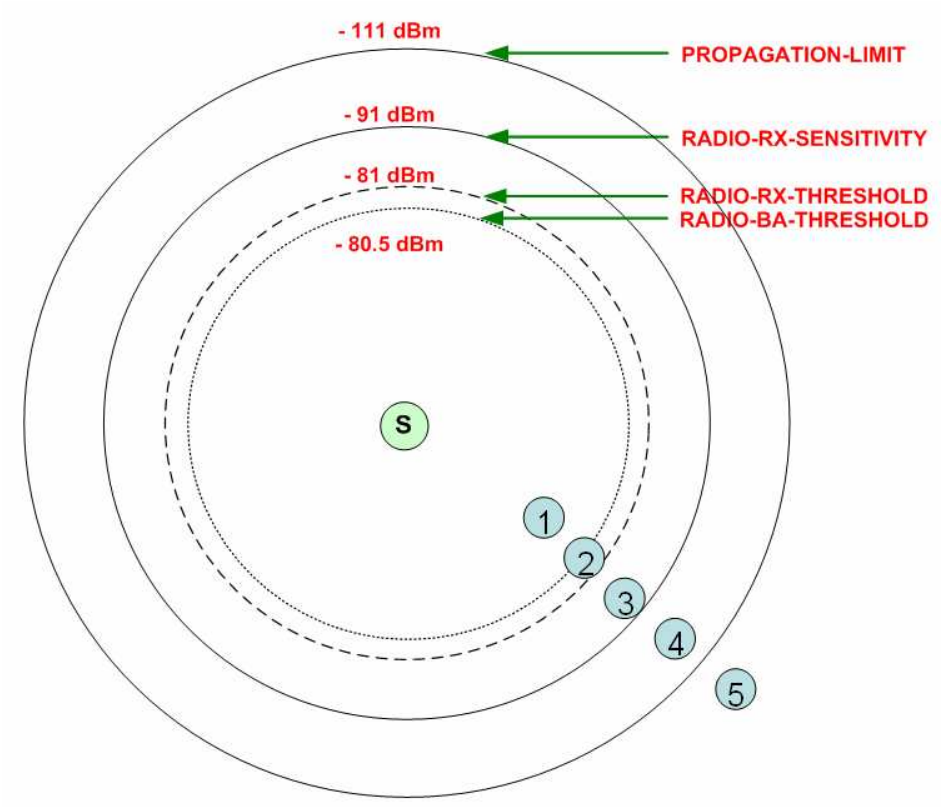
จากรูป PROPAGATION-LIMIT (in dBm) หมายถึง ค่าความแรงของสัญญาณจากเครื่องส่งส่งสัญญาณออกสู่อากาศ ตั้งแต่ -91 dBm ถึง -111 dBm เป็นระดับสัญญาณที่เครื่องรับไม่สามารถรับสัญญาณใดๆ จากเครื่องส่งได้ ดังนั้นเครื่องรับจึงไม่สามารถทำการประมวลผลสัญญาณดังกล่าว และค่านี้ต้องมีค่าน้อยกว่าค่า RADIO-RX-SENSITIVITY + RADIO-ANTENNA-GAIN ของเครื่องรับที่อยู่ในเครือข่าย โดยในที่นี้มีค่า default เท่ากับ -111 dBm เป็นต้น

RADIO-RX-SENSITIVITY (in dBm) หมายถึง ค่าความแรงของสัญญาณจากเครื่องส่งส่งสัญญาณออกสู่อากาศ ตั้งแต่ -81 dBm ถึง -91 dBm เป็นระดับสัญญาณที่เครื่องรับสามารถรับสัญญาณได้เบาบางมาก ดังนั้นจึงไม่สามารถทำการประมวลผลสัญญาณดังกล่าว แต่

เครื่องรับยังคงรับทราบสัญญาณ request-to-send (RTS) และ clear-to-send (CTS) ได้ โดยในที่นี้มีค่า default เท่ากับ -91 dBm เป็นต้น

RADIO-RX-THRESHOLD (in dBm) หมายถึง ค่าขีดเริ่มเปลี่ยนของกำลังงานที่มีค่าต่ำสุดสำหรับแพคเกตที่รับได้ ทั้งนี้ระดับความแรงของสัญญาณที่มากกว่าค่าขีดเริ่มนั้น เครื่องรับสามารถทำการประมวลผลสัญญาณที่รับได้ โดยในที่นี้มีค่ามาตรฐานเท่ากับ-81 dBm เป็นต้น

RADIO-BA-THRESHOLD (in dBm) หมายถึง ค่าขีดเริ่มเปลี่ยนของกำลังงานที่กำหนดให้เครื่องรับนำไปใช้เพื่อตรวจสอบปัญหาที่อาจขาดการเชื่อมต่อระหว่างเครื่องส่งและเครื่องรับ โดยค่านี้ต้องมีค่ามากกว่าค่า RADIO-RX-THRESHOLD ทั้งนี้เพื่อให้เครื่องรับสามารถทำการการประมวลผลสัญญาณที่รับได้ตามปกติ ในที่นี้กำหนดค่าเบื้องต้นเท่ากับ -80.5 dBm เป็นต้น



ภาพที่ 17 ค่าความแรงของสัญญาณจากเครื่องส่งไปยังเครื่องรับ

สัญลักษณ์ S หมายถึง เครื่องส่งที่ทำการส่งสัญญาณออกสู่อากาศได้ระยะทาง 250 เมตร ด้วยกำลังส่งสัญญาณ (radio transmition power) 7.87395 dBm

สัญลักษณ์หมายเลข 1 หมายถึง เครื่องรับที่อยู่ในตำแหน่งที่เครื่องส่งส่งสัญญาณออกสู่อากาศมีค่ากำลังงานในระดับที่เครื่องรับสามารถรับสัญญาณได้ดี และสามารถทำการติดต่อสื่อสารรับส่งแพคเกตข้อมูลกันได้ตามปกติ

สัญลักษณ์หมายเลข 2 หมายถึง เครื่องรับที่อยู่ในตำแหน่งที่เครื่องส่งส่งสัญญาณออกสู่อากาศมีค่ากำลังงานในระดับที่เครื่องรับสามารถรับสัญญาณได้ดี และตรวจพบว่าอาจจะขาดการเชื่อมต่อเกิดขึ้นได้ ดังนั้นเครื่องรับจึงเริ่มดำเนินการตามขั้นตอนวิธีของโพรโทคอล DSR-BA โดยยังคงติดต่อสื่อสารและรับส่งแพคเกตข้อมูลกันได้ตามปกติ

สัญลักษณ์หมายเลข 3 หมายถึง เครื่องรับที่อยู่ในตำแหน่งที่เครื่องส่งส่งสัญญาณออกสู่อากาศมีค่ากำลังงานในระดับที่เครื่องรับสามารถรับสัญญาณได้แต่สัญญาณไม่มากพอที่เครื่องรับจะสามารถทำการประมวลผลสัญญาณได้ จึงทำให้เกิดการสูญหายแพคเกตขึ้น

สัญลักษณ์หมายเลข 4 หมายถึงระยะที่เครื่องส่งส่งสัญญาณออกสู่อากาศมีค่ากำลังงานที่มีระยะความแรงของสัญญาณส่งถึงเครื่องรับอย่างเบาบาง โดยเครื่องรับยังคงสามารถรับ RTS และ CTS ที่ส่งกันอยู่ในช่องสัญญาณได้ แต่ไม่สามารถทำการติดต่อสื่อสารรับส่งแพคเกตข้อมูลกัน

สัญลักษณ์หมายเลข 5 หมายถึงระยะที่เครื่องส่งส่งสัญญาณออกสู่อากาศมีค่ากำลังงานที่มีระยะความแรงของสัญญาณส่งไม่ถึงเครื่องรับ ทำให้โหนดขาดการเชื่อมต่อกัน

2.2 เขียนโปรแกรมด้วยภาษา PARSEC (Terwilliger and Meyer, 2001) และภาษา C เพื่อจำลองการทำงานของโพรโทคอล DSR-BA ที่ใช้ในเครือข่ายเฉพาะกิจด้วยโปรแกรมจำลองการทำงานเครือข่ายไร้สาย GloMoSim ที่จำลองการทำงานของเครือข่ายเฉพาะกิจ โดยทำการปรับปรุงโพรโทคอล DSR ด้วยระบบปฏิบัติการลีนุกซ์ในห้องทดลองระบบปฏิบัติการ เก็บผลการจำลองการทำงานของโพรโทคอล DSR และโพรโทคอล DSR-BA นำผลที่ได้มาทำการวิเคราะห์เปรียบเทียบกับสมมติฐานที่ตั้งไว้

2.3 โปรแกรมจำลองการทำงานของเครือข่ายไร้สาย ในที่นี้ใช้โปรแกรมจำลองการทำงานเครือข่ายไร้สาย GloMoSim ซึ่งเป็นโปรแกรมการจำลองการทำงานเครือข่ายที่ออกแบบมาเฉพาะให้เหมาะกับเครือข่ายไร้สายโดยโปรแกรมจำลองการทำงาน GloMoSim ใช้ภาษา Parsec เป็นเครื่องมือช่วยในการจัดระบบการทำงานหลายๆ อย่างพร้อมๆ กันแบบ parallel processing อีกทั้งโปรแกรมจำลองการทำงาน GloMoSim เป็นโปรแกรมที่นักวิจัยทั้งในประเทศไทยและต่างประเทศนิยมใช้ในงานวิจัยทางด้านเครือข่ายไร้สายแบบต่างๆ ซึ่งโปรแกรมจำลองการทำงาน GloMoSim มีทั้งแบบที่เป็นโปรแกรมทางการพาณิชย์ในชื่อว่า Qualnet และโปรแกรมทางการศึกษาเพื่อการวิจัยและพัฒนาในชื่อว่า GloMoSim โดยโปรแกรมทางการศึกษาเพื่อการวิจัยและพัฒนาจะไม่มีการคิดค่าใช้จ่ายเกี่ยวกับลิขสิทธิ์ใดๆ ทั้งสิ้น

การใช้งานโปรแกรมจำลองการทำงานเครือข่ายไร้สาย GloMoSim เพื่อการทดลองนั้น จำเป็นต้องมีการกำหนดค่าเบื้องต้นเพื่อใช้ในการจำลองการทำงานของโพรโทคอล DSR และ DSR-BA โดยกำหนดได้เป็น 2 ส่วนคือ

2.3.1 การกำหนดค่าคงที่

ก. กำหนดระยะเวลาในการจำลองการทำงานไว้ 600 วินาที
ข. กำหนดตำแหน่งของโหนดแบบ RANDOM
ค. กำหนดรูปแบบการเคลื่อนที่ของโหนดแบบ RANDOM-WAYPOINT
ง. กำหนดขนาดการส่งข้อมูลแพคเกต 11 Mbps
จ. กำหนดรัศมีการส่ง 250 เมตร
ฉ. กำหนดขนาดแพคเกต 512 ไบต์
ช. กำหนดความถี่ 2.4 กิกะเฮิร์ต
ซ. กำหนดแอปปลิเคชั่นแบบ constant bit rate (CBR)
ฌ. กำหนดให้แต่ละโหนดส่งแพคเกตข้อมูล 4 แพคเกตต่อวินาที
ญ. กำหนดจำนวนโหนดในเครือข่าย 50 โหนด

2.3.2 การกำหนดค่าแปรผัน

ก. กำหนดขนาดพื้นที่เครือข่าย
   1) 1500 x 300 เมตร
ข. จำนวนโหนดที่ส่งแพคเกตข้อมูล
   1) จำนวน 10 โหนด
   2) จำนวน 20 โหนด
   3) จำนวน 30 โหนด
ค. กำหนดความเร็วสูงสุดในการเคลื่อนที่ของแต่ละโหนด
   1) 0 เมตรต่อวินาที (โหนดไม่เคลื่อนที่)
   2) 5 เมตรต่อวินาที
   3) 10 เมตรต่อวินาที
   4) 15 เมตรต่อวินาที
   5) 20 เมตรต่อวินาที

2.4 การวัดประสิทธิภาพ

เนื่องจากวัตถุประสงค์หลักของการทำงานวิจัยฉบับนี้เพื่อเพิ่มประสิทธิภาพการทำงานของโพรโทคอล DSR ดังนั้นวิธีการวัดประสิทธิภาพประกอบด้วย 3 ส่วนหลักๆ คือ

2.4.1 การวัดจำนวน Routing control packets ที่เกิดขึ้นในเครือข่าย เนื่องจากโพรโทคอล DSR-BA มีการใช้แพคเกตชนิดควบคุมเพิ่มขึ้นอีกสองชนิด จึงจำเป็นต้องทำการวัดประสิทธิภาพจำนวน Routing control packet ที่แต่ละโพรโทคอลได้ใช้ไปในเครือข่าย ทั้งนี้เพื่อนำมาเปรียบเทียบความสิ้นเปลืองที่เกิดขึ้นจากแต่ละโพรโทคอล

2.4.2 ค่า Packet delivery radio คือ ค่าอัตราส่วนระหว่างจำนวนแพคเกต CBR ที่โหนดปลายทางรับได้ต่อจำนวนแพคเกต CBR ที่โหนดเริ่มต้นสร้างขึ้นมา โดยที่ CBR เป็นแอปปลิเคชัน (application) รูปแบบหนึ่งที่ใช้งานในการติดต่อสื่อสารของโหนดต่างๆ ในเครือข่าย ถ้าหากค่า Packet delivery ratio มีค่ามากแสดงถึงประสิทธิภาพของโพรโทคอลในการรับส่งแพคเกตในเครือข่ายได้ดี ตัวอย่างเช่นถ้าโหนดเริ่มต้นสร้างแพคเกต CBR จำนวน 100 แพคเกต

เมื่อมีการรับส่งแพคเกตกันในเครือข่ายกับโหนดปลายทางปรากฏว่าโหนดปลายทางสามารถรับแพคเกต CBR ได้จำนวน 99 แพคเกต ดังนั้นค่า Packet delivery ratio มีค่า 0.99 ถ้าคิดเป็นเปอร์เซ็นต์จะได้ 99% แสดงว่าโพรโทคอลนี้มีประสิทธิภาพในการรับส่งแพคเกตในเครือข่ายได้ดีถึง 99%

2.4.3 ค่า Average end-to-end delay of data packets (Das *et al*., 2000) คือค่าความหน่วง end-to-end เฉลี่ยของแพคเกต CBR ทั้งหมด โดยค่าความหน่วงที่เกิดขึ้นมานั้น อาจเกิดขึ้นได้จากหลายสาเหตุ อย่างเช่น ช่วงเวลาในการเข้าถึงขณะทำการค้นหาเส้นทาง (buffering during route discovery latency) การเข้าเรียงลำดับตามลำดับของการอินเตอร์เฟซ (queuing at the interface queue) ค่าความหน่วงที่เกิดขึ้นจากการส่งแพคเกตซ้ำอีกที่เกิดขึ้น ณ ชั้น MAC ค่าเวลาที่ใช้ในการแพร่กระจายคลื่น (propagation times) และค่าเวลาที่ใช้ในการส่งผ่านข้อมูล (transfer times) เป็นต้น ซึ่งค่า Average end-to-end delay of data packets นี้มีค่ายิ่งน้อยยิ่งดีแสดงให้เห็นถึงประสิทธิภาพของโพรโทคอลในการรับส่งแพคเกตภายในเครือข่ายมีการรับส่งแพคเกตที่รวดเร็ว

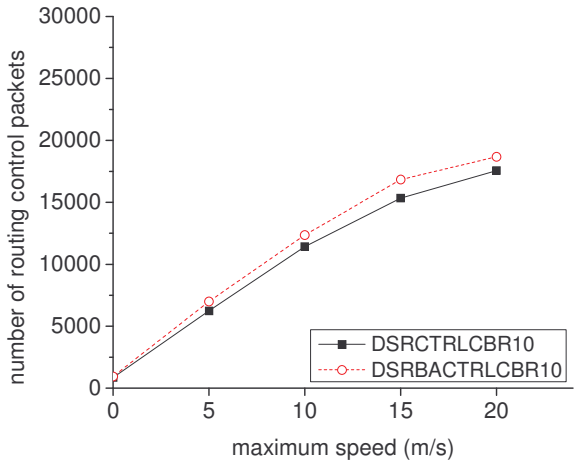
ผลและการวิจารณ์

ค่า Routing control packets

1. <u>ขนาดเครือข่าย 1500 x 300 ตารางเมตร</u>

1.1 ความเร็วสูงสุดในการเคลื่อนที่ของแต่ละโหนด 0 10 20 และ 30 เมตรต่อวินาที

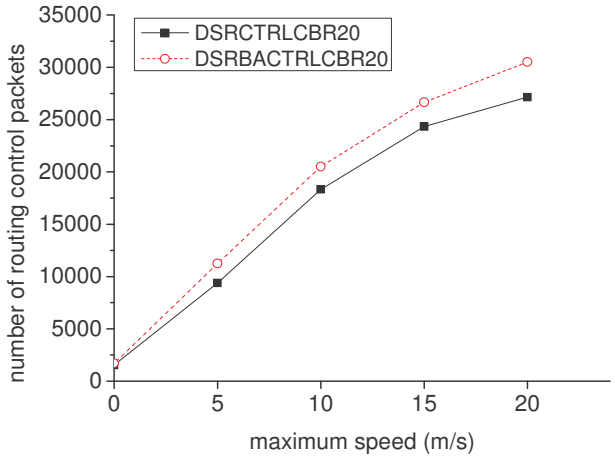
1.1.1 จำนวนโหนดที่ส่งแพคเกตข้อมูล 10 โหนด



<u>ภาพที่ 18</u> จำนวนของแพคเกตควบคุมที่เกิดขึ้น เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 10 โหนด ภายในเครือข่ายขนาด 1500×300 ตารางเมตร

จากผลการทดลองภาพที่ 18 แสดงให้เห็นถึงการทำงานของโพรโทคอลทั้ง DSR และ DSR-BA พบว่า Routing control packets ที่ช่วงความเร็วสูงสุด 0 ถึง 5 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 2.709% ที่ช่วงความเร็วสูงสุด 5 ถึง 10 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 2.839% ที่ช่วงความเร็วสูงสุด 10 ถึง 15 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 5.525% ที่ช่วงความเร็วสูงสุด 15 ถึง 20 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 3.945% จากผลการทดลองในภาพรวมพบว่า เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 10 โหนดนั้น DSR-BA มีค่า Routing control packets เพิ่มขึ้นประมาณ 2.71–5.53%

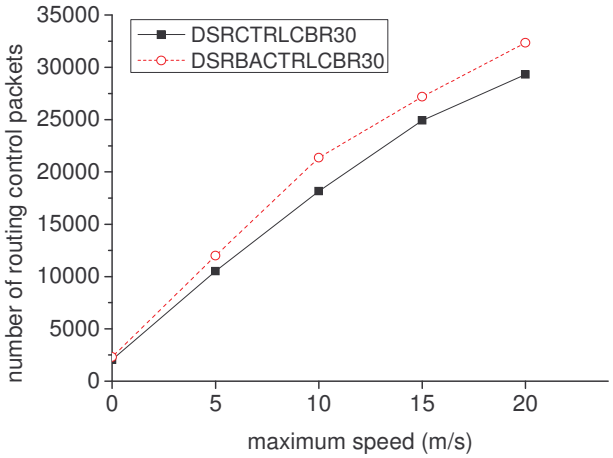
1.1.2 จำนวนโหนดที่ส่งแพคเกตข้อมูล 20 โหนด



ภาพที่ 19 จำนวนของแพคเกตควบคุมที่เกิดขึ้น เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 20 โหนด ภายในเครือข่ายขนาด 1500×300 ตารางเมตร

จากผลการทดลองภาพที่ 19 แสดงให้เห็นถึงการทำงานของโพรโทคอลทั้ง DSR และ DSR-BA พบว่า Routing control packets ที่ช่วงความเร็วสูงสุด 0 ถึง 5 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 4.987% ที่ช่วงความเร็วสูงสุด 5 ถึง 10 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 6.600% ที่ช่วงความเร็วสูงสุด 10 ถึง 15 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 6.221% ที่ช่วงความเร็วสูงสุด 15 ถึง 20 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 9.288% จากผลการทดลองในภาพรวมพบว่า เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 20 โหนดนั้น DSR-BA มีค่า Routing control packets เพิ่มขึ้นประมาณ 4.99–9.29%

1.1.3 จำนวนโหนดที่ส่งแพคเกตข้อมูล 30 โหนด



ภาพที่ 20 จำนวนของแพคเกตควบคุมที่เกิดขึ้น เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 30 โหนด ภายในเครือข่ายขนาด 1500×300 ตารางเมตร

จากผลการทดลองภาพที่ 20 แสดงให้เห็นถึงการทำงานของโพรโทคอลทั้ง DSR และ DSR-BA พบว่า Routing control packets ที่ช่วงความเร็วสูงสุด 0 ถึง 5 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 4.246% ที่ช่วงความเร็วสูงสุด 5 ถึง 10 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 7.387% ที่ช่วงความเร็วสูงสุด 10 ถึง 15 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 5.516% ที่ช่วงความเร็วสูงสุด 15 ถึง 20 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 7.163% จากผลการทดลองในภาพรวมพบว่า เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 30 โหนดนั้น DSR-BA มีค่า Routing control packets เพิ่มขึ้นประมาณ 4.25–7.39%

จากผลการทดลองภาพที่ 18 19 และ 20 แสดงให้เห็นถึงการทำงานของโพรโทคอลทั้ง DSR และ DSR-BA ในสภาพเครือข่ายที่โหนดมีการเคลื่อนที่ด้วยความเร็วต่ำสุด 0 เมตรต่อวินาทีและความเร็วสูงสุด 5 10 15 และ 20 เมตรต่อวินาที ภายในเครือข่ายขนาด 1500x300 ตารางเมตร และโหนดเริ่มต้นจำนวน 10 20 และ 30 โหนดตามลำดับ โดยแต่ละโหนดเริ่มต้นมีการส่งแพคเกตข้อมูลขนาด 512 ไบต์จำนวน 4 แพคเกตทุกๆ 1 วินาทีในช่วงระยะเวลา 600 วินาที

เมื่อพิจารณาในภาพรวมการทำงานของทั้งสองโพรโทคอลพบว่า DSR-BA มีค่า Routing control packets มากกว่า DSR ในทุกๆ ช่วงความเร็วในการเคลื่อนที่ของโหนด โดยแปรผันตามจำนวนโหนดเริ่มต้นที่เพิ่มขึ้นจาก 10 20 และ 30 โหนดตามลำดับ

เมื่อพิจารณาที่ความเร็วในการเคลื่อนที่สูงสุดของแต่ละโหนด จากภาพที่ 18 19 และ 20 ในช่วงความเร็วสูงสุดจาก 5 10 15 และ 20 เมตรต่อวินาที แพคเกตควบคุมที่ใช้นั้นจะเพิ่มขึ้นเรื่อยๆ ตามช่วงความเร็วสูงสุดที่เพิ่มขึ้น ทั้งนี้เพราะการที่โหนดมีการเคลื่อนที่อยู่ตลอดเวลานั้น ทำให้เส้นทางการเคลื่อนที่ของโหนดเปลี่ยนแปลงไป นั่นคือแต่ละเส้นทางที่โหนดใช้ในการติดต่อสื่อสารกันอาจจะขาดการติดต่อสื่อสารได้ง่าย ส่งผลถึงจำนวนการตรวจพบการเชื่อมต่อที่โหนดอาจจะขาดการติดต่อสื่อสารกันได้บ่อยครั้ง

ในขณะที่โหนดตรวจพบการเชื่อมต่อที่โหนดอาจจะขาดการติดต่อสื่อสารกัน DSR-BA จะใช้แพคเกตควบคุมเพิ่มขึ้นอีกสองชนิดนอกเหนือจากแพคเกตควบคุมที่ DSR ใช้งานอยู่ คือ RREF และ RFRP ที่จะถูกใช้ไปเพื่อตรวจสอบเส้นทางใหม่ในแคชเส้นทางหลังจากที่โหนดได้ตรวจพบการเชื่อมต่อที่อาจจะขาดการติดต่อสื่อสารกัน นอกจากสองแพคเกตควบคุมดังกล่าวแล้ว DSR-BA ใช้แพคเกตควบคุมอีกหนึ่งชนิดคือ RERRBA เพื่อแจ้งไปยังโหนดเริ่มต้นด้วย ดังนั้นยิ่งถ้าโหนดมีการเคลื่อนที่เร็วมากขึ้นการเชื่อมต่อสื่อสารระหว่างโหนดอาจจะขาดการสื่อสารได้ง่ายกว่าการที่โหนดมีการเคลื่อนที่ช้า ทำให้ DSR-BA ต้องทำการตรวจสอบการเชื่อมต่อของเส้นทางบ่อยครั้งขึ้น จึงส่งผลให้จำนวนแพคเกตควบคุมที่ใช้ไปเพื่อกระบวนการนี้จึงมากขึ้นตามไปด้วย

เมื่อพิจารณาที่ช่วงความเร็วสูงสุด 0 เมตรต่อวินาที นั่นคือสภาวะที่โหนดไม่มีการเคลื่อนที่ พบว่าแพคเกตควบคุมที่ทั้งสองโพรโทคอลใช้ไปนั้นมีจำนวนที่ใกล้เคียงกันโดย DSR-BA มีจำนวนมากกว่าเล็กน้อย เนื่องจากการจัดวางตำแหน่งของโหนดในเครือข่ายนั้นจัดวางอย่างสุ่ม (random) ซึ่งโหนดอาจจะวางตัวอยู่ในระยะของการตรวจพบการเชื่อมต่อที่โหนดอาจจะขาดการเชื่อมต่อกันได้ โหนดจึงส่งแพคเกตไปตรวจสอบเส้นทางใหม่ที่มีอยู่ในแคชเส้นทาง แต่การตรวจสอบนี้จะกระทำน้อยครั้งกว่าการที่โหนดมีการเคลื่อนที่ตลอดเวลา ดังนั้นแพคเกตควบคุมที่ใช้ไปจึงมีจำนวนไม่มากนัก

เมื่อพิจารณาที่จำนวนโหนดเริ่มต้น 10 20 และ 30 โหนดตามลำดับ จากภาพที่ 18 19 และ 20 นั้นพบว่าแพคเกตควบคุมที่ใช้จะเพิ่มมากขึ้นตามจำนวนการเพิ่มของโหนดเริ่มต้น นั่นคือยิ่งมีจำนวนโหนดเริ่มต้นมาก จำนวนของการจราจรของแพคเกตข้อมูลก็จะมากขึ้น จำนวนเส้นทางการส่งแพคเกตในเครือข่ายก็จะมีความต้องการใช้เส้นทางสูงขึ้นด้วย ดังนั้นเมื่อโหนดเคลื่อนที่ เส้นทางที่ใช้งานอยู่อาจจะขาดได้หลายเส้นทางมากขึ้น จากเดิม 10 เส้นทางเพิ่มเป็น 20 เส้นทาง และ 30เส้นทางหรือมากกว่านั้น ซึ่งจำนวนเส้นทางดังกล่าวที่เพิ่มขึ้นนี้ ส่งผลถึงการดูแล

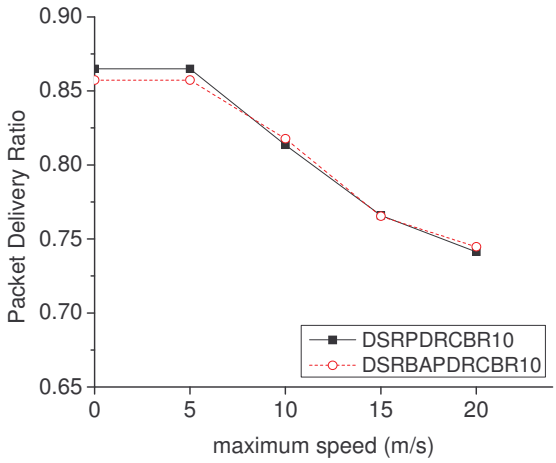
รักษาเส้นทางที่เพิ่มขึ้นด้วย นั่นคือจำนวนแพคเกตควบคุมที่ใช้เพื่อการดูแลรักษาเส้นทางจะมีความต้องการใช้งานที่สูงมาก ทั้งนี้ก็เพื่อรองรับการทำงานดังกล่าวให้สามารถส่งต่อแพคเกตข้อมูลได้นั่นเอง

ค่า Packet delivery ratio

1. <u>ขนาดเครือข่าย 1500 x 300 ตารางเมตร</u>

1.1 ความเร็วสูงสุดในการเคลื่อนที่ของแต่ละโหนด 0 10 20 และ 30 เมตรต่อวินาที

1.1.1 จำนวนโหนดที่ส่งแพคเกตข้อมูล 10 โหนด

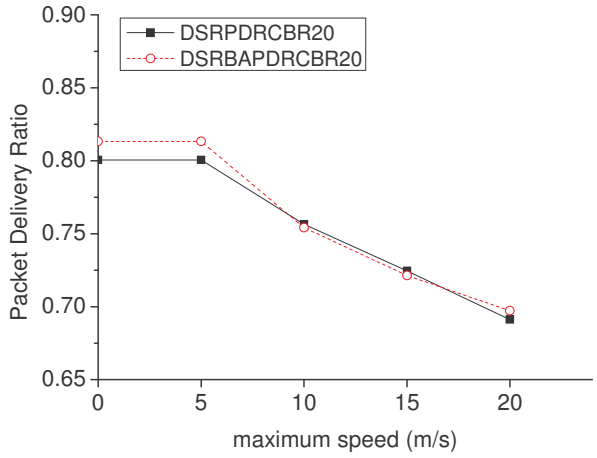


<u>ภาพที่ 21</u> ค่าของ Packet delivery ratio เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 10 โหนด ภายในเครือข่ายขนาด 1500×300 ตารางเมตร

จากผลการทดลองภาพที่ 21 แสดงให้เห็นถึงการทำงานของโพรโทคอลทั้ง DSR และ DSR-BA พบว่า Packet delivery ratio ที่ช่วงความเร็วสูงสุด 0 ถึง 5 เมตรต่อวินาที DSR-BA มีค่าน้อยกว่า DSR ประมาณ 0.0073% ที่ช่วงความเร็วสูงสุด 5 ถึง 10 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 0.0028% ที่ช่วงความเร็วสูงสุด 10 ถึง 15 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 0.0001% ที่ช่วงความเร็วสูงสุด 15 ถึง 20 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 0.0076% จากผลการทดลองในภาพรวมพบว่า เมื่อมีจำนวนโหนดที่ส่ง

แพคเกตข้อมูล 10 โหนดนั้น DSR-BA มีค่า Packet delivery ratio น้อยกว่าเล็กน้อยเมื่อโหนดมีการเคลื่อนที่ช้า และมีค่าเพิ่มขึ้นมากกว่า DSR เล็กน้อยเมื่อความเร็วในการเคลื่อนที่ของโหนดสูงขึ้น

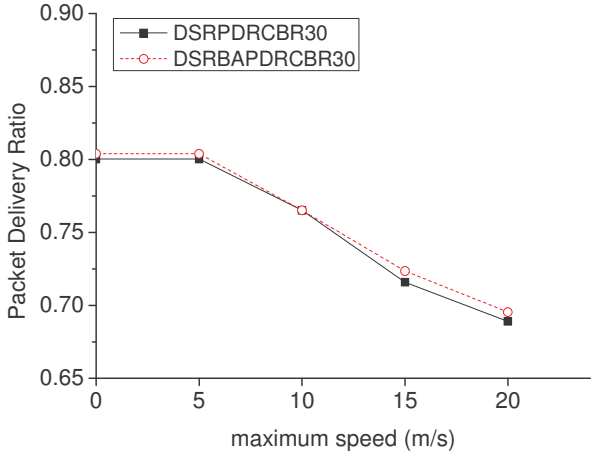
1.1.2 จำนวนโหนดที่ส่งแพคเกตข้อมูล 20 โหนด



ภาพที่ 22 ค่าของ Packet delivery ratio เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 20 โหนด ภายในเครือข่ายขนาด 1500×300 ตารางเมตร

จากผลการทดลองภาพที่ 22 แสดงให้เห็นถึงการทำงานของโพรโทคอลทั้ง DSR และ DSR-BA พบว่า Packet delivery ratio ที่ช่วงความเร็วสูงสุด 0 ถึง 5 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 0.0119% ที่ช่วงความเร็วสูงสุด 5 ถึง 10 เมตรต่อวินาที DSR-BA มีค่าน้อยกว่า DSR ประมาณ 0.0019% ที่ช่วงความเร็วสูงสุด 10 ถึง 15 เมตรต่อวินาที DSR-BA มีค่าน้อยกว่า DSR ประมาณ 0.0032% ที่ช่วงความเร็วสูงสุด 15 ถึง 20 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 0.0043% จากผลการทดลองในภาพรวมพบว่า เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 20 โหนดนั้น DSR-BA มีค่า Packet delivery ratio มากกว่าเมื่อโหนดมีการเคลื่อนที่ช้าและมีค่ามากกว่าเล็กน้อยเมื่อโหนดการเคลื่อนที่เร็วมาก แต่ในช่วงที่โหนดมีความเร็วในการเคลื่อนที่ปานกลาง DSR-BA มีค่าน้อยกว่า DSR เล็กน้อย

1.1.3 จำนวนโหนดที่ส่งแพคเกตข้อมูล 30 โหนด



ภาพที่ 23 ค่าของ Packet delivery ratio เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 30 โหนด ภายในเครือข่ายขนาด 1500×300 ตารางเมตร

จากผลการทดลองภาพที่ 23 แสดงให้เห็นถึงการทำงานของโพรโทคอลทั้ง DSR และ DSR-BA พบว่า Packet delivery ratio ที่ช่วงความเร็วสูงสุด 0 ถึง 5 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 0.0075% ที่ช่วงความเร็วสูงสุด 5 ถึง 10 เมตรต่อวินาที DSR-BA มีค่าน้อยกว่า DSR ประมาณ 0.0044% ที่ช่วงความเร็วสูงสุด 10 ถึง 15 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 0.0054% ที่ช่วงความเร็วสูงสุด 15 ถึง 20 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 0.0041% จากผลการทดลองในภาพรวมพบว่า เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 30 โหนดนั้น DSR-BA มีค่า Packet delivery ratio มีแนวโน้มมากกว่าในทุกช่วงความเร็วในการเคลื่อนที่ของโหนด โดย DSR-BA มีค่าน้อยกว่า DSR เล็กน้อยเมื่อโหนดมีการเคลื่อนที่ปานกลาง

จากผลการทดลองภาพที่ 21 22 และ 23 แสดงให้เห็นการทำงานของโพรโทคอลทั้ง DSR และ DSR-BA ในสภาพเครือข่ายที่โหนดมีการเคลื่อนที่ด้วยความเร็วต่ำสุด 0 เมตรต่อวินาทีและความเร็วสูงสุด 5 10 15 และ 20 เมตรต่อวินาที ภายในเครือข่ายขนาด 1500x300 ตารางเมตร และโหนดเริ่มต้นจำนวน 10 20 และ 30 โหนดตามลำดับ โดยแต่ละโหนดเริ่มต้นมีการส่งแพคเกตข้อมูลขนาด 512 ไบต์จำนวน 4 แพคเกตทุกๆ 1 วินาทีในช่วงระยะเวลา 600 วินาที

เมื่อพิจารณาในภาพรวมการทำงานของทั้งสองโพรโทคอลพบว่า DSR-BA มีค่า Packet Delivery Ratio น้อยกว่า DSR เมื่อมีความเร็วในการเคลื่อนที่ของโหนดปานกลาง แต่มีค่ามากกว่าเมื่อโหนดมีการเคลื่อนที่ช้ามากและเร็วมากในขณะที่มีจำนวนโหนดเริ่มต้นเพิ่มขึ้นด้วย

เมื่อพิจารณาที่ความเร็วในการเคลื่อนที่สูงสุดของแต่ละโหนด จากภาพที่ 21 ช่วงความเร็วสูงสุด 0 และ 5 เมตรต่อวินาที นั่นคือในสภาวะที่โหนดไม่มีการเคลื่อนที่หรือเคลื่อนที่ช้าๆ และมีจำนวนโหนดเริ่มต้นเพียง 10 โหนดพบว่า DSR ทำการรับส่งแพคเกตข้อมูลได้ดีกว่า DSR-BA เนื่องจากการที่โหนดได้ทำการตรวจสอบเส้นทางหรือหลังจากที่โหนดเริ่มต้นรับทราบเส้นทางที่อาจจะขาดการเชื่อมต่อ โหนดดังกล่าวจะลบเส้นทางที่คาดว่าอาจจะขาดการเชื่อมต่อออกจากแคชเส้นทาง ซึ่งถ้าโหนดไม่เคลื่อนที่เส้นทางดังกล่าวนั้น DSR ยังคงใช้งานอยู่ แต่ DSR-BA ได้ทำการลบเส้นทางนั้น ถ้าโหนดไม่มีเส้นทางอื่นๆ ในแคชเส้นทางโหนดเริ่มต้นจะต้องทำการค้นหาเส้นทางใหม่และขณะเดียวกันจะได้เส้นทางใหม่เป็นเส้นทางเดิมที่ได้ลบทิ้งไปแล้วจากโหนดตัวกลางส่งตอบกลับมา จากนั้น DSR-BA ทำการตรวจสอบเส้นทางและทำการลบเส้นทางอีกครั้ง ในขณะที่ DSR ยังคงใช้เส้นทางเดิม ด้วยเหตุนี้การรับส่งด้วย DSR จึงทำได้ดีกว่า DSR-BA

แต่เมื่อโหนดเพิ่มความเร็วในการเคลื่อนที่ DSR-BA สามารถรับส่งแพคเกตข้อมูลได้ใกล้เคียงกันในช่วงความเร็วสูงสุด 15 เมตรต่อวินาทีและรับส่งแพคเกตข้อมูลได้ดีขึ้นอีกเล็กน้อยในช่วงความเร็วสูงสุด 10 และ 20 เมตรต่อวินาที เพราะการที่โหนดเคลื่อนที่ DSR-BA จะมีโอกาสได้เปลี่ยนเส้นทางการส่งแพคเกตข้อมูลได้มากกว่า DSR โดยที่โหนดจะลบเส้นทางที่อาจจะขาดการเชื่อมต่อทิ้งไป ยิ่งตรวจพบบ่อย ข้อมูลเส้นทางในแคชเส้นทางก็จะถูกลบบ่อยขึ้น การจะได้เส้นทางใหม่มาใช้ก็มีมากขึ้น ซึ่งโหนดจะสามารถหลีกเลี่ยงเส้นทางเก่าที่อาจจะขาดการเชื่อมต่อเหล่านี้ไป ทำให้เมื่อโหนดขาดการเชื่อมต่อขึ้นจริง โหนดยังสามารถเลือกใช้เส้นทางอื่นๆ ได้ โดยไม่ต้องใช้เส้นทางที่เตรียมไว้ซึ่งเป็นเส้นทางที่ใช้งานไม่ได้แล้ว ซึ่งจะทำให้แพคเกตข้อมูลสูญหายไปอย่างแน่นอน

เมื่อเพิ่มจำนวนโหนดเริ่มต้นจาก 10 โหนดเป็น 20 โหนดและ 30 โหนด ดังภาพที่ 22 และ 23 พบว่าช่วงความเร็วสูงสุด 0 และ 5 เมตรต่อวินาทีนั้น DSR-BA รับส่งแพคเกตข้อมูลได้ดีกว่า DSR ทั้งนี้เพราะโหนดมีเส้นทางในแคชเส้นทางเพิ่มขึ้น ซึ่งเมื่อโหนดทำการตรวจสอบเส้นทางแล้วหรือหลังจากที่โหนดเริ่มต้นรับทราบเส้นทางที่อาจจะขาดการเชื่อมต่อ โหนดจะลบเส้นทางดังกล่าวออกจากแคชเส้นทาง ซึ่งถ้าไม่มีเส้นทางอื่นๆ ให้ใช้ โหนดจะต้องทำการค้นหา

เส้นทางใหม่ แต่ถ้าโหนดมีเส้นทางอื่นๆ ให้ใช้อีก โหนดก็จะใช้เส้นทางที่มีอยู่นั้น ในขณะที่ DSR จะรอให้เส้นทางขาดเสียก่อนทำให้แพคเกตสูญหายได้เมื่อใช้เส้นทางที่เตรียมไว้แต่เส้นทางดังกล่าวเป็นเส้นทางที่ไม่สามารถจะใช้งานได้ ซึ่งเส้นทางดังกล่าว DSR-BA ได้กำจัดออกไปจากแคชเส้นทางแล้ว

ส่วนในช่วงความเร็วสูงสุด 10 15 และ 20 เมตรต่อวินาทีนั้น DSR-BA รับส่งแพคเกตข้อมูลได้ใกล้เคียงหรือดีกว่าเล็กน้อยในช่วงความเร็วที่สูงขึ้น ทั้งนี้เพราะ DSR-BA เมื่อพิจารณาในส่วนของกระบวนการตรวจสอบเส้นทางที่อาจจะขาดการเชื่อมต่อนั้น จะตรวจสอบจากการที่โหนดเคลื่อนที่ออกไปเป็นหลัก ซึ่งเมื่อตรวจพบเส้นทางดังกล่าว โหนดจะลบเส้นทางที่อาจจะขาดการเชื่อมต่อออกไป จากนั้นก็จะใช้เส้นทางใหม่ ซึ่งเส้นทางที่ขาดการเชื่อมต่อนี้ โหนดได้ลบทิ้งไปเนื่องจากได้ตรวจสอบแล้ว ขณะที่ DSR จะยังคงใช้เส้นทางนั้นก่อน เมื่อใช้ไม่ได้แพคเกตข้อมูลก็สูญหาย หลังจากที่แจ้งให้กับโหนดเริ่มต้นแต่โหนดไม่ทราบ เนื่องจาก RERR ส่งไปไม่ถึง ทำให้โหนดเริ่มต้นยังคงใช้เส้นทางที่มีการขาดการเชื่อมต่อนี้ แต่ DSR-BA ไม่เป็นเช่นนั้น เพราะโหนดจะแจ้งความผิดพลาดของเส้นทางที่ควรหลีกเลี่ยงไปถึงยังโหนดเริ่มต้นก่อนที่เส้นทางจะขาดการเชื่อมต่อดังนั้นโหนดจะเลือกใช้เส้นทางใหม่ได้ทันก่อนที่เส้นทางเดิมจะขาดการเชื่อมต่อ แต่เส้นทางใหม่ที่โหนดเลือกใช้เป็นเส้นทางที่ไม่สามารถใช้งานได้ แพคเกตก็จะสูญหาย แต่ถ้าเป็นเส้นทางที่ใช้งานได้แพคเกตจะสูญหายน้อยกว่า DSR

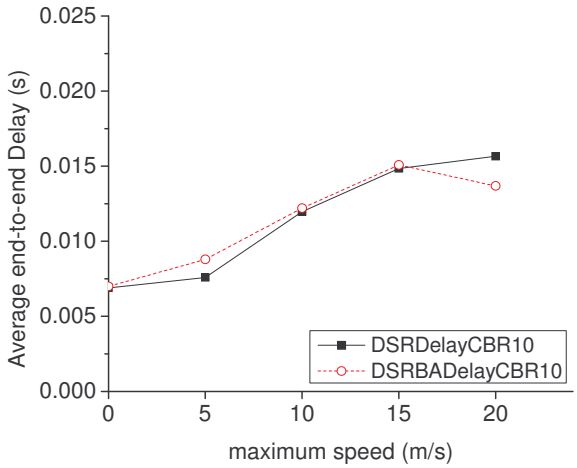
ดังนั้น DSR-BA จะใช้งานได้ดีเมื่อเส้นทางที่โหนดเตรียมไว้มีโหนดแรกเป็นโหนดที่ขาดการเชื่อมต่อไปแล้ว ซึ่งจะสามารถกำจัดเส้นทางนี้ออกจากแคชเส้นทางแล้วให้โหนดเลือกใช้เส้นทางลำดับถัดไป ซึ่ง DSR นั้นเมื่อโหนดเลือกใช้เส้นทางนี้ แพคเกตข้อมูลจะสูญหายทันที แต่ DSR-BA มีโอกาสเลือกใช้เส้นทางใหม่เพื่อนำพาแพคเกตข้อมูลนี้อีกครั้งก่อนที่แพคเกตข้อมูลนี้จะสูญหาย

ค่า Average end-to-end delay

1. <u>ขนาดเครือข่าย 1500 x 300 ตารางเมตร</u>

1.1 ความเร็วสูงสุดในการเคลื่อนที่ของแต่ละโหนด 0 10 20 และ 30 เมตรต่อวินาที

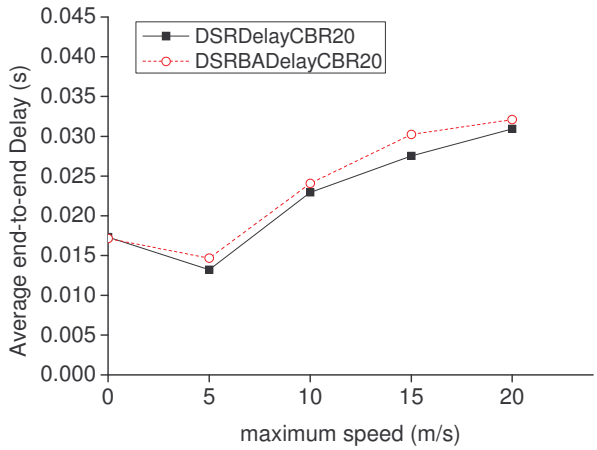
1.1.1 จำนวนโหนดที่ส่งแพคเกตข้อมูล 10 โหนด



<u>ภาพที่ 24</u> ค่าความหน่วง end-to-end เฉลี่ยของแพคเกต CBR ทั้งหมด เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 10 โหนด ภายในเครือข่ายขนาด 1500×300 ตารางเมตร

จากผลการทดลองภาพที่ 24 แสดงให้เห็นการทำงานของโพรโทคอลทั้ง DSR และ DSR-BA พบว่า Average end-to-end delay ที่ช่วงความเร็วสูงสุด 0 ถึง 5 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 2.52% ที่ช่วงความเร็วสูงสุด 5 ถึง 10 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 0.16% ที่ช่วงความเร็วสูงสุด 10 ถึง 15 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 1.44% ที่ช่วงความเร็วสูงสุด 15 ถึง 20 เมตรต่อวินาที DSR-BA มีค่าน้อยกว่า DSR ประมาณ 3.68% จากผลการทดลองในภาพรวมพบว่า เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 10 โหนดนั้น DSR-BA มีค่า Average end-to-end delay น้อยกว่าเมื่อโหนดมีการเคลื่อนที่เร็วมาก แต่เมื่อความเร็วในการเคลื่อนที่ของโหนดลดลงมีค่าสูงกว่าประมาณ 0.16-2.52%

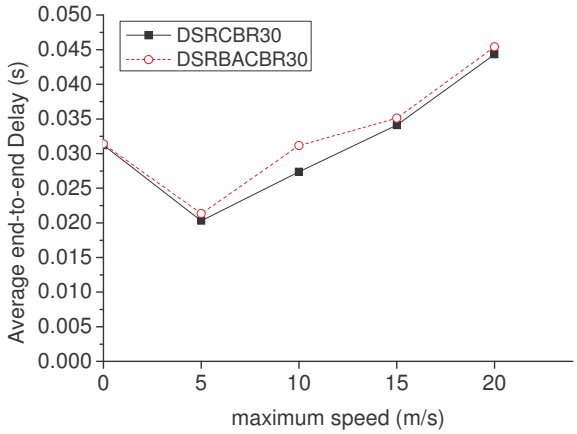
1.1.2 จำนวนโหนดที่ส่งแพคเกตข้อมูล 20 โหนด



ภาพที่ 25 ค่าความหน่วง end-to-end เฉลี่ยของแพคเกต CBR ทั้งหมด เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 20 โหนด ภายในเครือข่ายขนาด 1500×300 ตารางเมตร

จากผลการทดลองภาพที่ 25 แสดงให้เห็นการทำงานของโพรโทคอลทั้ง DSR และ DSR-BA พบว่า Average end-to-end delay ที่ช่วงความเร็วสูงสุด 0 ถึง 5 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 3.72% ที่ช่วงความเร็วสูงสุด 5 ถึง 10 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 3.64% ที่ช่วงความเร็วสูงสุด 10 ถึง 15 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 4.62% ที่ช่วงความเร็วสูงสุด 15 ถึง 20 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 2.50% จากผลการทดลองในภาพรวมพบว่า เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 20 โหนดนั้น DSR-BA มีค่า Average end-to-end delay มากกว่าในทุกช่วงความเร็วสูงสุดในการเคลื่อนที่ของโหนดประมาณ 2.50-4.62%

1.1.3 จำนวนโหนดที่ส่งแพคเกตข้อมูล 30 โหนด



ภาพที่ 26 ค่าความหน่วงเฉลี่ย end-to-end ของแพคเกต CBR ทั้งหมด เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 30 โหนด ภายในเครือข่ายขนาด 1500×300 ตารางเมตร

จากผลการทดลองภาพที่ 26 แสดงให้เห็นการทำงานของโพรโทคอลทั้ง DSR และ DSR-BA พบว่า Average end-to-end delay ที่ช่วงความเร็วสูงสุด 0 ถึง 5 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 3.40% ที่ช่วงความเร็วสูงสุด 5 ถึง 10 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 6.14% ที่ช่วงความเร็วสูงสุด 10 ถึง 15 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 2.88% ที่ช่วงความเร็วสูงสุด 15 ถึง 20 เมตรต่อวินาที DSR-BA มีค่ามากกว่า DSR ประมาณ 1.98% จากผลการทดลองในภาพรวมพบว่า เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 30 โหนดนั้น DSR-BA มีค่า Average end-to-end delay มากกว่าในทุกช่วงความเร็วสูงสุดในการเคลื่อนที่ของโหนดประมาณ 1.98-6.14%

จากผลการทดลองภาพที่ 24 25 และ 26 แสดงให้เห็นการทำงานของโพรโทคอลทั้ง DSR และ DSR-BA ในสภาพเครือข่ายที่โหนดมีการเคลื่อนที่ด้วยความเร็วต่ำสุด 0 เมตรต่อวินาทีและความเร็วสูงสุด 5 10 15 และ 20 เมตรต่อวินาที ภายในเครือข่ายขนาด 1500x300 ตารางเมตร และโหนดเริ่มต้นจำนวน 10 20 และ 30 โหนดตามลำดับ โดยแต่ละโหนดเริ่มต้นมีการส่งแพคเกตข้อมูลขนาด 512 ไบต์จำนวน 4 แพคเกตทุกๆ 1 วินาทีในช่วงระยะเวลา 600 วินาที

เมื่อพิจารณาในภาพรวมการทำงานของทั้งสองโพรโทคอลพบว่า DSR-BA มีความหน่วงเฉลี่ย end-to-end ของแพคเกต CBR ทั้งหมดที่เกิดขึ้นสูงกว่า DSR โดยค่าจะเพิ่มขึ้นตามความเร็วในการเคลื่อนที่ของโหนดที่เคลื่อนที่เร็วขึ้น และเพิ่มตามจำนวนโหนดเริ่มต้นที่เพิ่มขึ้นด้วย

เมื่อพิจารณาที่ช่วงความเร็วสูงสุด 0 5 และ10 เมตรต่อวินาที จากภาพที่ 24 มีโหนดเริ่มต้นจำนวน 10 โหนด พบว่าช่วงเวลาดังกล่าว DSR สามารถส่งแพคเกตข้อมูลไปถึงยังโหนดปลายทางได้มากกว่า จึงทำให้ค่าความหน่วงเฉลี่ย end-to-end ของแพคเกตน้อยกว่า ทั้งนี้เพราะเส้นทางที่ DSR ใช้นั้นเป็นเส้นทางที่สั้นกว่าเส้นทางที่ DSR-BA เลือกใช้ และ DSR มีการใช้งานเส้นทางเดิมนานกว่า DSR-BA ก่อนที่ DSR จะทำการเปลี่ยนเส้นทางหลังจากที่เส้นทางขาดการเชื่อมต่อไปแล้ว ในขณะที่ DSR-BA หลังจากที่ตรวจพบเส้นทางที่อาจจะขาดการเชื่อมต่อ แต่เนื่องจากเส้นทางดังกล่าวสามารถใช้ส่งแพคเกตได้อีกช่วงระยะเวลาหนึ่ง ก่อนที่เส้นทางดังกล่าวจะขาดการเชื่อมต่อ ซึ่ง DSR-BA ได้ทำการลบเส้นทางดังกล่าวไปแล้ว และเลือกใช้เส้นทางที่มีอยู่ในแคชเส้นทางซึ่งเป็นเส้นทางที่ยาวกว่า DSR จึงทำให้มีค่าความหน่วงเฉลี่ย end-to-end ของแพคเกตที่มากกว่านั่นเอง

แต่ในช่วงความเร็วสูงสุด 10 และ 15 เมตรต่อวินาที ทั้งโพรโทคอล DSR และ DSR-BA มีค่าความหน่วงเฉลี่ย end-to-end ของแพคเกตใกล้เคียงกัน เนื่องจากเมื่อโหนดเคลื่อนที่ด้วยความเร็วระดับหนึ่ง ช่วงเวลาที่เส้นทางขาดการเชื่อมต่อจะเร็วขึ้น ทำให้ DSR เปลี่ยนเส้นทางมาใช้เส้นทางเดียวกับ DSR-BA เร็วขึ้นด้วย เวลาที่โหนดใช้ในการส่งแพคเกตจึงใกล้เคียงกัน และถ้าโหนดเคลื่อนที่เร็วขึ้นอีกคือในช่วงความเร็วสูงสุด 15 และ 20 เมตรต่อวินาที DSR-BA จะเลือกใช้เส้นทางใหม่ได้รวดเร็วกว่าเนื่องจากเส้นทางจะขาดเร็วขึ้น จึงสามารถตรวจพบเส้นทางที่อาจจะขาดการเชื่อมต่อได้บ่อยขึ้น ทำให้เส้นทางที่มีอยู่จะถูกกำจัดได้มากขึ้น ถ้าเส้นทางที่เก็บไว้มีน้อยจนถูกกำจัดออกไปจนหมด โหนดจะเริ่มค้นหาเส้นทางใหม่ซึ่งจากการที่โหนดเคลื่อนที่ไปอย่างอิสระนั้น ทำให้เส้นทางที่ได้ใหม่นี้เป็นเส้นทางที่สั้นกว่าเส้นทางที่ DSR ยังคงใช้งานอยู่ ส่งผลให้ค่าความหน่วงเฉลี่ย end-to-end ของแพคเกตลดลง

เมื่อเพิ่มจำนวนโหนดเริ่มต้นเป็น 20 และ 30 โหนด จะทำให้การจราจรของแพคเกตข้อมูลมีสูงขึ้น แต่ละโหนดจะเก็บข้อมูลเส้นทางไว้ในแคชเส้นทางได้มากขึ้น นั่นคือโหนดจะมีหลายเส้นทางรองรับการเชื่อมต่อไปยังโหนดปลายทางได้หลายเส้นทาง ซึ่งก็มีข้อดีข้อเสียคือถ้าเส้นทางที่เก็บไว้นั้นยังคงใช้งานได้ โหนดจะไม่ต้องเสียเวลาในการค้นหาเส้นทางใหม่ แต่ถ้า

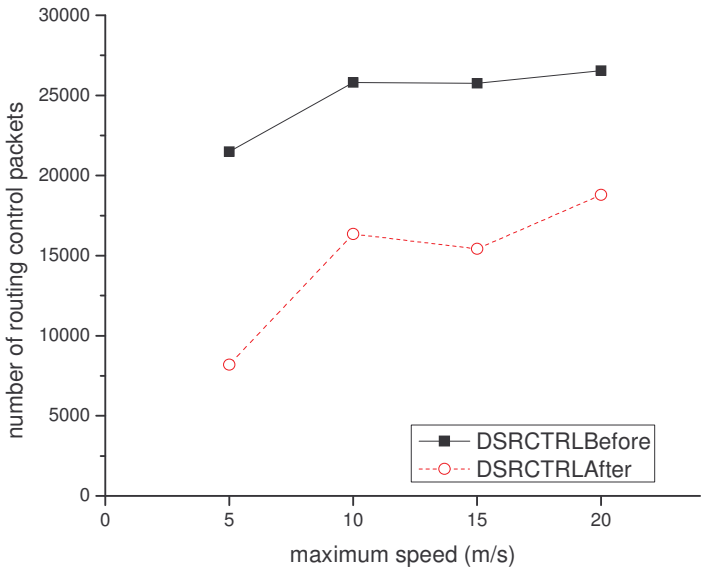
เส้นทางดังกล่าวใช้งานไม่ได้ แพคเกตข้อมูลก็จะสูญหายไป และถ้ามีจำนวนเส้นทางที่ใช้ไม่ได้ค้างอยู่ในแคชเส้นทางเป็นจำนวนมาก เนื่องจากเส้นทางไปโหนดปลายทางไปได้หลายเส้นทางมากกว่าจะได้เส้นทางที่ใช้งานได้จริง แพคเกตก็มีการสูญหายไปมากทีเดียว จากภาพที่ 25 และ 26 จะพบว่า DSR-BA มีค่าความหน่วงเฉลี่ย end-to-end ของแพคเกตที่สูงในทุกค่าความเร็วสูงสุดที่เพิ่มขึ้น ทั้งนี้เพราะเมื่อโหนดตรวจพบเส้นทางที่อาจจะขาดการเชื่อมต่อบ่อยครั้งขึ้น เส้นทางใหม่ในแคชเส้นทางก็จะถูกดึงออกมาใช้มากยิ่งขึ้น ซึ่งเส้นทางเหล่านี้มีเก็บไว้เป็นจำนวนมาก ประกอบกับเป็นเส้นทางที่ยาวขึ้น ทำให้ DSR-BA มีโอกาสมากกว่า DSR-BA ในการใช้เส้นทางที่ยาวขึ้นนี้ แต่ DSR-BA จะช่วยลดจำนวนเส้นทางที่ใช้ไม่ได้ที่มีอยู่ในแคชเส้นทางให้ลดลง และช่วยทำให้นำพาแพคเกตไปให้ถึงโหนดปลายทางได้เพิ่มขึ้นด้วย

ปัญหาและสิ่งที่ได้จากการทำวิจัย

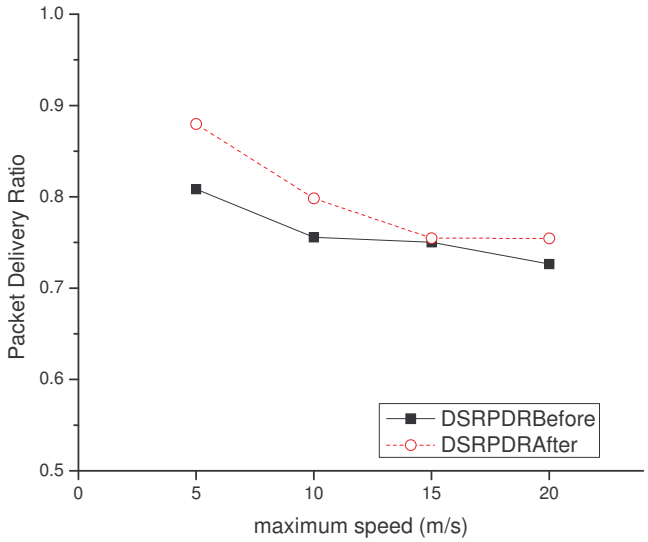
ปัญหาของโพรโทคอล DSR ในส่วนของโปรแกรมจำลองการทำงานของเครือข่ายไร้สาย จากการใช้โปรแกรมจำลองการทำงานเครือข่ายไร้สาย GloMoSim พบข้อผิดพลาดในตัวโปรแกรมที่จำลองการทำงานของโพรโทคอล DSR คือ โหนดตัวกลางที่ทำหน้าที่ส่งต่อ RERR ไม่ได้รับ RERR ตามที่แพคเกตได้ระบุโหนดที่จะทำหน้าที่ส่งแพคเกตดังกล่าวไว้ในส่วนหัวของแพคเกต ซึ่งในที่นี้มีเพียง next node เพียงหนึ่งฮอพเท่านั้นที่ได้รับแพคเกตดังกล่าวนี้ ที่เป็นเช่นนี้เพราะเกิดผิดพลาดที่ตัวโปรแกรมขณะกำหนดจำนวนฮอพเพื่อส่งไปยังโหนดที่ได้กำหนดไว้ให้ทำหน้าที่ในการส่งต่อแพคเกตนั่นเอง ความผิดพลาดครั้งนี้ส่งผลทำให้ RERR ส่งไปไม่ถึงยังโหนดปลายทางที่ระบุไว้ในแพคเกต ดังนั้นโหนดปลายทางดังกล่าวรวมถึงโหนดที่ดักฟังแพคเกตจึงไม่ได้ทำการลบเส้นทางที่ขาดการเชื่อมต่อนั้นออกจากแคชเส้นทาง ซึ่งจะทำให้ในแคชเส้นทางของโหนดยังคงมีเส้นทางที่ขาดการเชื่อมต่อนี้ อีกทั้งทำให้มีจำนวนแพคเกตควบคุมเพิ่มสูงขึ้นด้วยดังภาพที่ 27 แสดงจำนวนของแพคเกตควบคุมที่เกิดขึ้น เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 10 โหนด ภายในเครือข่ายขนาด 1500x300 ตารางเมตร จำนวน 10 รอบการจำลองโดยเปรียบเทียบโพรโทคอล DSR ระหว่างก่อนแก้ข้อผิดพลาดและหลังจากแก้ข้อผิดพลาด จะเห็นว่าจำนวนแพคเกตควบคุมที่ใช้ไปก่อนการแก้ไขข้อบกพร่องนั้นมีจำนวนที่มากกว่า และจากการที่โหนดไม่ได้กำจัดเส้นทางที่ขาดการเชื่อมต่อออกไปทำให้เส้นทางที่มีอยู่นั้นใช้งานไม่ได้ ดังนั้นเมื่อโหนดจำเป็นต้องใช้เส้นทางเพื่อส่งแพคเกตจึงไม่สามารถใช้งานได้ ทำให้แพคเกตสูญหาย ส่งผลถึงค่าของ Packet delivery ratio ดังภาพที่ 28 แสดงค่าของ Packet delivery ratio โดยที่ก่อนจะแก้ไขข้อผิดพลาดนั้นจะมีค่าที่ต่ำกว่าเมื่อนำไปเปรียบเทียบกับค่าของ Packet delivery ratio หลังจากที่ได้ทำการแก้ไขข้อผิดพลาด

ดังกล่าวแล้ว นอกจากนี้ยังส่งผลถึงค่าความหน่วงเฉลี่ย end-to-end ของแพคเกตที่ลดลงด้วยเมื่อได้ทำการแก้ไขแล้วดังภาพที่ 29 แสดงค่าความหน่วงเฉลี่ย end-to-end เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 10 โหนด ที่เป็นเช่นนี้เพราะว่าหลังจากได้แก้ไขแล้ว โหนดจะสามารถกำจัดเส้นทางที่ใช้งานไม่ได้ออกไปจากแคชเส้นทางของตนเองได้มากขึ้น ทำให้เส้นทางที่เลือกใช้งานสามารถใช้งานได้มากขึ้น โหนดจะเชื่อมต่อเส้นทางได้ไวโดยเฉพาะเมื่อโหนดต้องทำหน้าที่นำพาแพคเกตข้อมูลไปให้ถึงยังโหนดปลายทาง

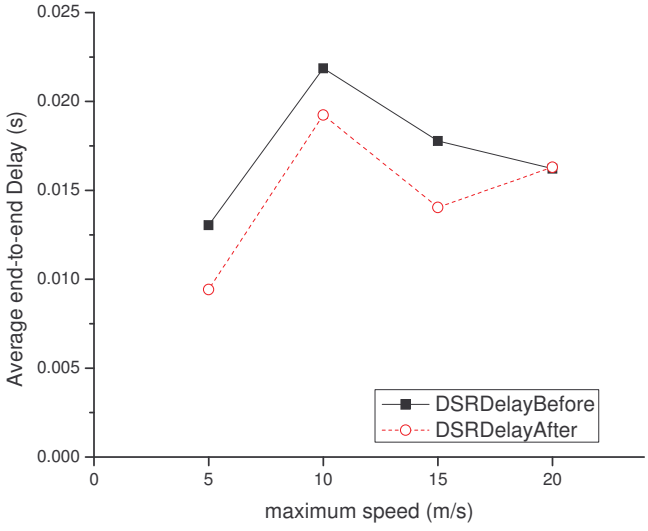
ปัญหาของโพรโทคอล DSR ในเรื่องการสูญหายแพคเกตข้อมูลนั้น จากการวิจัยพบว่าเป็นเพราะเส้นทางที่เก็บในแคชเส้นทางโดยส่วนใหญ่จะเป็นเส้นทางที่เก่าไม่สามารถใช้งานได้ โดยเฉพาะเมื่อโหนดมีการเคลื่อนที่ตลอดเวลาและด้วยความเร็วสูง ดังนั้นโหนดในเครือข่ายจึงมีแต่ข้อมูลเส้นทางที่อาจจะไม่สามารถใช้งานได้จำนวนมากเก็บอยู่ และด้วยกระบวนการของการดักฟังแพคเกตข้อมูลที่โหนดจะส่งเส้นทางที่สั้นกว่าไปยังโหนดเริ่มต้นเพื่อให้โหนดเปลี่ยนเส้นทางนั้น มีโอกาสที่โหนดจะส่งเส้นทางที่สั้นกว่าจริงแต่เป็นเส้นทางที่ใช้งานไม่ได้แล้วกลับไปให้ ดังนั้นเมื่อโหนดเริ่มต้นเปลี่ยนมาใช้เส้นทางนี้ จึงเกิดการสูญหายแพคเกต



ภาพที่ 27 จำนวนของแพคเกตควบคุมที่เกิดขึ้น เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 10 โหนด ภายในเครือข่ายขนาด 1500×300 ตารางเมตร จำนวน 10 รอบการจำลอง เปรียบเทียบโพรโทคอล DSR ระหว่างก่อนแก้ข้อผิดพลาดของแบบจำลองเครือข่ายต้นแบบและหลังจากแก้ข้อผิดพลาดของแบบจำลองเครือข่ายต้นแบบ



ภาพที่ 28 ค่าของ Packet delivery ratio เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 10 โหนด ภายในเครือข่ายขนาด 1500×300 ตารางเมตร จำนวน 10 รอบการจำลอง เปรียบเทียบโพรโทคอล DSR ระหว่างก่อนแก้ข้อผิดพลาดของแบบจำลองเครือข่ายต้นแบบและหลังจากแก้ข้อผิดพลาดของแบบจำลองเครือข่ายต้นแบบ



ภาพที่ 29 ค่าความหน่วงเฉลี่ย end-to-end เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 10 โหนด ภายในเครือข่ายขนาด 1500×300 ตารางเมตร จำนวน 10 รอบการจำลอง เปรียบเทียบโพรโทคอล DSR ระหว่างก่อนแก้ข้อผิดพลาดของแบบจำลองเครือข่ายต้นแบบและหลังจากแก้ข้อผิดพลาดของแบบจำลองเครือข่ายต้นแบบ

จากการที่โพรโทคอล ต้องการใช้ประโยชน์จากแคชเส้นทางให้มากที่สุดด้วยการส่งเส้นทางที่เก็บไว้ในแคชเส้นทางไปให้กับโหนดเริ่มต้นเมื่อโหนดเริ่มต้นส่ง RREQ มานั้น โหนดใดๆ ก็ตามที่มีเส้นทางไปยังโหนดปลายทางได้ โหนดดังกล่าวนั้นจะส่งเส้นทางกลับไปยังโหนดเริ่มต้นทันทีทั้งๆ ที่เส้นทางดังกล่าวเป็นเส้นทางที่ใช้ไม่ได้แล้ว ดังนั้นโหนดเริ่มต้นก็ยังคงได้เส้นทางที่ไม่สามารถส่งแพคเกตข้อมูลไปให้ถึงยังโหนดปลายทางได้ จึงทำให้แพคเกตสูญหายอีก

การใช้เส้นทางในแคชเส้นทางที่โหนดได้ทำการค้นหาในขณะเริ่มต้นนั้น อาจจะเป็นเส้นทางที่ยาวกว่า เมื่อโหนดเริ่มใช้งานเส้นทางดังกล่าวได้สักช่วงเวลาหนึ่ง เนื่องจากโหนดมีการเคลื่อนที่อย่างอิสระนั่นเอง ดังนั้นการใช้เส้นทางเดิมตลอดถ้าโหนดไม่ขาดการเชื่อมต่อหรือใช้เส้นทางจากแคชเส้นทางที่เก็บไว้ตั้งแต่เริ่มต้น อาจทำให้มีค่าการหน่วงของแพคเกตข้อมูลที่สูงกว่าการใช้เส้นทางที่สั้นๆ ได้

ในการส่ง RERR เมื่อเส้นทางขาดการเชื่อมต่อไปแล้ว โหนดยังทำได้ไม่ดีนัก เนื่องจากจะมีเฉพาะบางโหนดเท่านั้นที่รับทราบการแจ้งความผิดพลาดดังกล่าว ซึ่งมีบางโหนดที่เก็บเส้นทางที่ขาดการเชื่อมต่ออยู่นั้นได้เคลื่อนที่ออกไปแล้ว ทำให้ไม่ได้รับทราบการแจ้งความผิดพลาดนั้นๆ เมื่อโหนดได้รับ RREQ โหนดจะนำเส้นทางดังกล่าวไปแจ้งให้กับโหนดอื่นๆ ใช้ต่อไป ซึ่งหลักในการเลือกเส้นทางการส่งแพคเกตข้อมูลของโพรโทคอล DSR นั้น จะเลือกเส้นทางที่สั้นที่สุดที่เก็บอยู่ในแคชเส้นทาง ถ้าเส้นทางดังกล่าวเป็นเส้นทางที่ใช้ไม่ได้แล้วแต่เป็นเส้นทางที่สั้นที่สุด โหนดก็ยังคงเลือกใช้แม้จะมีเส้นทางในแคชเส้นทางให้เลือกมากมาย กระบวนการเก็บเส้นทางไว้ในแคชเส้นทางนั้น โหนดจะเก็บเส้นทางที่ได้ดักฟังจากแพคเกตตอบกลับทั้งที่ตอบกลับมาจากโหนดปลายทางและโหนดตัวกลาง ซึ่งถ้าตนเองมีเส้นทางที่สั้นที่สุดในแคชเส้นทางแต่เป็นเส้นทางที่ใช้ไม่ได้ แล้วได้รับทราบเส้นทางที่ยาวกว่าแต่ใช้งานได้ โหนดก็จะเก็บไว้แต่จะไม่ถูกเรียกใช้เป็นลำดับแรกโดยโหนดจะเรียกใช้เส้นทางที่สั้นที่สุดที่ตัวเองมีอยู่ก่อน และอีกกรณีโหนดมีเส้นทางที่ยาวกว่าแต่ยังสามารถใช้งานได้เก็บอยู่ในแคช เมื่อได้ยินเส้นทางจาก RREP ที่มาจากโหนดใดๆ ในเครือข่ายและเป็นเส้นทางที่สั้นกว่าเส้นทางในแคชเส้นทางที่ตนเองมีอยู่ แต่เส้นทางนี้เป็นเส้นทางที่ใช้งานไม่ได้แล้ว โหนดยังคงเก็บไว้ใช้เป็นลำดับแรกโดยไม่ทราบว่าเป็นเส้นทางที่ขาดการเชื่อมต่อไปแล้ว ซึ่งถ้าโหนดใช้เส้นทางนี้อาจจะมีการสูญหายแพคเกตได้ แม้เส้นทางนั้นจะเป็นเส้นทางที่สั้นกว่าก็ตาม

ทั้งนี้ทั้งนั้นที่กล่าวมาโหนดไม่สามารถบอกได้ว่าเส้นทางใดในแคชเส้นทางสามารถใช้งานได้ โดยจะบอกได้เมื่อโหนดได้ใช้เส้นทางนั้นแล้ว หรือมีกระบวนการตรวจสอบที่ชัดเจน ซึ่งยังคงต้องค้นคว้า วิจัยและพัฒนาต่อไป

เนื่องจากการที่โพรโทคอล DSR นี้ เน้นการใช้เส้นทางจากโหนดเริ่มต้นและใช้ประโยชน์จากแคชเส้นทาง ซึ่งผู้วิจัยมีความเห็นว่า ในการวิจัยและพัฒนาโพรโทคอล DSR ต่อๆ ไปนั้น ควรมีเป้าหมายอยู่ที่เส้นทางลำดับต้นๆ ที่โหนดจะเลือกใช้งานนั้น ควรเป็นเส้นทางที่สามารถใช้งานได้ทันทีโดยไม่ทำให้โหนดขาดการเชื่อมต่อ และควรทำให้เส้นทางที่โหนดเริ่มต้นเก็บไว้ในแคชเส้นทางเป็นเส้นทางที่สามารถพร้อมใช้งานให้มากที่สุดและสั้นที่สุด

จากเป้าหมายดังกล่าว ผู้วิจัยขอนำเสนอวิธีการเพิ่มเติมคือให้โหนดที่พบปัญหาทำการแจ้งเส้นทางดังกล่าวให้กับโหนดเริ่มต้นรับทราบความผิดพลาดของเส้นทางทุกครั้งที่โหนดอาจจะขาดการเชื่อมต่อทั้งที่เหตุการณ์เกิดขึ้นใกล้กับโหนดเริ่มต้นและโหนดปลายทาง รวมถึงเมื่อเส้นทางขาดการเชื่อมต่อจากการส่งแพคเกตควบคุมที่ใช้ตรวจสอบเส้นทางด้วย

จากนั้นเมื่อพิจารณาที่แคชเส้นทางในโหนดตัวกลางควรจะเป็นเส้นทางที่ใช้งานได้ทันทีเพื่อนำพาแพคเกตข้อมูลไปให้ถึงยังโหนดปลายทาง รวมถึงการเลือกส่งเส้นทางตอบกลับที่ใช้งานได้ส่งตอบกลับไปยังโหนดเริ่มต้นโดยนำเสนอวิธีการเลือกเส้นทางที่ next node เป็นโหนดที่เพิ่งได้รับทราบการคงอยู่ของโหนดในครั้งล่าสุดก่อนและเป็นเส้นทางที่สั้นที่สุดที่มีอยู่ในแคชเส้นทางโดยพิจารณาจากการดักฟังแพคเกตที่เพิ่งได้รับทราบการคงอยู่ของโหนด จากนั้นทำการตรวจสอบเส้นทางดังกล่าว หรือเลือกตอบกลับด้วยเส้นทางดังกล่าว

สรุปและข้อเสนอแนะ

สรุป

จากผลการทดลองแสดงให้เห็นถึงการทำงานของโพรโทคอลทั้ง DSR และ DSR-BA พบว่า DSR-BA มีการสิ้นเปลืองจำนวนแพคเกตควบคุมที่สูงกว่า DSR อีกทั้งมีค่าความหน่วงเฉลี่ย end-to-end ของแพคเกต CBR ทั้งหมดที่เกิดขึ้นมากกว่า แต่ค่า Packet delivery ratio ดีกว่าเล็กน้อยในบางกรณี ได้แก่ กรณีที่โหนดมีการเคลื่อนที่ในช่วงความเร็วสูงสุด 20 เมตรต่อวินาที เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 10 20 และ 30โหนด และกรณีที่โหนดมีการเคลื่อนที่ในช่วงความเร็วสูงสุด 0-5 เมตรต่อวินาที และ 15-20 เมตรต่อวินาที เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 20 และ 30โหนด เป็นต้น โดย DSR-BA ไม่เหมาะที่จะนำไปใช้ในกรณีที่โหนดมีการเคลื่อนที่ในช่วงความเร็วสูงสุด 5-15 เมตรต่อวินาที เมื่อมีจำนวนโหนดที่ส่งแพคเกตข้อมูล 10 20 และ 30โหนด ซึ่ง DSR-BA จะเหมาะกับเครือข่ายที่โหนดมีการเคลื่อนที่ไม่มากนัก และมีปริมาณการรับส่งข้อมูลสูงภายในเครือข่าย

เมื่อพิจารณาจากช่วงความเร็วสูงสุดในการเคลื่อนที่ของโหนดพบว่า DSR-BA จะทำงานได้ดีในสภาวะที่โหนดมีการเคลื่อนที่ในช่วงความเร็วสูงสุด 20 เมตรต่อวินาที ที่จำนวนโหนดเริ่มต้น 10 โหนด เนื่องจากในสภาพการณ์ดังกล่าว โหนดมีการเคลื่อนที่เร็วทำให้เส้นทางการติดต่อสื่อสารขาดการเชื่อมต่อได้ง่าย ซึ่ง DSR-BA ทำการตรวจสอบเส้นทางบ่อยครั้ง ทำให้โหนด ลบเส้นทางได้บ่อยขึ้น ข้อมูลเส้นทางในแคชเส้นทางจึงมีเส้นทางที่อาจจะขาดการเชื่อมต่อหรือใช้ไม่ได้ลดลง แต่การได้มาซึ่งประโยชน์ในส่วนนี้จะต้องแลกมาด้วยค่าแพคเกตควบคุมที่มากขึ้นและค่าความหน่วงเฉลี่ย end-to-end ของแพคเกต CBR ที่เพิ่มขึ้นมาจากเส้นทางใหม่ในแคชเส้นทางที่ยาวกว่าเส้นทางเดิม แต่สามารถนำพาแพคเกตข้อมูลไปยังโหนดปลายทางได้มากขึ้น

ในขณะเดียวกันด้วยจำนวนโหนดเริ่มต้นที่มีจำนวนน้อย ทำให้การจราจรของแพคเกตข้อมูลมีไม่มากนัก นั่นทำให้เส้นทางที่เก็บในแคชเส้นทางไปยังโหนดปลายทางมีจำนวนไม่มากนัก เมื่อโหนดทำการลบเส้นทางบ่อยครั้งจนไม่มีเส้นทางไปยังโหนดปลายทาง นี่จะเป็นโอกาสที่โหนดเริ่มต้นจะทำการค้นหาเส้นทางใหม่และได้มาซึ่งเส้นทางที่สั้นกว่าเดิม ส่งผลให้ค่าความหน่วงเฉลี่ย end-to-end ของแพคเกต CBR มีค่าลดลง

งานวิจัยที่จะทำต่อคือการแจ้งให้โหนดเริ่มต้นทราบถึงเส้นทางที่อาจขาดการเชื่อมต่อ ทั้งนี้เพื่อให้โหนดเริ่มต้นสามารถค้นหาเส้นทางใหม่ที่สั้นกว่าโดยเร็วที่สุด ก่อนที่เส้นทางที่ใช้อยู่จะขาดการเชื่อมต่อ ทำได้โดยการส่งแพคเกตแจ้งความผิดพลาดไปยังโหนดเริ่มต้นหลังจากที่โหนดตรวจพบเส้นทางที่อาจขาดการเชื่อมต่อ หลังจากโหนดเริ่มต้นรับทราบโหนดจะทำการตรวจสอบเส้นทางที่สั้นที่สุดที่มีอยู่ก่อน หลังจากตรวจสอบเสร็จสิ้นและเส้นทางใช้งานได้ โหนดเริ่มต้นจึงจะเริ่มใช้เส้นทางนั้นในการส่งแพคเกตข้อมูล

ข้อเสนอแนะ

งานวิจัยฉบับนี้ได้จำลองการทำงานของโพรโทคอล DSR-BA ด้วยการนำเสนอการตรวจพบเส้นทางที่อาจจะขาดการเชื่อมต่อ และทำการค้นหาเส้นทางใหม่จากแคชเส้นทางมาใช้ก่อนที่เส้นทางเดิมจะขาดการเชื่อมต่อ เนื่องจากในขณะที่โหนดได้ตรวจพบปัญหาที่อาจจะเกิดขึ้นกับเส้นทาง เริ่มต้นจากกระบวนการตรวจสอบนั้นมีเรื่องของเวลาและระยะทางการเคลื่อนที่ของโหนดที่ส่งผลถึงเส้นทางการเชื่อมต่ออาจจะขาดการติดต่อสื่อสารได้เข้ามามีส่วนเกี่ยวข้อง ในงานวิจัยนี้ได้มีการกำหนดค่าขีดเริ่มเปลี่ยนเบื้องต้นเพื่อทำการตรวจพบเส้นทางที่อาจจะขาดการเชื่อมต่อ ซึ่งค่าขีดเริ่มเปลี่ยนดังกล่าวนี้สามารถปรับเปลี่ยนได้เพื่อหาระยะที่เหมาะสมเพื่อแจ้งให้โหนดทราบและเริ่มกระบวนการตรวจสอบเส้นทาง ทั้งนี้เพื่อให้โพรโทคอลมีประสิทธิภาพในการตรวจพบเส้นทางที่อาจจะขาดการเชื่อมต่อที่ดีขึ้น นอกจากนี้หลังจากผ่านกระบวนการตรวจสอบแล้ว เมื่อโหนดได้รับแจ้งให้ทำการตรวจสอบเส้นทาง ในงานวิจัยนี้กำหนดให้โหนดเริ่มทำกระบวนการตรวจสอบและใช้เส้นทางใหม่ทันทีหลังจากที่โหนดได้รับการแจ้งปัญหาที่อาจจะเกิดขึ้นกับเส้นทางที่ใช้งานอยู่ ซึ่งกระบวนการนี้สามารถปรับเปลี่ยนได้โดยการรอให้เส้นทางขาดการเชื่อมต่อก่อนจากนั้นจึงใช้เส้นทางที่ได้ทำการตรวจสอบนี้ ทั้งนี้เพื่อให้เส้นทางที่ใช้งานอยู่นั้น ยังคงสามารถส่งแพคเกตได้อีกจำนวนหนึ่งด้วยจำนวนฮอพที่น้อยกว่าการใช้เส้นทางใหม่ทันทีซึ่งเป็นเส้นทางที่มีจำนวนฮอพที่มากกว่า ส่งผลให้มีค่าความหน่วงของแพคเกตลดลงได้ เนื่องจากเส้นทางเดิมที่ใช้งานอยู่นั้น แม้ว่าโหนดได้ตรวจพบเส้นทางอาจจะขาดการเชื่อมต่อแต่บางครั้งเส้นทางดังกล่าว ไม่ขาดการเชื่อมต่อโดยทันที เนื่องจากโหนดในเครือข่ายมีการเคลื่อนที่แบบสุ่มไม่สามารถคาดการณ์ได้แน่นอน และบางครั้งเส้นทางดังกล่าวไม่ได้ขาดการเชื่อมต่อขึ้นจริง นอกจากนี้ในส่วนของแคชเส้นทางยังคงมีปัญหาเรื่องของเส้นทางเก่าที่เก็บไว้นานแล้ว และไม่สามารถใช้งานได้ ซึ่งข้อแนะนำวิธีการจัดการเส้นทางในแคชเส้นทางนอกเหนือจากผู้วัจัยหลายท่านได้นำเสนอมาก่อนนี้ โดยการกำหนดลำดับความสำคัญของเส้นทางที่มีโหนดถัดไปเป็นโหนดที่ได้รับทราบการคงอยู่ล่าสุดก่อนซึ่งแน่นอนว่าการต่อแพคเกตไปยังโหนดถัดไปจะสามารถส่งไปได้สำเร็จ กระบวนการรับทราบการคงอยู่นี้อาศัยโหมดของการดักฟังปกติที่โพรโทคอล DSR ใช้งานอยู่

เอกสารและสิ่งอ้างอิง

ศิวรักษ์ ศิวโมกษธรรม. 2546. **มาตรฐาน IEEE 802.11 WLAN: ความรู้เบื้องต้น ช่องโหว่ และการรักษาความปลอดภัย (ตอนที่ 1).** ศูนย์ประสานงานการรักษาความปลอดภัยคอมพิวเตอร์ประเทศไทย. แหล่งที่มา: http://thaicert.nectec.or.th/paper/wireless/IEEE80211_1.php, 10 สิงหาคม 2547.

Cheng, C., R. Riley, S.P.R. Kumar and J.J. Garcia-Luna-Aceves. 1989. A loop-free Bell-Ford routing protocol without bouncing effect. **ACM SIGCOMM.** 1989: 224-237.

Chlamtac, I., M. Conti and J. Liu. 2003. Mobile Ad hoc Networking: Imperatives and Challenges. **Elsevier Ad Hoc Network Journal.** 1(1): 13-64.

Das, S.R., C.E. Perkins and E.E. Royer. 2000. Performance Comparison of two on-demand routing protocols for ad hoc networks. **INFOCOM.** 2000: 3-12.

Johnson, D., D. Maltz, Y-C. Hu and J. Jetcheva. 2003. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Mobile Ad Hoc Networking Working Group, Internet Draft. Available Source: http://tools.ietf.org/wg/manet/draft-ietf-manet-dsr/draft-ietf-manet-dsr-09.txt, August 10, 2004.

Marina, M. and S. Das. 2001. Performance of route caching strategies in dynamic source routing. **Proceedings of the 21st International Conference on Distributed Computing Systems**, **ICDCSW.** 2001: 425.

Perkins, C.E. 2001. **AD HOC NETWORKING.** Addison-Wesley, United States of America.

Perkins, C.E., E.M. Royer and S. Das. 2002. Ad hoc On-demand Distance Vector (AODV) Routing. Mobile Ad Hoc Networking Working Group, Internet Draft. Available Source:

http://tools.ietf.org/wg/manet/draft-ietf-manet-aodv/draft-ietf-manet-aodv-11.txt, August 10, 2004.

Perkins, C.E. and P. Bhagwat. 1994. Highly Dynamic Destination-Sequenced Distance Vector (DSDV) for Mobile Computers. **ACM SIGCOMM.** 1994 : 234-244.

Royer, E.M. and C.K. Toh. 1999. A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks. **IEEE Personal Communications Magazine.** 6(2): 46-55.

Sengul, C. 2003. **Local Route Recovery in Mobile Ad Hoc Networks.** M.S. thesis, University of Illinois at Urbana-Champaign.

Aumted, S. and Chavalit S. 2006. Route Break Avoidance in DSR. **Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON.** 2006: May 10-13, 2006.

Tauchi, M., T. Ideguchi and T. Okuda. 2005. Ad-Hoc Routing Protocol Avoiding Route Breaks Based on AODV. **Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS.** 2005: 322a.

Terwilliger, M. and R. A. Meyer. 2001. GloMoSim: Global Mobile Information Systems Simulation Library. UCLA Parallel Computing Laboratory. University of California at Los Angeles. GloMoSim. Available Source: http://pcl.cs.ucla.edu/projects/glomosim, August 10, 2004.

Terwilliger, M. and R. A. Meyer. 2001. PARSEC: Parallel Simulation Environment for Complex System. UCLA Parallel Computing Laboratory. University of California at Los Angeles. PARSEC. Available Source: http://pcl.cs.ucla.edu/projects/parsec, August 10, 2004.

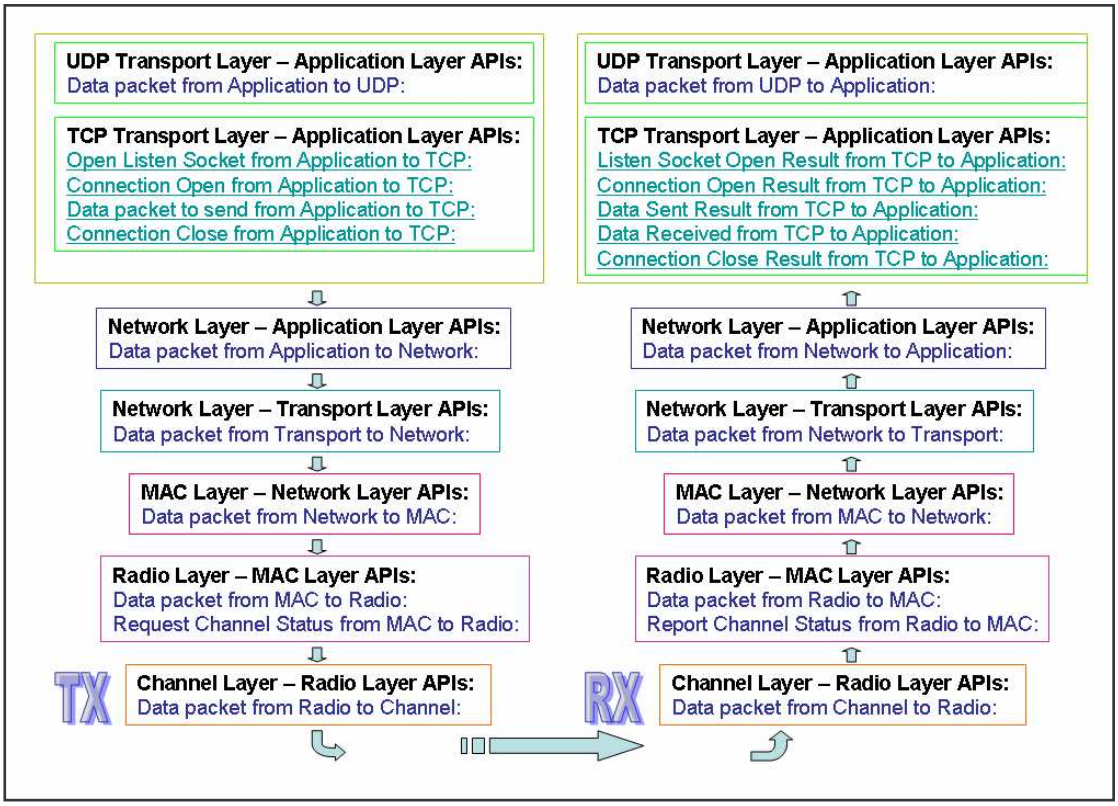
Wu, S.-L., T.-K. Lin, Y.-C. Tseng and J.-P. Sheu. 1999. Route Optimization on Wireless Mobile Ad-Hoc Networks. **The 5th Mobile Computing Workshop.** 1999: 143-150.

Yu, X. and Z. Kedem. 2005. A distributed adaptive cache update algorithm for the Dynamic Source Routing protocol. **Proceedings of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM.** 2005: 730-739.

ภาคผนวก

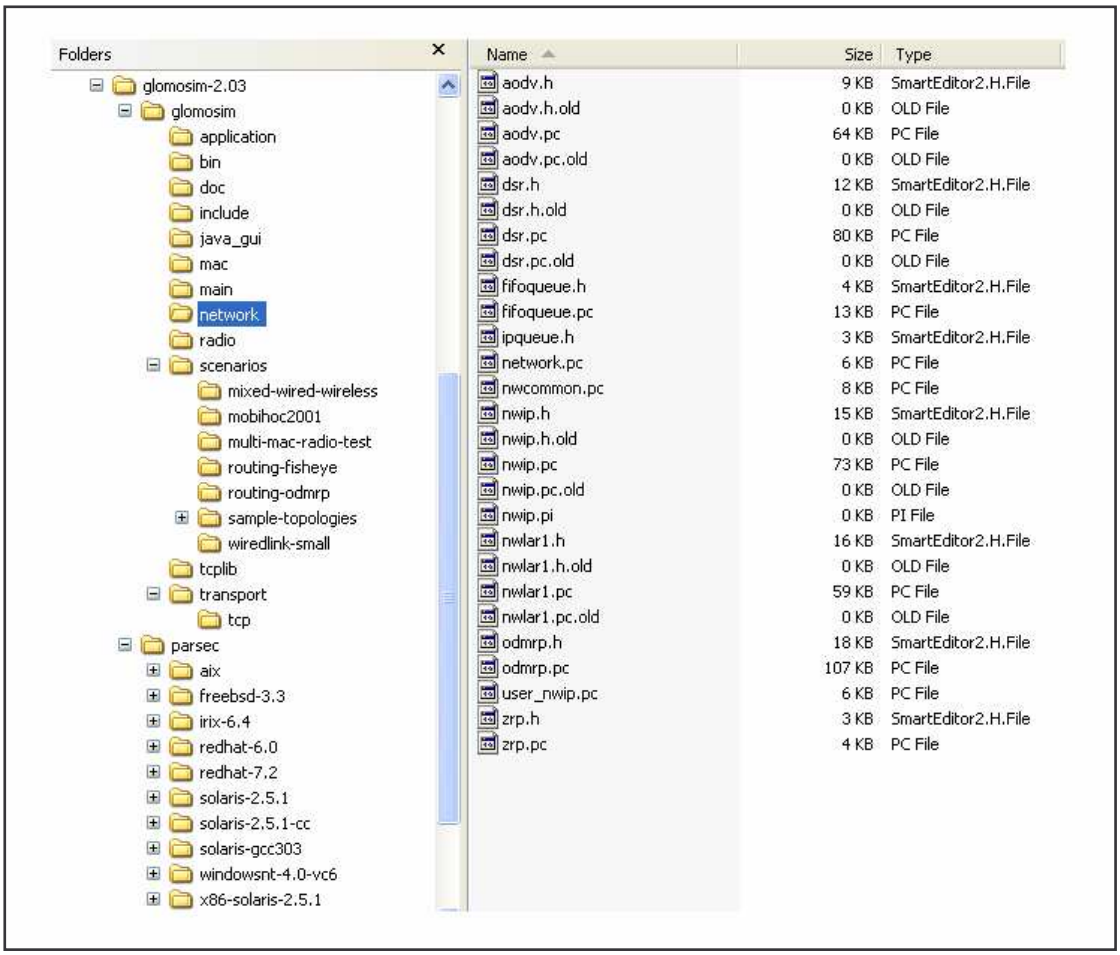
การใช้งานโปรแกรมการจำลองเครือข่าย GloMoSim

1. โครงสร้างของโปรแกรมการจำลองเครือข่าย GloMoSim



ภาพที่ 1 โครงสร้างการส่งแพคเกตข้อมูลด้วยโปรแกรมการจำลองเครือข่าย GloMoSim

ภาพที่ 1 แสดงลำดับการส่งแพคเกตข้อมูลจากชั้น Application และชั้น Transport ซึ่งในชั้น Application ได้จัดเตรียม FTP, TELNET, CBR และ HTTP เพื่อให้สะดวกต่อการใช้งาน ส่วนชั้น Transport ได้จัดเตรียม UDP และ TCP เมื่อแพคเกตส่งมาจากทั้งสองชั้นดังกล่าวจะส่งต่อไปยังชั้น Network ในชั้นนี้ได้จัดเตรียมโพรโทคอลสำหรับการค้นหาเส้นทางไว้หลายชนิด ได้แก่ AODV, DSR, BELLMANFORD, LAR1, OSPF, WRP, FISHEYE, NS_DSDV, STATIC เป็นต้น จากนั้นแพคเกตจะถูกส่งต่อไปยังชั้น MAC มีโพรโทคอลเช่น IEEE 802.11 เป็นต้น จากนั้นแพคเกตจะส่งต่อไปที่ชั้น Radio และชั้น Channel เพื่อส่งออกอากาศต่อไป



ภาพที่ 2 โครงสร้างการจัดวางโปรแกรมในชั้นต่างๆ ของโปรแกรมการจำลองเครือข่าย GloMoSim

ภาพที่ 2 แสดงให้เห็นถึงโครงสร้างในการจัดวางโปรแกรมในชั้นต่างๆ ของโปรแกรมการจำลองเครือข่าย GloMoSim โดยชั้น Application รวบรวมอยู่ในโฟลเดอร์ application ชั้น Transport รวบรวมอยู่ในโฟลเดอร์ transport ชั้น Network รวบรวมอยู่ในโฟลเดอร์ network ชั้น MAC รวบรวมอยู่ในโฟลเดอร์ mac ชั้น Radio และ Channel รวบรวมอยู่ในโฟลเดอร์ radio สำหรับโฟลเดอร์ bin ใช้สำหรับกระทำการแฟ้มข้อมูลเข้าและแฟ้มข้อมูลออก โฟลเดอร์ doc ได้รวบรวมเอกสารที่เป็นประโยชน์ต่อการใช้งานโปรแกรมการจำลองเครือข่าย GloMoSim โฟลเดอร์ include รวบรวมแฟ้มข้อมูล header ต่างๆ (header files) ที่จำเป็นต้องใช้ร่วมกันกับแฟ้มข้อมูลอื่นๆ โฟลเดอร์ java_gui มีข้อมูลเกี่ยวกับการจำลองการทำงานของเครือข่ายเสมือนจริง โฟลเดอร์ main บรรจุการออกแบบ framework พื้นฐานของการจำลองเครือข่าย โฟลเดอร์ scenarios มีตัวอย่างการกำหนดค่าการจำลองเครือข่ายในสถานการณ์ต่างๆ และโฟลเดอร์ tcplib มี libraries สำหรับ TCP

2. <u>การติดตั้งโปรแกรมการจำลองเครือข่าย GloMoSim</u>

2.1 เข้าสู่ระบบปฏิบัติการ Linux

2.2 ทำการคัดลอกแฟ้มข้อมูล glomosim-2.03.tar.gz ลงในเครื่อง

2.3 พิมพ์คำสั่ง gunzip –dfv glomosim-2.03.tar.gz

2.4 พิมพ์คำสั่ง tar –xfv glomosim-2.03.tar

2.5 พิมพ์คำสั่ง cd glomosim-2.03

2.6 ทำการคัดลอกโฟลเดอร์ parsec/redhat-7.2 ไว้ที่ /usr/local/parsec/redhat-7.2

2.7 พิมพ์คำสั่ง cd glomosim-2.03/glomosim/main

2.8 พิมพ์คำสั่ง export PCC_DIRECTORY=/usr/local/parsec/redhat-7.2/

2.9 พิมพ์คำสั่ง export PATH=$PATH:/usr/local/parsec/redhat-7.2/bin

2.10 พิมพ์คำสั่ง make แล้วไม่มีข้อความแสดงการผิดพลาด

2.11 กลับสู่โฟลเดอร์หลักของ GloMoSim คือ ./glomosim-2.03/

2.12 พิมพ์คำสั่งเพื่อทดสอบโปรแกรม cd glomosim-2.03/glomosim/bin

2.13 ./glomosim config.in

2.14 ไม่มีข้อความแสดงการผิดพลาดใดๆ

3. <u>การเพิ่มโพรโทคอลลงในโปรแกรมการจำลองเครือข่าย GloMoSim</u>

3.1 สร้างแฟ้มข้อมูลของโพรโทคอล DSR-BA ด้วย dsrba.pc และ dsrba.h

3.2 กำหนด IP ของโพรโทคอล DSR-BA ด้วย 165 ลงในแฟ้มข้อมูล nwcommon.h

3.3 ประกาศโพรโทคอล DSR-BA ลงในแฟ้มข้อมูล network.h

3.4 เพิ่มโพรโทคอล DSR-BA ลงในแฟ้มข้อมูล nwip.pc

3.5 เพิ่มโพรโทคอล DSR-BA ลงในแฟ้มข้อมูล application.pc

3.6 เพิ่มโพรโทคอล DSR-BA ลงในแฟ้มข้อมูล config.in

```
 *
 * File: nwCommon.h
 * By: Teresa Yan (tyan@cs.ucla.edu)
 * Objective: for routing protocol to update IP forwarding table and
 *            layer on top of IP to encapsulate IP header by setting
 *            certain fields in IP.
 * Date: 2/22/1999.
 */

#ifndef _NWCOMMON_H_
#define _NWCOMMON_H_

#include "message.h"

/* protocol number for IP */
#define IPPROTO_TCP  6
#define IPPROTO_UDP  17
#define IPPROTO_OSPF 87
#define IPPROTO_BELLMANFORD 520
#define IPPROTO_FISHEYE 530
#define IPPROTO_AODV 123
#define IPPROTO_DSR 135
#define IPPROTO_DSRBA 165
#define IPPROTO_ODMRP 145
#define IPPROTO_LAR1 110
#define IPPROTO_ZRP 133
#define NETWORK_UNREACHABLE   -2
#define DEFAULT_INTERFACE 0


/*  FUNCTION    NetworkInitForwardingTable
 *  PURPOSE     initialize the fowarding table, allocate enough memory for
 *              number of rows, used by ip
 *  PARAMETER   node
```

ภาพที่ 3 ตัวอย่างการกำหนด IP ของโพรโทคอล DSR-BA ลงในแฟ้มข้อมูล nwcommon.h

```
#define NETWORK_IP_DELAY                1 * MICRO_SECOND

typedef enum {
    NETWORK_PROTOCOL_IP = 0,
    ROUTING_PROTOCOL_AODV,
    ROUTING_PROTOCOL_DSR,
    ROUTING_PROTOCOL_DSRBA,
    ROUTING_PROTOCOL_LAR1,
    ROUTING_PROTOCOL_ODMRP,
    ROUTING_PROTOCOL_OSPF,
    ROUTING_PROTOCOL_ZRP,
    ROUTING_PROTOCOL_ALL,
    ROUTING_PROTOCOL_NONE
} NetworkRoutingProtocolType;


struct glomo_network_ip_struct;

/*
 * typedef to GlomoNetwork in main.h
 */

struct glomo_network_str
{
    //NETWORK_PROTOCOL networkProtocol;
    BOOL networkStats;
    BOOL guiOption;

    struct glomo_network_ip_struct* networkVar;
    void *routingVar;
};
```

ภาพที่ 4 ตัวอย่างการประกาศโพรโทคอล DSR-BA ลงในแฟ้มข้อมูล network.h

```
//
// FUNCTION   NetworkIpLayer
// PURPOSE    To handle Network IP layer events, incoming messages and
//            messages sent to itself (timers,etc).
// PARAMETERS
//    message - Abstract GloMoSim Message (Packet) to handle.
//

void NetworkIpLayer(GlomoNode *node, Message *msg) {
   switch (msg->protocolType) {

   case NETWORK_PROTOCOL_IP:
{
      switch(msg->eventType) {
...
...
   case ROUTING_PROTOCOL_AODV: {
      RoutingAodvHandleProtocolEvent(node, msg);
      break;   }
   case ROUTING_PROTOCOL_DSR: {
      RoutingDsrHandleProtocolEvent(node, msg);
      break;
   }
   case ROUTING_PROTOCOL_DSRBA: {
      RoutingDsrbaHandleProtocolEvent(node, msg);
      break;
   }
   case ROUTING_PROTOCOL_ZRP: {
      RoutingZrpHandleProtocolEvent(node, msg);
      break;
   }
...
...
   default:
      NetworkIpUserHandleProtocolEvent(node, msg);
      break;
   }//switch//
}//NetworkIpLayer
```

ภาพที่ 5 ตัวอย่างการเพิ่มโพรโทคอล DSR-BA ที่ฟังก์ชัน NetworkIpLayer() ในแฟ้มข้อมูล nwip.pc

```
//
// FUNCTION   NetworkIpInit
// PURPOSE    Initialize variables, being called once for each node
//            at the beginning.
// PARAMETERS
//            nodeInput -
//

void NetworkIpInit(GlomoNode* node, const GlomoNodeInput* nodeInput)
{
   GlomoNetworkIp* ipLayer = (GlomoNetworkIp *)node->networkData.networkVar;
   ...
   ...
   else if (strcmp(protocolString, "AODV") == 0) {
      ipLayer->routingProtocolChoice = ROUTING_PROTOCOL_AODV;
      RoutingAodvInit(
         node, (GlomoRoutingAodv**)&ipLayer->routingProtocol, nodeInput);
   }
   else if (strcmp(protocolString, "DSR") == 0) {
      ipLayer->routingProtocolChoice = ROUTING_PROTOCOL_DSR;
      RoutingDsrInit(
         node, (GlomoRoutingDsr**)&ipLayer->routingProtocol, nodeInput);
   }
   else if (strcmp(protocolString, "DSRBA") == 0) {
      ipLayer->routingProtocolChoice = ROUTING_PROTOCOL_DSRBA;
      RoutingDsrbaInit(node, (GlomoRoutingDsrba**)&ipLayer->routingProtocol, nodeInput);
   }
   else if (strcmp(protocolString, "ZRP") == 0) {
      ipLayer->routingProtocolChoice = ROUTING_PROTOCOL_ZRP;
      RoutingZrpInit(
         node, (GlomoRoutingZrp**)&ipLayer->routingProtocol, nodeInput);
   }
   ...
   ...
}//NetworkIpInit
```

ภาพที่ 6 ตัวอย่างการเพิ่มโพรโทคอล DSR-BA ที่ฟังก์ชัน NetworkIpInit() ในแฟ้มข้อมูล nwip.pc

```
//
// FUNCTION   NetworkIpFinalize()
// PURPOSE    Print out statistics, being called at the end.
//

void NetworkIpFinalize(GlomoNode *node)
{
   GlomoNetworkIp* ipLayer = (GlomoNetworkIp*)node->networkData.networkVar;
...
...
        break;
   case ROUTING_PROTOCOL_AODV:
      RoutingAodvFinalize(node);
      break;
   case ROUTING_PROTOCOL_DSR:
      RoutingDsrFinalize(node);
      break;
   case ROUTING_PROTOCOL_DSRBA:
      RoutingDsrbaFinalize(node);
      break;
   case ROUTING_PROTOCOL_ZRP:
      RoutingZrpFinalize(node);
      break;
...
...
   if (node->networkData.networkStats == TRUE) {
     NetworkIpPrintStats(node);
   }
}//NetworkIpFinalize
```

ภาพที่ 7 ตัวอย่างการเพิ่มโพรโทคอล DSR-BA ที่ฟังก์ชัน NetworkIpFinalize() ในแฟ้มข้อมูล nwip.pc

```
//
// FUNCTION  ProcessPacketForMeFromMac()
//
// PURPOSE   Direct the packet either to the transport layer or
//           to a routing protocol.
//

static void ProcessPacketForMeFromMac(GlomoNode *node, Message *msg, NODE_ADDR srcBA)
{
   GlomoNetworkIp* ipLayer = (GlomoNetworkIp *)node->networkData.networkVar;
...
...
   case IPPROTO_DSR: {
      RoutingDsrHandleProtocolPacket(node, msg, sourceAddress,
          destinationAddress, ttl);
      break;
   }
   case IPPROTO_DSRBA: {
      RoutingDsrbaHandleProtocolPacket(node, msg, sourceAddress,
        destinationAddress, ttl);
      break;
   }
   case IPPROTO_ZRP: {
      RoutingZrpHandleProtocolPacket(node, msg, sourceAddress,
          destinationAddress);
      break;
   }
...
...
   default:
      NetworkIpUserHandleProtocolPacket(node, msg, IpProtocol, sourceAddress,
          destinationAddress, ttl);
      break;
   }//switch//
}//ProcessPacketForMeFromMac//
```

ภาพที่ 8 ตัวอย่างการเพิ่มโพรโทคอล DSR-BA ที่ฟังก์ชัน ProcessPacketForMeFromMac() ในแฟ้มข้อมูล nwip.pc

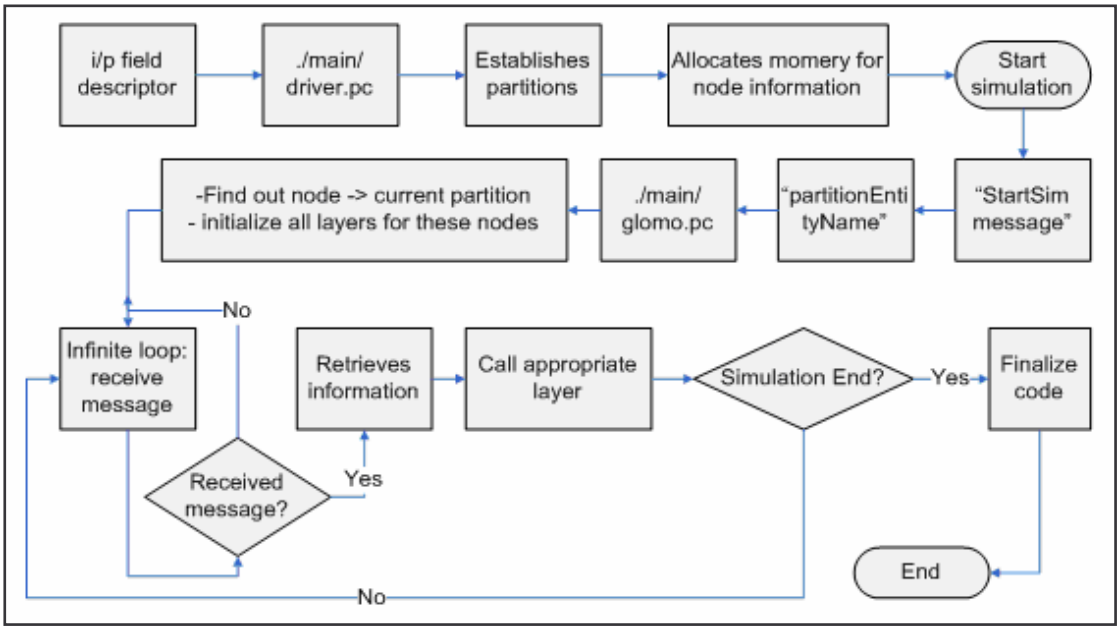
```
/*
 * NAME:        GLOMO_AppInit.
 * PURPOSE:     start applications on nodes according to user's
 *              specification.
 * PARAMETERS:  node - pointer to the node,
 *              nodeInput - configuration information.
 * RETURN:      none.
 */
void
GLOMO_AppInit(GlomoNode *node, const GlomoNodeInput *nodeInput)
{
    BOOL retVal;
    char buf[GLOMO_MAX_STRING_LENGTH];
...
...
    else if (strcmp(buf, "AODV") == 0) {
    }
    else if (strcmp(buf, "DSR") == 0) {
    }
    else if (strcmp(buf, "DSRBA") == 0) {
    }
...
...
    }
    /* Setting up applications. */

    GLOMO_AppInitApplications(node, nodeInput);

    AppLayerInitUserApplications(
       node, nodeInput, &node->appData.userApplicationData);

}//GLOMO_AppInit
```

ภาพที่ 9 ตัวอย่างการเพิ่มโพรโทคอล DSR-BA ในแฟ้มข้อมูล application.pc

```
##############################
#


#ROUTING-PROTOCOL    BELLMANFORD
#ROUTING-PROTOCOL    AODV
#ROUTING-PROTOCOL    DSR
ROUTING-PROTOCOL    DSRBA
#ROUTING-PROTOCOL    LAR1
#ROUTING-PROTOCOL    WRP
#ROUTING-PROTOCOL    FISHEYE

#ROUTING-PROTOCOL    ZRP
#ZONE-RADIUS         2

#ROUTING-PROTOCOL    STATIC
#STATIC-ROUTE-FILE   ROUTES.IN
```

ภาพที่ 10 ตัวอย่างการเพิ่มโพรโทคอล DSR-BA ในแฟ้มข้อมูล config.in

4. การทำงานของโปรแกรมการจำลองเครือข่าย GloMoSim



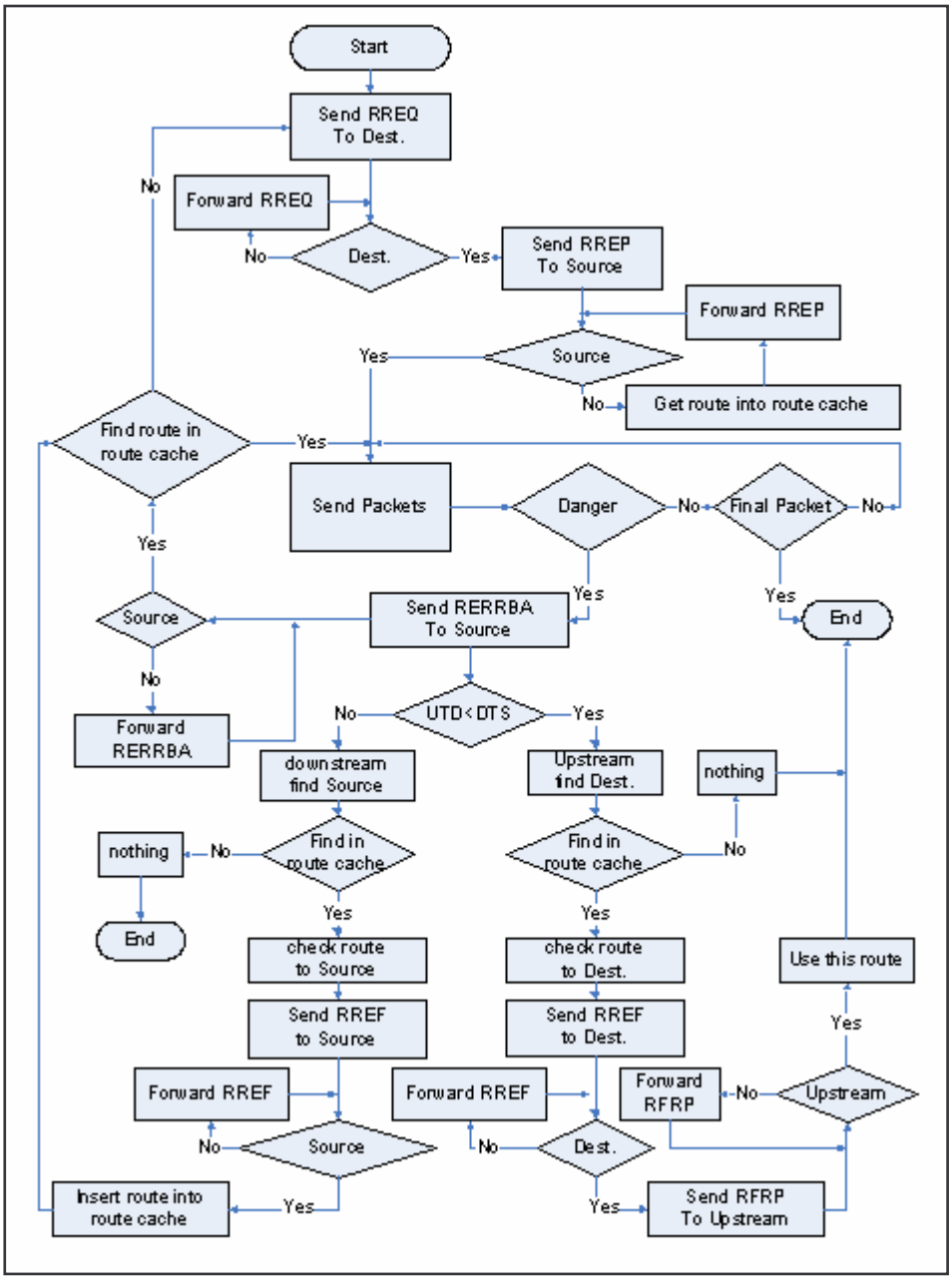
ภาพที่ 11 ภาพรวมการทำงานของโปรแกรมการจำลองเครือข่าย GloMoSim

ภาพที่ 11 แสดงภาพรวมการทำงานของโปรแกรมการจำลองเครือข่าย GloMoSim โดยโปรแกรมเริ่มต้นการทำงานด้วยคำสั่ง ./glomosim config.in โดยทำการอ่านค่า configuration ที่กำหนดไว้จากแฟ้มข้อมูล config.in จากนั้นจึงนำข้อมูลที่ได้มาใช้ในแฟ้มข้อมูล driver.pc ซึ่งจะทำการเตรียมเครือข่ายเพื่อให้พร้อมใช้สำหรับการเริ่มการทำงานจำลองเครือข่าย เมื่อได้รับข้อความ "StartSim message" โปรแกรมเริ่มต้นทำการจำลองเครือข่าย โดยเริ่มทำการแบ่งเครือข่ายเป็นส่วนๆ ซึ่งนำไปใช้ในแฟ้มข้อมูล glomo.pc จากนั้นจึงนำข้อมูลที่ได้มาทั้งหมดทำการกำหนดค่าเริ่มของทุกชั้นการสื่อสารได้แก่ GLOMO_GlobalPropInit, GLOMO_NetworkPreInit, GLOMO_PropInit, GLOMO_RadioInit, GLOMO_MacInit, GLOMO_NetworkInit, GLOMO_TransportInit, GLOMO_AppInit และ GLOMO_MobilityInit เป็นต้น จากนั้นโปรแกรมจะรอรับข้อความต่างๆ เพื่อทำการจำลองเครือข่ายตามคำสั่งที่รับข้อความเข้ามา และทำงานตามชั้นการสื่อสารต่างๆ ตามข้อมูลที่แต่ละโหนดได้รับ จนกว่าการจำลองการทำงานของเครือข่ายได้สิ้นสุดลง เมื่อสิ้นสุดการจำลองการทำงานแล้วจึงจัดทำผลการจำลองเครือข่ายทั้งหมดในทุกชั้นการสื่อสารได้แก่ GLOMO_RadioFinalize, GLOMO_MacFinalize, GLOMO_NetworkFinalize, GLOMO_TransportFinalize, GLOMO_AppFinalize และ

GLOMO_MobilityFinalize เป็นต้น และผลที่ได้นั้นนำไปจัดเก็บไว้ในแฟ้มข้อมูล glomo.stat เป็นการสิ้นสุดการจำลองการทำงานเครือข่าย

5. การเขียนโปรแกรมเพิ่มลงในโปรแกรมการจำลองเครือข่าย GloMoSim

5.1 ขั้นตอนการทำงานของโพรโทคอล DSR-BA



ภาพที่ 12 ขั้นตอนการทำงานของโพรโทคอล DSR-BA

ภาพที่ 12 แสดงขั้นตอนการทำงานของโพรโทคอล DSR-BA โดยเริ่มจากการที่โหนดเริ่มต้นส่ง RREQ ไปยังโหนดปลายทาง ถ้าไม่ใช่โหนดปลายทางจะทำการส่งต่อ RREQ ถ้าใช้โหนดปลายทาง โหนดจะตอบกลับโหนดเริ่มต้นด้วย RREP โหนดที่ได้รับ RREP ถ้าไม่ใช่โหนดเริ่มต้นจะส่งต่อ RREP ถ้าใช่โหนดเริ่มต้นจะนำข้อมูลเส้นทางได้นี้มาใช้ในการส่งแพคเกตข้อมูล ถ้าโหนดที่ทำการส่งแพคเกตข้อมูลไม่ตรวจพบเส้นทางอาจจะขาดการเชื่อมต่อ โหนดยังคงใช้เส้นทางเดิมในการส่งแพคเกตข้อมูลจนสิ้นสุดการส่งแพคเกตสุดท้าย แต่ถ้าโหนดตรวจพบเส้นทางอาจจะขาดการเชื่อมต่อ โหนดทำการแจ้งกลับไปยังโหนดเริ่มต้นด้วย RERRBA และทำการส่งต่อจนกระทั่งถึงโหนดเริ่มต้น และให้โหนดเริ่มต้นจัดการกับเส้นทางที่ใช้อยู่เดิม หลังจากที่โหนดส่ง RERRBA ออกไป โหนดจะทำการเลือกว่าโหนด upstream หรือโหนด downstream ทำหน้าที่ดูแลรักษาเส้นทาง ถ้าเป็นโหนด downstream โหนดทำการค้นหาเส้นทางจากแคชเส้นทางไปยังโหนดเริ่มต้น ถ้าไม่มีเส้นทางโหนดจะไม่ทำอะไร ถ้ามีเส้นทางโหนดส่ง RREF ไปยังโหนดเริ่มต้นด้วยเส้นทางใหม่นี้ จนกระทั่ง RREF ส่งไปถึงโหนดเริ่มต้น เส้นทางที่ตรวจสอบแล้วนี้ จะถูกเก็บลงในแคชเส้นทางเพื่อให้โหนดเริ่มต้นได้เลือกใช้ในโอกาสต่อไป แต่ถ้าโหนด upstream ทำหน้าที่ดูแลรักษาเส้นทาง โหนด upstream ทำการค้นหาเส้นทางจากแคชเส้นทางไปยังโหนดปลายทาง ถ้าไม่มีเส้นทางโหนดจะไม่ทำอะไร ถ้ามีเส้นทางโหนดส่ง RREF ไปยังโหนดปลายทางด้วยเส้นทางใหม่นี้ จนกระทั่ง RREF ส่งไปถึงโหนดปลายทาง แล้วจะตอบกลับด้วย RFRP ไปยังโหนด upstream ที่ส่ง RREF เมื่อโหนด upstream ได้รับ RFRP นี้ โหนดจะเปลี่ยนมาใช้เส้นทางนี้ทันที และใช้จนกว่าโหนดเริ่มต้นจะเปลี่ยนเส้นทางไปใช้เส้นทางใหม่

5.2 ขั้นตอนการเพิ่มฟังก์ชันการตรวจสอบ

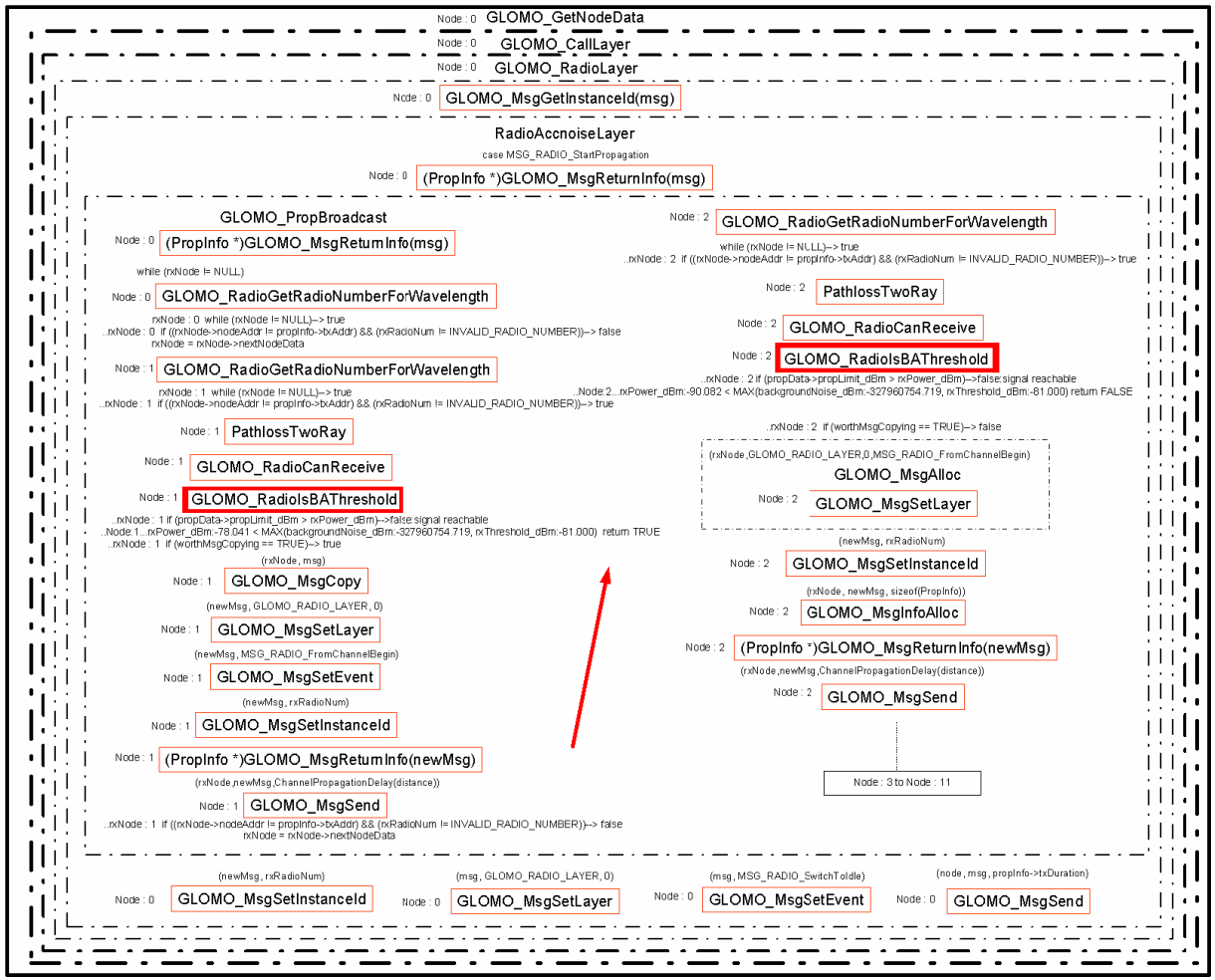
เพิ่มตัวแปรตรวจสอบเส้นทางด้วย RADIO_DEFAULT_BA_THRESHOLD, baThreshold_mW และ baThreshold_dBm ลงในแฟ้มข้อมูล radio.h เพิ่มฟังชันในการเรียกใช้ตัวแปรการตรวจสอบในแฟ้มข้อมูล radio.pc เพิ่มฟังชันในการเรียกใช้ตัวแปรการตรวจสอบในแฟ้มข้อมูล nwip.pc และ nwip.h เพิ่มฟังชันในการเรียกใช้ตัวแปรการตรวจสอบในแฟ้มข้อมูล dsrba.pc และ dsrba.h

```
/*
 * This is an aggregated delay at the radio PHY.
 * The value should be reduced as we define more detailed delay times.
 */

#define RADIO_RX_TX_TURNAROUND_TIME  (5 * MICRO_SECOND)
#define RADIO_PHY_DELAY RADIO_RX_TX_TURNAROUND_TIME

#define RADIO_DEFAULT_TX_POWER 15.0
#define RADIO_DEFAULT_ANTENNA_GAIN 0.0
#define RADIO_DEFAULT_ANTENNA_HEIGHT 1.5
#define RADIO_DEFAULT_RX_SENSITIVITY -91.0
#define RADIO_DEFAULT_RX_THRESHOLD   -81.0
#define RADIO_DEFAULT_BA_THRESHOLD   -80.0

/*
 * Different radio types supported.
 */
typedef enum {
    RADIO_ACCNOISE,
    RADIO_FOR_SIRCIM,
    RADIO_NONOISE
} RADIO_TYPE;
```

ภาพที่ 13 ตัวอย่างการเพิ่มตัวแปรตรวจสอบเส้นทาง RADIO_DEFAULT_BA_THRESHOLD ลงในแฟ้มข้อมูล radio.h

```
/*
 * Structure for radio layer.
 * Can be used for radio with and without capture ability.
 *
 * typedef to GlomoRadio in main.h
 */
struct glomo_radio_str {
    RADIO_TYPE radioType;
    int        radioNumber;
    int        macInterfaceIndex;
    double     txPower_mW;
    double     txPower_dBm;
    double     rxSensitivity_mW;
    double     rxSensitivity_dBm;
    double     rxThreshold_mW;
    double     rxThreshold_dBm;
    double     baThreshold_mW;
    double     baThreshold_dBm;
    float      antennaGain_dB;
    float      antennaHeight;
    int        bandwidth;
    double     backgroundNoise_mW;   // Pre-calculated to include bandwidth.
    double     backgroundNoise_dBm;  //
    float      wavelength;
    BOOL       radioStats;
    BOOL       guiOption;

    /*
     * The following varialbe will be interpreted differently
     * by different radio layers.
     */
    void *radioVar;
};
```

ภาพที่ 14 ตัวอย่างการเพิ่มตัวแปรตรวจสอบเส้นทาง baThreshold_mW และ baThreshold_dBm ลงในแฟ้มข้อมูล radio.h



ภาพที่ 15 โครงสร้างชั้น Radio ของโปรแกรมการจำลองเครือข่าย GloMoSim และตำแหน่งการ เพิ่มฟังก์ชันที่เรียกใช้ตัวแปรตรวจสอบเส้นทาง

6. <u>ตัวอย่างวิธีการพิมพ์เส้นทางที่โหนดใช้ในการติดต่อสื่อสารและแคชเส้นทางของโปรแกรมการจำลองเครือข่าย GloMoSim</u>

6.1 ตัวอย่างฟังก์ชันการพิมพ์เส้นทางที่โหนดใช้ในการติดต่อสื่อสาร

```
/*
 * PrintPath
 * Print : Source Route
 */

void PrintPath (GlomoNode *node, NODE_ADDR *path,int NumAddresses)
{
   int i;

   printf("\n..NodeSYD:%d NumAddresses:%d ..start.....PrintPath...\n",
               node->nodeAddr,NumAddresses);

   printf("..NodeSYD:%d ",node->nodeAddr);

   for (i = 0; i< NumAddresses; i++)
   {
       printf(" path[i:%d]:%d",i,path[i]);
   }

   printf("\n..Node:%d...end....PrintPath...\n",node->nodeAddr);

}//PrintPath
```

<u>ภาพที่ 16</u> ตัวอย่างฟังก์ชันการพิมพ์เส้นทางที่โหนดใช้ในการติดต่อสื่อสารเพิ่มเติมลงในโปรแกรมการจำลองเครือข่าย GloMoSim

6.2 ผลการทำงานของฟังก์ชัน PrintPath()

```
..NodeSYD:16 destAddr:21 ...start..end..RoutingDsrbaTransmitData..--------

..NodeSYD:16 NumAddresses:1 ..start.....PrintPath...
..NodeSYD:16  path[i:0]:21
..Node:16...end....PrintPath...

..NodeSYD:3 destAddr:34 ...start..end..RoutingDsrbaTransmitData..--------

..NodeSYD:3 NumAddresses:7 ..start.....PrintPath...
..NodeSYD:3  path[i:0]:13 path[i:1]:45 path[i:2]:8 path[i:3]:2 path[i:4]:39 path[i:5]:11 path[i:6]:34
..Node:3...end....PrintPath...

..NodeSYD:38 destAddr:32 ...start..end..RoutingDsrbaTransmitData..--------

..NodeSYD:38 NumAddresses:2 ..start.....PrintPath...
..NodeSYD:38  path[i:0]:22 path[i:1]:32
..Node:38...end....PrintPath...
```

<u>ภาพที่ 17</u> ตัวอย่างผลการพิมพ์เส้นทางที่โหนดใช้ในการติดต่อสื่อสารด้วยฟังก์ชัน PrintPath() ที่เพิ่มเติมลงในโปรแกรมการจำลองเครือข่าย GloMoSim

จากภาพที่ 17 การพิมพ์เส้นทางที่โหนดใช้ในการติดต่อสื่อสารกัน โดยมีการส่งแพคเกตข้อมูลเมื่อโหนดเริ่มต้นคือโหนด 16 และโหนดปลายทางคือโหนด 21 ทำการส่งแพคเกตข้อมูลด้วยเส้นทาง (D: S, …, D) ในที่นี้คือ (21: 16, 21) ด้วยจำนวน 1 ฮอพ การส่งแพคเกตข้อมูลเมื่อโหนดเริ่มต้นคือโหนด 3 และโหนดปลายทางคือโหนด 34 ทำการส่งแพคเกตข้อมูลด้วยเส้นทาง (D: S, …, D) ในที่นี้คือ (34: 3, 13, 45, 8, 2, 39, 11, 34) ด้วยจำนวน 7 ฮอพ และการส่งแพคเกตข้อมูลเมื่อโหนดเริ่มต้นคือโหนด 38 และโหนดปลายทางคือโหนด 32 ทำการส่งแพคเกตข้อมูลด้วยเส้นทาง (D: S, …, D) ในที่นี้คือ (32: 38, 22, 32) ด้วยจำนวน 2 ฮอพ

6.3 ตัวอย่างฟังก์ชันการพิมพ์แคชเส้นทาง

```
/*
 * PrintRC
 * Print : Route Cache
 */

void PrintRC(GlomoNode *node, DSRBA_RouteCache *routeCache)
{
   DSRBA_RouteCacheEntry *current;
   int i;

   printf("..\nNode:%d ..start.....PrintRC...\n",node->nodeAddr);

    for (current = routeCache->head;
         current != NULL ;
         current = current->next)
    {
        printf("\n");
        printf("..Node:%d current->destAddr:%d \n",node->nodeAddr,current->destAddr);
        for (i = 0; i< current->hopCount; i++)
        {
         printf("->[%d]:%d",i,current->path[i]);
        }
         printf("\n  .....weakState:%d   notUse:%d  routeAvailable:%d",
                     current->weakState,current->notUse,current->routeAvailable);
    }

   printf("\n..Node:%d...end....PrintRC...\n",node->nodeAddr);

}//PrintRC
```

ภาพที่ 18 ตัวอย่างฟังก์ชันการพิมพ์แคชเส้นทางที่เพิ่มเติมลงในโปรแกรมการจำลองเครือข่าย GloMoSim

6.4 ผลการทำงานของฟังก์ชัน PrintRC()

```
Node:14 ..start.....PrintRC...

..Node:14 current->destAddr:0
->[0]:35->[1]:9->[2]:5->[3]:0
  .....weakState:0   notUse:0  routeAvailable:0
..Node:14 current->destAddr:0
->[0]:48->[1]:25->[2]:16->[3]:0
  .....weakState:0   notUse:0  routeAvailable:0
  ..Node:14 current->destAddr:0
->[0]:40->[1]:35->[2]:9->[3]:5->[4]:0
  .....weakState:0   notUse:0  routeAvailable:0
..Node:14 current->destAddr:1
->[0]:8->[1]:1
  .....weakState:0   notUse:0  routeAvailable:0
..Node:14 current->destAddr:1
->[0]:48->[1]:1
  .....weakState:0   notUse:0  routeAvailable:0
..Node:14 current->destAddr:2
->[0]:8->[1]:2
  .....weakState:0   notUse:0  routeAvailable:0
..Node:14 current->destAddr:3
->[0]:3
..Node:14 current->destAddr:47
->[0]:38->[1]:3->[2]:47
->[0]:49
  .....weakState:0   notUse:0  routeAvailable:0
..Node:14 current->destAddr:49
->[0]:22->[1]:49
  .....weakState:0   notUse:0  routeAvailable:0
..Node:14...end....PrintRC...
```

ภาพที่ 19 ตัวอย่างผลการพิมพ์แคชเส้นทางด้วยฟังก์ชัน PrintRC() ที่เพิ่มเติมลงในโปรแกรมการจำลองเครือข่าย GloMoSim

จากภาพที่ 19 การพิมพ์แคชเส้นทางของโหนด 14 โดยมีเส้นทางไปยังโหนดปลายทางในแคชเส้นทางดังนี้คือ เส้นทางการส่งแพคเกตข้อมูลไปยังโหนดปลายทางคือโหนด 0 ด้วยเส้นทาง (D: S, …, D) ในที่นี้คือ (0: 14, 35, 9, 5, 0) สถานะของเส้นทางทั้ง weakState notUse and routeAvailable ยังไม่มีการเปลี่ยนแปลงใดๆ และมีอีกสองเส้นทางการส่งแพคเกตข้อมูลไปยังโหนดปลายทางคือโหนด 0 ด้วยเส้นทาง (0: 14, 48, 25, 16, 0) และ (0: 14, 40, 35, 9, 5, 0) โดยที่มีสถานะของเส้นทางทั้ง weakState, notUse and routeAvailable ยังไม่มีการเปลี่ยนแปลงใดๆ นอกจากนี้โหนด 14 มีเส้นทางไปยังโหนดปลายทางอื่นๆ อีกในที่นี้ได้แก่ โหนด 1 2 3 47 และ 49 เป็นต้น

6.5 ตัวอย่างการใช้ฟังก์ชัน PrintPath() และฟังก์ชัน PrintRC()

```
else /* data packet */
{
    NODE_ADDR path[DSRBA_MAX_SR_LEN + 1];
    int length;
    int current;

    printf("..Node:%d nextHop:%d src:%d dest:%d datapacket ...DropNoti...\n",
                node->nodeAddr,nextHopAddress,ipHeader->ip_src,ipHeader->ip_dst);

    ExtractIpSourceAndRecordedRoute(newMsg, path, &length, &current);
    assert(length <= (DSRBA_MAX_SR_LEN+1));

    option = GetPtrToDsrbaIpOptionField(newMsg);
    option->segmentLeft = option->segmentLeft - 1;

    PrintPath(node, path,  length);

    unreachableAddr = path[current - 1];

    /* Delete the routes that use the broken link */
    RoutingDsrbaDeleteRouteCache(node, node->nodeAddr,
                        unreachableAddr, &dsrba->routeCacheTable);

    PrintRC(node, &dsrba->routeCacheTable);

    targetAddr = ipHeader->ip_dst;

    if (option->salvagedBit == TRUE &&
        node->nodeAddr != path[0])
```

ภาพที่ 20 ตัวอย่างการนำฟังก์ชัน PrintPath() และฟังก์ชัน PrintRC() ไปใช้งานในโปรแกรมการจำลองเครือข่าย GloMoSim

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	นางสาวศิญดา อ่ำเทศ
วัน เดือน ปี ที่เกิด	วันที่ 6 เมษายน 2522
สถานที่เกิด	พะเยา
ประวัติการศึกษา	วศ.บ. (วิศวกรรมไฟฟ้า) สถาบันเทคโนโลยีราชมงคล (พ.ศ. 2544)
ผลงานทางวิชาการ	ตีพิมพ์ผลงาน เรื่อง “Route Break Avoidance in DSR” ในงานประชุมวิชาการ Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON 2006) ที่จังหวัดอุบลราชธานี ประเทศไทย ระหว่างวันที่ 10-13 พฤษภาคม พ.ศ. 2549