การปรับปรุงการแพร่ข้อมูลแบบเชื่อถือได้บนเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะที่เชื่อมต่อแบบอสมมาตรด้วยพื้นฐานของอัลกอริทึมการโหวตอาร์เอสเอสไอ

นายณัฐวิทย์ กมลธรรม

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

IMPROVING RELIABLE BROADCAST OVER ASYMMETRIC VANETS BASED ON A RSSI-VOTING ALGORITHM

Mr.Nattavit Kamoltham

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Computer Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2011

หัวข้อวิทยานิพนธ์	การปรับปรุงการแพร่ข้อมูลแบบเชื่อถือได้บนเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะที่เชื่อมต่อแบบอสมมาตรด้วยพื้นฐานของอัลกอริทึมการโหวตอาร์เอสเอสไอ
โดย	นายณัฐวิทย์ กมลธรรม
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร.กุลธิดา โรจน์วิบูลย์ชัย

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

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

คณะกรรมการสอบวิทยานิพนธ์

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

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร.กุลธิดา โรจน์วิบูลย์ชัย)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.เฉลิมเอก อินทนากรวิวัฒน์)

..... กรรมการภายนอกมหาวิทยาลัย
(ดร.ภาสกร ประถมบุตร)

ณัฐวิทย์ กมลธรรม : การปรับปรุงการแพร่ข้อมูลแบบเชื่อถือได้บนเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะที่เชื่อมต่อแบบอสมมาตรด้วยพื้นฐานของอัลกอริทึมการโหวตอาร์เอสเอสไอ (IMPROVING RELIABLE BROADCAST OVER ASYMMETRIC VANETS BASED ON A RSSI-VOTING ALGORITHM) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร.กุลธิดา โรจน์วิบูลย์ชัย, 78 หน้า.

การพัฒนาเครือข่ายไร้สายแบบแอดฮอกสำหรับรถยนต์นั้นมักทำบนโปรแกรมจำลองเครือข่ายซึ่งยังไม่ครอบคลุมปัจจัยต่างๆ ที่มีอยู่บนเครือข่ายจริง ทำให้การพัฒนานั้นต้องทำการพัฒนาในส่วนการทดลองบนเครือข่ายจำลองและพัฒนาในส่วนของการทดลองบนเครือข่ายจริงแยกจากกันทำให้เสียเวลาอย่างน้อยสองครั้ง โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 จึงเป็นอีกทางเลือกในการทำวิจัยเพราะว่าสามารถทำงานได้ทั้งการจำลองเครือข่ายและการทำงานบนระบบจริงด้วยการพัฒนาเพียงครั้งเดียว อีกทั้งโครงสร้างการทำงานของโปรแกรมที่สอดคล้องกับหลักการทำงานของโครงสร้างของเครือข่ายทำให้สามารถนำความรู้ความเข้าใจมาพัฒนาได้โดยตรง

วิทยานิพนธ์นี้ได้ทำการทดลองบนเครือข่ายจริงด้วยโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 และพบปัญหาการเชื่อมต่อแบบอสมมาตรที่บางโหนดส่งข้อมูลให้กับผู้อื่นไม่ได้ ทำให้การทำงานของโพรโทคอลที่มีการกระจายแบบเชื่อถือได้มีประสิทธิภาพที่ต่ำลง ซึ่งจากการตรวจสอบพบว่าปัญหาเกิดจากการที่ผู้ส่งข้อมูลไม่สามารถส่งข้อมูลไปยังผู้รับได้ และการทำงานในการเลือกผู้กระจายข้อมูลลำดับถัดไปของโพรโทคอลผิดพลาดเพราะโหนดที่ถูกเลือกอาจมีความแรงของสัญญาณต่ำทำให้ไม่สามารถกระจายข้อมูลไปให้กับเพื่อนบ้านโดยรอบได้ แต่ในสถานการณ์ที่มีการเชื่อมต่อแบบอสมมาตรนั้นเป็นไปได้ยากที่จะทำให้การส่งข้อมูลระหว่างโหนดสำเร็จ 100 % จึงเน้นแก้ไขในกระบวนการเลือกผู้กระจายข้อมูลลำดับถัดไปของโพรโทคอล ซึ่งในวิทยานิพนธ์นี้เลือกโพรโทคอล DECA ที่มีการทำงานอย่างง่าย รวดเร็ว และลดความซ้ำซ้อนของข้อมูลเป็นตัวอย่างในการศึกษา โดยเสนออัลกอริทึมการโหวตอาร์เอสเอสไอเป็นกระบวนการในการแก้ไขซึ่งโหนดทุกโหนดจะโหวตว่าเพื่อนบ้านโหนดไหนที่มีค่าความแรงของสัญญาณในการส่ง หรือค่าอาร์เอสเอสไอมากที่สุด แล้วแนบไปกับ Beacon message เพื่อแลกเปลี่ยนข้อมูลกับเพื่อนบ้าน เมื่อผู้ส่งต้องการกระจายข้อมูลก็จะเลือกโหนดที่ได้รับการโหวตมากที่สุดเป็นผู้กระจายข้อมูลลำดับถัดไป โดยการแก้ไขนี้สามารถเพิ่มค่าความน่าเชื่อถือของโพรโทคอลได้สูงสุดถึง 17 % และลดค่าใช้จ่ายได้สูงสุดถึง 28 %

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

สาขาวิชา .....วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อ อ. ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา .....2554.....

5470200421 : MAJOR COMPUTER ENGINEERING

KEYWORDS : VEHICULAR NETWORKS/ BROADCAST/ AD HOC NETWORKS/ NETWORK SIMULATOR 2/ NETWORK SIMULATOR 3/ ASYMMETRIC LINK

NATTAVIT KAMOLTHAM : IMPROVING RELIABLE BROADCAST OVER ASYMMETRIC VANETS BASED ON A RSSI-VOTING ALGORITHM.

ADVISOR : ASST. PROF. KULTIDA ROJVIBOONCHAI, Ph.D., 78 pp.

Many researches of vehicular ad-hoc networks (VANETs) are usually implement on simulator programs but that implementation cannot be reused or tested on a real system. So researchers need to do more works or make new implementation to evaluate their works on the real system. Network Simulator 3 (NS-3) is the new simulator that has been developed with whole new idea. NS-3 supports on both simulation and emulation (real implementation). This can reduce time of implementation because researchers can implement their works only one time then they can use the same source codes to evaluate their works on both simulation and emulation.

This thesis implemented VANET on emulation of NS-3 and then found an asymmetric link problem that reduces the performance of reliable broadcasting protocols. After investigation, we found that the problem is caused by a sender who cannot send data to a receiver and a protocol node selection which cannot select an efficient forwarder node within a broadly transmission range and vastly neighbor coverage area. In this thesis, it is impossible to achieve a 100 % of delivery ratio in an asymmetric scenario. Therefore, we focus on the selection of a forwarder node with a long transmission range to cover the maximum number of neighbors. We choose a DECA which is simple, fast and less redundant data, as a case study. Then we propose a RSSI-Voting Algorithm (RVA) as a solution of this problem. The mechanism of RVA is that all neighbors will vote for a neighbor with the highest RSSI level and then attach to beacon messages. Every node exchanges the beacon message and updates information. If sender wants to broadcast a data, it will select a neighbor that gains the highest voting from neighbors as the forwarder node. The simulation results show that our mechanism can improve protocol performance up to 17 % and decrease its retransmission overhead up to 28 %.

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

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

Academic Year: .....2011.....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ได้รับคำแนะนำ ข้อคิดเห็น และแนวทางในการทำวิจัยที่ดีต่างๆ จากอาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร.กุลธิดา โรจน์วิบูลย์ชัย ที่คอยช่วยเหลือให้ความรู้ทางด้านที่จำเป็นต่อวิทยานิพนธ์ และคำปรึกษาที่สามารถแก้ไขปัญหาต่างๆ ที่เกิดขึ้นได้ นอกจากนั้นยังถ่ายทอดประสบการณ์ต่างๆ ให้เป็นข้อคิด และแนวทางที่ดีต่อการทำงานต่อไป ผู้วิจัยจึงขอกราบขอบพระคุณเป็นอย่างสูงมา ณ ที่นี้

กราบขอบพระคุณคณะกรรมการสอบวิทยานิพนธ์ ผู้ช่วยศาสตราจารย์ ดร.เกริก ภิรมย์โสภา ผู้ช่วยศาสตราจารย์ ดร.เฉลิมเอก อินทนากรวิวัฒน์ และ ดร.ภาสกร ประถมบุตร ที่สละเวลามาให้ข้อเสนอแนะ และแนวคิดต่างๆ ที่เป็นประโยชน์ต่อการพัฒนาวิทยานิพนธ์ฉบับนี้

ขอขอบคุณจุฬาลงกรณ์มหาวิทยาลัยที่ให้ความรู้ และประสบการณ์ที่ดีต่างๆ ที่เป็นส่วนหนึ่งในการใช้ชีวิตในมหาวิทยาลัย รวมทั้งขอขอบคุณอาจารย์ทุกๆท่านที่ได้อบรม สั่งสอนให้ผู้วิจัยได้รับความรู้ในวิชาต่างๆ ที่เป็นประโยชน์ในการทำงาน อีกทั้งเพื่อนๆ พี่ๆ น้องๆ ที่ดี ที่คอยช่วยเหลือและเป็นกำลังใจให้ในช่วงเวลาต่างๆ

และสุดท้าย ขอขอบคุณทุกคนในครอบครัวทั้ง คุณพ่อ คุณแม่ พี่ชาย พี่สาว และญาติๆ ที่คอยสนับสนุนให้ผู้ทำวิจัยได้เรียนรู้ และทำในสิ่งที่ดี คอยเป็นกำลังใจให้กันและกันตลอดมา

สารบัญ

หน้า

หน้า

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

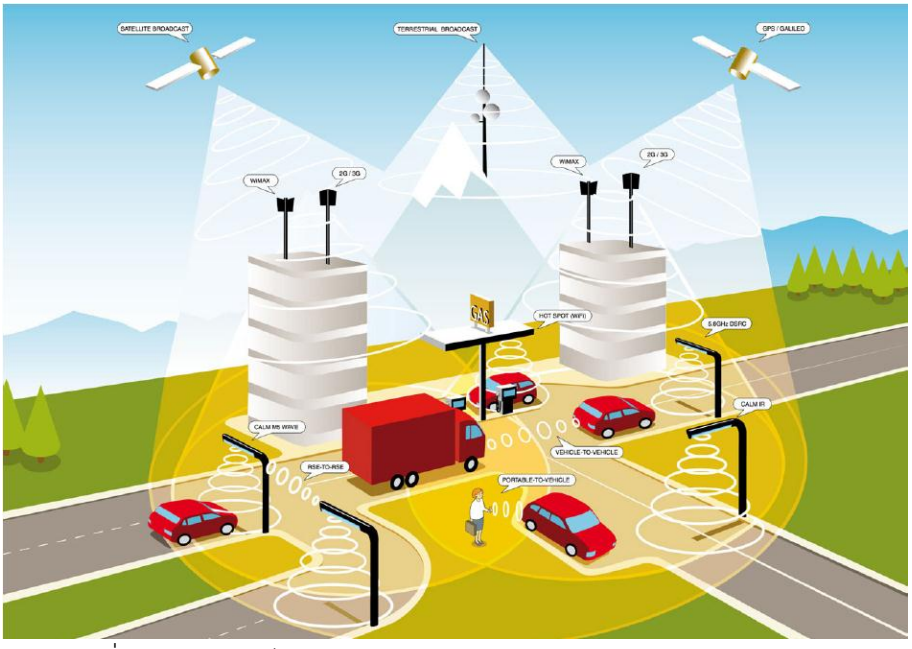
หน้า

บทที่ 1

บทนำ

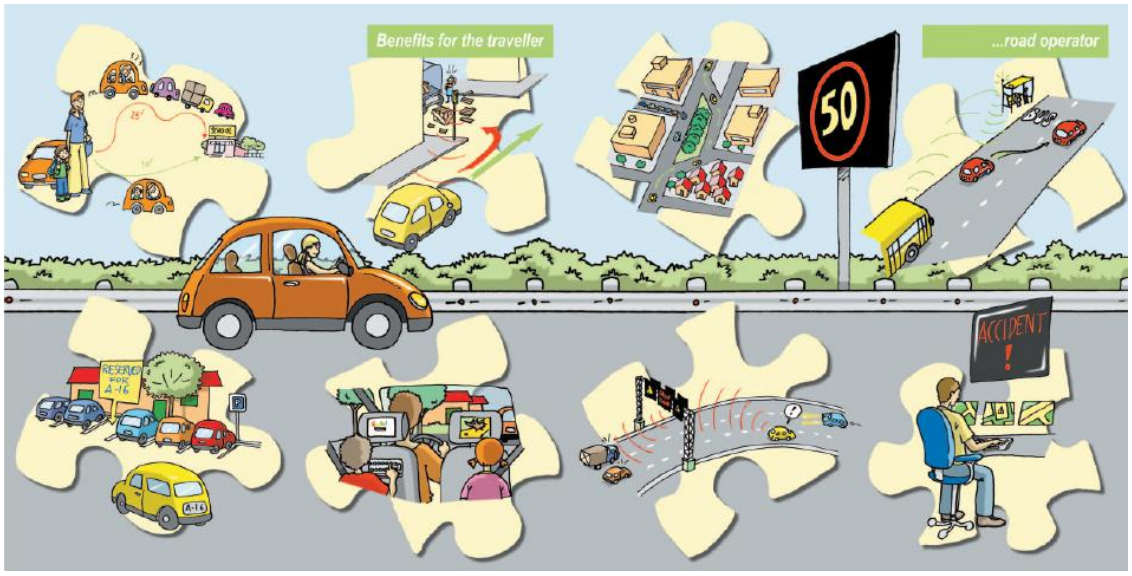
1.1 ที่มาและความสำคัญของปัญหา

เครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะ (VANET : Vehicular Ad-hoc Networks) ดังภาพที่ 1.1 เป็นเทคโนโลยีที่ได้รับความสนใจอย่างแพร่หลายในปัจจุบัน [1] โดยถูกออกแบบเพื่อใช้งานกับระบบจราจรอัจฉริยะ (ITS : Intelligent Transportation System) [2] ดังภาพที่ 1.2 ที่ช่วยเพิ่มความปลอดภัยในการจราจร อำนวยความสะดวกในการขับขี่บนท้องถนน และเพื่อความบันเทิงต่างๆ ที่เป็นส่วนเพิ่มเติมในชีวิตประจำวันของหลายๆ คนในตอนนี้ ซึ่งเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะนี้เป็นส่วนหนึ่งของการสื่อสารบนเครือข่ายไร้สายแบบแอดฮอก (MANET : Mobile Ad-hoc Networks) โดยทั้งเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะ และเครือข่ายไร้สายแบบแอดฮอกนั้นทำงานได้โดยไม่ต้องมีโครงสร้างพื้นฐาน แต่ใช้การติดต่อสื่อสารระหว่างอุปกรณ์ไร้สายหรืออุปกรณ์ที่ฝังอยู่บนยานพาหนะให้ส่งข้อมูลหากันได้เอง การทำงานแบบไม่ต้องพึ่งโครงสร้างพื้นฐานจำพวกเสาส่งสัญญาณ หรืออุปกรณ์ที่เป็นตัวกลางในการเชื่อมต่อเครือข่ายระหว่างอุปกรณ์ไร้สายนี้ทำให้ข้อมูลในการติดต่อสื่อสารนั้นเป็นปัจจุบัน และใช้งานได้ครอบคลุมพื้นที่ที่ต้องการใช้งาน อีกทั้งยังเป็นการลดต้นทุนของระบบในการทำงานและยังได้ประสิทธิภาพการใช้งานของระบบที่ดีกว่า



ภาพที่ 1.1 เครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะ [20]

แต่เครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะนั้นมีความแตกต่างจากเครือข่ายไร้สายแบบแอดฮอก ตรงที่การเคลื่อนที่ของยานพาหนะนั้นมักเป็นไปในทิศทางที่คาดเดาได้ และเปลี่ยนแปลงไว อย่างเช่นการเคลื่อนที่ของยานพาหนะบนถนนที่วิ่งเป็นทางยาว หรือเป็นทางแยก ซึ่งยานพาหนะมักเคลื่อนที่ไปทางเดียวกัน หรือสวนทางกัน นอกจากนั้นยังมีความแตกต่างในส่วนความสำคัญของข้อมูลที่ส่ง ในบางสถานการณ์การส่งข้อมูลของยานพาหนะจำเป็นที่ต้องส่ง ณ ขณะนั้น (Real-time) ตัวอย่างเช่น ระบบหลีกเลี่ยงอุบัติเหตุ หรือบริการแจ้งเตือนเหตุฉุกเฉิน เป็นต้น และข้อมูลต้องถูกส่งไปเป็นวงกว้างให้กับยานพาหนะในละแวกอย่างทั่วถึงและทำงานได้กับรูปแบบตำแหน่งที่เปลี่ยนแปลงตลอดเวลา อีกทั้งบริการเหล่านี้ยังต้องการการแพร่ข้อมูลที่มีความเชื่อถือได้ (Reliable Broadcast) เป็นพื้นฐานในการทำงาน เพื่อความปลอดภัยในการใช้งานด้วย อย่างไรก็ตามการทำงานของระบบจราจรอัจฉริยะนั้นยังอยู่ในขั้นพัฒนา ทำให้เทคโนโลยีนี้ยังสามารถที่จะเติบโตต่อไปในหลายทิศทาง และมีหลายหัวข้อที่เป็นที่น่าสนใจในการทำวิจัยทั้งในประเทศ และต่างประเทศ



ภาพที่ 1.2 ตัวอย่างบริการในระบบจราจรอัจฉริยะ [20]

การพัฒนาเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะในหลายๆ งานวิจัยล้วนแล้วแต่พัฒนาและทดลองด้วยการจำลองเครือข่าย (Simulation) โดยใช้โปรแกรมจำลองเครือข่าย เช่น โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 (NS-2 : Network Simulator 2) [3] โปรแกรม OmNeT++ [23] โปรแกรม QualNet [24] และโปรแกรม OPNET [25] เป็นต้น โดยโปรแกรมเหล่านี้รองรับแค่การทำงานแบบการจำลองเครือข่าย ซึ่งจำลองเครือข่ายในการทดสอบการส่งข้อมูลของยานพาหนะซึ่งสามารถกำหนดปัจจัยที่เกี่ยวข้องในการพัฒนาได้ให้เป็นรูปแบบที่ต้องการ แต่การพัฒนาเพียงแค่การจำลองเครือข่ายนั้นยังได้ผลลัพธ์ที่แตกต่างและไม่เพียงพอเมื่อเปรียบเทียบกับการทดลองบนระบบจริง

(Real world experiment) ซึ่งถ้าต้องการวัดประสิทธิภาพการทำงานของงานวิจัยที่พัฒนานั้นให้ใกล้เคียงกับการใช้งานจริงมากขึ้น จึงควรเพิ่มการทดสอบบนระบบจริง

แต่การใช้งานโปรแกรมจำลองเครือข่ายที่กล่าวมานั้นไม่รองรับการทำงานในระบบจริง ซึ่งหากผู้วิจัยต้องการที่จะพัฒนางานวิจัยของตนให้สามารถทำงานได้บนระบบจริงนั้นจะต้องทำการเริ่มพัฒนางานใหม่อีกครั้งเพื่อให้สามารถทำงานบนระบบปฏิบัติการคอมพิวเตอร์ที่ใช้ (Kernel) และเชื่อมต่อส่วนประสาน (Interface) ของฮาร์ดแวร์ที่ใช้งาน ทำให้การพัฒนานั้นยากขึ้น และเสียเวลาในการทำวิจัยที่ต้องพัฒนางานของตนสองครั้งหรือมากกว่านั้น เพื่อทดสอบด้วยการจำลองเครือข่ายและระบบจริง จึงควรหาวิธีที่ทำให้ปัญหาดังกล่าวกลายเป็นเรื่องที่ง่ายขึ้น

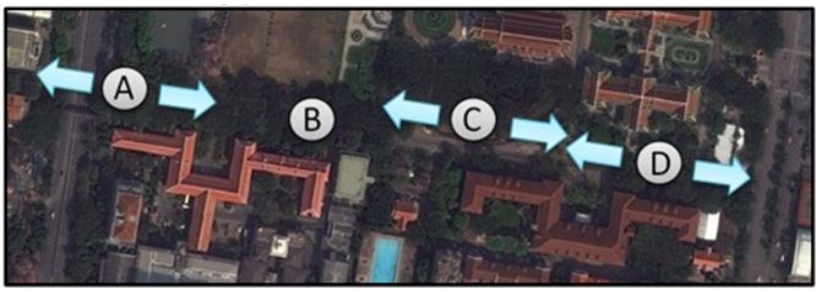
โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 (NS-3 : Network Simulator 3) [4] ซึ่งเป็นโปรแกรมจำลองเครือข่ายที่ถูกพัฒนาโครงสร้างใหม่ให้แตกต่างจากโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 เดิม จึงเป็นอีกทางเลือกในการทำวิจัย [5][6] เพราะรูปแบบโครงสร้างการทำงานของโปรแกรมที่รองรับการพัฒนาครั้งเดียวแต่สามารถทำงานได้ทั้งแบบการจำลองเครือข่ายและการทำงานบนระบบจริง ทำให้เกิดความง่ายและรวดเร็วในการพัฒนา นอกจากนั้นโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 มีรูปแบบการทำงานเป็นการเขียนโปรแกรมเชิงวัตถุ (Object-oriented) ทำให้การพัฒนาโครงสร้างของโปรแกรมใกล้เคียงกับการพัฒนาบนระบบจริงมากขึ้น ซึ่งสามารถนำความรู้ ความเข้าใจทางด้านเครือข่ายมาใช้พัฒนาได้ง่าย อีกทั้งโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 สามารถพัฒนาได้ด้วยภาษา C++ หรือ Python เพียงอย่างเดียว แตกต่างจากโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ที่ต้องใช้ภาษา Tcl Script ควบคู่กับ C++ ในการพัฒนา ทำให้โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ได้เปรียบในการใช้งานกว่าโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 แต่กระนั้นแล้วการพัฒนาด้วยภาษา C++ เพียงอย่างเดียวสำหรับโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ทำให้ผู้พัฒนาต้องมีความเข้าใจในการทำงานส่วนต่างๆ เป็นอย่างดี เพราะว่าการเขียนโปรแกรมต้องเรียกใช้ทุกฟังก์ชั่นทุกส่วนเองด้วยภาษา C++ และต้องใช้งานได้เหมือนกับการพัฒนาระบบเครือข่ายของจริงด้วยความรู้และเข้าใจที่ถูกต้อง แตกต่างจากโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ที่มีภาษา Tcl Script ในการช่วยในการเรียกใช้ฟังก์ชันของโปรแกรม และเชื่อมประสานการทำงานของโปรแกรมกับภาษา C++ ที่พัฒนาไว้

ต่อมาจึงทำการทดสอบการพัฒนาโพรโทคอลบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ด้วยโพรโทคอล DECA (Density-Aware Reliable Broadcasting in Vehicular Ad-hoc Network) [7] เพราะเป็นโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้ (Reliable Broadcasting Protocol) ในชั้นเครือข่าย (Network Layer) ซึ่งทำงานได้เร็ว ง่าย ใช้ข้อมูลน้อย และลดความซ้ำซ้อนของข้อมูล อีกทั้งได้มีการพัฒนาบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 อยู่แล้วจึงเหมาะสมกับการนำมาเปรียบเทียบผล

การทำงานของโพรโทคอลกับผลลัพธ์ที่พัฒนาจากโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ซึ่งในบทที่ 3 ของวิทยานิพนธ์นี้จะกล่าวเกี่ยวกับการเปรียบเทียบการทำงานของโพรโทคอล DECA บนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 เปรียบเทียบกับการทำงานของโพรโทคอล DECA บนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2

โดยการทำงานของ DECA นั้นมีรูปแบบเก็บและส่งออกไป (Store-and-Forward) ใช้ลักษณะการส่งข้อมูลแบบกระจายข้อมูล (Broadcast) แต่มีการเลือกผู้กระจายข้อมูลลำดับถัดไป (Forwarder node) เพื่อลดความซ้ำซ้อนและการชนกันของข้อมูล ซึ่งหลักการในการเลือกผู้ถูกเลือกคือการใช้ค่าความหนาแน่นของเพื่อนบ้าน โดยเพื่อนบ้านไหนที่มีจำนวนเพื่อนบ้านรอบตัวเองที่ติดกันมากที่สุดก็จะถูกเลือกให้เป็นผู้กระจายข้อมูลลำดับถัดไป นอกจากนั้นเพื่อนบ้านที่ไม่ได้เป็นผู้ถูกเลือกจะทำการตั้งเวลาถอยหลังแบบสุ่มไม่ให้เท่ากัน เพื่อตรวจสอบความสำเร็จในการกระจายข้อมูลของผู้ถูกเลือก ซึ่งถ้าผู้ถูกเลือกทำการกระจายข้อมูลสำเร็จ เวลาที่นับไว้ก็จะถูกยกเลิกหมด แต่หากว่าไม่มีการกระจายข้อมูลจากผู้ถูกเลือก เพื่อนบ้านที่นับเวลาหมดก่อนก็จะทำการกระจายข้อมูลนั้นแทน

(ก) รูปแบบโหนดอยู่นิ่ง



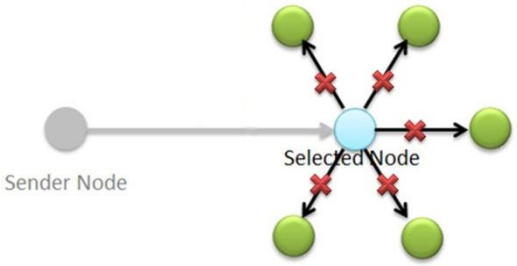
(ข) รูปแบบโหนดเคลื่อนที่

ภาพที่ 1.3 การทดสอบระบบจริงในบริเวณจุฬาลงกรณ์มหาวิทยาลัย

หลังจากได้พัฒนา DECA ลงบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 แล้วจึงเริ่มทำการทดสอบการทำงานบนระบบจริง (Emulation) โดยทำการวางคอมพิวเตอร์พกพา (Laptop) จำนวนสี่เครื่องในจุดต่างๆของจุฬาลงกรณ์มหาวิทยาลัย ดังภาพที่ 1.3 โดยในภาพที่ 1.3(ก) นั้นให้คอมพิวเตอร์พกพาแต่ละเครื่องตั้งอยู่กับที่ แล้วทำการส่งข้อมูลด้วยโพรโทคอล DECA ส่วนในภาพที่ 1.3(ข) นั้นให้

คอมพิวเตอร์พกพาสามเครื่องเคลื่อนที่ตามระยะทางที่กำหนด ส่วนอีกหนึ่งเครื่องนั้นให้ตั้งอยู่กับที่ แล้วทำการส่งข้อมูลด้วยโพรโทคอล DECA ผลลัพธ์ที่ได้พบว่าทั้งสองสถานการณ์นั้นมีการส่งข้อมูลได้ทั่วถึงครบทุกเครื่อง แต่มีสิ่งที่ผิดสังเกตคือค่าใช้จ่าย (Overhead) ของการทดลองนั้นสูงมาก ซึ่งแตกต่างจากการทดสอบด้วยการจำลองเครือข่าย ซึ่งเมื่อตรวจสอบการทำงานและผลการทดลองอย่างละเอียดพบว่าปัญหาที่เกิดขึ้นเกิดจากการเชื่อมต่อแบบอสมมาตร (Asymmetric link) ที่เกิดจากคอมพิวเตอร์แต่ละเครื่องที่ใช้ในการทดลองนั้นมีความสามารถของอุปกรณ์ในการส่งข้อมูลที่แตกต่างกัน ทำให้ความแรงของสัญญาณและระยะในการส่งข้อมูลนั้นแตกต่างกัน ส่งผลให้การส่งข้อมูลจากเครื่องหนึ่งนั้นไม่สามารถส่งไปให้กับอีกเครื่องหนึ่งได้ดังภาพที่ 1.4(ก) รวมทั้งปัญหาในการทำงานของโพรโทคอลที่เลือกผู้กระจายข้อมูลลำดับถัดไปที่มีความหนาแน่นสูงสุด (เชื่อมต่อกับเพื่อนบ้านมากที่สุด) แต่มีความแรงของสัญญาณต่ำ ทำให้ระยะในการส่งข้อมูลสั้นจึงไม่สามารถแพร่กระจายข้อมูลให้กับเพื่อนบ้านได้อย่างทั่วถึงดังแสดงในภาพที่ 1.4(ข)

(ก) ผู้ส่งข้อมูลผิดพลาด



(ข) ผู้ถูกเลือกในการกระจายข้อมูลผิดพลาด

ภาพที่ 1.4 ความผิดพลาดที่เกิดขึ้นในรูปแบบการเชื่อมต่อแบบอสมมาตร

ดังนั้นการแก้ไขการทำงานของโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้บนสถานการณ์ที่มีการเชื่อมต่อแบบอสมมาตร (Asymmetric link scenarios) จึงควรพัฒนาที่กระบวนการเลือกผู้กระจายข้อมูลลำดับถัดไป (Node selection algorithm) ของโพรโทคอล เพื่อให้ยานพาหนะหรือโหนดที่ถูกเลือกนั้นสามารถกระจายข้อมูลต่อไปให้กับเพื่อนบ้านได้อย่างทั่วถึงและมีจำนวนมากที่สุด โดยมีงานวิจัย [13] ซึ่งใช้ค่าวัดความแรงสัญญาณที่ได้รับ (RSSI : Received Signal Strength Indication) ซึ่งเป็นค่าความแรงของสัญญาณของผู้ส่งข้อมูลที่วัดได้ที่ผู้รับข้อมูลนั้นไปใช้ในการวิเคราะห์สัญญาณของผู้ส่งข้อมูล (Carrier sensing) และการรบกวนของสัญญาณระหว่างโหนดต่างๆ ในระบบ ซึ่งทำให้ทราบว่าค่าดังกล่าวนั้นมีประโยชน์ต่อการนำไปใช้ในการเลือกโหนดหรือยานพาหนะที่มีความแรงของสัญญาณมากในการในการส่งข้อมูล

ดังนั้นในวิทยานิพนธ์นี้จึงใช้ค่าวัดความแรงของสัญญาณที่ได้รับควบคู่กับกระบวนการโหวต (Voting algorithm) เกิดเป็นกระบวนการโหวตแบบอาร์เอสเอสไอ (RVA : RSSI-Voting algorithm) ซึ่งช่วยพัฒนากระบวนการเลือกผู้กระจายข้อมูลลำดับถัดไปของโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้ให้ทำงานได้อย่างมีประสิทธิภาพ เพราะข้อมูลของผู้ถูกเลือกนั้นเกิดจากการโหวตของยานพาหนะในละแวกซึ่งต่อเชื่อมกันโดยตรง ทำให้ผลที่ออกมานั้นเป็นจริงในขณะเวลานั้นและสามารถเลือกยานพาหนะที่มีความแรงของสัญญาณในการส่งข้อมูลสูง โดยการวัดผลการทำงานของโพรโทคอลนั้นใช้ตัวแปรสองอย่างคือความเชื่อถือได้ (Reliability, %) ที่วัดอัตราส่วนของจำนวนโหนดที่รับข้อมูลกับจำนวนโหนดทั้งหมด และค่าใช้จ่าย (Overhead, byte) ที่วัดค่าใช้จ่ายที่เกิดขึ้นในการส่งข้อความทั้งหมดในระบบ เพื่อเปรียบเทียบประสิทธิภาพการทำงานของโพรโทคอลในแต่ละสถานการณ์

1.2 วัตถุประสงค์ของการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อปรับปรุงกระบวนการเลือกผู้กระจายข้อมูลลำดับถัดไปของโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้เพื่อให้สามารถเลือกโหนดที่มีความแรงของสัญญาณสูง มีระยะในการส่งข้อมูลมาก และสามารถส่งข้อมูลให้กับเพื่อนบ้านได้มากที่สุด เพื่อเพิ่มประสิทธิภาพของโพรโทคอลให้มีความน่าเชื่อถือมากขึ้นและมีค่าใช้จ่ายน้อยลง

1.3 ขอบเขตของงานวิจัย

1) การพัฒนาและการทดลองทำบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3.10 (NS-3.10) โดยนำผลการทดลองมาหาค่าเฉลี่ยเพื่อแสดงผลการทดลอง และเปรียบเทียบผลการทดลองนั้นกับผลการทดลองบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2.34 (NS-2.34)
2) การจำลองพฤติกรรมของรถยนต์ทำบนโปรแกรมจำลอง SUMO โดยใช้ถนนยาวตรงแทนการทดลองบนทางหลวง และถนนตารางจัตุรัสแทนการทดลองในเมือง
3) การแพร่ข้อมูลจะกระทำต่อไปจนกว่าข้อความนั้นจะหมดอายุตามที่ผู้เริ่มส่งข้อมูลกำหนด (Source node)
4) โหนดมีการแลกเปลี่ยนข้อมูลกับเพื่อนบ้าน (Neighbor node) ในระดับการรับส่งสัญญาณของตัวเองเท่านั้น โพรโทคอลจะใช้ข้อมูลของเพื่อนบ้านในการตัดสินใจเลือกโหนดที่จะเป็นผู้ถูกเลือกต่อไป
5) การทดลองการทำงานของระบบจะมีการสูญเสียหรือการผิดพลาดของข้อมูลในช่องสัญญาณไร้สาย (Wireless channel) บนพื้นฐานของ IEEE 802.11

1.4 ขั้นตอนและวิธีดำเนินการวิจัย

1) ศึกษาการแพร่ข้อมูลที่มีความเชื่อถือได้สำหรับเครือข่ายไร้สาย และงานวิจัยที่เกี่ยวข้อง
2) ศึกษาการทำงานของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2.34 และเวอร์ชั่น 3.10
3) ทดสอบการทำงานของโพรโทคอล DECA บนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3.10 ทั้งระบบจำลองเครือข่าย (Simulation) และระบบจริง (Emulation)
4) ศึกษาและออกแบบความแตกต่างในการพัฒนาโพรโทคอล DECA บนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2.34 และเวอร์ชั่น 3.10
5) ออกแบบวิธีการแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตรบนโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้
6) พัฒนากระบวนการในการแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตรบนโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้
7) ทดสอบและเก็บข้อมูลการทำงานของกระบวนการในการแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตรบนโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้
8) วิเคราะห์ผลการทดลองและแก้ไขเพื่อพัฒนาให้กระบวนการที่แก้ไขนั้นทำงานได้ดีกับโพรโทคอล
9) ประเมินผลและสรุปผลจากการศึกษาความเป็นไปได้ในการแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตร
10) สรุปผลและเรียบเรียงวิทยานิพนธ์

1.5 คุณค่าทางวิชาการ

1) ทำให้การทดลองเครือข่ายนั้นพัฒนาได้บนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ซึ่งสามารถทดลองได้ทั้งระบบจำลองเครือข่าย (Simulation) และระบบจริง (Emulation) ซึ่งเป็นการทดลองที่ทำให้ได้รับประสิทธิภาพเท่าเทียมการทดลองจริง
2) นำไปเป็นส่วนหนึ่งในการแก้ไขปัญหาระยะการส่งข้อมูลที่มีการระยะการส่งไม่เท่ากัน ซึ่งมักถูกมองข้ามไปในการทดลองแบบจำลองเครือข่าย
3) เพื่อให้การทำงานของระบบจราจรอัจฉริยะสามารถทำงานได้อย่างมีประสิทธิภาพ และมีความเชื่อถือได้มากขึ้น

1.6 ผลงานตีพิมพ์จากวิทยานิพนธ์

1) หัวเรื่อง “Improving Reliable Broadcast over Asymmetric VANETs Based on a RSSI-Voting Algorithm” โดย ณัฐวิทย์ กมลธรรม และ กุลิสร์ ณ นคร และ กุลธิดา โรจน์วิบูลย์ชัย ในบันทึกการประชุม “Intelligent Signal Processing and Communication Systems (ISPACS 2011)” ซึ่งจัดขึ้น ณ เชียงใหม่ ประเทศไทย ระหว่างวันที่ 7-9 ธันวาคม 2554
2) หัวเรื่อง “From NS-2 to NS-3 – Implementation and Evaluation” โดย ณัฐวิทย์ กมลธรรม และ กุลิสร์ ณ นคร และ กุลธิดา โรจน์วิบูลย์ชัย ในบันทึกการประชุม “Computer, Communications & Applications Conference (COMCOMAP 2012)” ซึ่งจัดขึ้น ณ ฮ่องกง ประเทศจีน ระหว่างวันที่ 11-13 มกราคม 2555

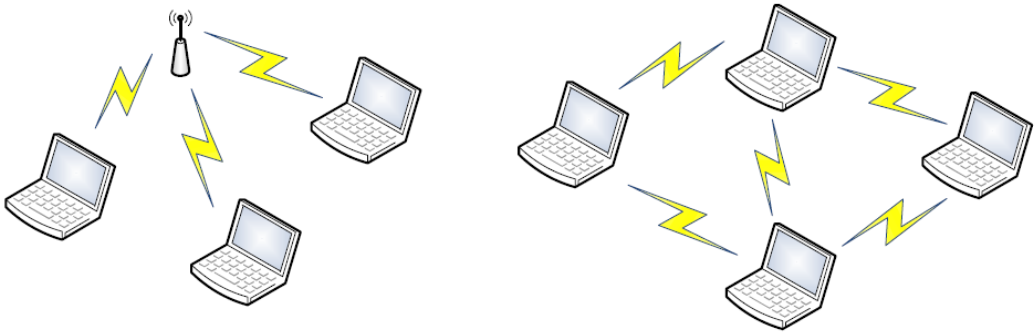
บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 การสื่อสารบนเครือข่ายไร้สายแบบแอดฮอก

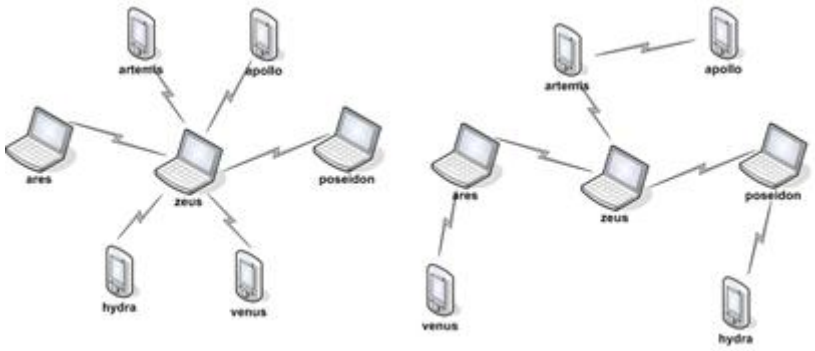
เครือข่ายไร้สายแบบแอดฮอกเป็นเครือข่ายที่อุปกรณ์ไร้สายต่างๆ สามารถติดต่อสื่อสารกันเป็นกลุ่มย่อยๆ โดยไม่มีโครงสร้างพื้นฐานเป็นตัวกลางในการติดต่อสื่อสารทำให้การสื่อสารระหว่างกันนั้นสามารถใช้งานได้ในสถานการณ์ที่โครงสร้างพื้นฐานนั้นติดตั้งไม่ได้ เช่น พื้นที่สงคราม หรือพื้นที่ภัยธรรมชาติทำลายโครงสร้างพื้นฐาน เป็นต้น ทำให้การสื่อสารด้วยเครือข่ายไร้สายแบบแอดฮอกนั้นสามารถเหมาะสมกับการพัฒนาเพื่อใช้ในการสื่อสารในปัจจุบันนี้ ซึ่งประหยัดทั้งค่าใช้จ่ายในการติดตั้งอุปกรณ์ และพัฒนาใช้งานได้อย่างรวดเร็ว โดยการใช้งานนั้นสามารถส่งข้อมูลได้ทั้งแบบ single-hop ซึ่งเป็นการส่งข้อมูลให้กับอุปกรณ์ที่อยู่ติดกันในหนึ่งการส่ง และการส่งแบบ multi-hop ซึ่งเป็นการส่งข้อมูลให้กับอุปกรณ์ที่ผู้ส่งมองไม่เห็นที่อยู่ห่างกันไปหลายครั้งการส่ง ดังแสดงในภาพที่ 2.1



(ก) การสื่อสารไร้สายแบบมีโครงสร้างพื้นฐาน (ข) การสื่อสารไร้สายแบบแอดฮอก

ภาพที่ 2.1 ความแตกต่างของเครือข่ายไร้สาย

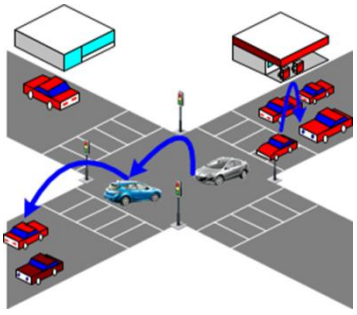
จากที่กล่าวไปข้างต้น ด้วยคุณสมบัติที่ดีของเครือข่ายไร้สายแบบแอดฮอกทำให้การศึกษาและวิจัยทางด้านนี้มีหลายแขนง ซึ่งแบ่งออกเป็นประเภทต่างๆ ที่ใช้งานในสถานการณ์ที่แตกต่างกันออกไป อย่างเช่น เครือข่ายไร้สายแบบแอดฮอกสำหรับอุปกรณ์ไร้สายเคลื่อนที่ (MANET : Mobile Ad-hoc Network) หรือเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะ (VANET : Vehicular Ad-hoc Network) เป็นต้น



(ก) สื่อสารแบบ single-hop (ข) การสื่อสารแบบ multi-hop

ภาพที่ 2.2 เครือข่ายไร้สายแบบแอดฮอก [21]

2.1.2 การสื่อสารบนเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะ



ภาพที่ 2.3 เครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะ

เครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะ เป็นเครือข่ายที่ใช้ในการติดต่อสื่อสารระหว่างยานพาหนะ ซึ่งเป็นส่วนหนึ่งของระบบจราจรอัจฉริยะที่ช่วยให้เกิดความสะดวกสบายในการขับขี่ยานพาหนะบนท้องถนน โดยเครือข่ายนี้จะแตกต่างจากเครือข่ายไร้สายแบบแอดฮอกสำหรับอุปกรณ์ไร้สายเคลื่อนที่ เพราะว่าการเคลื่อนที่ของยานพาหนะนั้นจะเป็นไปในทิศทางที่ค่อนข้างแน่นอน เพราะว่าถนนในแต่ละจุดนั้นยานพาหนะจะต้องเคลื่อนที่ไปในทิศทางเดียวกัน หรือทิศทางตรงข้ามกัน นอกจากนั้นความเร็วในการเคลื่อนที่ของยานพาหนะยังเป็นส่วนสำคัญในการทำงานของเครือข่ายนี้ เพราะว่าโพรโทคอลที่จะนำมาใช้งานในเครือข่ายนั้นจะต้องสามารถรองรับการเกิดการแยกตัวของเครือข่ายที่เกิดจากการเปลี่ยนแปลงตำแหน่งของยานพาหนะ และความหนาแน่นของยานพาหนะที่เกิดขึ้น

การติดต่อสื่อสารบนเครือข่ายนี้จึงจำเป็นที่จะต้องมีการส่งข้อมูลที่เป็นปัจจุบันและเชื่อถือได้ เพราะหากเกิดอุบัติเหตุขึ้นยานพาหนะทุกคันในละแวกจึงควรได้รับข้อมูลทันทีเพื่อที่จะหลีกเลี่ยงการใช้เส้นทางในการเดินทางนั้น ดังนั้นโพรโทคอลในการส่งข้อมูลของระบบจึงควรเป็นโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้ จึงจะทำให้เครือข่ายนั้นมีประสิทธิภาพในการทำงาน

2.1.3 มาตรฐาน IEEE 802.11p

มาตรฐานการติดต่อสื่อสารที่แตกต่างจากมาตรฐานเดิม เหมาะสำหรับการเชื่อมต่อและส่งข้อมูลเพียงสั้นๆ เพื่อสอดคล้องกับการทำงานบนเครือข่ายไร้สายที่มีการเคลื่อนที่อย่างรวดเร็วและซับซ้อน โดยตอบสนองทั้งทางด้านการส่งข้อมูลสื่อบันเทิง และทางด้านความปลอดภัย การติดต่อสื่อสารในมาตรฐานนี้ใช้ความถี่ 5.9 GHz สามารถติดต่อสื่อสารได้ในระดับความเร็วอย่างน้อย 200 กิโลเมตรต่อชั่วโมง โดยมีขอบเขตของสัญญาณอยู่ที่ระยะ 1000 เมตร

ตารางที่ 2.1 ตารางเปรียบเทียบมาตรฐาน IEEE 802.11p กับ มาตรฐานอื่นๆ

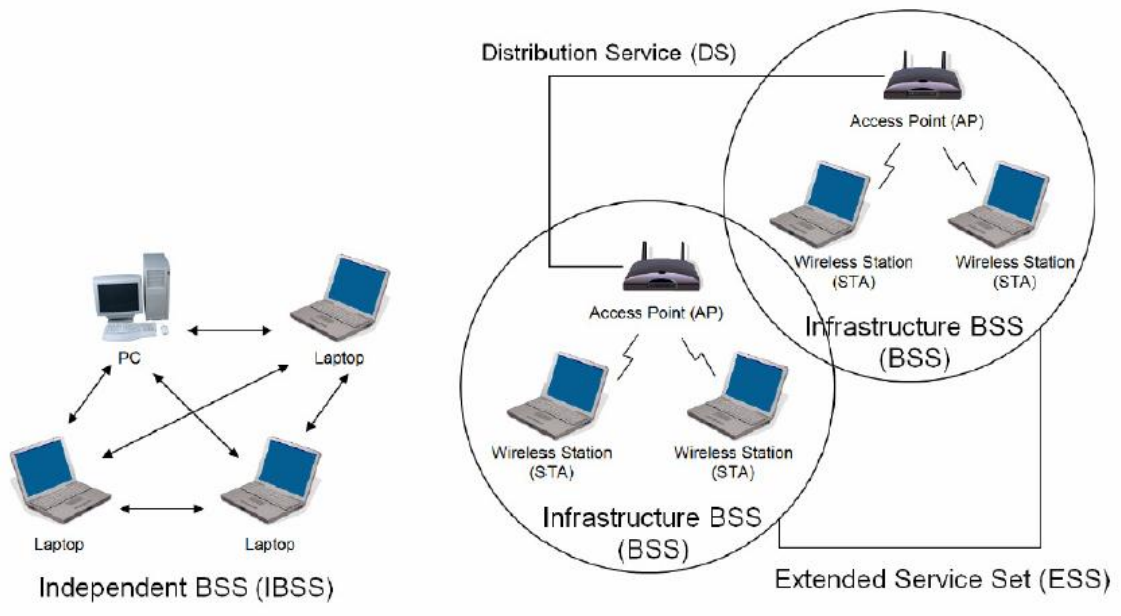
IEEE	Release[23]	Frequency (GHz)	Bandwidth (MHz)	Data Rate (Mbps)	Modulation	Outdoor Distance (m.)
802.11a	Sep. 1999	5	20	54	OFDM	120
802.11b	Sep. 1999	2.4	20	11	DSSS	140
802.11g	Sep. 2003	2.4	20	54	OFDM,DSSS	140
802.11n	Oct. 2009	2.4/5	20/40	72.2/150	OFDM	250
802.11p	In progress, IEEE-SA Sponsors Ballots (Last Schedule, Nov. 2010)	5.9	10	6-27	OFDM (doubling 802.11a parameter)	Up to 1000

ในส่วนของความแตกต่างระหว่าง IEEE 802.11p กับ IEEE 802.11 อื่นๆ นั้นแบ่งได้ออกเป็นความแตกต่างในส่วนของระดับชั้น MAC และระดับชั้น Physical

ระดับชั้น MAC

โดยทั่วไปการเชื่อมต่อของเครือข่ายไร้สายในมาตรฐาน IEEE 802.11 นั้นจะมีกระบวนการ Basic Service Set (BSS) ที่ Access Point นั้นใช้กำจัดสัญญาณอื่นๆ ที่ไม่เกี่ยวข้อง และโดยมี Extended Service Set ที่รวม BSS เป็นแบบ Logical Link Control (LLC) หรือ IBSS (Independent BSS) ที่ใช้งานกับเครือข่ายไร้สายแบบแอดฮอก แต่ขั้นตอนดังกล่าวนั้นใช้เวลามากเกินตั้งแต่ขั้นตอนการ Beacon message ของ Access Point จนถึงขั้นตอนยืนยันตัวตนและเชื่อมต่อ ทำ

ให้กระบวนการนี้ไม่เหมาะสมกับการใช้งานเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะ ที่ความรวดเร็วเป็นสิ่งจำเป็นในการทำงานของระบบอย่างมาก



ภาพที่ 2.4 Independent Basic Service และ Extended Service Set [22]

การทำงานของ IEEE 802.11p จึงแตกต่างออกไปด้วยการมี “WAVE mode” ที่ให้มีการแลกเปลี่ยนข้อมูลโดยค่าของ BSSID เป็น “*” โดยไม่ต้องอยู่ใน BSS ใดๆ ทำให้การแลกเปลี่ยนข้อมูลนั้นทำได้ทันทีเมื่ออยู่ในช่องสัญญาณเดียวกัน แต่จะไม่สามารถเป็นสมาชิก BSS หรือ IBSS ได้ และไม่สามารถยืนยันตัวตนและเชื่อมต่อ MAC ได้

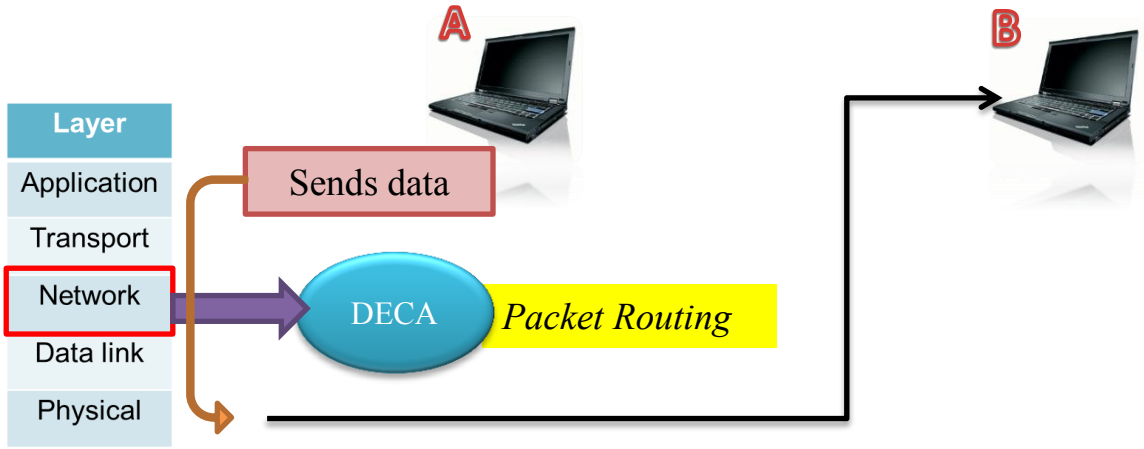
ระดับชั้น Physical

การทำงานในระดับชั้นนี้ยังใช้ IEEE 802.11a เป็นมาตรฐานในการพัฒนา ซึ่งมีความกว้างของช่องสัญญาณ 20 MHz แทน 10MHz แต่ใช้ความถี่ที่ 5.9 GHz แต่ยังมีสัญญาณรบกวนอยู่ในการรับส่งข้อมูล [22] ทำให้ต้องปรับปรุงต่อไปในการผลิตอุปกรณ์ตามมาตรฐานนี้

2.1.4 โครงสร้างเครือข่าย

โครงสร้างของเครือข่ายนั้นแบ่งออกเป็นห้าชั้นโดยเริ่มจากชั้น Application ที่โหนดผู้ส่งข้อมูลทำการกระจายข้อมูลออกไปให้กับทุกโหนด โดยจะส่งข้อมูลลงมาในแต่ละลำดับชั้นของเครือข่าย ซึ่งในชั้น Network นั้นโพรโทคอล DECA ที่เป็นโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้ทำงานในชั้นนี้เพื่อค้นหาเส้นทางในการส่งข้อมูลเพื่อกระจายข้อมูลไปให้กับทุกโหนด และเลือกผู้กระจายข้อมูลลำดับถัดไปที่เป็นการลดการซ้ำซ้อนของข้อมูลในการส่ง หลังจากนั้นข้อมูลก็จะถูกส่งลงไปในชั้นล่างของ

เครือข่ายและทำการส่งข้อมูลไปให้กับโหนดอื่นๆ ซึ่งโหนดที่รับข้อมูลก็จะรับขึ้นไปตามลำดับชั้นต่างๆ เพื่อนำข้อมูลไปใช้ต่อไป



ภาพที่ 2.5 โครงสร้างเครือข่าย

2.2 งานวิจัยที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวข้องกับวิทยานิพนธ์นี้นั้นแบ่งได้เป็น 2 ประเภทหลักๆ คือ งานวิจัยโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้ ซึ่งในวิทยานิพนธ์นี้นำเอาโพรโทคอล Density-Aware Reliable Broadcasting in Vehicular Ad-hoc Network หรือ DECA มาเป็นตัวอย่างในการวิจัย ส่วนอีกประเภทของงานวิจัยคือ งานวิจัยการแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตรบนเครือข่ายไร้สายแบบแอดฮอก โดยมีรายละเอียดดังนี้

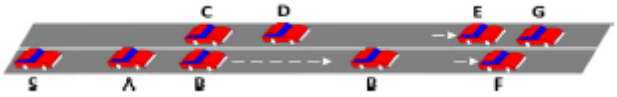
2.2.1 งานวิจัยโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้

2.2.1.1 Density-Aware Reliable Broadcasting in Vehicular Ad-hoc Network: DECA

เทคโนโลยีที่ใช้ในการแพร่กระจายข้อมูลทางจราจรนั้นใช้โพรโทคอล Density-Aware Reliable Broadcasting in Vehicular Ad-hoc Network หรือ DECA [7] ซึ่งเป็นโพรโทคอลบนเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะประเภทที่ป้องกันความซ้ำซ้อนของข้อมูลและการสูญหายของข้อมูล มีรายละเอียดดังนี้

ภาพรวมของโพรโทคอล

การทำงานของโพรโทคอลเป็นการเก็บและส่งออกไป ซึ่งจะเลือกยานพาหนะที่มีความหนาแน่นสูงหรือเชื่อมต่อกับยานพาหนะคันอื่นๆมากที่สุดเป็นผู้กระจายข้อมูลลำดับถัดไป เพราะว่าการเลือกให้ยานพาหนะที่มีความหนาแน่นสูงกระจายข้อมูลจะทำให้ลดความซ้ำซ้อนของข้อมูลที่ส่ง และการไม่ใช้ GPS ซึ่งเทียบกับโพรโทคอลอื่นๆที่ทุกคันต่างกระจายข้อมูลออกไปหมดทำให้มีข้อมูลเดียวกันนั้นกระจายไปทั่ว ทุกคันได้รับแล้วก็ได้รับอีกและทำการส่งต่อไปอีกครั้งทำให้เกิดความซ้ำซ้อนของข้อมูลมาก ดังนั้นหากเลือกให้ผู้ส่งคนเดียว และส่งได้มากที่สุดย่อมเป็นทางเลือกที่ดีกว่า ส่วนผู้ที่ไม่ได้เป็นผู้ถูกเลือกในการกระจายข้อมูลลำดับถัดไปเมื่อได้รับข้อมูลแล้วก็จะทำการจับเวลาถอยหลังแบบสุ่มเพื่อตรวจสอบการทำงานของผู้ถูกเลือกว่าทำการกระจายข้อมูลสำเร็จหรือไม่ หากพบว่ามีการกระจายข้อมูลออกมาก็จะทำการยกเลิกการจับเวลาถอยหลัง แต่หากว่าไม่มีการกระจายข้อมูลออกมาจากผู้ถูกเลือกยานพาหนะที่ทำการนับเวลาถอยหลังหมดก่อนก็จะทำการกระจายข้อมูลออกมาแทนให้กับยานพาหนะคันอื่นๆ เพื่อป้องกันความผิดพลาดที่อาจจะเกิดขึ้นจากผู้ถูกเลือกที่อาจจะได้รับผลกระทบจากช่องสัญญาณที่มีการรบกวนกันหรือการสูญหายของข้อมูล



ภาพที่ 2.6 สถานการณ์ในการส่งข้อมูลของยานพาหนะ [7]

จากภาพที่ 2.6 ยานพาหนะ S จะรู้ว่า A, B, C และ D เป็นเพื่อนบ้านของตนเองที่เชื่อมต่อกันอยู่ โดยจะรู้ว่า D เป็นยานพาหนะที่มีการเชื่อมต่อกับยานพาหนะคันอื่นในละแวกมากที่สุดจากการแลกเปลี่ยนข้อมูลระหว่างกันด้วยข้อมูล Beacon โดย S นั้นก็จะเลือกให้ D เป็นผู้ถูกเลือกในการกระจายข้อมูลลำดับถัดไป ส่วน A, B และ C ก็รับข้อมูลไปแต่จะไม่กระจายข้อมูลต่อ และนับเวลาถอยหลัง ซึ่งหากว่า D ไม่ได้กระจายข้อมูลออกมา แล้ว B เป็นผู้ที่นับถอยหลังสั้นกว่า A และ C ทำให้ B ทำการส่งข้อมูลออกมาแทน D และเมื่อ A กับ C ได้ยินการส่งข้อมูลออกมาจาก B ก็จะทำการยกเลิกการนับเวลาถอยหลัง และไม่มีการกระจายข้อมูลออกมา

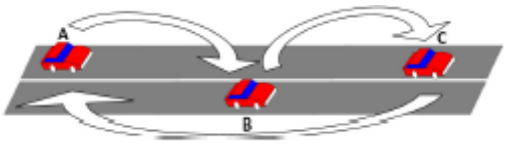
หลังจากนั้น B ก็จะเคลื่อนที่ต่อจนเจอกับ E, F และ G โดยที่ทั้งสามนั้นไม่ได้เป็นเพื่อนบ้านของ D มาก่อน และไม่อยู่ในรัศมีของ D ทำให้ไม่ได้รับข้อมูลจาก D เลย เมื่อ B ผ่านมาใกล้ๆ ก็จะทำการแลกเปลี่ยนข้อมูลกันด้วยข้อมูล Beacon ที่แลกเปลี่ยนออกมาเป็นระยะๆ แล้ว B พบว่า E, F และ G ยังไม่มีข้อมูล ทำให้ B ต้องส่งข้อมูลออกมาให้ทั้งสาม เนื่องจากอยู่ในเงื่อนไขว่าเพื่อนบ้านที่ไม่มีข้อมูลนั้นๆ จะต้องทำการส่งข้อมูลไปให้ ต่อจากนั้น E, F และ G ก็จะทำการนับเวลาถอยหลังแบบสุ่ม ซึ่งผู้ที่

นับเวลาสั้นที่สุดก็จะเป็นผู้กระจายข้อมูลต่อไป เพราะว่า E, F และ G ย่อมรู้จักกับเพื่อนบ้านในระแวกของตนเองมากกว่า B เพราะว่าอยู่ในละแวกนั้นมาก่อน

รายละเอียดของโพรโทคอล

ยานพาหนะหรือโหนดที่ใช้งานโพรโทคอล DECA นั้นจะมีรายการอยู่ 2 รายการ คือ รายการเพื่อนบ้าน (Neighbor list) กับรายการกระจายข้อมูล (Broadcast list) ซึ่งอันแรกเป็นรายการที่เก็บว่าโหนดไหนเป็นเพื่อนบ้านกับตนเองบ้าง โดยรายจากนี้จะปรับปรุงข้อมูลเป็นระยะๆ ด้วยการส่งข้อมูล Beacon ออกไปเพื่อทำการตรวจสอบข้อมูลของเพื่อนบ้านที่ใกล้เคียง ส่วนรายการหลังนั้นคือรายการเก็บข้อมูล (Message) ที่จะทำการส่งออกไปให้กับโหนดอื่น โดยข้อมูลนั้นจะถูกตัดออกจากรายการด้วย 2 วิธี คือ หนึ่ง การหมดอายุของข้อมูลนั้น หากว่าข้อมูลนั้นอยู่มานานและโหนดนั้นได้กระจายข้อมูลออกไปแล้ว สองคือเมื่อเครื่องนั้นได้รับข้อมูลจากโหนดอื่นทำให้รู้ว่าผู้ถูกเลือกได้กระจายข้อมูลได้อย่างถูกต้องก็จะลบข้อมูลนั้นออกจากรายการกระจายข้อมูล

DECA นั้นจะใช้การส่ง Beacon ออกมาเป็นระยะๆ โดย Beacon นั้นประกอบด้วยข้อมูลความหนาแน่นของแต่ละยานพาหนะ และข้อมูลของข้อมูลที่ได้รับมาทั้งหมด ทำให้แต่ละโหนดรู้ว่าเพื่อนบ้านของตนเองนั้นเป็นเช่นไร ใครมีความหนาแน่นมากที่สุด และใครได้รับข้อมูลอะไรแล้วบ้าง หากจะทำการแพร่กระจายข้อมูลออกไปก็จะรู้ว่าจะเลือกใครเป็นผู้กระจายข้อมูลลำดับถัดไป และทำให้สามารถตรวจสอบได้ว่าแต่ละโหนดนั้นยังไม่มีข้อมูลไหน ทำให้สามารถที่จะบอกออกไปว่าตนเองนั้นยังไม่มีข้อมูลนั้นๆ เพื่อให้เพื่อนบ้านทำการส่งข้อมูลดังกล่าวกลับมาให้



ภาพที่ 2.7 สถานการณ์การเกิดปัญหาการส่งข้อมูลวนซ้ำ [7]

จากภาพที่ 2.7 จะเห็นว่าการส่ง Beacon ออกมานั้นก็อาจจะเกิดการส่งวนซ้ำ (Loop) เพราะว่า A ส่งข้อมูลไปให้กับ B แล้ว B ก็ส่งข้อมูลไปให้กับ C หลังจากนั้น C พบว่า A เป็นผู้ถูกเลือกจึงส่งข้อมูลกลับไปให้ A ทำให้เกิดการวนซ้ำ การแก้ปัญหาดังกล่าวจึงต้องมีการระบุว่าใครได้รับข้อมูลนั้นแล้ว เพื่อที่จะไม่ส่งไปให้อีกซ้ำๆ นี่ก็คือประโยชน์อีกอย่างที่ควรมี Beacon ด้วย

แต่เมื่ออยู่ในสถานการณ์การเชื่อมต่อแบบอสมมาตรหลักการเลือกผู้กระจายข้อมูลลำดับถัดไปของโพรโทคอล DECA นั้นไม่สามารถทำงานได้อย่างมีประสิทธิภาพ เพราะการเลือกโหนดที่มีความหนาแน่นของเพื่อนบ้านเยอะแต่ว่าโหนดนั้นอาจจะมีความแรงของสัญญาณในการส่งข้อมูลต่ำ ทำให้

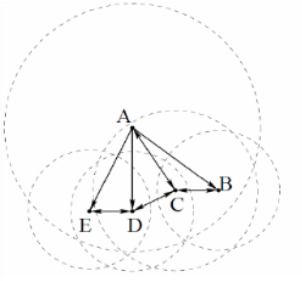
ระยะการส่งข้อมูลนั้นสั้นซึ่งมีผลทำให้ไม่สามารถส่งข้อมูลไปให้กับเพื่อนบ้านที่อยู่ติดกันได้ ดังแสดงในภาพที่ 1.4(ข) ทำให้ต้องหาวิธีแก้ไขในส่วนของการเลือกผู้กระจายข้อมูลลำดับถัดไป

2.2.2 งานวิจัยการแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตรบนเครือข่ายไร้สายแบบแอดฮอก

2.2.2.1 Modified AODV with QoS for Asymmetric MANET

งานวิจัยของ Jairath A et al ในปี 2010 ชื่อว่า Modified AODV with QoS for Asymmetric MANET [8] เสนอวิธีแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตรบนเครือข่ายไร้สายแบบแอดฮอกสำหรับอุปกรณ์ไร้สายเคลื่อนที่ (MANET : Mobile Ad-hoc Network) ด้วยวิธีการแก้ไขโพรโทคอล AODV ซึ่งเป็นโพรโทคอลประเภท Unicast โดยแต่ละโหนดต่อเชื่อมกันในระบบ ซึ่งการส่งข้อมูลถ้าหากว่าเส้นทางในการส่งข้อมูลนั้นปกติก็ยังส่งข้อมูลตามเส้นทางที่ได้จากการหาเส้นทาง (Routing) แต่หากว่าเส้นทางในการส่งข้อมูลนั้นเกิดขาดหรือมีปัญหาขึ้น ก็จะทำการแก้ไขปัญหาที่เกิดขึ้นด้วยการหาเส้นทางใหม่ที่ย้อนกลับทางที่ต้องการส่งข้อมูลเรียกว่า Reverse path โดยเส้นทางนั้นจะต้องมีค่าใช้จ่ายน้อยและไม่เกิดการวนซ้ำ ซึ่งเมื่อได้เส้นทางดังกล่าวแล้วก็จะทำการส่งข้อมูลอีกครั้งไปทางใหม่ ด้วยหลักการนี้ทำให้สามารถแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตรได้

แต่การแก้ไขปัญหาดังกล่าวนั้นเป็นการแก้ไขบนโพรโทคอลที่มีการส่งข้อมูลบนเส้นทางเดียว (Unicasting protocol) ซึ่งเป็นโพรโทคอลที่มีเส้นทางในการส่งข้อมูลบนเส้นทางหลักทางเดียว ซึ่งทำให้การแก้ไขนั้นสามารถหาเส้นทางอื่นในการส่งข้อมูลแทนได้ทำให้การส่งข้อมูลนั้นทำได้บนสถานการณ์ที่มีการเชื่อมต่อแบบอสมมาตร

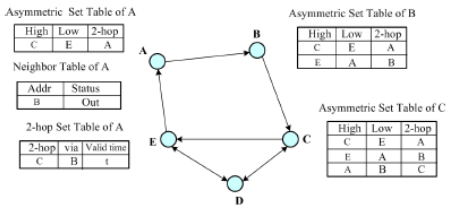


ภาพที่ 2.8 การแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตรบนโพรโทคอล AODV [8]

2.2.2.2 Extended Multicast Optimized Link State Routing Protocol in MANETs with Asymmetric Links

งานวิจัย Extended Multicast Optimized Link State Routing Protocol in MANETs with Asymmetric Links [9] ของ Yong Bai ในปี 2007 นั้นเป็นการแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตรสำหรับเครือข่ายไร้สายแบบแอดฮอกสำหรับอุปกรณ์ไร้สายเคลื่อนที่ด้วยการแก้ไขโพรโทคอล Multicast-OLSR (MOLSR) ให้ทำงานได้บนสถานการณ์ดังกล่าว ซึ่งโพรโทคอลใหม่ที่แก้ไขมีชื่อว่า MOLSR-ASYM วิธีแก้ไขนั้นทำโดยการใช้ Hello message ให้ตรวจสอบการเชื่อมต่อของแต่ละโหนดว่าเกิดปัญหาดังกล่าวขึ้นที่จุดไหน ซึ่งหากพบว่ามีเส้นทางที่ขาดจากกันขึ้นก็จะทำการเปลี่ยนเส้นทางในการส่งข้อมูลไปยังเส้นทางอื่นที่ไม่เกิดปัญหาดังกล่าวแทน ทำให้โพรโทคอลนี้สามารถทำงานได้บนสถานการณ์ที่มีการเชื่อมต่อแบบอสมมาตร

การแก้ไขดังกล่าวนั้นเป็นการทำงานบนโพรโทคอล MOLSR ที่แตกต่างจากโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้ (Reliable Broadcasting Protocol) ซึ่งการแก้ไขนั้นสามารถทำได้โดยการหาเส้นทางใหม่แทน ทำให้สามารถส่งข้อมูลได้



ภาพที่ 2.9 การแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตรบนโพรโทคอล MOLSR [9]

2.2.2.3 Using Asymmetric Link to Improve SSR's Routing Performance

งานวิจัย [10] ของ Birnstill et al ในปี 2010 นี้เป็นงานที่เสนอวิธีในการแก้ไขการทำงานของโพรโทคอล Scalable Source Routing Protocol (SSR) ให้ทำงานได้บนสถานการณ์การเชื่อมต่อแบบอสมมาตร โดยการพัฒนา Hello message หลักการทำงานคือ กลุ่มโหนดที่เชื่อมต่อกันในรูปแบบของต้นไม้จะถือว่าเชื่อมต่อกันด้วย Inbound path แต่เมื่อการเชื่อมต่อดังกล่าวเกิดปัญหาการเชื่อมต่อแบบอสมมาตรขึ้น โพรโทคอลที่ได้รับการแก้ไขแล้วจะแก้ไขด้วยการหาเส้นทางใหม่หรือที่เรียกว่า Outbound path เพื่อใช้ในการส่งข้อมูลแทนเส้นทางเดิมที่เสียไป ทำให้โพรโทคอลทำงานได้ดีขึ้น

แต่การแก้ไขปัญหาดังกล่าวนั้นทำบนโพรโทคอลที่มีการส่งข้อมูลบนเส้นทางเดียว ซึ่งมีเส้นทางในการส่งข้อมูลหลัก ทำให้การแก้ปัญหาการเชื่อมต่อแบบอสมมาตรที่เกิดขึ้นนั้นทำได้โดยการหา

เส้นทางใหม่ที่ดีกว่าในการส่งข้อมูลแทน ทำให้โพรโทคอลสามารถทำงานได้บนสถานการณ์ที่เกิดปัญหานั้นๆ

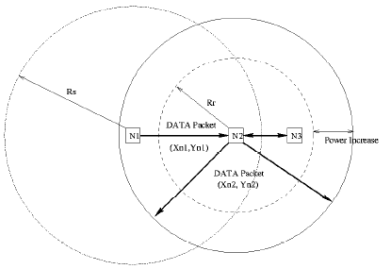
2.2.2.4 On Supporting Link Asymmetric in Mobile Ad hoc Networks

งานวิจัย [11][12] ของ Dongkyun K. ในปี 2001 และ 2000 ที่เกี่ยวข้องกับการเสนอวิธีแก้ไขให้โพรโทคอลบนเครือข่ายไร้สายแบบแอดฮอกสำหรับอุปกรณ์ไร้สายเคลื่อนที่ นั้นสามารถทำงานได้บนสถานการณ์การเชื่อมต่อแบบอสมมาตร โดยมีการเสนอแนวทางแก้ไขสำหรับโพรโทคอลในสองลักษณะดังนี้

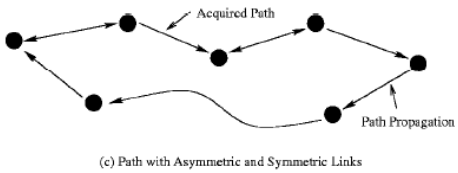
1. Link level ประเภทของโพรโทคอลที่แต่ละโหนดแบ่งกลุ่มออกเป็นลำดับชั้น ตัวอย่างเช่น
    a. โพรโทคอล GPS-based Hop by hop Acknowledgement (GAHA)
    b. โพรโทคอล GPS-based Passive Acknowledgement (GAPA)

ซึ่งเมื่อโหนดในลำดับชั้นบนพบว่าโหนดในลำดับชั้นล่างนั้นไม่สามารถที่จะส่งข้อมูลมาให้กับโหนดบนได้ โหนดในลำดับชั้นล่างนั้นจะเพิ่มความแรงของสัญญาณในการส่งข้อมูลเพื่อให้ข้อมูลนั้นสามารถส่งถึงโหนดในลำดับชั้นบนได้ ซึ่งทำให้สามารถแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตรได้

2. End-to-end ประเภทของโพรโทคอลที่โหนดเชื่อมต่อกันและต้องการส่งข้อมูลไปให้กันระหว่างโหนด ซึ่งมีการสร้างเส้นทางในการส่งข้อมูลสองเส้นทาง โดยเมื่อเส้นทางหลักที่ใช้ส่งข้อมูลนั้นขาด ก็สามารถแก้ไขด้วยการส่งข้อมูลไปยังอีกเส้นทางที่เตรียมไว้ เพื่อแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตร ตัวอย่างเช่น
    a. โพรโทคอล Routing Protocol based on Source-initiated (RODA)



ภาพที่ 2.10 โพรโทคอลประเภทที่แบ่งโหนดเป็นลำดับชั้น [11]



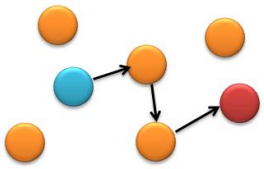
(c) Path with Asymmetric and Symmetric Links

ภาพที่ 2.11 โพรโทคอลประเภทที่โหนดเชื่อมต่อระหว่างกัน [11]

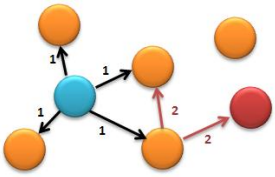
2.2.3 ความแตกต่างระหว่างงานวิจัยการแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตร

จากงานวิจัยที่เกี่ยวข้องพบว่าโพรโทคอล DECA นั้นไม่สามารถทำงานได้อย่างมีประสิทธิภาพเมื่ออยู่ในสถานการณ์การเชื่อมต่อแบบอสมมาตร ซึ่งพบว่าส่วนที่เป็นปัญหาคือการเลือกผู้กระจายข้อมูลลำดับถัดไป ทำให้ต้องการวิธีการแก้ไขเพื่อให้การเลือกผู้ถูกเลือกนั้นเป็นไปได้อย่างมีประสิทธิภาพ แต่เมื่อศึกษางานวิจัยที่เกี่ยวข้องกับการแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตรพบว่างานวิจัยส่วนใหญ่นั้นพัฒนาบนโพรโทคอลที่มีการส่งข้อมูลบนเส้นทางเดียว ซึ่งเป็นโพรโทคอลที่มีเส้นทางหลักในการส่งข้อมูลเพียงหนึ่งเส้นทาง ดังภาพที่ 2.12 ดังนั้นเมื่อเกิดปัญหาดังกล่าวขึ้นจึงสามารถแก้ไขปัญหาด้วยการหาเส้นทางอื่นในการส่งข้อมูลแทน ทำให้สามารถแก้ไขการทำงานของโพรโทคอลที่มีการส่งข้อมูลบนเส้นทางเดียว บนสถานการณ์การเชื่อมต่อแบบอสมมาตรได้

แตกต่างจากโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้ อย่างเช่น DECA [7], POCA [16], EAEP [17], AckPBSM [18] และ DV-Case [19] ซึ่งใช้หลักการกระจายข้อมูลออกไปในการส่งข้อมูลทำให้ไม่มีเส้นทางหลักในการส่งข้อมูล แต่ต้องกระจายออกไปทั่วๆ รอบโหนดของตนเอง ดังภาพที่ 2.13 ทำให้หลักการแก้ไขปัญหาการเชื่อมต่อแบบอสมมาตรด้วยการหาเส้นทางอื่นนั้นเป็นไปไม่ได้ จึงต้องหากระบวนการในการแก้ไขปัญหาดังกล่าวในลักษณะอื่นแทน



ภาพที่ 2.12 โพรโทคอลที่มีการส่งข้อมูลบนเส้นทางเดียว



ภาพที่ 2.13 โพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้

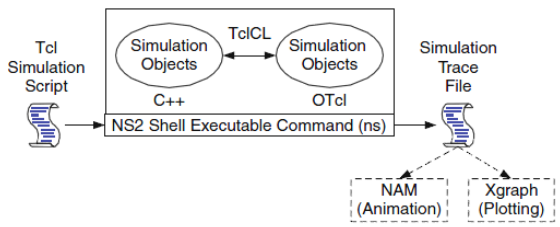
บทที่ 3

การพัฒนาโพรโทคอลจากโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ไปยังเวอร์ชั่น 3

3.1 โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2

โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 (NS-2 : Network Simulator 2) [3] เป็นโปรแกรมจำลองสถานการณ์เชิงเหตุการณ์ที่มีไว้เพื่อศึกษาพฤติกรรมของการทำงานของระบบเครือข่ายทั้งแบบมีสายและแบบไร้สาย ซึ่งความยืดหยุ่นของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ทำให้ได้รับความนิยมอย่างแพร่หลายตั้งแต่ที่เริ่มมีโปรแกรมนี้มาในปี พ.ศ. 2532 โดยพื้นฐานของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ได้มาจากโปรแกรมจำลองสถานการณ์ระบบเครือข่าย REAL ที่ถูกพัฒนาขึ้นโดย University of California ร่วมกับ Cornell University โดยที่ปัจจุบันโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ได้ Defense Advanced Research Projects Agency (DARPA) ที่ได้พัฒนาผ่าน Virtual InterNetwork Testbed (VINT) และ National Science Foundation (NSF) เป็นผู้ดูแลเรื่องการพัฒนาโปรแกรมนี้ต่อไป

โดยสถาปัตยกรรมพื้นฐานจะเป็นไปดังภาพที่ 2.14 โดยโปรแกรมนั้นจะทำการรับภาษาในการจำลองเครือข่าย Tcl (Tcl Simulation Script) มาเพื่อสร้างไฟล์จำลองจราจร (Simulation trace file) เพื่อนำไปสร้างกราฟหรือภาพเคลื่อนไหวต่อไป โดยที่โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 จะประกอบไปด้วยภาษาที่ใช้ในการทำงานสองภาษาหลัก คือ ภาษา C++ และObjected-oriented Tool Command Language (OTcl) โดยที่ภาษา C++ จะทำหน้าที่คิดกลไกการทำงาน ส่วน OTcl จะเป็นตัวสร้างสถานการณ์จำลองขึ้น ซึ่งทั้งสองภาษานี้จะถูกเชื่อมต่อกันโดย TclCL ซึ่งทำหน้าที่สร้างความสัมพันธ์ระหว่าง C++ และ OTcl ขึ้น โดยหลังจากการจำลองสถานการณ์แล้วโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 จะให้ผลลัพธ์ออกมาเป็นทั้งแบบข้อความและภาพเคลื่อนไหวโดยที่มีเครื่องมือ เช่น NAM (Network AniMator) และ XGraph เป็นเครื่องมือแสดงผล

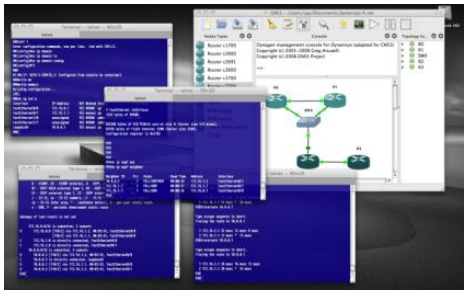


ภาพที่ 3.1 สถาปัตยกรรมพื้นฐานของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 [3]

3.2 โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3

โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 (NS-3 : Network Simulator 3) [4] เป็นโปรแกรมจำลองสถานการณ์เหมือนกับโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 แต่ไม่ได้เป็นการวิวัฒนาการมาจากเวอร์ชั่นเดิม เนื่องจากมีการเริ่มพัฒนาใหม่ทั้งหมด โดยจะไม่สามารถนำไปใช้ร่วมกับโปรแกรมเวอร์ชั่นเก่าได้ โดยที่สร้างขึ้นเพื่อระบบอินเตอร์เน็ต และเพื่อการศึกษางานวิจัยเป็นหลัก โดยที่มีส่วนเปลี่ยนแปลงไปอย่างน่าสนใจ ดังนี้

- สามารถใช้ภาษา Python เป็นตัวช่วยในการพัฒนาโปรแกรมได้นอกเหนือจาก C++
- มีการปรับเปลี่ยนโครงสร้างในการทำงานใหม่ ให้มีลักษณะเป็นโครงสร้างวัตถุ (Objected-oriented)
- มีการใช้งานเหมือนกับระบบเครือข่ายจริง
- เน้นในด้านการผสมผสานของซอฟต์แวร์
- มีระบบการทำงานจริง (Emulation) ทำให้สามารถทดลองบนระบบเครือข่ายจริงได้
- มีคู่มือการใช้งานระบบ Attribute ที่ดีขึ้น



ภาพที่ 3.2 ลักษณะของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 [4]

3.3 ความแตกต่างระหว่างโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 กับเวอร์ชั่น 3

กล่าวถึงโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 แล้วเป็นโปรแกรมจำลองเครือข่ายที่มีประสิทธิภาพ ได้เริ่มมีการพัฒนาโครงการโปรแกรมจำลองเครือข่ายมาตั้งแต่ปี พ.ศ. 2532 โดยพัฒนาต่อยอดมาจากระบบจำลองเครือข่าย REAL โดย ISI (Information Sciences Institute) และใช้กันอย่างแพร่หลายมาหลายปี จะเห็นได้จากการที่มีผู้ใช้มากกว่าครึ่งหนึ่งที่เลือกใช้โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ในการทำวิจัยของ ACM และ IEEE ในระหว่างปี พ.ศ. 2543 และ พ.ศ. 2547

โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 เป็นโปรแกรมจำลองเครือข่ายการทำงานของระบบเครือข่ายแบบจำลองเหตุการณ์ไม่ต่อเนื่อง (Discrete event simulator) ซึ่งสนับสนุนการจำลอง

เครือข่ายมากมาย เช่น การเลือกเส้นทางในการขนส่งสินค้า จำลองการทำงานของโพรโทคอล เช่น UDP, TCP, RTP ทั้งที่อยู่บนเครือข่ายแบบมีสายและแบบไร้สาย อีกทั้งยังสนับสนุนโพรโทคอลแบบหลากหลาย (Multiple protocol) และสามารถแสดงรายละเอียดการจราจรของระบบเครือข่ายออกมาในรูปแบบของกราฟฟิก และสนับสนุนกระบวนการในการหาเส้นทาง (Routing) และการลำดับข้อมูลต่างๆ ด้วย โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 เป็นโปรแกรมเปิดข้อมูล (Open source) ซึ่งสามารถทำงานได้บนทั้ง Linux, FreeBSD, SunOS, Solaris และ Windows โดยจะใช้ภาษา C++ ในการเขียนส่วนของโปรแกรมที่เป็นการกำหนดกระบวนการต่างๆรวมถึงการสร้างโพรโทคอล และใช้ OTcl ในการจำลองสถานการณ์ขึ้น โดยกำหนดคุณลักษณะต่างๆ ซึ่งจะเขียนออกมาในรูปแบบของ TCL Script

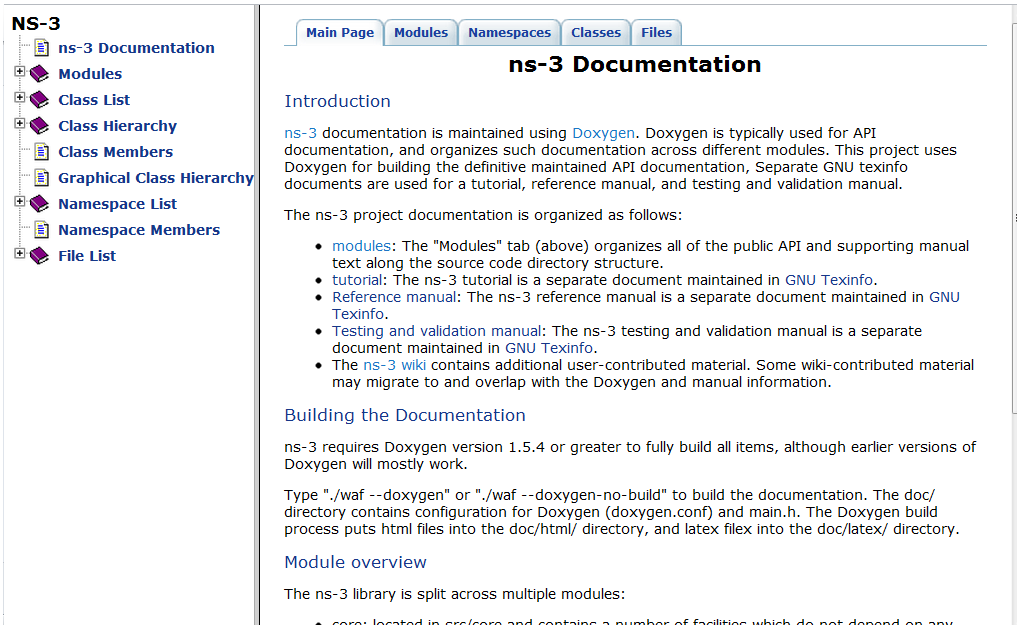
แต่ถึงแม้โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 เป็นโปรแกรมที่มีประสิทธิภาพในการทำงานสูงและได้รับความนิยมอย่างแพร่หลายก็ตาม ก็ยังคงมีปัญหาในการใช้งานอยู่หลายประการ เช่น

1. การออกแบบรหัสเป็นแบบเก่า ไม่สามารถใช้งานได้กับการออกแบบรหัสใหม่ได้
2. ความสามารถในการรองรับจำนวนเครื่องในเครือข่ายที่ทดลองมีไม่มาก
3. การใช้งานระบบสืบข้อมูล (Tracing) ใช้งานได้ค่อนข้างยาก

ทั้งนี้ทั้งนั้นเหตุผลที่โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 มีปัญหาด้านการใช้งานก็มาจากเหตุผลหลักๆ คือ การที่ไม่ได้รับการดูแลเป็นเวลานาน ซึ่งจะเห็นได้จากการที่โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 มีการแก้ไขครั้งล่าสุดคือ เวอร์ชั่น 2.34 ในปี พ.ศ. 2552 ซึ่งรุ่นก่อนหน้านั้นคือ เวอร์ชั่น 2.33 ได้ออกมาในปี พ.ศ. 2551 จะเห็นได้ว่าการเปลี่ยนแปลงเป็นไปอย่างช้าและใช้เวลานาน จึงทำให้ความนิยมในการใช้โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ก็มีน้อยลงไปตามลำดับ

ในปี พ.ศ. 2549 ได้มีการเปิดตัวโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ขึ้นโดยที่โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 นั้นเป็นโครงการที่ตั้งใจจะใช้เวลาพัฒนา 4 ปีแต่ก็ยังพัฒนาเรื่อยมาจนถึงปัจจุบัน โดยที่การพัฒนานั้นไม่ได้ต่อเนื่องจากโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 แต่ทำการพัฒนาใหม่ทั้งหมดจากเริ่มต้น โดยที่มีมหาวิทยาลัย เช่น University of Washington, Georgia Tech University รวมถึง บริษัทอื่นร่วมในการให้ทุนในการวิจัยโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ขึ้น เป้าหมายหลักของการพัฒนาคือ การพัฒนาสภาพแวดล้อมการจำลองที่เป็นที่นิยมและเปิดกว้างสำหรับผู้วิจัยทางด้านเครือข่าย และเป็นโครงงานที่เปิดกว้างเพื่อสร้างเครือข่ายผู้ใช้งานที่จะร่วมกันสร้างและมีคนช่วยกันเป็นจำนวนมากเพื่อให้โครงการนี้ออกมาสมบูรณ์แบบ โดยในปัจจุบันการพัฒนาจะค่อนข้างเน้นไปทางเดียวกับที่การวิจัยอินเตอร์เน็ตทำ ดังต่อไปนี้

1. การที่แกนของซอฟต์แวร์สามารถเพิ่มเติมได้
    a. เขียนในภาษา C++โดยมี Python เป็นส่วนต่อประสาน นอกจากนั้นยังมีเอกสารประกอบ API โดยใช้ Doxygen



ภาพที่ 3.3 ตัวอย่างเอกสารประกอบ API ของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 [4]

    b. โครงการนี้จะทำการดูแลในส่วนของแกนหลักและคงไว้ซึ่งการออกแบบเดิม ในส่วนของกลุ่มผู้ใช้งานจะทำหน้าที่ในการพัฒนาส่วนเสริมนอกเหนือจากที่โครงการนี้ดูแล
    c. พยายามหลีกเลี่ยงปัญหาที่เกิดขึ้นในโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 เช่น ปัญหาการเข้ากันไม่ได้ระหว่างแบบจำลอง และการจัดการหน่วยความจำที่ทำได้ไม่ดีพอ
2. มีความเสมือนจริงและสามารถใช้งานจริงได้

ปัญหาที่มีความสำคัญมากในการวิจัยหลักๆ แล้วคือ การที่งานวิจัยจำเป็นต้องผสมผสานการทำงานระหว่างการจำลองสถานการณ์และการใช้งานจริงของเครือข่าย ซึ่งการจำลองเครือข่ายไม่สามารถทำได้เพียงพอ ทำให้เป็นการยากหากต้องการนำโปรแกรมนั้นไปใช้จริงและยากในการเปรียบเทียบผลลัพธ์ที่ได้กับผลลัพธ์ที่ควรเป็นในระบบเครือข่ายจริง โดยโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ได้รับการออกแบบให้คล้ายกับการทำงานบนระบบเครือข่ายจริงให้มากที่สุด เช่น การออกแบบโหนดมาเพื่อสอดคล้องกับคอมพิวเตอร์จริงๆ การใช้ระบบซอกเก็ต (Socket) และส่วนประสานผู้ใช้ด้วย IP (IP/Device driver interface) การนำกลับมาใช้ใหม่ของ Kernel และรหัสต้นฉบับต่างๆ ได้

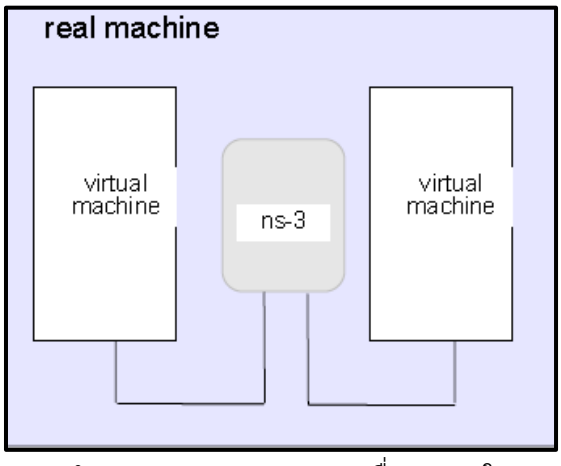
3. การบูรณาการของซอฟต์แวร์

ในส่วนนี้ปัญหาเดิมของการจำลองเครือข่ายคือ การที่โปรแกรมจะสร้างตัวแบบและเครื่องมือใหม่ขึ้นทั้งๆ ที่มีเครื่องมือเปิดข้อมูล (Open source) ต่างๆ ที่สามารถนำมาใช้ได้ให้เลือกมากมายในปัจจุบัน ดังนั้นโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 จึงคงไว้ซึ่งข้อมูลขาเข้า (Input) และข้อมูลขาออก (Output) ที่เป็นมาตรฐาน เพื่อที่เครื่องมืออื่นๆ สามารถมาใช้กันได้ เช่น Pcap trace output, NS-2 mobility script เป็นต้น นอกจากนี้โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ยังมีการเพิ่มส่วนช่วยเหลือในการทำงานของรหัส เช่น Network Simulator Cradle ที่สามารถทำงานได้กับ Linux TCP code

4. รองรับการทำงานของการสร้างมโนภาพและการทดสอบโปรแกรม

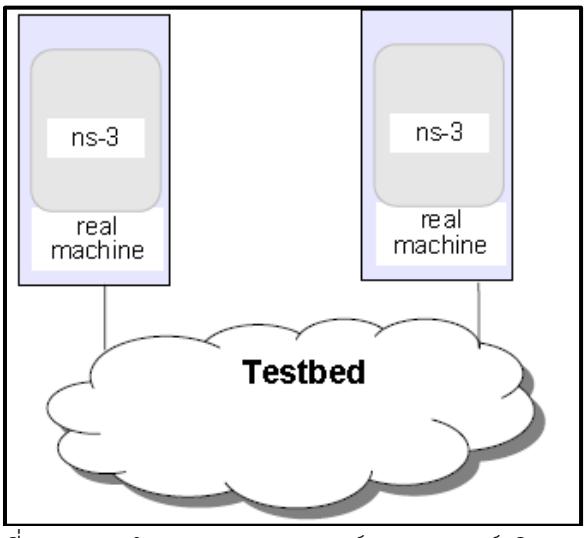
เนื่องจากนักวิจัยต้องการย้ายการพัฒนาระหว่างการจำลองไปเป็นการทดลองบนระบบจริง และยังไม่มีโปรแกรมจำลองเครือข่ายที่สนับสนุนนัก โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ได้แก้ไขปัญหานี้โดยการสร้างรูปแบบการทำงานอีก 2 แบบ คือ

- Virtual machines ทำงานบนอุปกรณ์ และช่องสัญญาณของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3



ภาพที่ 3.4 การจำลอง Virtual machine เพื่อทดลองในระบบจริง [4]

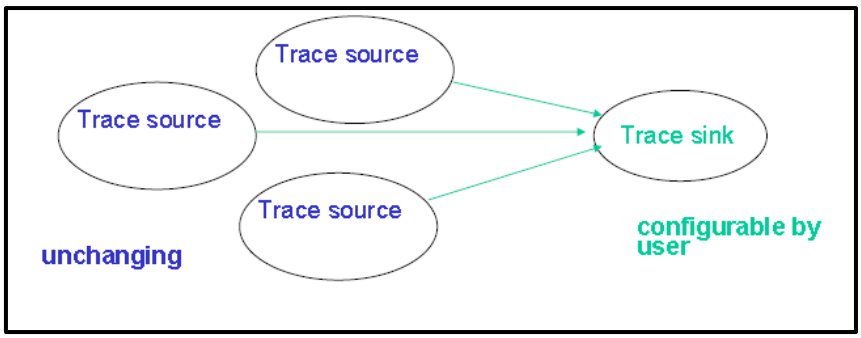
- ใช้กองซ้อน (Stack) ในการทำงานแบบสถานการณ์จริง (Emulation) ซึ่งสามารถส่งข้อมูลได้บนระบบ และอุปกรณ์จริง



ภาพที่ 3.5 การทำงานบนสถานการณ์และอุปกรณ์จริง [4]

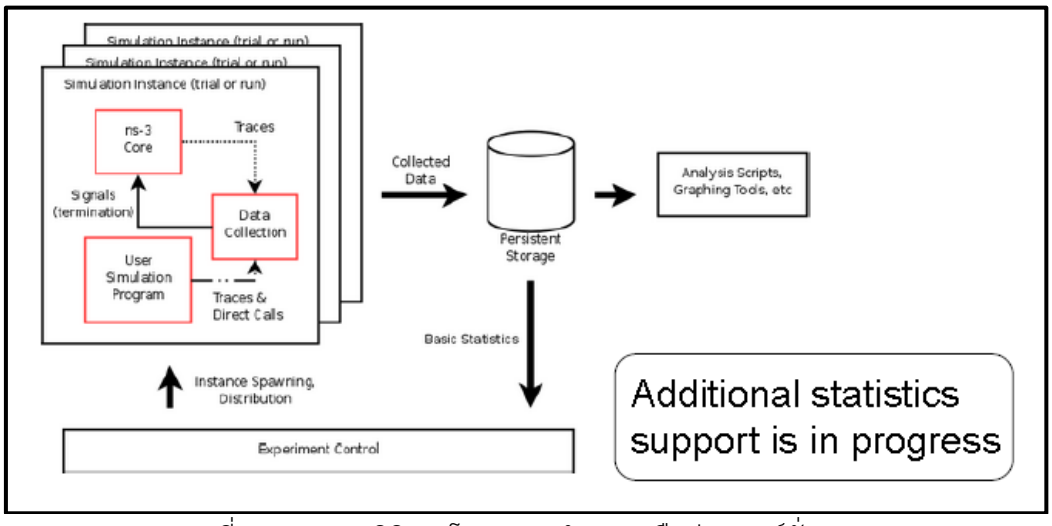
5. การที่มีระบบสืบข้อมูล (Tracing) และสถิติที่มีความยืดหยุ่นสูง

- โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ได้ทำการแยก trace source ออกจาก trace sink เพื่อให้ผู้ใช้งานสามารถปรับแต่ง trace sink ได้

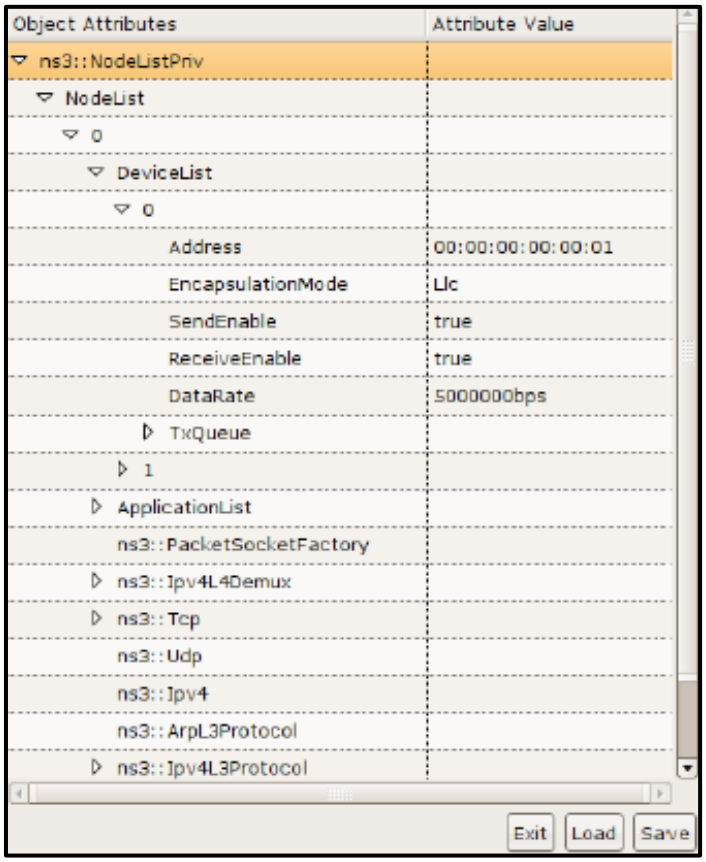


ภาพที่ 3.6 ระบบสืบข้อมูล (tracing) ที่ปรับแต่ง trace sink ได้ [4]

- โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ได้มีการสร้าง trace sources ให้เลือกใช้มากมาย เช่น Packet reception, state machine transitions
- ระบบสืบข้อมูลยังรองรับการทำงานสถิติและการจัดข้อมูลโดยการทำงานเป็นดังนี้
  - จัดการการทำงานหลายๆ สถานการณ์ที่ไม่เกี่ยวข้องกันได้
  - จัดรูปแบบข้อมูลให้ออกมาในหลายรูปแบบของข้อมูลขาออก
  - ทำงานเกี่ยวเนื่องกับ trace source
  - ข้อมูลทางสถิติสามารถเชื่อมต่อกับการจำลองเครือข่ายได้ในขณะกำลังทำงานอยู่ เช่น หลักการทำงานของการจำลองเครือข่ายถ้าหากตัวนับ (Counter)ถึงค่าหนึ่งๆ



ภาพที่ 3.7 ระบบสถิติของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 [4]



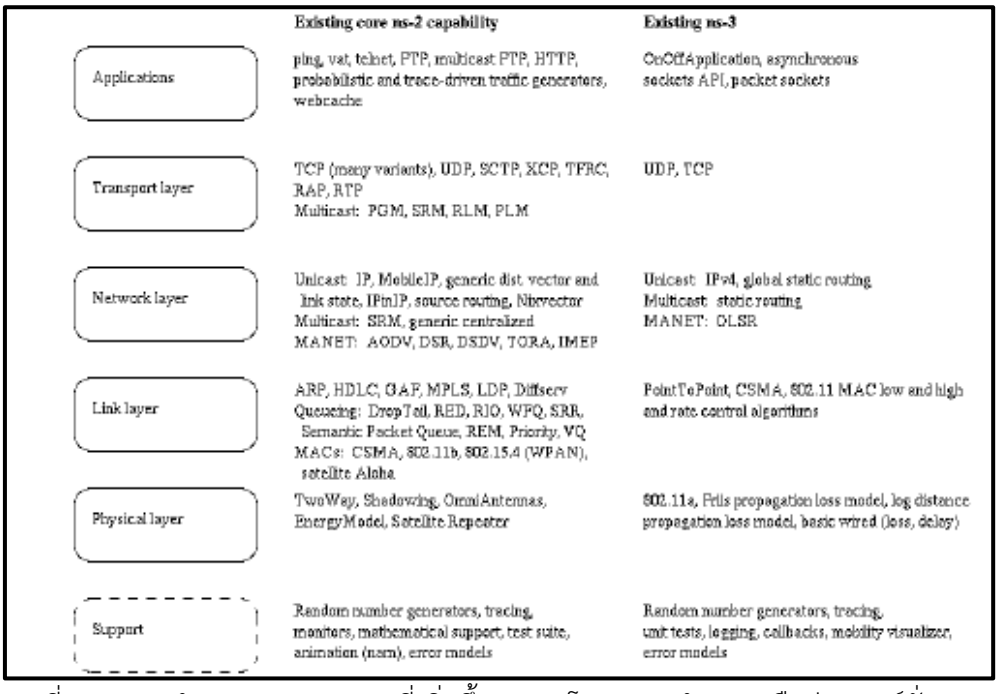
ภาพที่ 3.8 ตัวอย่างกราฟฟิกการกำหนดค่าในโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 [4]

6. มีระบบคุณลักษณะ (Attribute)

เนื่องจากนักวิจัยต้องการที่จะรู้ค่าที่เปลี่ยนแปลงในระหว่างการจำลองเครือข่ายและสามารถจัดการได้อย่างง่าย ในโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 จึงได้มีการสร้างชุดคุณลักษณะขึ้น โดยที่จะประกอบด้วย ชื่อและข้อความช่วยเหลือ ชนิด และค่าเริ่มต้น ซึ่งจะทำหน้าที่ดังนี้

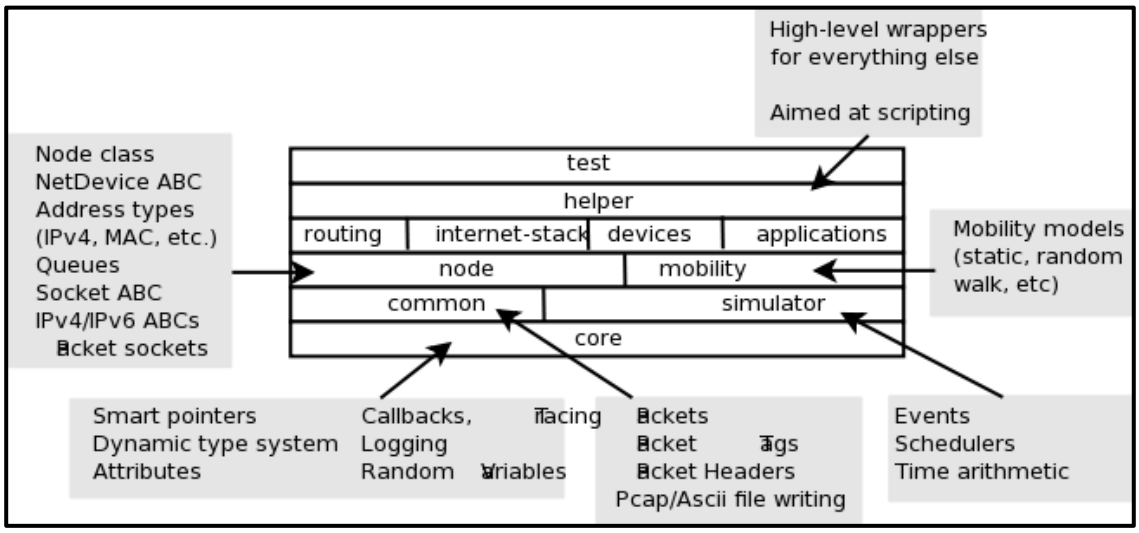
- จัดการตัวแปรของการจำลองเครือข่ายสำหรับวัตถุนิ่ง (Static object) ทุกตัว
- ทำการเก็บข้อมูล (dump) และอ่านข้อมูลที่เก็บทั้งหมดในไฟล์การปรับค่า (Configuration)
- สร้างมโนภาพโดยใช้ส่วนประสานกราฟฟิกกับผู้ใช้ (Graphic user interface)
- ทำให้ง่ายต่อการตรวจสอบตัวแปรของการจำลองเครือข่าย
- มีการสร้างเอกสารประกอบโดยใช้ Doxygen
- มีเครื่องมือกำหนดที่เป็นกราฟฟิก ดังภาพที่ 3.8

7. มีแบบจำลองใหม่



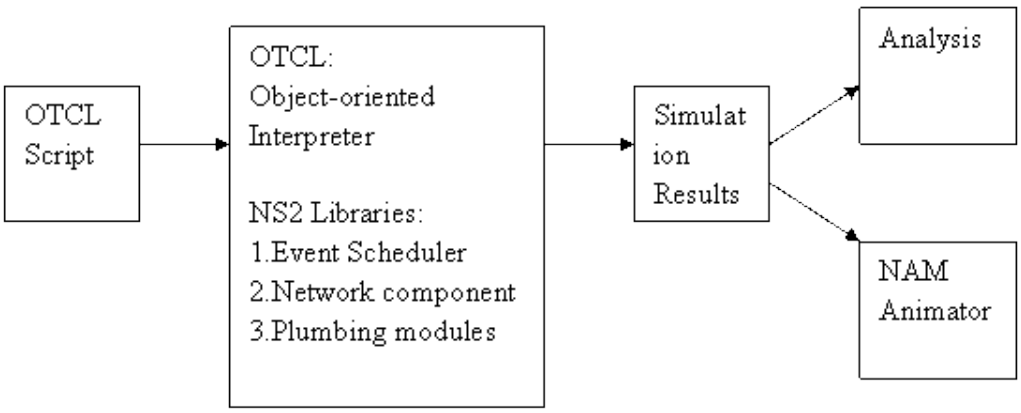
ภาพที่ 3.9 แบบจำลองความสามารถที่เพิ่มขึ้นมาของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 [6]

นอกจากนั้นการทำงานของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ยังได้แบบการทำงานออกเป็นส่วนจำเพาะ (Module) ออกจากกันเพื่อให้ง่ายต่อการพัฒนา และสอดคล้องกับการทำงานในระบบเครือข่ายจริง ดังภาพที่ 3.10

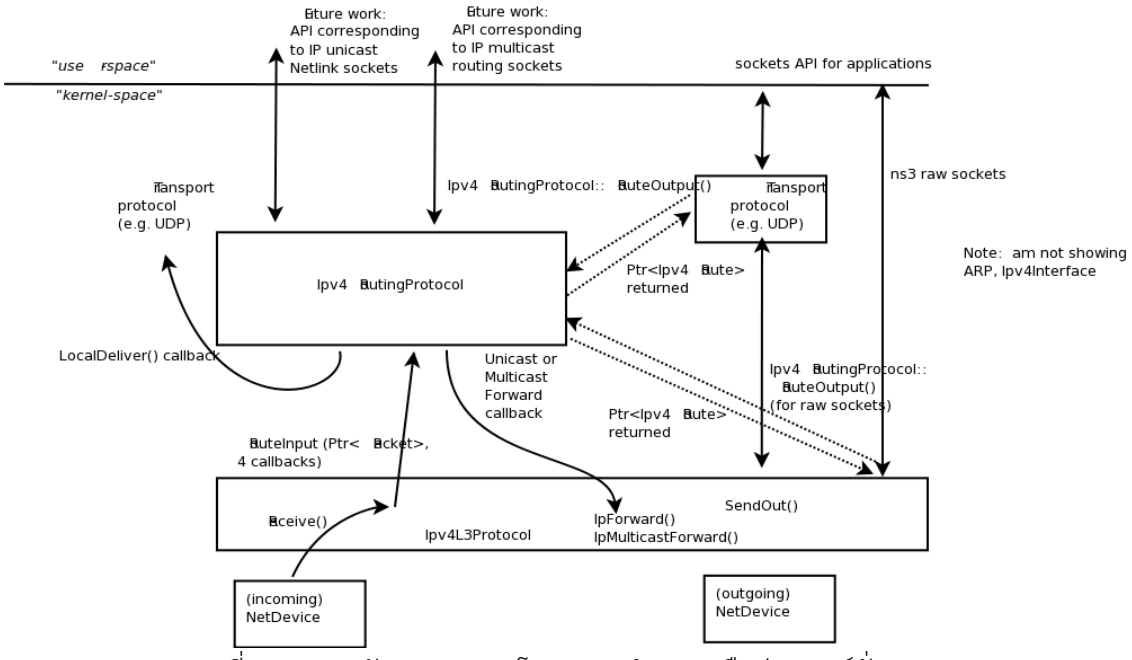


ภาพที่ 3.10 รูปแบบของส่วนจำเพาะของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 [6]

ซึ่งจากที่กล่าวมาทั้งหมดนั้นทำให้เห็นภาพได้ว่าการพัฒนาของโปรแกรมจำลองเครือข่ายทั้งสองเวอร์ชั่นนั้นแตกต่างกันออกไปโดยสิ้นเชิง เพราะการต้องการปรับเปลี่ยนระบบให้สอดคล้องกับการทำงานในระบบจริงมากขึ้น ดังนั้นผู้ใช้งานจึงต้องเข้าใจในการทำงานของโปรแกรมทั้งสองเพื่อความง่ายในการใช้งานโปรแกรมจำลองเครือข่ายในการพัฒนางานวิจัยของตน โดยสถาปัตยกรรมของแต่ละโปรแกรมเป็นดังภาพที่ 3.11 สำหรับโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 และภาพที่ 3.12 สำหรับโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3



ภาพที่ 3.11 สถาปัตยกรรมของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 [3]



ภาพที่ 3.12 สถาปัตยกรรมของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 [4]

3.4 เปรียบการพัฒนาโพรโทคอลบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 กับ เวอร์ชั่น 3

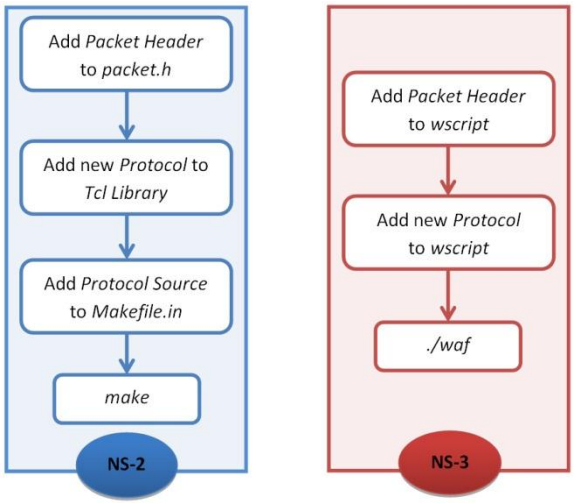
เมื่อศึกษาการทำงานของแต่ละโปรกรมทั้งโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 กับเวอร์ชั่น 3 และศึกษาความแตกต่างของโครงสร้างโปรแกรมจากหัวข้อที่แล้ว ทำให้พบว่าโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 สามารถทำงานได้ทั้งการจำลองเครือข่ายและการทำงานบนระบบจริง ซึ่งแตกต่างจากเวอร์ชั่น 2 ดังนั้นการเลือกใช้งานเวอร์ชั่น 3 ในการทำงานวิจัยจึงเป็นทางเลือกที่ดีกว่าที่ง่ายต่อการศึกษาการทำงานในระบบจริงของโพรโทคอล นอกจากนี้ยังมีงานวิจัย [5][6] ที่ได้ศึกษาการทำงานของโปรแกรมจำลองเครือข่ายแต่ละประเภท และพบว่าโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 นั้นทำงานได้อย่างมีประสิทธิภาพมากที่สุด ทั้งในเชิงความเร็วของการทำงาน ความง่ายในการใช้งานของโพรโทคอล และความสามารถในการจัดการหน่วยความจำที่เป็นระบบมากขึ้น

ดังนั้นก่อนที่จะทำการทดลองการทำงานของโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้บนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 จึงต้องทำการศึกษาความแตกต่างในการใช้งานโปรแกรมทั้งสองเวอร์ชั่นก่อน โดยในวิทยานิพนธ์นี้เน้นในการเปรียบเทียบความแตกต่างในสองส่วนหลักๆ คือ การเพิ่มโพรโทคอลใหม่บนโปรแกรม และโครงสร้างการทำงานของโปรแกรมอย่างกว้างของทั้งโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 และเวอร์ชั่น 3

การเพิ่มโพรโทคอลใหม่บนโปรแกรมจำลองเครือข่าย

สำหรับโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 การเพิ่มโพรโทคอลใหม่นั้นเริ่มต้นจากการเพิ่ม Packet Header ของโพรโทคอลใหม่ที่ packet.h จากนั้นทำการเพิ่มโพรโทคอลใหม่เข้าสู่ Tcl Library และเพิ่ม Packet Source ที่ Makefile.in แล้วทำการสั่งคำสั่ง make ก็เป็นการสิ้นสุดการเพิ่มโพรโทคอลใหม่บนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ดังภาพที่ 3.13 ด้านซ้าย

แตกต่างจากโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ที่มีขั้นตอนที่ต่างกันออกไป เริ่มจากการเพิ่ม Packet Header และโพรโทคอลใหม่ที่พัฒนาเข้าสู่ wscript ในแฟ้มข้อมูลของแต่ละส่วนของโปรแกรมนั้นๆ เพื่อทำให้วัตถุเหล่านั้นเชื่อมต่อเป็นส่วนหนึ่งกับ NS-3 Library หลังจากนั้นก็สั่ง ./waf ก็เสร็จการเพิ่มโพรโทคอล ขั้นตอนในความแตกต่างแสดงในภาพที่ 3.13 ด้านขวา



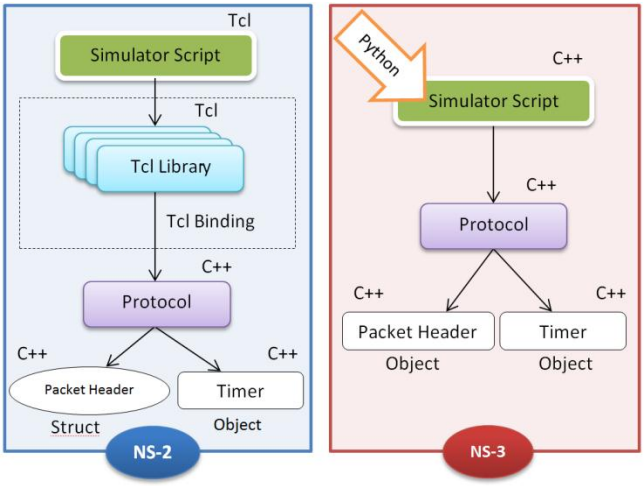
ภาพที่ 3.13 การเพิ่มโพรโทคอลใหม่บนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 และเวอร์ชั่น 3

โครงสร้างการทำงานของโปรแกรมอย่างกว้างของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 และเวอร์ชั่น 3

ในส่วนของความแตกต่างต่อมานั้นคือ การเปรียบเทียบโครงสร้างภาพรวมของการทำงานของทั้งสองโปรแกรม เริ่มจากบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 เมื่อทำการสั่งให้โปรแกรมทำงานทาง Tcl Script ก็จะมีการสร้าง Tcl Binding ไปสู่โพรโทคอลที่พัฒนาด้วยภาษา C++ โดยวิทยานิพนธ์นี้เน้นที่โพรโทคอลในชั้นเครือข่าย หลังจากนั้นโพรโทคอลก็ทำการเรียก Packet Header ที่เป็น Struct รวมทั้งการเรียกใช้ Timer ที่เป็นวัตถุได้ เป็นต้น

แต่สำหรับโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 นั้นหลังจากสั่งคำสั่ง ./waf ในส่วนของ Simulator Script ที่ถูกพัฒนาด้วย C++ ทั้งหมดก็จะเรียกใช้โพรโทคอลได้โดยตรง ไม่ต้องมี Tcl

Binding จากนั้นโพรโทคอลก็เรียกใช้ Packet Header และ Timer ที่เป็นวัตถุทั้งคู่ได้ตามปกติ ความแตกต่างของทั้งสองโปรแกรมนั้นแสดงในภาพที่ 3.14



ภาพที่ 3.14 โครงสร้างการทำงานของโปรแกรมอย่างกว้างของทั้งสองโปรแกรม

การสร้างรูปแบบการเคลื่อนที่ของยานพาหนะ

โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 นั้นสามารถเรียกใช้งานไฟล์รูปแบบการเคลื่อนที่ (Mobility trace file) ซึ่งเป็นไฟล์ที่สร้างรูปแบบการเคลื่อนที่ของยานพาหนะที่ใช้สำหรับโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ได้ โดยเรียกใช้งานฟังก์ชั่น NS2MobilityHelper ที่จะทำการเปลี่ยนรูปแบบของการเคลื่อนที่สำหรับโปรแกรมเวอร์ชั่น 2 ให้กลายเป็นรูปแบบของโปรแกรมเวอร์ชั่น 3 ซึ่งในเวอร์ชั่น 2 นั้นยานพาหนะหรือโหนดจะเคลื่อนที่ตามค่าเวกเตอร์ความเร็วที่ถูกกำหนดไว้ แต่เวอร์ชั่น 3 นั้นโหนดจะใช้ความแตกต่างของอัตราเร็วในแต่ละแกน x, y และ z แทนเวกเตอร์ความเร็ว เมื่อทำการทดสอบการทำงานของโพรโทคอลด้วยสภาพแวดล้อมรูปแบบถนนทางหลวง (Highway) โดยการเรียกใช้ไฟล์รูปแบบการเคลื่อนที่ดังกล่าว พบว่าประสิทธิภาพที่ออกมานั้นต่ำกว่าการทดลองบนโปรแกรมเวอร์ชั่น 2 ทุกรูปแบบ ซึ่งมีความผิดพลาดเกิดขึ้น 6-7 % และเมื่อทำการทดลองการเคลื่อนที่ของโหนดทำให้ทราบว่าการเคลื่อนที่ของโหนดในการทดลองบนโปรแกรมเวอร์ชั่น 3 นั้นแตกต่างจากการเคลื่อนที่ของโหนดในการทดลองบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ทั้งๆ ที่ใช้ไฟล์รูปแบบการเคลื่อนที่เดียวกัน ซึ่งตำแหน่งการเคลื่อนที่ที่แตกต่างกันนั้นมีระยะถึง 200 เมตร ทำให้เกิดปัญหาในการทดลองการทำงานของโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้ และเมื่อตรวจสอบอย่างละเอียดพบว่าปัญหานั้นเกิดจาก Ns2MobilityHelper ของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ที่มีการกำหนดให้มีการเคลื่อนที่ของโหนดนั้นมีการหยุดในช่วงเวลาสั้นๆ ที่สังเกตไม่เห็น ทำให้พฤติกรรมการ

เคลื่อนที่โดยรวมนั้นแตกต่างจากที่ควรเป็น ซึ่งการแก้ปัญหานี้จึงทำได้ด้วยการเปลี่ยนให้โหนดเคลื่อนที่อย่างต่อเนื่องไม่มีการหยุดในช่วงเวลาสั้นๆ ก่อนการเปลี่ยนทิศทาง ซึ่งทำการยกเลิกการหยุดใน Ns2MobilityHelper.cc บรรทัดที่ 614 ดังแสดงในภาพที่ 3.15

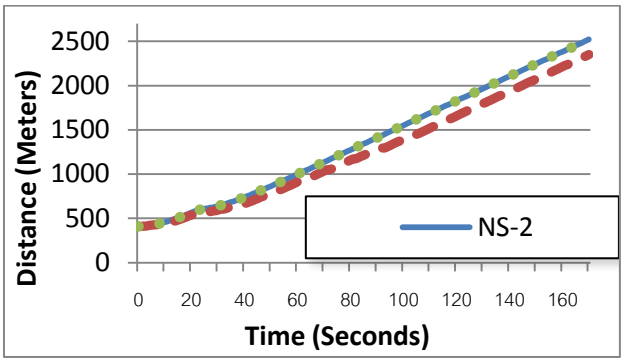
```
if (time >= 0)
   {
      Simulator::Schedule (Seconds (at + time),
      &ConstantVelocityMobilityModel::SetVelocity,model, Vector (0, 0, 0));
   }
```

ภาพที่ 3.15 ส่วนของโปรแกรมที่ทำให้เกิดการหยุดของโหนดในการเคลื่อนที่

ตารางที่ 3.1 ค่าความผิดพลาดในการเคลื่อนที่เฉลี่ย

	ความหนาแน่นของยานพาหนะ (คัน/กิโลเมตร)				
	2	10	30	60	120
NS-3 ต้นแบบ Ns2MobilityHelper	6.6853	6.1845	6.1958	7.8375	7.4393
NS-3 ปรับปรุง Ns2MobilityHelper	0.0013	0.0009	0.0019	0.0036	0.3843

เพื่อทดลองการทำงานของโพรโทคอลหลังจากแก้ไขเรื่องการเคลื่อนที่จึงใช้โปรแกรมจำลองการจราจร Simulation of Urban and Mobility (SUMO) [14] ในการสร้างสถานการณ์ในการเคลื่อนที่ของโหนดและเปลี่ยนเป็นรูปแบบที่โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ใช้งานด้วยโปรแกรม Traffic and Network Simulation Environment (TraNS) [15] โดยใช้สภาพแวดล้อมรูปแบบถนนทางหลวง ซึ่งเป็นเส้นทาง 2 เลน ความยาว 4 กิโลเมตร และแสดงการเคลื่อนที่ของทุกโหนดบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 และเวอร์ชั่น 3 ด้วยความหนาแน่น 30 ยานพาหนะต่อกิโลเมตร ดังภาพที่ 3.16 ส่วนตารางที่ 3.1 แสดงค่าเฉลี่ยในการเคลื่อนที่ของทุกโหนดเปรียบเทียบระหว่างโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 กับเวอร์ชั่น 3 ซึ่งแสดงค่าว่าเมื่อทำการปรับเปลี่ยนการเคลื่อนที่ให้ไม่มีการหยุดในการเคลื่อนที่แล้ว ค่าผิดพลาดที่เกิดขึ้นมีน้อยมากทำให้การเคลื่อนที่ใกล้เคียงกับการเคลื่อนที่บนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2



ภาพที่ 3.16 การเคลื่อนที่ของโหนดในโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 และเวอร์ชั่น 3

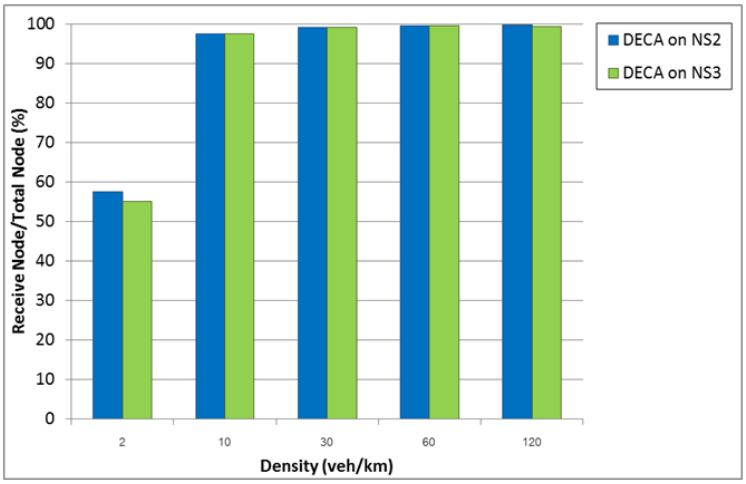
การเปรียบเทียบการทำงานของโพรโทคอลบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 และเวอร์ชั่น 3

หลังจากที่การทำงานของโพรโทคอลบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 สามารถเรียกใช้ไฟล์รูปแบบการเคลื่อนที่ได้แล้ว เพื่อเปรียบเทียบประสิทธิภาพการทำงานของโพรโทคอลบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 และเวอร์ชั่น 3 จึงทำให้การทดลองโดยใช้โพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้ ซึ่งในวิทยานิพนธ์นี้ได้นำเอาโพรโทคอล DECA ที่เคยมีการทดลองอยู่แล้วบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 มาเป็นตัวอย่างในการศึกษา เพราะว่าโพรโทคอล DECA นั้นมีลักษณะการทำงานที่ง่าย เร็ว และลดความซ้ำซ้อนของข้อมูล ให้ทำงานบนสภาพแวดล้อมรูปแบบถนนทางหลวง ที่สร้างโดย SUMO และเปลี่ยนให้เป็นรูปแบบที่ใช้งานกับโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ด้วย TraNS เป็นตัวกำหนดการเคลื่อนที่ของโหนดในการทดลอง การทดลองนี้ทำงานบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ที่ปรับเปลี่ยนการเคลื่อนที่ของโหนดแล้วให้ไม่มีการหยุด เพื่อการทำงานจะได้มีประสิทธิภาพใกล้เคียงกับการใช้งานบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 โดยแต่ละโหนดมีระยะในการส่งข้อมูล 250 เมตร โดยมีการตั้งค่าพารามิเตอร์ต่างๆ ดังตารางที่ 3.2

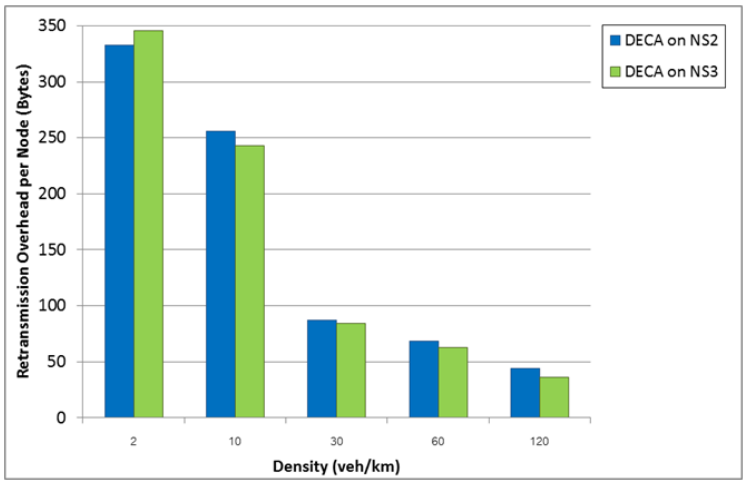
ตารางที่ 3.2 การตั้งค่าพารามิเตอร์ทดสอบประสิทธิภาพโพรโทคอล DECA

อายุของข้อความ	10 วินาที
จำนวนข้อความ	5
ขนาดของข้อความ	512 ไบต์
เวลาในการทดลอง	50 วินาที
ความเร็วสูงสุดของยานพาหนะ (กิโลเมตร/ชั่วโมง)	50, 80
ความหนาแน่นของยานพาหนะ(คัน/กิโลเมตร)	ถนนทางหลวง (Highway) 2, 10, 30, 60, 120
โปรแกรมจำลองเครือข่าย	NS-2.34, NS-3.10

การทำงานนั้นโหนดจะเริ่มต้นส่งข้อมูลทุกๆ 10 วินาที จำนวน 5 ข้อมูล ผลลัพธ์ที่ได้แสดงดังภาพที่ 3.17 เป็นค่าความเชื่อถือได้ (Reliability) ระหว่างการทดสอบโพรโทคอล DECA บนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 กับเวอร์ชั่น 3 โดยในแกน x แทนความหนาแน่นของยานพาหนะต่อกิโลเมตร และแกน y แทนค่าความเชื่อถือได้เป็นเปอร์เซนต์ ซึ่งพบว่าผลที่ออกมามีค่าใกล้เคียงกัน แตกต่างกันเพียงแค่ในสถานการณ์ที่มีความหนาแน่นของยานพาหนะที่ 2 คันต่อกิโลเมตร แต่ว่าค่ายังอยู่ในเกณฑ์ที่ยอมรับได้ ส่วนภาพที่ 3.18 แสดงผลค่าใช้จ่ายของโพรโทคอลทั้งสองโปรแกรม แกน x นั้นยังแทนความหนาแน่นของยานพาหนะ ส่วนแกน y นั้นแทนค่าใช้จ่ายของโพรโทคอล ผลลัพธ์ที่ได้ออกมานั้นมีค่าใกล้เคียงกันทั้งสองโปรแกรม ดังนั้นสรุปว่าประสิทธิภาพการทำงานของโพรโทคอลบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 กับเวอร์ชั่น 3 นั้นมีค่าเท่ากัน



ภาพที่ 3.17 ค่าความเชื่อถือได้



ภาพที่ 3.18 ค่าใช้จ่าย

บทที่ 4

การปรับปรุงการแพร่ข้อมูลแบบเชื่อถือได้บนเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะที่เชื่อมต่อแบบอสมมาตร

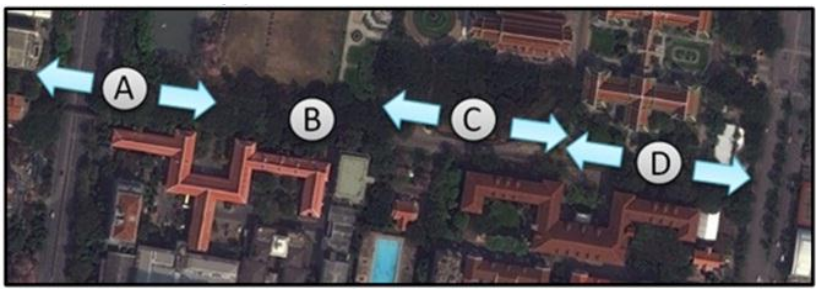
การปรับปรุงการแพร่ข้อมูลแบบเชื่อถือได้บนเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะที่เชื่อมต่อแบบอสมมาตรนั้นถูกออกแบบเพื่อปรับปรุงประสิทธิภาพของโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้ ซึ่งเป็นโพรโทคอลที่ออกแบบมาเพื่อเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะ ในวิทยานิพนธ์นี้นั้นเลือกโพรโทคอล DECA ซึ่งเป็นโพรโทคอลที่ทำงานง่าย รวดเร็ว และลดความซ้ำซ้อนของข้อมูล มาเป็นโพรโทคอลตัวอย่างในการวิจัย เพื่อหากระบวนการในการแก้ไขปัญหาการทำงานของโพรโทคอลบนสถานการณ์ที่มีการเชื่อมต่อแบบอสมมาตร

4.1 แนวคิดในการออกแบบ

จากการทดลองการทำงานบนระบบจริงของโพรโทคอล DECA ด้วยโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ดังภาพที่ 4.1 ซึ่งจัดวางคอมพิวเตอร์พกพาจำนวนสี่เครื่องบนตำแหน่งที่แตกต่างกันภายในจุฬาลงกรณ์มหาวิทยาลัยแล้วกำหนดสถานการณ์สองแบบ อย่างแรกคือให้ทุกเครื่องอยู่นิ่งในตำแหน่งของแต่ละเครื่อง แล้วให้ทุกเครื่องทำงานโพรโทคอล DECA พร้อมกันและส่งข้อมูลหากันและกัน กับอีกแบบคือให้เครื่องจำนวนสามเครื่องเคลื่อนที่ไปในทิศทางที่กำหนด แต่อีกเครื่องหนึ่งยังอยู่นิ่งแล้วทำการส่งข้อมูลหากัน จากการทดลองพบว่าผลลัพธ์ของการทำงานของโพรโทคอลเป็นดังนี้ ค่าความเชื่อถือได้ออกมาเป็น 100 % เพราะว่าทุกเครื่องได้รับข้อมูลที่ส่งออกมาจากผู้ส่งข้อมูลเหมือนกันหมด แต่ว่าค่าใช้จ่ายในการทำงานของโพรโทคอลนั้นมีค่าสูงมาก ซึ่งแตกต่างจากการทดลองบนการจำลองเครือข่ายถึง 5 เท่า ทำให้เกิดปัญหาขึ้นในการทำงานของโพรโทคอลบนระบบจริง หลังจากนั้นได้ทำการทดลองและตรวจสอบถึงผลที่เกิดขึ้นทำให้พบว่าเกิดปัญหาขึ้นในการทำงานของโพรโทคอลบนระบบเครือข่ายจริง ปัญหาที่ว่าก็คือปัญหาการเชื่อมต่อแบบอสมมาตร เพราะว่าในการทดลองนั้นคอมพิวเตอร์พกพาแต่ละเครื่องมีความสามารถในการส่งข้อมูลของอุปกรณ์รับส่งสัญญาณไร้สายที่แตกต่างกัน ทำให้บางเครื่องมีความแรงของสัญญาณสูงซึ่งสามารถส่งข้อมูลได้ไกล แต่บางเครื่องมีความแรงของสัญญาณต่ำทำให้ไม่สามารถส่งข้อมูลไปยังเครื่องอื่นได้ทำได้เพียงรับข้อมูลจากเครื่องอื่น ซึ่งสาเหตุนี้ส่งผลกับการทำงานของโพรโทคอล DECA อย่างมาก เพราะว่าเครื่องที่มีความแรงของสัญญาณต่ำ ก็เปรียบเสมือนรถยนต์ที่อยู่ในสถานการณ์ที่กำหนดขึ้นและมีความแรงของสัญญาณที่น้อย ทำให้การส่งข้อมูลไปยังผู้อื่นทำได้ยากทำได้แต่เพียงรับข้อมูลมาจากผู้ที่สัญญาณแรงและส่งมาถึง ในเมื่อโหนดนี้ได้รับข้อมูลมาแล้วไม่สามารถทำการส่งข้อมูลต่อได้ ทำให้เครื่องที่ควรได้รับข้อมูลแต่ไม่ได้รับก็ส่ง

Beacon message ออกมาว่ายังไม่ได้รับข้อมูลนั้นๆ เมื่อส่งออกมาเรื่อยๆ ประกอบกับเครื่องที่มีความแรงของสัญญาณต่ำพยายามส่งข้อมูลนั้นออกไปให้ ก็ทำให้ค่าใช้จ่ายในการทำงานของระบบมีมากขึ้น เกิดการชนกันของข้อมูลมากขึ้น ส่งผลให้ประสิทธิภาพการทำงานของโพรโทคอลต่ำลง ดังนั้นถ้าระบบมีจำนวนยานพาหนะที่มากขึ้นก็ทำให้ค่าความเชื่อถือได้ก็ตกลงเช่นกัน

(ก) รูปแบบโหนดอยู่นิ่ง



(ข) รูปแบบโหนดเคลื่อนที่

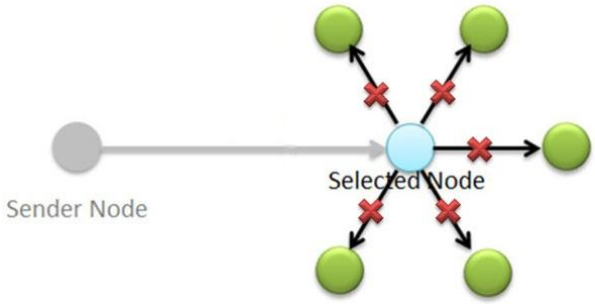
ภาพที่ 4.1 การทดสอบระบบจริงในบริเวณจุฬาลงกรณ์มหาวิทยาลัย

จากปัญหาที่เกิดขึ้นดังกล่าวทำให้ทราบว่าการทำงานของโพรโทคอล DECA นั้นไม่สามารถทำงานได้ด้วยเหตุผลสองอย่าง คือ เหตุผลแรกการส่งข้อมูลนั้นส่งไม่สำเร็จ เพราะในสถานการณ์การเชื่อมต่อแบบอสมมาตรนั้นนอกจากปัญหาจะเกิดจากปัจจัยภายในคือความสามารถของอุปกรณ์รับส่งสัญญาณไร้สายที่มีความแรงของสัญญาณต่ำแล้ว ยังเกิดจากปัจจัยภายนอกด้วย อย่างเช่น สัญญาณที่ส่งออกไปนั้นอาจจะถูกเบี่ยงเบน แทรกซอด หรือดูดซับ ทำให้การส่งข้อมูลนั้นทำได้ไม่สำเร็จ ส่วนเหตุผลที่สองที่ทำให้โพรโทคอลทำงานแย่ลงคือ การเลือกผู้กระจายข้อมูลลำดับถัดไปที่ไม่มีประสิทธิภาพเพียงพอ เพราะผู้ถูกเลือกนั้นอาจเป็นโหนดที่มีความแรงของสัญญาณต่ำทำให้การส่งข้อมูลไปให้กับเพื่อนบ้านโดยรอบนั้นไม่สำเร็จ ดังภาพที่ 4.2 ส่งผลให้เกิดปัญหาการไม่ได้รับข้อมูล และค่าใช้จ่ายในระบบที่สูงเกินไป

ดังนั้นจากที่กล่าวไปทำให้ทราบว่าในสถานการณ์การเชื่อมต่อแบบอสมมาตรการที่ผู้ส่งจะส่งข้อมูลสำเร็จ 100 % นั้นเป็นไปได้ยาก เพราะความแรงของสัญญาณในการส่งนั้นมีความแตกต่างกันไป

ในแต่ละโหนด หรืออาจเกิดขึ้นเพราะปัจจัยภายนอก ดังนั้นการแก้ไขปัญหาให้โหนดสามารถส่งข้อมูลได้สำเร็จ 100 % นั้นเป็นไปได้ยากในสถานการณ์การเชื่อมต่อแบบอสมมาตร วิทยานิพนธ์นี้จึงเน้นในการพัฒนากระบวนการในการแก้ไขการเลือกผู้กระจายข้อมูลลำดับถัดไป เพราะว่าการเลือกโหนดที่มีความแรงของสัญญาณสูงทำให้มีระยะในการส่งข้อมูลไกลและครอบคลุมการเชื่อมต่อกับเพื่อนบ้านมากที่สุด จะทำให้การส่งข้อมูลให้ทุกโหนดนั้นเป็นไปได้มาก

(ก) ผู้ส่งข้อมูลผิดพลาด



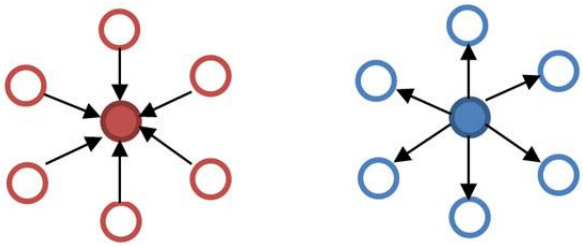
(ข) ผู้ถูกเลือกในการกระจายข้อมูลผิดพลาด

ภาพที่ 4.2 ความผิดพลาดที่เกิดขึ้นในรูปแบบการเชื่อมต่อแบบอสมมาตร

กระบวนการในการเลือกผู้กระจายข้อมูลลำดับถัดไปนั้นมีแนวคิดในการพัฒนามาจากหลักการทำงานเดิมของโพรโทคอล DECA ในการเลือกโหนดดังภาพที่ 4.3(ก) การเลือกโหนดของ DECA เดิมนั้นมีหลักการเลือกโดยเลือกโหนดที่ได้รับ Beacon message จากเพื่อนบ้านหลายโหนดมากที่สุด ซึ่งจะถือว่าโหนดนั้นมีความหนาแน่นของเพื่อนบ้านมากที่สุดทำให้โหนดนั้นถูกเลือกให้เป็นผู้กระจายข้อมูลลำดับถัดไป แต่สิ่งนั้นไม่สามารถยืนยันได้ว่าโหนดที่ถูกเลือกจะสามารถส่งข้อมูลไปให้กับเพื่อนบ้านที่เชื่อมต่อได้สำเร็จ เพราะว่าโหนดนั้นทำได้แค่เพียงรับ beacon มาจากเพื่อนบ้านแต่อาจจะมีความแรงของสัญญาณต่ำ ทำให้การส่งข้อมูลไปไม่ถึงเพื่อนบ้านเหล่านั้น ด้วยเหตุนี้ทำให้ DECA ไม่สามารถทำงานได้อย่างมีประสิทธิภาพในสถานการณ์การเชื่อมต่อแบบอสมมาตร

แต่หากมองในมุมตรงข้าม การเลือกโหนดที่สามารถส่งข้อมูลให้กับเพื่อนบ้านได้จริง จึงเป็นทางเลือกที่ดีกว่าในการทำงานของโพรโทคอล ดังภาพที่ 4.3 (ข) โดยมีหลักการว่าโหนดที่สามารถส่ง beacon ให้กับเพื่อนบ้านได้มากที่สุดก็ถือว่าโหนดนั้นจะสามารถส่งข้อมูลไปให้กับเพื่อนบ้านได้มากที่สุดเช่นกัน ซึ่งการทำงานคือโหนดนั้นจะส่ง beacon ไปให้กับเพื่อนบ้านโดยรอบ และหากเพื่อนบ้านส่วนใหญ่ในละแวกนั้นทำการโหวตออกมาว่าโหนดที่ส่ง beacon มานั้นมีความแรงของสัญญาณมาก

หากได้รับผลโหวตมากที่สุดก็ทำให้โหนดนั้นได้รับการเลือกเป็นผู้ถูกเลือกในการกระจายข้อมูลลำดับถัดไป



(ก) DECA แบบเดิม (ข) กระบวนการเลือกแบบใหม่
(ลูกศรแทนทิศทางการส่ง Beacon สำเร็จ)

ภาพที่ 4.3 กระบวนการในการเลือกโหนด

นอกจากกระบวนการโหวตของเพื่อนบ้านแล้วการแก้ไขปัญหาในการเลือกของโพรโทคอลนั้นยังใช้งานควบคู่กับการค่าวัดความแรงสัญญาณที่ได้รับหรือค่า RSSI โดยค่านี้จะเป็นตัวแปรหลักที่ใช้ในการเลือกผู้ส่งสัญญาณที่แรงที่สุดเพื่อที่จะเป็นผู้กระจายข้อมูลลำดับถัดไปที่ใช้ควบคู่กับการโหวตอีกด้วย จึงได้ออกมาเป็นกระบวนการเลือกผู้กระจายลำดับถัดไปที่เรียกว่า อัลกอริทึมการโหวตแบบอาร์เอสเอสไอ (RVA : RSSI-Voting algorithm)

4.2 หลักการทำงาน

การทำงานของกระบวนการในการพัฒนาให้โพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้นั้นได้เน้นไปที่การแก้ไขกระบวนการเลือกผู้กระจายข้อมูลลำดับถัดไปดังที่กล่าวไว้ก่อนหน้านี้ โดยกระบวนการอื่นๆ ในการทำงานของโพรโทคอลนั้นยังคงการทำงานในส่วนหลักๆ ไว้ อย่างเช่น การแลกเปลี่ยน beacon message เพื่อปรับข้อมูลของเพื่อนบ้านให้ทันสมัย การนับเวลาของแต่ละข้อมูลที่ส่งเพื่อให้ทันสมัย เป็นต้น

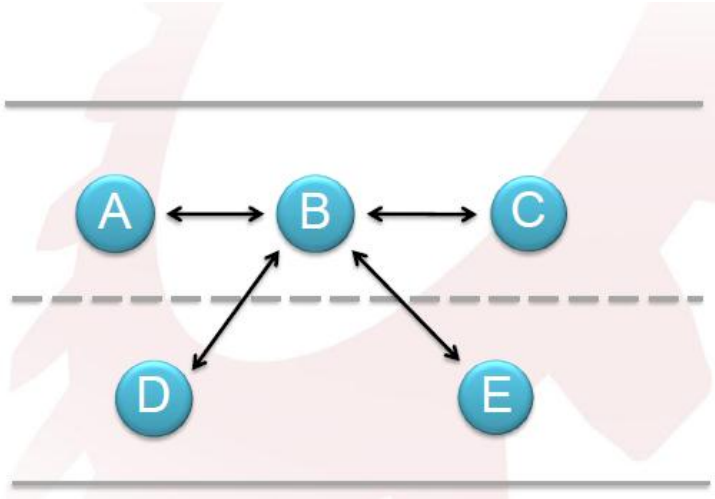
โดยในวิทยานิพนธ์นี้ได้นำเอาโพรโทคอล DECA มาเป็นตัวอย่างในการวิจัย ซึ่งส่วนของการทำงานหลักของโพรโทคอลนั้นก็ยังคงการทำงานของโพรโทคอล DECA ไว้ ส่วนที่แก้ไขคือกระบวนการเลือกผู้กระจายข้อมูลลำดับถัดไป

4.2.1 หลักการทำงานของกระบวนการที่แก้ไขกับโพรโทคอล DECA

1. แต่ละโหนดทำการโหวตเพื่อนบ้านของตนว่าโหนดไหนมีค่า RSSI มากที่สุด (RSSI ได้มาจากการแลกเปลี่ยน Beacon ครั้งก่อนหน้า) แล้วแนบผลโหวตไปกับ Beacon โดยหนึ่งโหนดสามารถโหวตได้แค่ 1 โหนดเท่านั้น เพื่อทำให้ข้อมูลที่เพิ่มเข้าไปใน Beacon ไม่เยอะเกินไป
2. เมื่อโหนดได้รับ Beacon มาจากเพื่อนบ้านก็จะเก็บข้อมูลของเพื่อนบ้านพร้อมผลโหวตไว้ในตารางเพื่อนบ้านของตน
3. หากโหนดใดต้องการส่งข้อมูลก็จะทำการนับว่าเพื่อนบ้านในตารางเพื่อนบ้านของตนนั้นโหวตโหนดไหนมากที่สุดและโหนดที่ได้รับการโหวตนั้นเป็นเพื่อนบ้านของตน ก็จะเลือกโหนดที่ได้รับการโหวตมากที่สุดให้เป็นผู้ถูกเลือกต่อไป

4.2.2 ตัวอย่างในการทำงาน

1. ขั้นตอนแรกนั้นคือกระบวนการที่แต่ละโหนดจะแลกเปลี่ยน Beacon ของตนเองกับเพื่อนบ้าน ตัวอย่างด้านล่างในภาพที่ 4.4 เป็นตัวอย่างตารางเพื่อนบ้านของโหนด A ซึ่งพบว่าเพื่อนบ้านของ A ประกอบด้วย B กับ D ซึ่ง A พบว่า D มีค่า RSSI มากที่สุดจากการแลกเปลี่ยน Beacon ครั้งก่อนหน้านี้ ทำให้ A เลือกที่จะโหวต D แนบไปกับการ Beacon ครั้งต่อไป



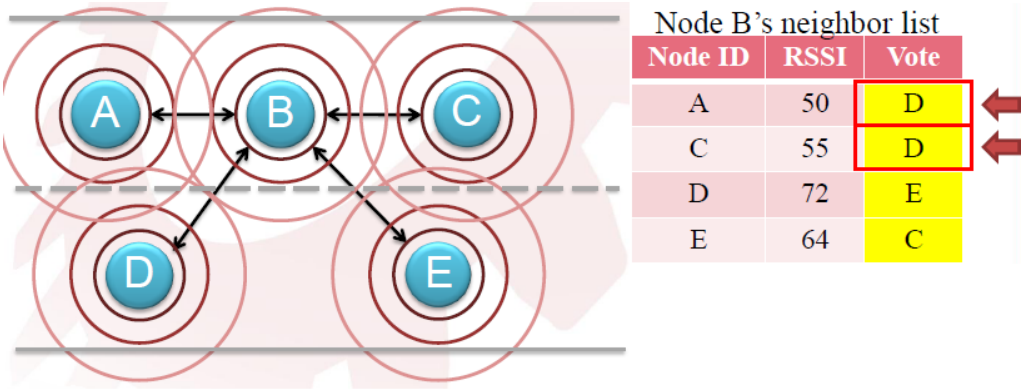
Node A's neighbor list

Node ID	RSSI
B	60
D	71

ภาพที่ 4.4 ขั้นตอนที่หนึ่งของอัลกอริทึมการโหวตแบบอาร์เอสเอสไอ

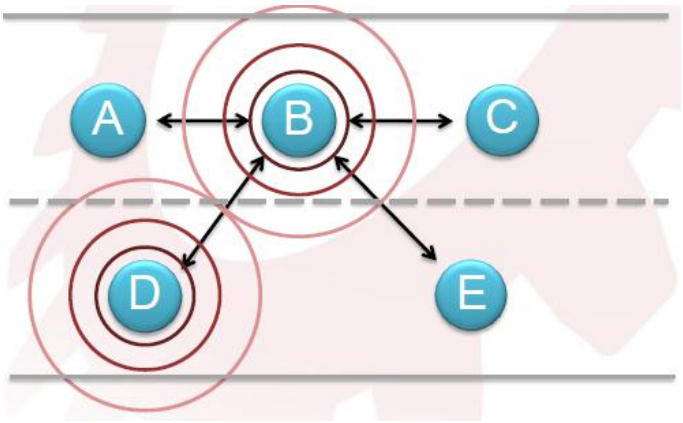
2. จากนั้นทุกโหนดทำการแลกเปลี่ยน Beacon กัน และแก้ไขข้อมูลของเพื่อนบ้านของตนในตารางเพื่อนบ้านให้ทันสมัย จากตัวอย่างด้านล่างในภาพที่ 4.5 เป็นตัวอย่างของตารางเพื่อน

บ้านของโหนด B ซึ่งได้รับข้อมูลเพื่อนบ้านของตนในละแวกมาและแก้ไขให้ทันสมัยแล้ว ซึ่งในตารางจะมีการระบุค่า RSSI ของเพื่อนบ้านแต่ละโหนด นอกจากนั้นยังมีข้อมูลว่าเพื่อนบ้านของตนนั้นทำการโหวตโหนดใดว่ามีความแรงของสัญญาณมากที่สุด ซึ่ง B พบว่าโหนด D นั้นมีผลโหวตจากเพื่อนบ้านมากที่สุดถึง 50 % ของการโหวตจากเพื่อนบ้านทั้งหมด



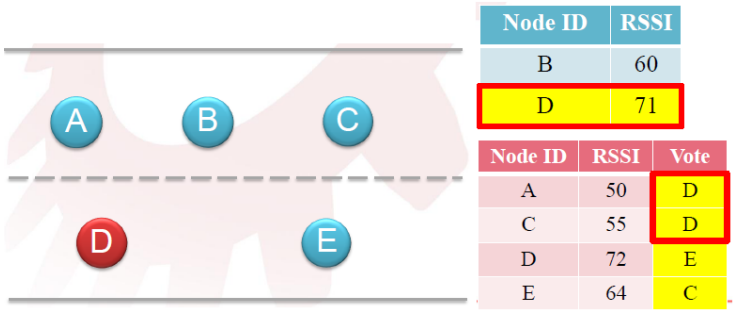
ภาพที่ 4.5 ขั้นตอนที่สองของอัลกอริทึมการโหวตแบบอาร์เอสเอสไอ

3. ดังนั้นหากโหนด B ต้องการกระจายข้อมูลไปให้กับเพื่อนบ้านทั้งหมด ก็จะทำการเลือกให้โหนด D เป็นผู้กระจายข้อมูลลำดับถัดไป ดังตัวอย่างด้านล่างต่อไปนี้ในภาพที่ 4.6



ภาพที่ 4.6 ขั้นตอนที่สามของอัลกอริทึมการโหวตแบบอาร์เอสเอสไอ

ดังนั้นสรุปได้ว่าโหนดที่มีความแรงของสัญญาณของผู้ส่งมากที่สุด (ค่า RSSI มากที่สุด) และได้รับการโหวตจากเพื่อนบ้านมากที่สุดจะถูกเลือกให้เป็นผู้กระจายข้อมูลในลำดับถัดไป ดังตัวอย่างด้านล่างต่อไปนี้ดังภาพที่ 4.7

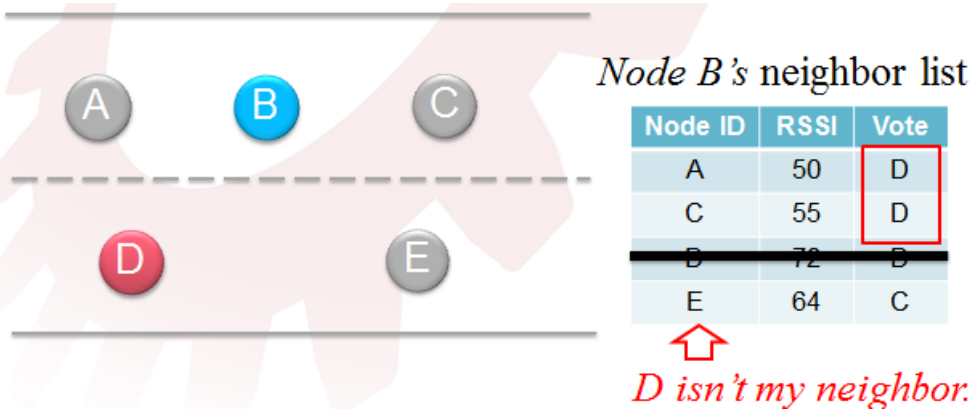


ภาพที่ 4.7 สรุปการเลือกโหนดของอัลกอริทึมการโหวตแบบอาร์เอสเอสไอ

4.3 ปัญหาที่เกิดขึ้นจากอัลกอริทึมการโหวตอาร์เอสเอสไอ และวิธีแก้ไข

4.3.1 โหนดที่ได้รับผลโหวตอันดับหนึ่งไม่ใช่เพื่อนบ้านของผู้ส่งข้อมูล

จากการทดลองบนสถานการณ์ที่มีจำนวนยานพาหนะมากพบว่าโหนดที่ถูกเลือกนั้นจะได้รับการโหวต 50-80 % โดยประมาณจากการโหวตของเพื่อนบ้านโดยรอบทั้งหมด ดังนั้นถ้าโหนดที่ได้รับการโหวตนี้ไม่ได้เป็นเพื่อนบ้านกับผู้ส่งข้อมูล ดังภาพที่ 4.8 ซึ่งโหนด D เป็นโหนดที่ได้รับการโหวตมากที่สุด แต่โหนด D นั้นไม่ได้เป็นเพื่อนบ้านกับโหนด B ซึ่งเป็นผู้ส่งข้อมูล ทำให้โหนดอื่นที่ได้รับการโหวตในอันดับต่อๆ มาซึ่งได้รับการโหวตจากเพื่อนบ้านโดยรอบเพียงต่ำสุดแค่ 20 % โดยประมาณ นั้นไม่สามารถทำงานในส่วนของการเป็นผู้กระจายข้อมูลลำดับถัดไปแทนได้ เพราะว่าโหนดนั้นอาจจะมีความแรงของสัญญาณต่ำและครอบคลุมเพื่อนบ้านน้อย ทำให้การส่งข้อมูลไปให้กับเพื่อนบ้านนั้นจะสำเร็จน้อยและเพื่อนบ้านได้รับข้อมูลไม่ทั่วถึง นอกจากนั้นมีการทดลองพบว่าการเลือกให้สามารถใช้โหนดที่ได้รับการโหวตลำดับรองจากลำดับหนึ่งในการเป็นผู้ถูกเลือกได้แทนนั้นเกิดปัญหาคำตอบที่ดีที่สุดเฉพาะที่ (Local Optimum) ทำให้เกิดการกระจุกตัวของข้อมูลเฉพาะที่ ส่งผลให้ข้อมูลนั้นไม่กระจายไปทั่วระบบซึ่งโหนดอื่นๆ นั้นจะไม่ได้รับข้อมูลส่งผลให้ประสิทธิภาพในการทำงานของโพรโทคอลต่ำลง



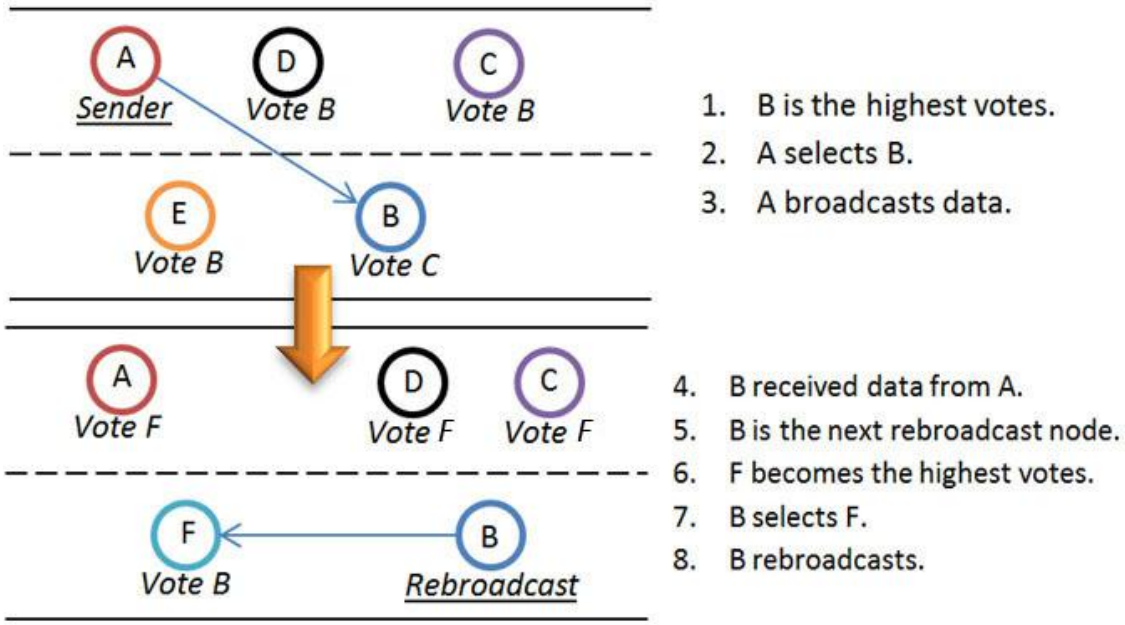
ภาพที่ 4.8 ปัญหาโหนดที่ได้รับการโหวตอันดับหนึ่งไม่ได้เป็นเพื่อนบ้านกับผู้ส่งข้อมูล

การแก้ไขปัญหาดังกล่าวนี้ทำโดยให้ผู้ส่งข้อมูลนั้นกระจายข้อมูลออกไปโดยไม่มีการเลือกผู้กระจายข้อมูลลำดับถัดไปเพื่อให้เพื่อนบ้านที่ได้รับข้อมูลนี้ทำการนับเวลาถอยหลังแบบสุ่ม แล้วเมื่อเพื่อนบ้านที่ทำการนับเวลาหมดก่อนก็จะทำการกระจายข้อมูลลำดับถัดไปโดยจะทำการเลือกว่าจะให้เพื่อนบ้านโหนดใดเป็นผู้กระจายข้อมูลลำดับถัดไปด้วย โดยกระบวนการนี้จะทำให้การทำงานกลับไปสู่กระบวนการของโพรโทคอลตามปกติ ทำให้ข้อมูลนั้นสามารถกระจายไปทั้งระบบอย่างทั่วถึงมากขึ้น

4.3.2 การเลือกตัวเองเป็นผู้กระจายข้อมูลลำดับถัดไป

แบบโครงสร้าง (Topology) ของสถานการณ์การเคลื่อนที่ของยานพาหนะในการทดลองวิทยานิพนธ์นี้นั้นสามารถแบบออกเป็นสองโครงสร้าง คือ แบบโครงสร้างที่เปลี่ยนแปลงเร็ว กับแบบโครงสร้างที่เปลี่ยนแปลงช้า ซึ่งโครงสร้างแบบหลังนี้จะทำให้เกิดปัญหาการเลือกตัวเองเป็นผู้กระจายข้อมูลลำดับถัดไปส่งผลให้การทำงานของโพรโทคอลนั้นมีประสิทธิภาพที่ต่ำลง โดยทั้งสองโครงสร้างมีรายละเอียดดังนี้

4.3.2.1 แบบโครงสร้างที่เปลี่ยนแปลงเร็ว (Fast-changing topology)



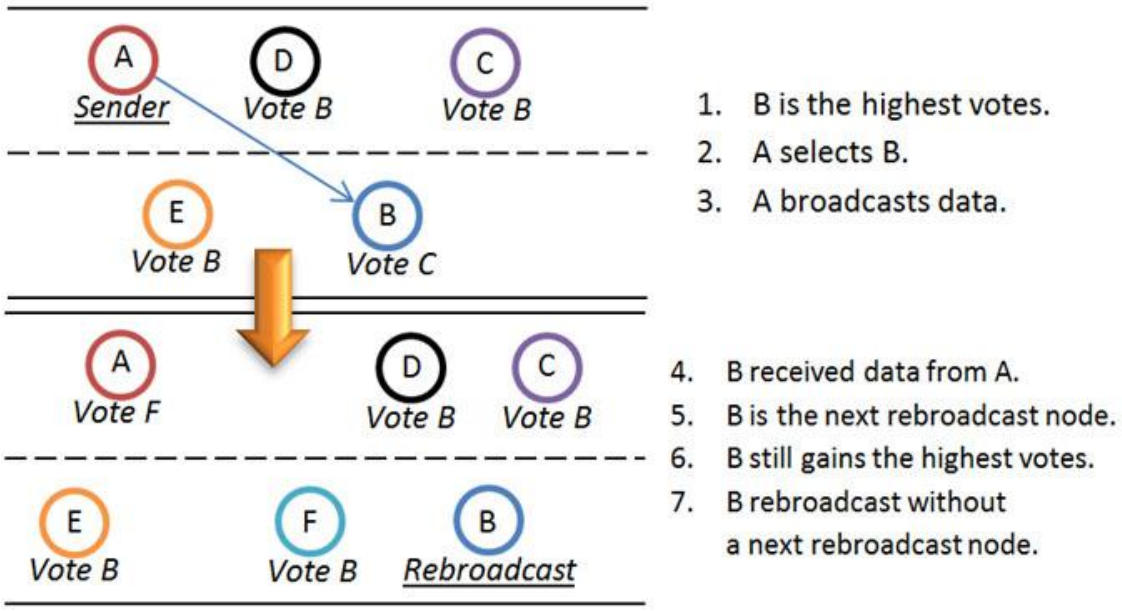
ภาพที่ 4.9 แบบโครงสร้างที่เปลี่ยนแปลงเร็ว

แบบโครงสร้างนี้เกิดจากการที่ยานพาหนะนั้นเคลื่อนที่ด้วยความเร็วสูง หรือมีความหนาแน่นของยานพาหนะที่ต่ำ ทำให้ตำแหน่งของยานพาหนะนั้นจะมีการเปลี่ยนแปลงที่เร็วเคลื่อนที่ไปยังตำแหน่งอื่นๆ ต่างจากจุดเดิม ดังภาพที่ 4.9 ซึ่งโหนด A นั้นเป็นผู้ส่งข้อมูลแล้ว A พบว่าโหนด B ได้รับการโหวตจากเพื่อนบ้านมากที่สุด ต่อมา A จึงกระจายข้อมูลไปให้กับเพื่อนบ้านและเลือก B ให้เป็นผู้

กระจายข้อมูลลำดับถัดไป ซึ่งในขณะนั้นทุกโหนดก็มีการเคลื่อนที่ทำให้เกิดการเปลี่ยนตำแหน่งของโหนด หลังจากที่ B ได้รับข้อมูลจาก A แล้ว B ก็จะทำการกระจายข้อมูลต่อ โดย B พบว่าผลของการโหวตจากเพื่อนบ้านนั้นได้เปลี่ยนแปลงแล้ว ซึ่ง F ได้รับการโหวตจากเพื่อนบ้านในละแวกให้เป็นผู้กระจายข้อมูลลำดับถัดไป ดังนั้น B จึงกระจายข้อมูลโดยมี F กระจายข้อมูลต่อ

แบบโครงสร้างนี้นั้นพบว่าการทำงานของโพรโทคอลจะเป็นปกติ ลำดับในการกระจายข้อมูลจะเปลี่ยนไปเรื่อยในแต่ละโหนด ทำให้ประสิทธิภาพในการทำงานของโพรโทคอลนั้นดี

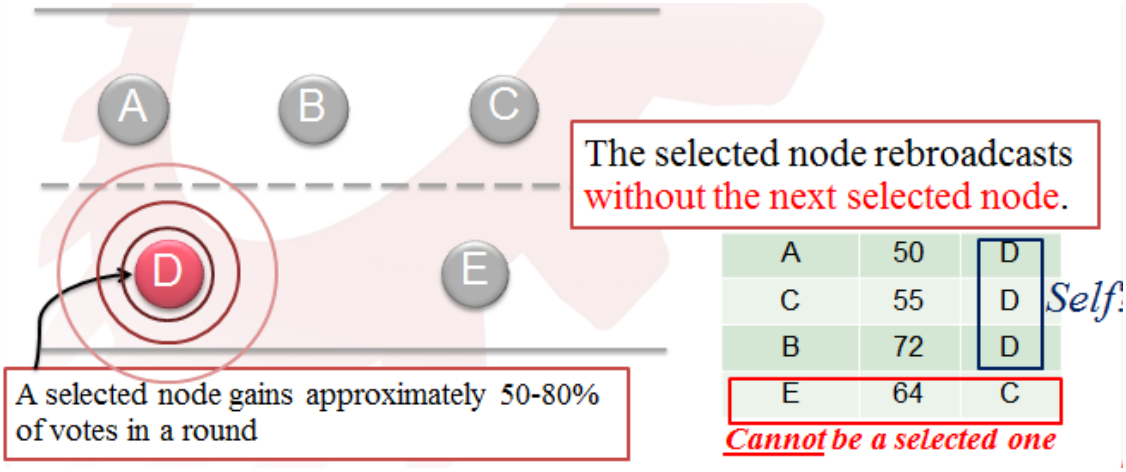
4.3.2.2 แบบโครงสร้างที่เปลี่ยนแปลงช้า (Slow-changing topology)



ภาพที่ 4.10 แบบโครงสร้างที่เปลี่ยนแปลงช้า

แบบโครงสร้างนี้เกิดจากการที่ยานพาหนะนั้นมีการเคลื่อนที่ช้า หรือมีความหนาแน่นของยานพาหนะสูง ทำให้ตำแหน่งของยานพาหนะนั้นแทบจะไม่เปลี่ยนแปลงไปจากเดิมแตกต่างจากแบบโครงสร้างที่กล่าวไปก่อนหน้า ซึ่งส่งผลให้เกิดปัญหาในการทำงานของโพรโทคอล ปัญหาดังกล่าวคือปัญหาการเลือกตัวเองเป็นผู้กระจายข้อมูลลำดับถัดไป ดังภาพที่ 4.10 โหนด A นั้นพบว่าโหนด B ได้รับการโหวตมากที่สุด และ A ทำการเลือก B ให้เป็นผู้กระจายข้อมูลลำดับถัดไป ซึ่งการทำงานในส่วนนี้ยังเป็นปกติ แต่การเคลื่อนที่ของโหนดช้า หรือมีความหนาแน่นของโหนดสูง ทำให้ตำแหน่งของโหนดไม่เกิดการเปลี่ยนแปลงมากนัก เมื่อ B ได้รับข้อมูลจาก A แล้ว B ก็ยังพบว่าเพื่อนบ้านในระแวกของตนเองนั้นยังคงเลือก B เองให้เป็นผู้กระจายข้อมูลลำดับถัดไป ทำให้ B พบปัญหาว่าตัวเองนั้นไม่อยู่ในตารางเพื่อนบ้านของตน ซึ่งปัญหานี้ก็คล้ายคลึงกับปัญหาก่อนหน้า เพราะการเลือกผู้ถูกเลือกลำดับรองลงมานั้นส่งผลให้การกระจายข้อมูลไม่มีประสิทธิภาพและเกิดปัญหาคำตอบที่ดีที่สุดเฉพาะที่

(Local Optimum) ทำให้เกิดการกระจุกตัวของข้อมูลเฉพาะที่ ส่งผลให้ข้อมูลนั้นไม่กระจายไปทั่วระบบซึ่งโหนดอื่นๆ ภาพที่ 4.11 ก็เป็นอีกตัวอย่างของการเกิดปัญหานี้ขึ้นหากโหนด D ต้องการกระจายข้อมูลต่อแต่พบว่าตนเองเป็นผู้ถูกเลือกลำดับถัดไป



ภาพที่ 4.11 ปัญหาการเลือกตัวเองเป็นผู้กระจายข้อมูลลำดับถัดไป

ดังนั้นการแก้ไขปัญหาดังกล่าวทำโดยให้ผู้กระจายข้อมูลที่พบว่าตนเองเป็นผู้ส่งข้อมูลลำดับถัดไปนั้นกระจายข้อมูลออกไป โดยไม่เลือกผู้กระจายข้อมูลลำดับถัดไปเพื่อให้เพื่อนบ้านที่ได้รับข้อมูลนี้ทำการนับเวลาถอยหลังแบบสุ่ม แล้วเมื่อเพื่อนบ้านที่ทำการนับเวลาหมดก่อนก็จะทำการกระจายข้อมูลลำดับถัดไป โดยจะทำการเลือกว่าจะให้เพื่อนบ้านโหนดใดเป็นผู้กระจายข้อมูลลำดับถัดไปด้วย ซึ่งจะทำให้การทำงานของโพรโทคอลนั้นกลับไปเป็นปกติ และกระจายข้อมูลให้ทั่วทั้งเครือข่ายไม่เกิดการกระจุกตัวของข้อมูลเฉพาะที่

บทที่ 5

ผลการทดลองและวิเคราะห์ผล

บทนี้ทำการวิเคราะห์สมรรถนะของการปรับปรุงการแพร่ข้อมูลของโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้บทสถานการณ์ที่เชื่อมต่อแบบอสมมาตร โดยเริ่มจากการกำหนดตัววัดสมรรถนะของโพรโทคอล (Performance Metrics) ในด้านต่างๆ ดังนี้ ความเชื่อถือได้ ค่าใช้จ่าย กำหนดเครื่องมือ โปรแกรมจำลองเครือข่าย และสภาพแวดล้อมในการทดลอง ผลการทดลองเปรียบเทียบสมรรถนะในด้านต่างๆ

5.1 ตัววัดสมรรถนะของโพรโทคอล (Performance Metrics)

งานวิจัยนี้มีตัววัดสมรรถนะในการทำงานของโพรโทคอล 2 ตัว ดังนี้

1. ความเชื่อถือได้ (Reliability) วัตถุประสงค์ในการทำงานของโพรโทคอลคือ สามารถส่งผ่านข้อมูลให้กับยานพาหนะในบริเวณพื้นที่หนึ่งๆ ให้ได้รับข้อมูลที่มีความเชื่อถือได้ ดังนั้นค่าความเชื่อถือได้จะคำนวณจากจำนวนข้อมูลที่ได้รับหารด้วยจำนวนข้อมูลทั้งหมดแล้ววัดผลออกมาเป็นเปอร์เซ็นต์
2. ค่าใช้จ่าย (Overhead) ในการแพร่ที่มีความเชื่อถือสูงย่อมมีค่าใช้จ่ายที่สูงกว่าดังนั้นในการทดลองจึงวัดค่าจำนวนไบต์ (Byte) ของข้อมูลที่ส่งทั้งหมดรวมทั้ง Beacon message ที่มีการส่งเพื่อแลกเปลี่ยนข้อมูลระหว่างโหนดทั้งหมดด้วย แล้วนำมาคิดเป็นต่อหนึ่งโหนดต่อหนึ่งข้อมูล เพื่อแสดงประสิทธิภาพของโพรโทคอล

5.2 เครื่องมือในการวัดสมรรถนะของโพรโทคอล

เนื่องจากโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้ นั้นถูกออกแบบมาเพื่อทำงานบนเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะ ซึ่งเป็นเรื่องยากที่จะทำการทดลองด้วยการนำเอาโหนดแต่ละโหนดนั้นวิ่งบนถนนจริงๆ อีกทั้งค่าใช้จ่ายที่สูงและความยากลำบากในการทดลอง ดังนั้นเพื่อให้ได้สภาพแวดล้อมในการทดลองที่เหมือนกันจึงใช้โปรแกรมจำลองเครือข่าย ซึ่งประกอบด้วยโปรแกรมจำลองเครือข่าย ดังนี้

1. โปรแกรมจำลอง Simulator of Urban Mobility (SUMO) ซึ่งจำลองพฤติกรรมการเคลื่อนที่ของยานพาหนะบนถนน โดยสามารถกำหนดรูปแบบการเคลื่อนที่ต่างๆ ได้ ทั้ง สภาพแวดล้อม

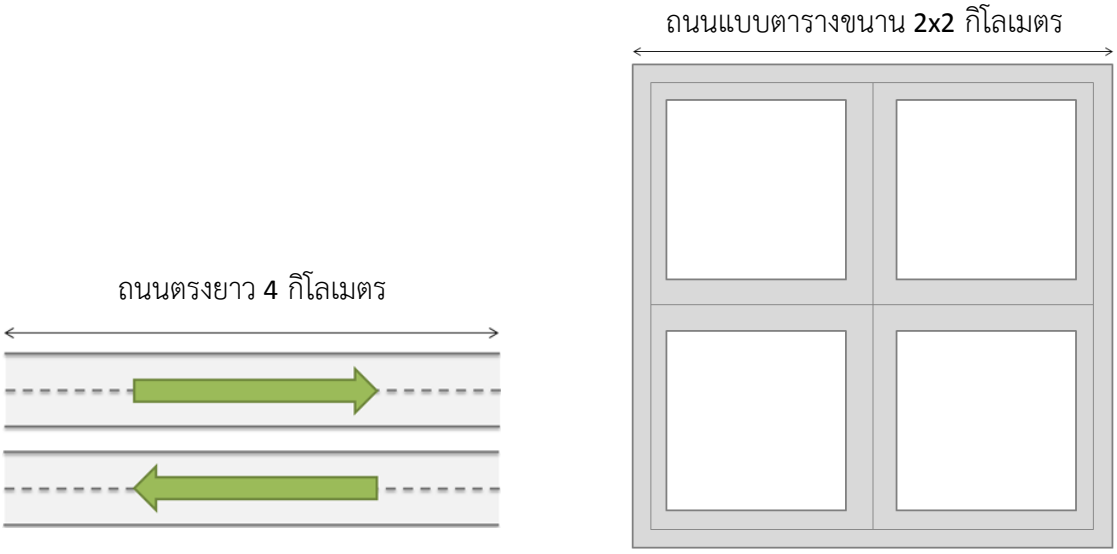
แบบถนนในเมือง หรือถนนทางหลวง อีกทั้งยังสามารถกำหนดความหนาแน่นของยานพาหนะได้ด้วย

2. โปรแกรม Traffic and Network Simulation Environment (TraNS) ซึ่งเป็นโปรแกรมที่ใช้ในการเปลี่ยนรูปแบบการเคลื่อนที่ของยานพาหนะที่ได้จากโปรแกรม SUMO ให้อยู่ในรูปแบบที่โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 สามารถนำไปใช้งานได้
3. โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2.34 (NS-2.34 : Network Simulator 2.34) เป็นโปรแกรมจำลองเครือข่ายในลักษณะของการทำงานเครือข่ายเสมือนจริง
4. โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3.10 (NS-3.10 : Network Simulator 3.10) เป็นโปรแกรมจำลองเครือข่ายในลักษณะของการทำงานเครือข่ายเสมือนจริง ซึ่งสามารถใช้งานได้ทั้งการทดลองระบบจำลองเครือข่าย และการทดลองระบบจริง

ในการทดลองนั้นเริ่มต้นด้วยการจำลองพฤติกรรมการเคลื่อนที่ของยานพาหนะ โดยใช้โปรแกรมจำลอง SUMO ในทั้งสภาพแวดล้อมบนถนนทางหลวงและสภาพแวดล้อมถนนในเมือง จากนั้นใช้โปรแกรม TraNS เพื่อที่จะเปลี่ยนรูปแบบของการเคลื่อนที่ดังกล่าวไปเป็นรูปแบบที่โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 นั้นสามารถเรียกใช้งานได้ ซึ่งโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 นั้นก็สามารถที่จะเรียกใช้รูปแบบดังกล่าวสำหรับเวอร์ชั่น 2 ด้วยการเรียกใช้ฟังก์ชั่น Ns2MobilityHelper

5.3 สภาพแวดล้อมที่ใช้ในการทดลอง

สภาพแวดล้อมในการทดลองแบบออกเป็นสองรูปแบบคือ สภาพแวดล้อมของถนนทางหลวงและสภาพแวดล้อมของถนนในเมือง ซึ่งถนนทางหลวงนั้นมีลักษณะเป็นถนนทางตรงยาว 4 กิโลเมตร มีช่องทางจราจรขนาด 4 ช่องทาง ส่วนถนนในเมืองมีลักษณะเป็นตารางขนาด 2x2 ตารางกิโลเมตร ประกอบด้วยสี่แยก 1 แยก และสามแยกจำนวน 4 แยก ทั้งหมดนั้นไม่มีไฟจราจร โดยแต่ละแยกมีระยะห่างกัน 1 กิโลเมตร มีช่องทางจราจรขนาด 2 ช่องทาง ดังภาพที่ 5.1 ความหนาแน่นของยานพาหนะโดยเฉลี่ยจะถูกแบ่งออกเป็นขนาดต่างๆ ดังตารางที่ 5.1 เพื่อทดลองการทำงานของโพรโทคอลและพฤติกรรมของยานพาหนะในสถานการณ์ต่างๆ ที่แตกต่างกัน เพื่อพิจารณาถึงผลของประสิทธิภาพในการทำงานของโพรโทคอล ซึ่งสภาพการจราจรนั้นถูกจำลองโดยใช้โปรแกรมจำลอง SUMO รุ่น 0.10.3 และเปลี่ยนแปลงรูปแบบให้เหมาะสมกับโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ด้วยโปรแกรม TraNS รุ่น 1.0



(ก) สภาพแวดล้อมรูปแบบถนนทางหลวงยาว (ข) สภาพแวดล้อมรูปแบบถนนในเมือง

ภาพที่ 5.1 ลักษณะถนนที่ใช้ในการทดลอง

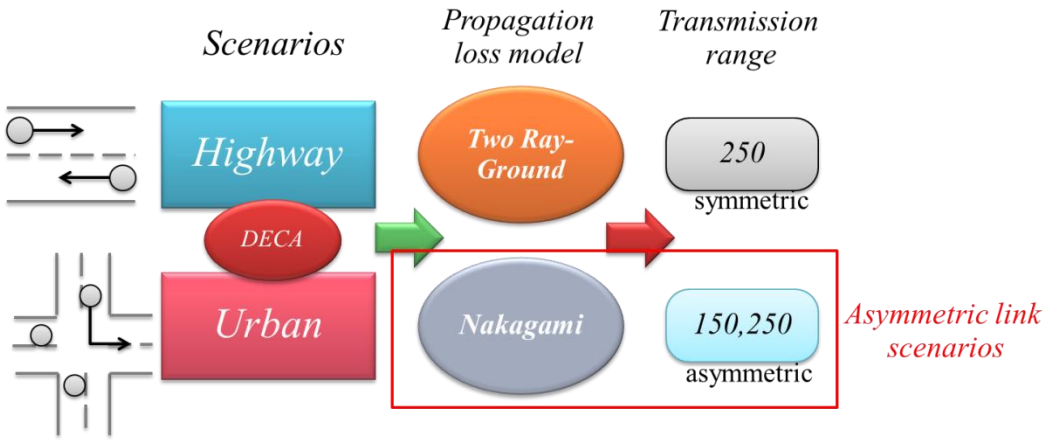
ตารางที่ 5.1 การตั้งค่าพารามิเตอร์เพื่อทดลองประสิทธิภาพ

อายุของข้อความ	10วินาที
จำนวนของข้อความ	5
ขนาดของข้อความ	512 ไบต์
ระยะเวลาในการทดลอง	50 วินาที
ความเร็วสูงสุดของยานพาหนะ (กิโลเมตร/ชั่วโมง)	50, 80
ความหนาแน่นของยานพาหนะ (คัน/กิโลเมตร)	2, 10, 30, 60, 120
โปรแกรมจำลองเครือข่าย	NS-3.10
สถานการณ์การสื่อสารไร้สาย (ระยะในการส่งข้อมูลสูงสุด (เมตร)/ Propagation Loss Model)	250 เมตร/ Two-Ray Ground 250 เมตร/ Nakagami 150, 250 เมตร/ Two-Ray Ground 150, 250 เมตร/ Nakagami

แต่ในสถานการณ์การเชื่อมต่อแบบอสมมาตรนั้นมีผลกระทบอย่างมากในรูปแบบยานพาหนะที่มีความหนาแน่นต่ำ เพาะว่าโหนดนั้นจะมีระยะห่างระหว่างกันมากซึ่งเมื่อความแรงสัญญาณต่ำทำให้ระยะการส่งข้อมูลนั้นสั้นกว่าระยะห่างระหว่างโหนดทำให้เห็นผลกระทบที่เกิดขึ้นอย่างชัดเจน แต่ในรูปแบบของโหนดที่มีความหนาแน่นมากตำแหน่งของแต่ละโหนดนั้นใกล้กันจนเมื่อความแรงของ

สัญญาณต่ำ ระยะส่งข้อมูลสั้นก็ไม่มีผลกระทบต่อการทำงานเพราะว่าระยะห่างระหว่างโหนดใกล้มากในระยะที่ยังส่งข้อมูลได้อยู่ ดังนั้นงานวิจัยนี้จึงเน้นรูปแบบของยานพาหนะที่มีความหนาแน่นต่ำโดยใช้สภาพแวดล้อมรูปแบบถนนทางหลวงที่มีความหนาแน่น 2 6 10 30 60 120 คันต่อกิโลเมตร ซึ่งมากกว่าสภาพแวดล้อมรูปแบบถนนในเมืองที่ใช้ความหนาแน่นในการทดลองเพียงแค่ 2 6 10 คันต่อกิโลเมตร เพราะรูปแบบถนนทางหลวงนั้นมีความหนาแน่นของโหนดที่น้อยและลักษณะการเคลื่อนที่ของโหนดที่เร็วกว่าทำให้ไม่เกิดการกระจุกตัว ทำให้เหมาะกับการทดลองสถานการณ์ที่มีการเชื่อมต่อแบบอสมมาตร

นอกจากนั้นการทดลองนี้ยังเปลี่ยน Propagation Loss Model ซึ่งเดิมโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 นั้นใช้ Two-Ray Ground Propagation Loss Model ที่มีค่าแน่นอนในการใช้งานและมีความเป็นไปได้ในการส่งข้อมูลที่ไม่เป็นความจริง เพราะว่าสามารถกำหนดระยะในการส่งข้อมูลได้แน่นอนซึ่งหากเลยระยะนั้นไปแล้วจะไม่สามารถส่งข้อมูลได้เลย ซึ่งหากต้องการให้การส่งข้อมูลของเครือข่ายนั้นเสมือนจริงจึงต้องเปลี่ยนเป็น Nakagami Propagation Loss Model แทน เพราะมีรูปแบบที่คล้ายกับการทำงานบนระบบจริง เพาะว่าสัญญาณในระบบนั้นมีการถูกรบกวนและบิดเบือนทำให้ในแต่ละช่วงของระยะที่ส่งข้อมูลได้นั้นจะมีความเป็นไปได้ไม่ถึง 100 % และถ้าผู้รับห่างจากผู้ส่งมากขึ้นความเป็นไปได้ในการส่งข้อมูลนั้นก็จะน้อยลง หากกำหนดระยะการส่งข้อมูลไว้ที่ระยะหนึ่งในการทดลองนี้จะกำหนดให้ระยะนั้นความเป็นไปได้ในการส่งข้อมูลจะมีเพียง 50 % และหากเลยระยะการส่งข้อมูลไปแล้วก็ยังมีความเป็นไปได้ในการส่งข้อมูลสำเร็จแต่จะมีค่าที่น้อย



ภาพที่ 5.2 เงื่อนไขในการจำลองสถานการณ์ในการทดลอง

อีกทั้งการทดลองยังประกอบไปด้วยการจำลองสถานการณ์ระยะการส่งสัญญาณสองแบบคือการเชื่อมต่อสมมาตรและการเชื่อมต่อแบบอสมมาตร ซึ่งในสถานการณ์แรกแบบสมมาตรนั้นจะให้ทุกโหนดในเครือข่ายนั้นมีความแรงของสัญญาณเท่ากันทำให้มีระยะการส่งข้อมูลอยู่ที่ 250 เมตร ส่วนอีก

สถานการณ์จะให้โหนดแบ่งออกเป็นสองกลุ่ม คือ กลุ่มที่มีระยะการส่งข้อมูลอยู่ที่ 150 เมตร กับ 250 เมตร เพื่อแทนลักษณะของยานพาหนะที่มีความแตกต่างของระยะการส่งสัญญาณ ซึ่งจากภาพที่ 5.2 จะสังเกตได้ว่าเมื่อใช้ Nakagami Propagation Loss Model กับสถานการณ์แบบอสมมาตร ซึ่งทำให้สอดคล้องกับสถานการณ์การเชื่อมต่อแบบอสมมาตรเสมือนการทำงานบนระบบจริง

ในการทดลองนั้นจะแบ่งสถานการณ์ในการทดลองเป็น 5 สถานการณ์ ดังนี้

1. **DECA Symmetric with Two-Ray Ground** สถานการณ์จำลองที่แทนลักษณะของสถานการณ์แบบสมมาตรซึ่งทำงานด้วยโพรโทคอล DECA โดยให้แต่ละโหนดมีระยะการส่งข้อมูลที่ 250 เมตร และใช้ Two-Ray Ground Propagation Loss Model ซึ่งเป็นลักษณะการทำงานของอุปกรณ์ไร้สายแบบอุดมคติ ในสถานการณ์นี้นั้นมักถูกใช้ในการทดลองจำลองเครือข่ายของงานวิจัยส่วนใหญ่
2. **DECA Symmetric with Nakagami** สถานการณ์จำลองที่แทนลักษณะของสถานการณ์แบบสมมาตรซึ่งทำงานด้วยโพรโทคอล DECA โดยให้แต่ละโหนดมีระยะการส่งข้อมูล 250 เมตร และใช้ Nakagami Propagation Loss Model ซึ่งให้ค่าที่ใกล้เคียงกับสัญญาณในระบบจริง ทำให้การทดลองนั้นใกล้เคียงระบบจริงมากขึ้น
3. **DECA Asymmetric with Two-Ray Ground** สถานการณ์จำลองที่แทนลักษณะของสถานการณ์แบบอสมมาตรซึ่งทำงานด้วยโพรโทคอล DECA โดยให้โหนดแบ่งออกเป็นสองกลุ่ม คือ ระยะส่งสัญญาณ 150 เมตร กับ 250 เมตร และใช้ Two-Ray Ground Propagation Loss Model ทำให้การจำลองนั้นใกล้เคียงกับระบบจริงที่แต่ละโหนดมีความแรงของสัญญาณแตกต่างกันมากขึ้น
4. **DECA Asymmetric with Nakagami** สถานการณ์จำลองที่แทนลักษณะแบบสถานการณ์แบบอสมมาตรซึ่งทำงานด้วยโพรโทคอล DECA โดยแบ่งโหนดออกเป็นสองกลุ่ม คือระยะการส่งข้อมูล 150 เมตร กับ 250 เมตร และใช้ Nakagami Propagation Loss Model ซึ่งให้ค่าเสมือนการทำงานบนระบบจริง
5. **RVA Asymmetric with Nakagami** สถานการณ์จำลองที่แทนลักษณะแบบสถานการณ์อสมมาตรที่ทำงานด้วยอัลกอริทึมการโหวตแบบอาร์เอสเอสไอ (RSSI-Voting algorithm : RVA) ในการเลือกโหนด โดยแบ่งโหนดออกเป็นสองกลุ่มคือ ระยะส่งข้อมูล 150 เมตร กับ 250 เมตร และใช้ Nakagami Propagation Loss Model ซึ่งให้ค่าเสมือนกับการทำงานบนระบบจริง

ในการทดลองใช้โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3.10 แต่ละครั้งข้อมูลจะถูกแพร่กระจายจำนวน 1 ข้อความ เป็นจำนวน 5 ข้อความ ซึ่งมีขนาดข้อความละ 512 ไบต์ โดยอายุของข้อความนั้นคือ 10 วินาที ซึ่งเมื่อข้อความหมดอายุก็จะถูกลบออกจากระบบ และข้อความใหม่ก็จะถูกแพร่กระจายออกมาจากผู้ส่งข้อมูลทันที การทดลองระบบสื่อสารไร้สายของโหนดทำงานตามมาตรฐาน IEEE 802.11 โดยมีการชนกันของข้อมูลตามปกติ ดังตารางที่ 5.1

5.4 ผลการทดลองค่าความเชื่อถือได้ของโพรโทคอล

ค่าความเชื่อถือได้จากผลการทดลองบนถนนทางหลวง ดังภาพที่ 5.1(ก)

เมื่อพิจารณาในมุมมองของสถานการณ์จำลอง มีรายละเอียดดังนี้

1. **DECA Symmetric with Two-Ray Ground** สถานการณ์นี้มักถูกใช้ทั่วไปในการจำลองเครือข่ายของงานวิจัยทั่วไป ซึ่งค่าความเชื่อถือได้ที่ออกมานั้นมีค่าสูงเกินจริง เพราะว่า Two-Ray Ground นั้นให้ค่าของสัญญาณที่ไม่สมจริงอีกทั้งทุกโหนดในเครือข่ายยังมีความแรงของสัญญาณเท่ากัน ทำให้ค่าความเชื่อถือได้นั้นออกมาดี แต่ก็ไม่สามารถที่จะนำผลจากการทดลองนี้ไปใช้ได้บนระบบจริงเพราะว่าสถานการณ์ในการทดลองนั้นยังขาดอีกหลายปัจจัยที่มีบนระบบจริง
2. **DECA Symmetric with Nakagami** สถานการณ์รูปแบบนี้นั้นผลที่ออกมาจะแย่กว่าสถานการณ์แบบที่หนึ่ง เพราะว่า Nakagami นั้นให้ค่าของสัญญาณในการส่งข้อมูลที่เสมือนจริงมากขึ้น แต่ทุกโหนดในเครือข่ายยังคงมีความแรงของสัญญาณที่เท่ากันทำให้ค่าที่ออกมานั้นยังไม่เสมือนจริง
3. **DECA Asymmetric with Two-Ray Ground** สถานการณ์รูปแบบนี้ค่าความเชื่อถือได้ที่ออกมานั้นจะดีกว่าสถานการณ์แบบที่สอง แต่ยังแย่กว่าสถานการณ์แบบที่หนึ่งถึงแม้ว่าโหนดในเครือข่ายจะถูกแบ่งออกเป็นสองกลุ่มทำให้มีความแรงของสัญญาณที่แตกต่างกัน แต่ว่า Two-Ray Ground ยังให้ค่าที่ไม่เสมือนจริงทำให้ผลที่ออกมานั้นดีกว่า Nakagami
4. **DECA Asymmetric with Nakagami** สถานการณ์รูปแบบนี้นั้นมีค่าความเชื่อถือได้ที่ต่ำที่สุดเมื่อเทียบกับสถานการณ์ที่ทำการทดลองทั้งหมด เพราะว่าโหนดทั้งเครือข่ายนั้นมีความแรงของสัญญาณที่แตกต่างกันอีกทั้ง Nakagami ยังให้ค่าของสัญญาณในการส่งข้อมูลที่เสมือนจริงทำให้การทำงานของระบบนั้นคล้ายคลึงกับการทำงานบนระบบจริงมากที่สุด ซึ่งยานพาหนะแต่ละคันอาจมีความแรงของสัญญาณที่แตกต่างกันและสัญญาณนั้นอาจถูกบิดเบือนหรือดูดซับจากบรรยากาศภายนอกต่างๆ

5. **RVA Asymmetric with Nakagami** สถานการณ์ในการทดลองสุดท้ายนี้ทำงานด้วยอัลกอริทึมการโหวตแบบอาร์เอสเอสไอซึ่งแก้ไขปัญหาในการเลือกโหนดของโพรโทคอลแล้วทำให้ค่าความเชื่อถือได้ที่วัดออกมานั้นมีค่าที่มากขึ้นเมื่อเทียบกับสถานการณ์แบบที่สี่ที่เสมือนจริงมากที่สุด โดยค่าที่ออกมานั้นดีขึ้นมากสุดถึง 17 %

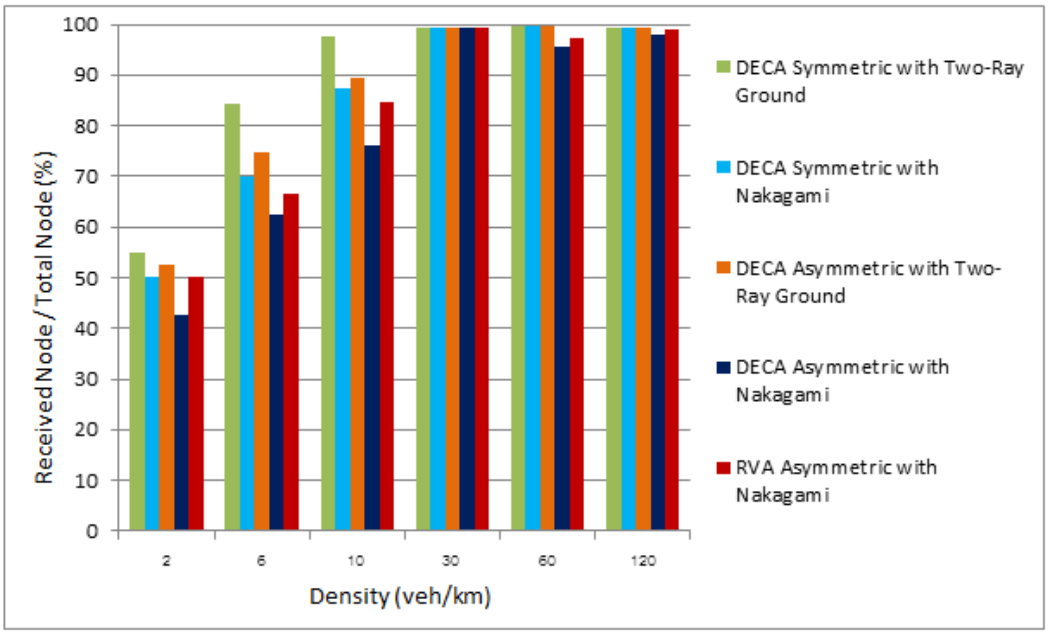
เมื่อพิจารณาในมุมมองของความหนาแน่นของยานพาหนะในเครือข่าย สามารถแยกได้เป็นสองความหนาแน่นหลักๆ ดังนี้

1. **ความหนาแน่นต่ำ (2-10 คัน/กิโลเมตร)** การทดลองนั้นค่าความเชื่อถือได้ที่วัดออกมานั้นจะมีค่าออกมาต่ำเมื่อเทียบกับความหนาแน่นที่สูงขึ้นเรื่อยๆ เพราะว่าในสถานการณ์ที่มีความหนาแน่นต่ำนั้นจะได้รับผลกระทบจากการเชื่อมต่อแบบอสมมาตรมากที่สุด โดยระยะในการส่งข้อมูลของแต่ละโหนดนั้นจะสั้นกว่าระยะห่างระหว่างโหนดแต่ละโหนด ทำให้เมื่อแต่ละโหนดมีความแรงของสัญญาณที่แตกต่างกันและค่าของสัญญาณที่เสมือนจริง ระยะในการส่งข้อมูลของแต่ละโหนดยิ่งน้อยลงกว่าเดิม ทำให้การส่งข้อมูลในเครือข่ายนั้นไม่ทั่วถึงแล้วยิ่งความหนาแน่นต่ำมากก็ทำให้ค่าความเชื่อถือที่ออกมานั้นจะต่ำมากเช่นกัน แต่เมื่อแก้ไขปัญหาด้วยอัลกอริทึมการโหวตแบบอาร์เอสเอสไอก็ทำให้ผลออกมาดีขึ้นโดยในความหนาแน่นที่ต่ำสุดนั้นสามารถเพิ่มค่าความเชื่อถือได้ให้เท่ากับสถานการณ์การเชื่อมต่อแบบสมมาตรในรูปแบบสัญญาณที่เป็น Nakagami เหมือนกันได้
2. **ความหนาแน่นสูง (20-120 คัน/กิโลเมตร)** การทดลองนี้ค่าความเชื่อถือได้จะมีค่ามากกว่าความหนาแน่นต่ำ เพราะว่าในสถานการณ์นี้จะไม่ได้รับผลกระทบจากการเชื่อมต่อแบบอสมมาตรเท่าไรนักเนื่องจากระยะในการส่งข้อมูลของแต่ละโหนดนั้นมีระยะที่ยาวกว่าระยะห่างของแต่ละโหนดแม้ว่าความแรงของสัญญาณที่ส่งจะน้อยก็ตามทำให้ค่าความเชื่อถือได้นั้นออกมาสูงกว่าสถานการณ์ที่มีความหนาแน่นต่ำ แต่อย่างไรก็ตามการเชื่อมต่อแบบอสมมาตรก็ส่งผลกระทบแต่ไม่มากนักซึ่งการแก้ไขด้วยอัลกอริทึมการโหวตอาร์เอสเอสไอก็ส่งผลให้ค่าความเชื่อถือได้ของระบบดีขึ้นตามมา

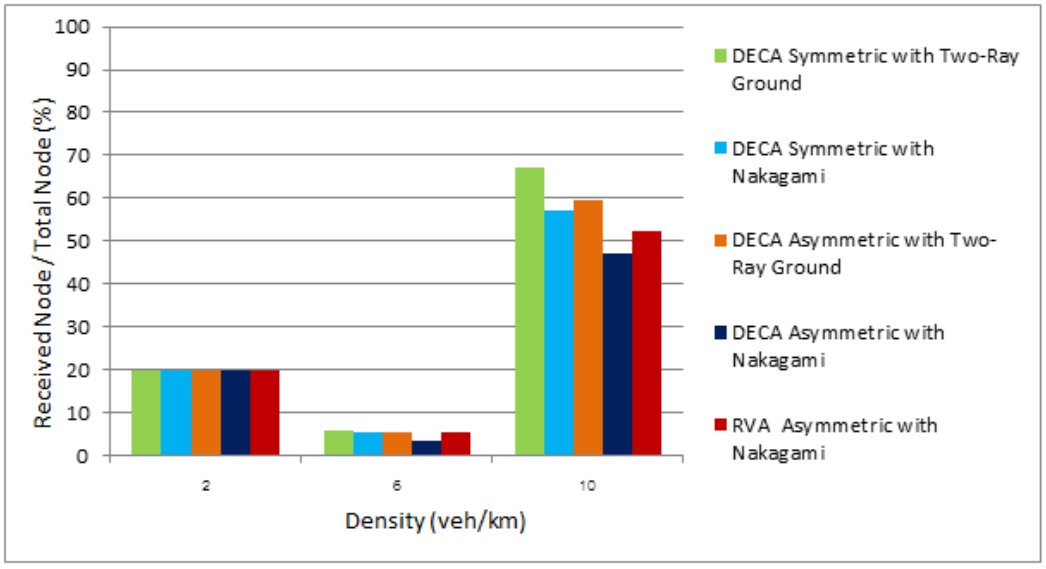
ค่าความเชื่อถือได้จากผลการทดลองบนถนนในเมือง ดังภาพที่ 5.1(ข)

การทดลองบนสภาพแวดล้อมแบบถนนในเมืองนั้นมีลักษณะเหมือนกันการทดลองสภาพแวดล้อมแบบถนนทางหลวงที่มีความหนาแน่นสูง เพราะว่าถนนในเมืองนั้นมีลักษณะที่เป็นตารางทำให้การเคลื่อนที่ของยานพาหนะนั้นจะกระจายไปตามเส้นทางที่เลี้ยวไปมา และมีการหยุดเพื่อรอให้รถที่อยู่สี่แยกหรือสามแยกเดินทางไปก่อนซึ่งการหยุดก็ทำให้สามารถส่งข้อมูลไปให้กับโหนดอื่นๆ ได้ ดังนั้นค่าความเชื่อถือได้ที่ออกมานั้นจึงมีผลที่แตกต่างกับถนนทางหลวง

ในส่วนของความหนาแน่น 2 คัน/กิโลเมตรนั้นเนื่องจากการกำหนดการเคลื่อนที่ของโหนดเป็นไปในทางที่ทำให้ยานพาหนะสามารถเจอกันได้หมดทำให้ค่าความเชื่อถือได้ในแต่ละสถานการณ์จำลองนั้นออกมาเท่าๆ กัน แต่ในส่วนของความหนาแน่นอื่นๆ อย่างเช่น 6 กับ 10 คัน/กิโลเมตร ค่าความเชื่อถือได้ของสถานการณ์เชื่อมต่อแบบอสมมาตรนั้นมีผลออกมาต่ำ ซึ่งแก้ไขด้วยอัลกอริทึมการโหวตแบบอาร์เอสเอสไอแล้วก็ทำให้การทำงานของโพรโทคอลนั้นมีประสิทธิภาพที่ดีขึ้นมากสุดถึง 5 %



ภาพที่ 5.3 กราฟแสดงค่าความเชื่อถือได้จากการทดลองบนถนนทางหลวง



ภาพที่ 5.4 กราฟแสดงค่าความเชื่อถือได้จากการทดลองบนถนนในเมือง

จากผลการทดลองสรุปได้ว่าค่าความเชื่อถือได้ของโพรโทคอลนั้นได้รับผลกระทบมากที่สุดจากสถานการณ์ที่โหนดมีความหนาแน่นต่ำ และโหนดแต่ละโหนดมีความแรงของสัญญาณที่แตกต่างกันตามการเชื่อมต่อแบบอสมมาตร อีกทั้งสัญญาณมีค่าแบบ Nakagami ซึ่งเสมือนสัญญาณของระบบจริง

5.5 ผลการทดลองค่าใช้จ่ายของโพรโทคอล

ค่าใช้จ่ายได้จากผลการทดลองบนถนนทางหลวงและถนนในเมือง ดังภาพที่ 5.1 (ก) และ 5.1 (ข) ตามลำดับ ซึ่งพิจารณาจากกราฟของค่าใช้จ่ายทั้งการทดลองบนถนนทางหลวงและถนนในเมืองจะพบความสอดคล้องกัน สามารถนำมาพิจารณาคล้ายกับค่าความเชื่อถือได้ดังนี้

เมื่อพิจารณาในมุมมองของสถานการณ์จำลอง มีรายละเอียดดังนี้

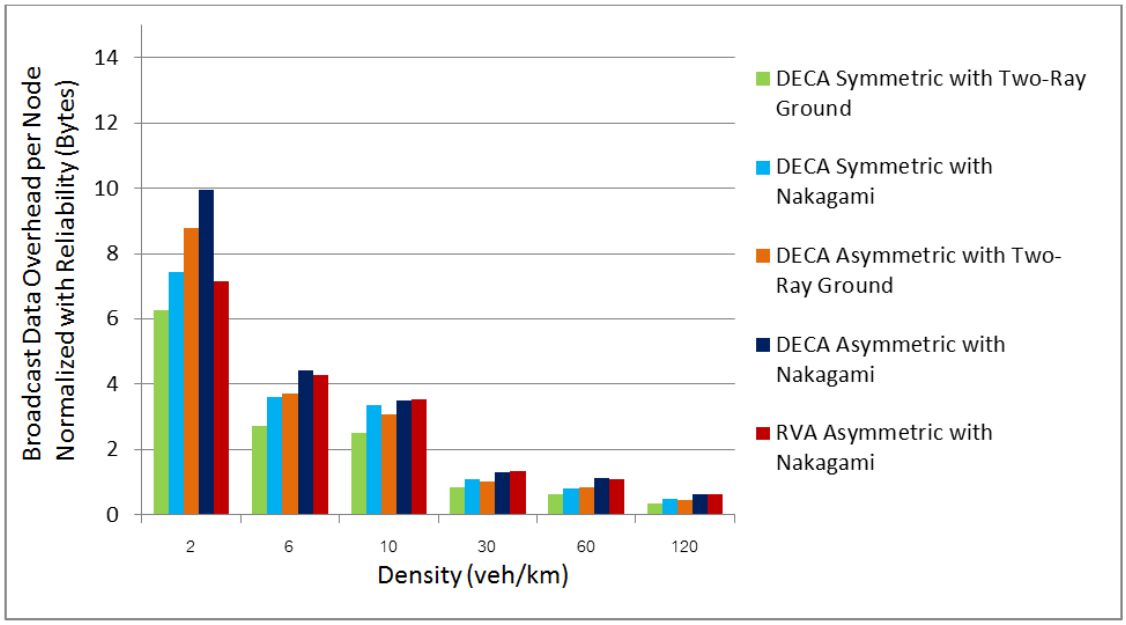
สถานการณ์จำลองที่เชื่อมต่อกันแบบสมมาตรและมีสัญญาณแบบ Two-Ray Ground นั้นจะสามารถส่งข้อมูลหากันได้ดีทำให้ค่าใช้จ่ายในการส่งข้อมูลทั้งหมดของระบบนั้นน้อยกว่าสถานการณ์แบบอื่นๆ แต่ว่าด้วยลักษณะที่ไม่เสมือนจริงทำให้ค่านั้นไม่เหมาะกับการใช้อ้างถึงประสิทธิภาพของโพรโทคอลได้ ส่วนสถานการณ์จำลองที่ใกล้เคียงกับระบบจริงมากขึ้นก็ทำให้ค่าใช้จ่ายของระบบนั้นมีมากขึ้นเนื่องจากเมื่อการส่งข้อมูลระหว่างโหนดเป็นไปได้ยากโหนดที่ไม่ได้รับข้อมูลก็ต้อง Beacon ออกมาหาเพื่อนบ้านให้ส่งข้อมูลนั้นๆ ให้กับตนเอง ส่งผลให้ค่าใช้จ่ายก็เพิ่มขึ้นเรื่อยๆ และขึ้นมากที่สุดในสถานการณ์จำลองการเชื่อมต่อแบบอสมมาตรที่มีสัญญาณแบบ Nakagami แต่เมื่อแก้ไขปัญหาด้วยอัลกอริทึมการโหวตก็ทำให้ค่าใช้จ่ายของการส่งข้อมูลในระบบนั้นน้อยลงมากสุดถึง 28 %

เมื่อพิจารณาในมุมมองของความหนาแน่นของยานพาหนะในเครือข่าย มีรายละเอียดดังนี้

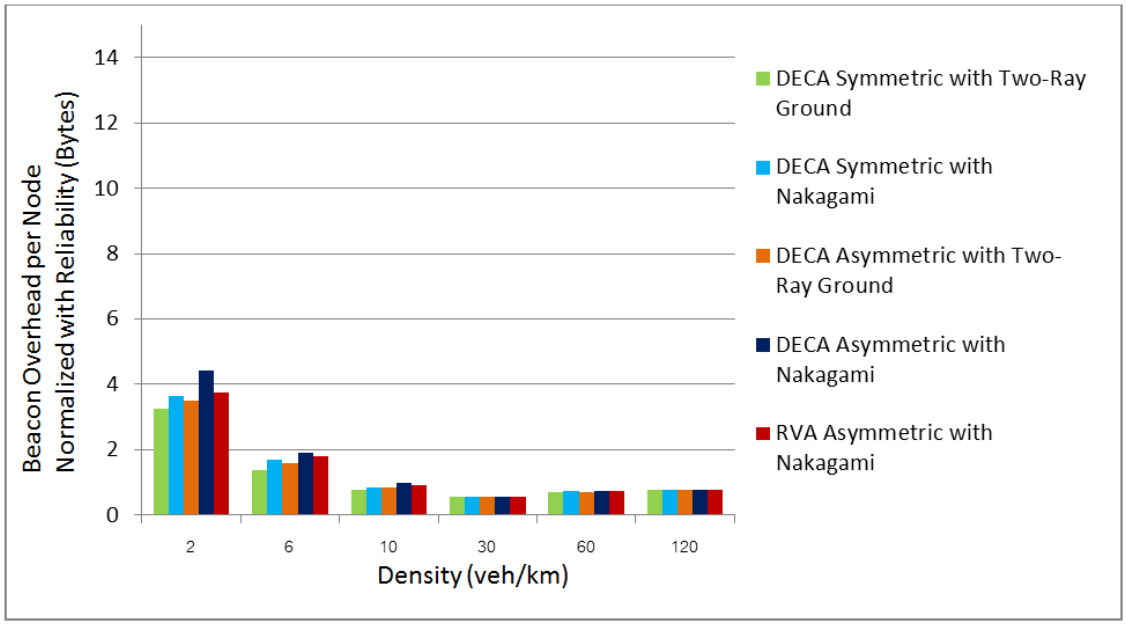
ในสภาพแวดล้อมของการทดลองที่มีความหนาแน่นต่ำนั้น การส่งข้อมูลระหว่างโหนดเป็นเรื่องที่ยาก เพราะระยะในการส่งข้อมูลนั้นจะสั้นกว่าระยะห่างระหว่างโหนดทำให้มีการไม่ได้รับข้อมูลเกิดขึ้นในบางโหนด ซึ่งโหนดนั้นก็ต้องส่ง Beacon ออกมาให้เพื่อนบ้านเพื่อให้ส่งข้อมูลนั้นกลับมายังตนเองอีก ส่งผลให้ค่าใช้จ่ายของระบบนั้นจะมาก ซึ่งเมื่อความหนาแน่นเพิ่มขึ้นทำให้ระยะระหว่างโหนดนั้นน้อยลง การส่งข้อมูลก็สำเร็จมากขึ้นก็ทำให้ค่าใช้จ่ายของระบบนั้นน้อยลงตามมาด้วยเช่นกัน แต่เมื่อสังเกตการทดลองบนถนนในเมืองที่ความหนาแน่น 2 คัน/กิโลเมตร นั้นค่าใช้จ่ายจะน้อยกว่าความหนาแน่น 6 คัน/กิโลเมตร เพราะว่าการกำหนดรูปแบบการเคลื่อนที่ของโหนดนั้นส่งผลให้การเคลื่อนที่ของโหนดสามารถที่จะส่งข้อมูลให้กับโหนดในระบบได้ทั้งหมดทำให้ค่าใช้จ่ายจึงออกมาน้อยกว่าความหนาแน่นลำดับถัดไป

โดยจากกราฟแสดงค่าใช้จ่ายนั้นสามารถแยกออกได้เป็น 3 แบบ ดังนี้

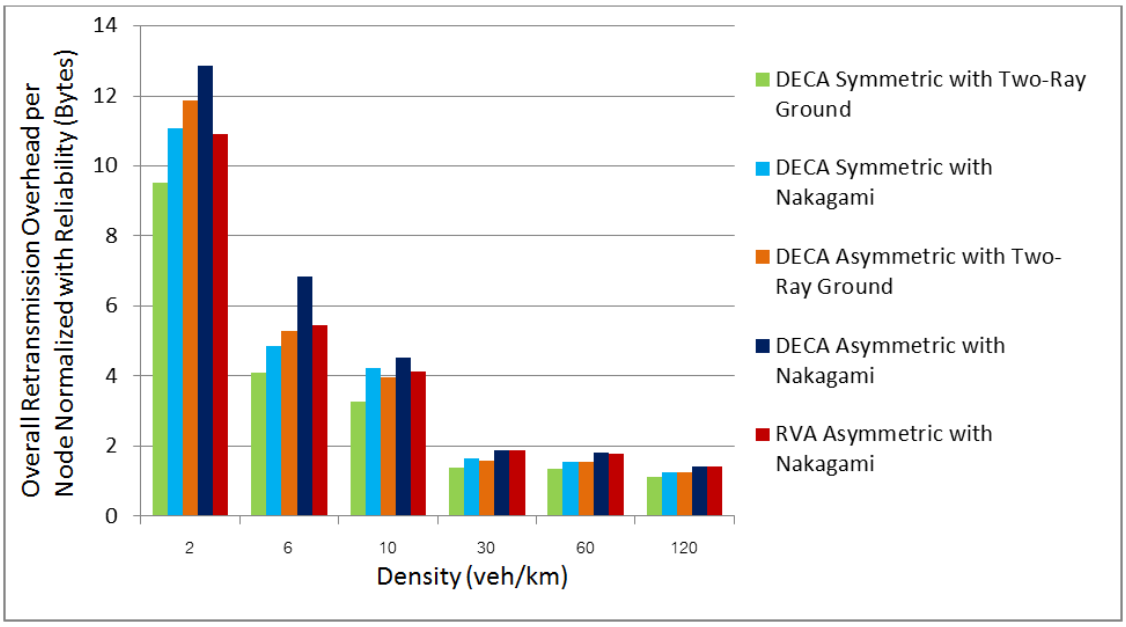
1. กราฟแสดงค่าใช้จ่ายในการส่งข้อความ เมื่อผู้ส่งเริ่มกระจายข้อความให้กับโหนดเพื่อนบ้าน ระบบก็จะทำการกระจายข้อความไปให้กับโหนดอื่น ๆ ที่อยู่ในละแวกต่อไป โดยข้อความนั้นจะมีขนาด 512 ไบท์ ซึ่งมากกว่าค่าใช้จ่ายในการแลกเปลี่ยนข้อมูลระหว่างเพื่อนบ้านในข้อความประเภท Beacon
2. กราฟแสดงค่าใช้จ่ายในการส่ง Beacon การส่ง Beacon นั้นแต่ละโหนดจะทำการส่งให้เพื่อนบ้านของตนเองเป็นระยะเพื่อแลกเปลี่ยนข้อมูลของของกันและกัน โดยข้อความนั้นจะประกอบด้วยข้อมูลที่บอกว่าได้รับข้อความใดแล้ว และผลของการโหวตเพื่อนบ้านที่มีค่าอาร์เอสเอสไอมากที่สุด
3. กราฟแสดงค่าใช้จ่ายรวม กราฟนี้เป็นกราฟแสดงผลรวมของค่าใช้จ่ายในการส่งข้อความกับค่าใช้จ่ายในการส่ง Beacon ซึ่งจะแสดงให้ทราบว่าค่าใช้จ่ายที่ใช้ไปจริงต่อหนึ่งโหนดต่อหนึ่งข้อความนั้นมีค่าเท่าใด



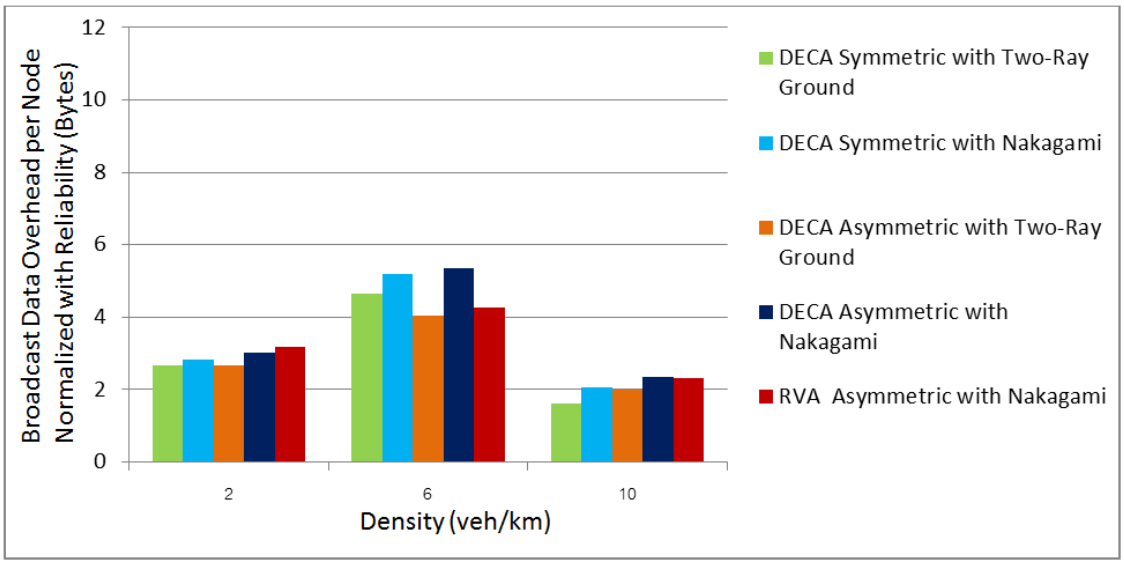
ภาพที่ 5.5 กราฟแสดงค่าใช้จ่ายในการส่งข้อความของระบบที่ได้จากการทดลองบนถนนทางหลวง



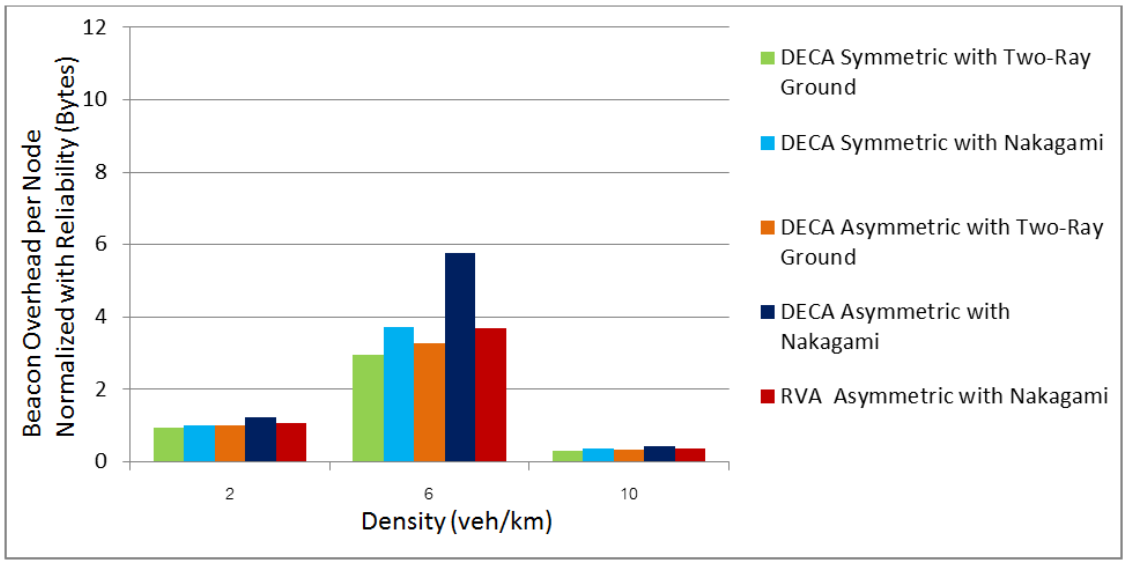
ภาพที่ 5.6 กราฟแสดงค่าใช้จ่ายในการส่ง Beacon ของระบบที่ได้จากการทดลองบนถนนทางหลวง



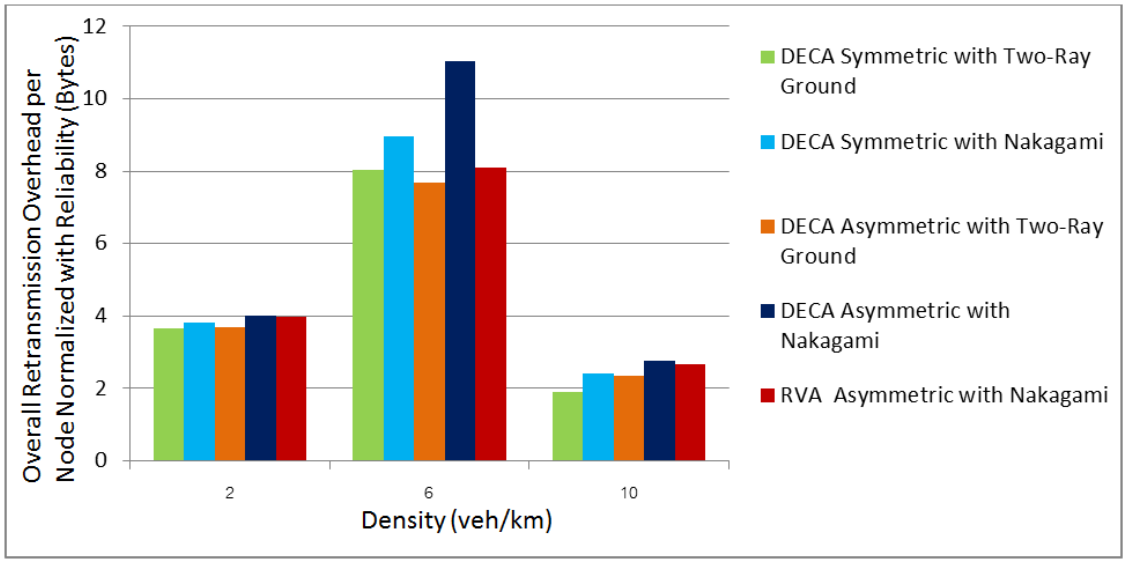
ภาพที่ 5.7 กราฟแสดงค่าใช้จ่ายรวมของระบบที่ได้จากการทดลองบนถนนทางหลวง



ภาพที่ 5.8 กราฟแสดงค่าใช้จ่ายในการส่งข้อความของระบบที่ได้จากการทดลองบนถนนในเมือง



ภาพที่ 5.9 กราฟแสดงค่าใช้จ่ายในการส่ง Beacon ของระบบที่ได้จากการทดลองบนถนนในเมือง



ภาพที่ 5.10 กราฟแสดงค่าใช้จ่ายรวมของระบบที่ได้จากการทดลองบนถนนในเมือง

บทที่ 6

สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

วิทยานิพนธ์ฉบับนี้สามารถแบ่งเนื้อหาออกเป็นสองส่วนหลัก ๆ ซึ่งประกอบด้วย ส่วนแรกคือการพัฒนาโพรโทคอลจากโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 ไปยังเวอร์ชั่น 3 และส่วนที่สองคือการปรับปรุงการแพร่ข้อมูลแบบเชื่อถือได้บนเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะที่เชื่อมต่อแบบอสมมาตรด้วยพื้นฐานของอัลกอริทึมการโหวตอาร์เอสเอสไอ โดยส่วนแรกนั้นได้ศึกษาการทำงานของแต่ละโปรแกรมและความแตกต่าง เพื่อเป็นแนวทางในการพัฒนาโพรโทคอลบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ซึ่งเป็นโปรแกรมจำลองเครือข่ายที่ถูกพัฒนาขึ้นใหม่ด้วยโครงสร้างที่แตกต่างจากเดิมทั้งหมด โดยเนื้อหาส่วนนี้นั้นอยู่บทที่ 3 กับภาคผนวก และส่วนที่สองที่เกี่ยวกับการแก้ไขปัญหาการทำงานของโพรโทคอลบนเครือข่ายที่เชื่อมต่อแบบอสมมาตรนั้นอยู่ในบทที่ 4 และวัดผลการทดลองกับวิเคราะห์ผลในบทที่ 5

งานวิจัยนี้ได้นำเสนอกระบวนการในการแก้ไขปัญหาของโพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้ในการทำงานบนสถานการณ์ที่มีการเชื่อมต่อแบบอสมมาตร โดยงานวิจัยได้เสนออัลกอริทึมการโหวตอาร์เอสเอสไอ (RVA : RSSI-Voting Algorithm) ซึ่งเป็นกระบวนการในการเลือกผู้กระจายข้อมูลลำดับถัดไปเพื่อแก้ไขปัญหาการทำงานของโพรโทคอลบนสถานการณ์ที่เป็นปัญหาดังกล่าว ซึ่งกระบวนการนี้สามารถเพิ่มประสิทธิภาพการทำงานของโพรโทคอลได้ดีขึ้น

ในงานวิจัยนี้ได้หาวิธีการพัฒนาเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะซึ่งแต่เดิมนั้นมักทำด้วยการจำลองเครือข่ายแต่ว่ายังไม่สามารถทดลองบนระบบจริงได้ทำให้การทดลองยังขาดปัจจัยหลายๆ อย่างที่เกิดขึ้นบนระบบจริง หลังจากวิเคราะห์อย่างละเอียดแล้วจึงเสนอโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 เป็นทางเลือกในการทำวิจัยเพราะความสามารถของโปรแกรมนั้นรองรับทั้งการทดลองด้วยการจำลองเครือข่ายและการทดลองบนระบบจริง หลังจากนั้นจึงทดลองการทำงานของโพรโทคอล DECA ที่เป็นโพรโทคอลที่ทำงานง่าย รวดเร็ว และลดความซ้ำซ้อนของข้อมูล เพื่อเป็นตัวอย่างในการศึกษาพัฒนาบนการทดลองบนระบบจริงของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3

จากการทดลองจึงพบปัญหาที่เกิดขึ้นบนระบบจริงแต่มักถูกข้ามในการทดลองจำลองเครือข่ายทั่ว ๆ ไปในหลาย ๆ งานวิจัย ปัญหานั้นคือการเชื่อมต่อแบบอสมมาตร ซึ่งปัญหานี้ทำให้การทำงานของโพรโทคอลนั้นมีประสิทธิภาพที่ต่ำลง โดยมีค่าความเชื่อถือได้ต่ำ และค่าใช้จ่ายในการส่งข้อมูลทั้งระบบ

ที่สูง สาเหตุมาจากการส่งข้อมูลที่ผิดพลาดของผู้ส่งทำให้ผู้รับไม่ได้รับข้อมูลจึงทำการ Beacon ออกมาเพื่อให้เพื่อนบ้านส่งข้อมูลให้กับตนเองใหม่ และการเลือกผู้กระจายข้อมูลลำดับถัดไปที่ไม่มีประสิทธิภาพเพราะอาจเป็นโหนดที่มีความแรงของสัญญาณต่ำ ทำให้ไม่สามารถส่งข้อมูลไปให้กับเพื่อนบ้านได้

จากการศึกษาและวิจัยทำให้ทราบว่าในสถานการณ์การเชื่อมต่อแบบอสมมาตรนั้นเป็นการยากที่จะทำให้ความสำเร็จของการส่งข้อมูลนั้นเป็น 100 % ดังนั้นงานวิจัยนี้จึงเน้นไปที่การแก้ไขกระบวนการเลือกผู้กระจายข้อมูลลำดับถัดไป ซึ่งงานวิจัยนี้ได้เสนออัลกอริทึมการโหวตอาร์เอสเอสไอที่แก้ไขการเลือกโดยเปลี่ยนแนวคิดจากผู้ถูกเลือกจะได้รับ Beacon จากเพื่อนบ้านมากที่สุดคือมีความหนาแน่นมากที่สุด เป็นผู้ถูกเลือกควรเป็นโหนดที่ส่ง Beacon ให้กับเพื่อนบ้านได้มากที่สุดและมีค่า RSSI สูงที่สุด หลังจากนั้นเพื่อนบ้านจะทำการโหวตว่าเพื่อนบ้านของตนโหนดไหนมีค่า RSSI สูงที่สุด ซึ่งโหนดที่ได้รับการโหวตมากที่สุดก็จะเป็นผู้ถูกเลือกในการกระจายข้อมูลลำดับถัดไป

6.2 ข้อจำกัด

การทำงานของโพรโทคอลจะได้รับผลกระทบจากปัญหาการชื่อมต่อแบบอสมมาตรมากที่สุดในสถานการณ์ที่มีความหนาแน่นของยานพาหนะในเครือข่ายต่ำ เพราะว่าระยะการส่งข้อมูลของโหนดนั้นจะสั้นกว่าระยะห่างระหว่างโหนด แต่ในกรณีความหนาแน่นสูงจะทำให้ระยะการส่งข้อมูลของโหนดนั้นอาจยาวกว่าระยะห่างระหว่างโหนดทำให้ได้รับผลกระทบจากปัญหาดังกล่าวน้อย ซึ่งกระบวนการแก้ไขปัญหาดังกล่าวด้วยอัลกอริทึมการโหวตอาร์เอสเอสไอจะเพิ่มประสิทธิภาพได้ไม่เท่ากับสถานการณ์ที่มีการเชื่อมต่อแบบสมมาตรด้วยรูปแบบสัญญาณเดียวกัน

การทำงานของกระบวนการแก้ไขดังกล่าวนั้นทดลองและให้ผลการทดลองดีบนการทดลองที่ส่งข้อความครั้งละ 1 ข้อความ แต่เมื่อมีการส่งข้อความเป็นจำนวนมากๆ อาจส่งผลให้การทำงานของโพรโทคอลเกิดปัญหาคอขวด (Bottle neck) เนื่องจากโหนดที่มีความแรงของสัญญาณสูงจะถูกเลือกให้กระจายข้อมูลบ่อยส่งผลให้อาจรับข้อความจำนวนมากจนเกิดการชนกัน หรือหน่วยเก็บความจำเต็มทำให้ข้อมูลอาจสูญหาย

ในการทดลองนี้พัฒนาบนเครือข่ายไร้สายแบบแอดฮอกสำหรับยานพาหนะแบบที่ไม่สนใจพลังงานในแต่ละโหนด ซึ่งถือว่าพลังงานมีไม่จำกัด แต่หากจะนำกระบวนการแก้ไขไปประยุกต์ใช้จริงกับงานทดลองหรือเครือข่ายที่สนใจเรื่องพลังงาน อาจทำให้เกิดปัญหาพลังงานหมดอย่างรวดเร็วได้ในโหนดที่มีสัญญาณแรงและถูกเลือกให้กระจายข้อมูลบ่อย

6.3 ข้อเสนอแนะ

โพรโทคอลที่มีการกระจายข้อมูลแบบเชื่อถือได้ที่พัฒนาบนเครือข่ายไร้สายแบบแอดฮอก สำหรับยานพาหนะส่วนใหญ่มักถูกออกแบบให้มีการกระจายข้อมูลแบบลดความซ้ำซ้อนของข้อมูล ซึ่งความแตกต่างของแต่ละโพรโทคอลก็คือ การเลือกผู้กระจายข้อมูลในลำดับถัดไปซึ่งจะช่วยให้โพรโทคอลสามารถกระจายข้อมูลไปให้กับโหนดต่าง ๆ ทั่วทั้งเครือข่าย อย่างไรก็ตามโพรโทคอลเหล่านั้นมักถูกพัฒนาด้วยการทดลองแบบจำลองเครือข่ายอย่างเดียว ทำให้ไม่ครอบคลุมการแก้ไขในการทำงานบนระบบจริงในบางปัจจัย ซึ่งปัญหาการเชื่อมต่อแบบอสมมาตรนั้นก็อาจส่งผลกระทบต่อการทำงานของโพรโทคอลต่างๆ เหล่านี้ เช่น POCA, EAEP, AckPBSM และ DV-Cast ดังนั้นกระบวนการในการแก้ไขปัญหาดังกล่าว ด้วยอัลกอริทึมการโหวตแบบอาร์เอสเอสไอนั้นก็สามารถที่จะนำไปประยุกต์ใช้ให้เข้ากับโพรโทคอลเหล่านั้นได้ทุกตัวที่มีการเลือกผู้กระจายข้อมูลในลำดับถัดไป

หากต้องการนำเอากระบวนการที่งานวิจัยนี้เสนอไปพัฒนาบนการทดลองที่มีการส่งข้อความจำนวนมาก ต้องมีการพิจารณาถึงหน่วยเก็บความจำในแต่ละโหนดเพื่อป้องกันการสูญหายของข้อความที่ส่งในระบบ อีกทั้งหากนำไปพัฒนาบนเครือข่ายที่สนใจเรื่องพลังงานก็ต้องมีการพิจารณาถึงการกระจายการใช้พลังงานในแต่ละโหนดให้เท่าๆ กันเพื่อการทำงานที่มีประสิทธิภาพของโพรโทคอล

รายการอ้างอิง

[1] M. Conti, and S. Giordano. Multihop Ad Hoc Networking: The Reality. IEEE Communications Magazine 45, 4, (April 2007): 88-95.

[2] PIARC. The intelligent transport system handbook, New York : Thomson Press, 2004.

[3] Varadhan, K. The network simulator NS-2 [Online]. 2011. Available from : http://nsnam.isi.edu/nsnam/ index.php/Main_Page [2011, March 13]

[4] Abraham, John. The network simulator NS-3 [Online]. 2011. Available from : http:/www.nsnam.org [2011, March 13]

[5] E. Weingartner, H.vom Lehn and K. Wehrle. A Performance Comparison of Recent Network Simulators. In IEEE International Conference on Communications (ICC'09), pp.1-5. 2009.

[6] T. R. Henderson, S. Roy, S. Floyd and G. F. Riley. ns-3 Project Goals [Online]. 2006. Available from : http://www.nsnam.org/docs/meetings/wns2/wns2-ns3.pdf [2006, October 10]

[7] N. Na Nakorn, and K. Rojviboonchai. DECA: Density-Aware Reliable Broadcasting in Vehicular Ad-Hoc Networks. In IEEE the 7th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON) 2010, Chiang Mai, Thailand, 2010.

[8] Jairath, A, and Khan, W.U. Modified AODV with QoS for Asymmetric MANET. In Computational Intelligence and Communication Networks (CICN), 2010, pp.265-268. 2010.

[9] Yong Bai, and Lan Chen. Extended Multicast Optimized Link State Routing Protocol in MANETs with Asymmetric Links. In Global Telecommunications Conference (GLOBECOM '07), 2007, pp.1312-1317, 2007.

[10] Birnstill Pascal, Di Pengfei, and Fuhrmann Thomas. Using asymmetric links to improve SSR's routing performance. In Ad Hoc Networking Workshop (Med-Hoc-Net), 2010, pp.1-6. 2010.

[11] Dongkyun Kim, Toh, C.-K., and Yanghee Choi. On supporting link asymmetry in mobile ad hoc networks. In Global Telecommunications Conference, 2001.(GLOBECOM '01), pp.2798-2803. 2001.

[12] Donkyun Kim, Toh, C.-K., and Yanghee Choi. RODA: a new dynamic routing protocol using dual paths to support asymmetric links in mobile ad hoc networks. In Computer Communications and Networks, 2000, Proceedings. pp.4-8. 2000.

[13] Jeongkeun Lee; and et al. RSS-based Carrier Sensing and Interference Estimation in 802.11 Wireless Networks. In Sensor, Mesh and Ad Hoc Communications and Networks (SECON '07), 2007, pp.491-500. 2007.

[14] German Aerospace Center (DLR). Simulation of Urban MObility (SUMO) [Online]. 2010. Available from : http://sumo.sourceforge.net [2011, September 28]

[15] Piorkowski, M. Traffic and Network Simulation Environment (TraNS) [Online]. 2008. Available from : http://trans.epfl.ch [2011, September 28]

[16] K. Na Nakorn and K. Rojviboonchai. POCA : Position-Aware Reliable Broadcasting in VANET. In Proc. the 2nd Asia-Pacific Conference of Information Processing (APCIP2010), 2010, Nanchang, China, 2010.

[17] M. Nekovee, and B. Bjarni Bogason. Reliable and efficient information dissemination in intermittenly connected vehicular ad hoc networks. In IEEE the 65th VTC'07-Spring, 2007, Dublin, Ireland, 2007.

[18] F. J. Ros, P. M. Ruiz, and I. Stojmenovic. Reliable and efficient broadcasting in vehicular ad hoc networks. In IEEE the 69th Vehicular Technology Conference (VTC'09), 2009, Barcelona, Spain, 2009.

[19] O. K. Tonguz, N. Wisitpongphan, F. Bai, P. Mudalige, and V. Sadekar. Broadcasting in VANET. In Proc. IEEE INFOCOM MOVE Workshop, 2007, Anchorage, USA, 2007.

[20] Kompfner, P. Cooperative Vehicle-Infrastructure System (CVIS) [online]. 2009. Available from: http://cvisproject.org [2010, October 28]

[21] Aggelou, George. Mobile Ad Hoc Networks. In McGraw-Hill Professional Engineering. New York: McGraw-Hill Professional Publishing, 2005.

[22] Jiang, D. and Delgrossi, L. IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments. In IEEE the 67th Vehicular Technology Conference (VTC'08-Spring), 2008, Singapore, 2008.

[23] A. Varga. The OMNeT++ discrete event simulation system [Online]. 2001. Available from : http://www.omnetpp.org [2011, December 11]

[24] QualNet, Scalable Network Technologies, Inc [Online]. 2008. Available from : http:// www.scalable-networks.com [2011, December 11]

[25] OPNET. Technologies Inc. OPNET modeler website [Online]. 2010. Available from : http://www.opnet.com/solution/network/_rd/modeler.html [2011, December 12]

ภาคผนวก

ภาคผนวก ก

การสร้างมอดูลใหม่บนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3

เนื่องจากโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 ถูกออกแบบมาเพื่อให้ทำงานได้ง่ายและเป็นระบบมากขึ้น ทำให้การออกแบบของโครงสร้างนั้นเป็นแบบเชิงวัตถุ การทำงานใส่ส่วนต่าง ๆ นั้นแบ่งกันออกมาอย่างชัดเจน ซึ่งการเพิ่มมอดูลใหม่บนโปรแกรมนั้นจึงต้องเพิ่มในส่วนต่าง ๆ ดังนี้

1. สร้างสารบบให้กับมอดูลใหม่ที่สร้างขึ้น โดยสร้างขึ้นภายใต้ contrib โดยจะต้องสร้างชื่อสารบบให้กับให้มีชื่อเดียวกับชื่อมอดูล เช่น ต้องการเพิ่มมอดูล awesome เข้าไปอยู่ที่ contrib คือ ~/tarballs/ns-allinone-3.9/ns-3.9/src/contrib นั้นจะต้องสร้างสารบบขึ้นดังนี้ ~/tarballs/ns-allinone-3.9/ns-3.9/src/contrib/awesome
2. เพิ่มคำสั่งเข้าไปใน waf script โดยสามารถเปิดไฟล์ได้จากที่อยู่ของ wscript เช่น ~/tarballs/ns-allinone-3.9/ns-3.9/src/wscript จะได้ออกมาเป็น

```
all_modules = (
    'core',
    'common',
    'simulator',
    'contrib',
    'node',
    'internet-stack',
    'devices/point-to-point',
    ...
    'mpi',
    'contrib/topology-read',
    'contrib/energy',
    )
```

ให้เพิ่มมอดูลเข้าไปเพิ่มต่อท้าย เช่น ต้องการเพิ่มมอดูล awesome เข้าไป ดังนี้

```
all_modules = (
    'core',
    'common',
    'simulator',
    'contrib',
    'node',
    'internet-stack',
    'devices/point-to-point',
    ...
    'mpi',
    'contrib/topology-read',
```

```
    'contrib/energy',
    'contrib/awesome', ← เพิ่มในส่วนนี้เข้าไป
    )
```

3. ไปที่ที่อยู่ของมอดูลที่เพิ่งสร้างใหม่และทำการสร้าง wscript ขึ้นในที่อยู่นั้น จากนั้นระบบจะทำการสร้างรายการของไฟล์ .cc และ .h

```
## -*- Mode: python; py-indent-offset: 4; indent-tabs-mode: nil;
coding: utf-8; -*-

def build(bld):
    obj = bld.create_ns3_module('dcf', ['node'])
    obj.source = [
          'one_of_your.cc',             ←
          'another_of_your.cc', ←
          ...
          ]
    headers = bld.new_task_gen('ns3header')
    headers.module = 'dcf'
    headers.source = [
         'one_of_your.h',               ←
         'another_of_your.h',    ←
         ...
    ]
```

4. หากต้องการสร้างไฟล์ helper ให้สร้างไฟล์นั้นโดยแบ่งออกเป็นสองไฟล์คือ ไฟล์ .cc และ .h โดยไฟล์ .cc มีตัวอย่างดังนี้

```
#include "deca-helper.h"
#include "ns3/deca-protocol.h"
#include "ns3/node-list.h"
#include "ns3/names.h"
#include "ns3/ipv4-list-routing.h"

namespace ns3 {

DecaHelper::DecaHelper() :
  Ipv4RoutingHelper ()
{
  m_agentFactory.SetTypeId ("ns3::deca::DecaProtocol");
}

DecaHelper*
DecaHelper::Copy (void) const
{
  return new DecaHelper (*this);
}
```

```
Ptr<Ipv4RoutingProtocol>
DecaHelper::Create (Ptr<Node> node) const
{
  Ptr<deca::DecaProtocol>                agent                =
m_agentFactory.Create<deca::DecaProtocol> ();
  node->AggregateObject (agent);
  return agent;
}

void
DecaHelper::Set (std::string name, const AttributeValue &value)
{
  m_agentFactory.Set (name, value);
}

}
```

ส่วนไฟล์ .h นั้นมีตัวอย่างดังนี้

```
#ifndef DECAHELPER_H_
#define DECAHELPER_H_

#include "ns3/object-factory.h"
#include "ns3/node.h"
#include "node-container.h"
#include "ipv4-routing-helper.h"

namespace ns3 {

class DecaHelper : public Ipv4RoutingHelper
{
public:
  DecaHelper();
/**
   * \internal
   * \returns pointer to clone of this OlsrHelper
   *
   *  This  method  is  mainly  for  internal  use  by  the  other
helpers;
   * clients are expected to free the dynamic memory allocated
by this method
   */
  DecaHelper* Copy (void) const;

 /**
   * \param node the node on which the routing protocol will run
   * \returns a newly-created routing protocol
   *
```

```
   *      This      method      will      be      called      by
ns3::InternetStackHelper::Install
   *
   *  TODO:  support  installing  DECA  on  the  subset  of  all
available IP interfaces
   */
  virtual   Ptr<Ipv4RoutingProtocol>   Create   (Ptr<Node>   node)
const;
/**
   * \param name the name of the attribute to set
   * \param value the value of the attribute to set.
   *
   *     This     method     controls     the     attributes     of
ns3::aodv::RoutingProtocol
   */
  void Set (std::string name, const AttributeValue &value);

private:
  ObjectFactory m_agentFactory;
};

}
#endif /* DECAHELPER_H_ */
```

นำไปใส่แฟ้ม helper แล้วแก้ไข wscript ในแฟ้มนั้นแล้วไปหา helper.source ดังต่อไปนี้

```
helper.source = [
        'node-container.cc',
        'net-device-container.cc',
        'wifi-helper.cc',
        'olsr-helper.cc',
```

ให้เพิ่มชื่อไฟล์ helper.cc ที่ได้สร้างขึ้นเข้าไปในส่วนนี้ และเพิ่มชื่อไฟล์ helper.h ที่ได้สร้างขึ้นในส่วนต่อไปนี้

```
headers.source = [
        'node-container.h',
        'net-device-container.h',
        'wifi-helper.h',
        'olsr-helper.h',
```

5. จากนั้นให้ใช้คำสั่ง ./waf build ในสารบบ ~/tarballs/ns-allinone-3.9/ns-3.9/ ระบบจะทำการสร้างไฟล์ awesome-module.h ขึ้นมาให้ การสร้างมอดูลใหม่ก็จะเสร็จสมบูรณ์พร้อมนำไปใช้งานได้

ภาคผนวก ข

ความแตกต่างของโพรโทคอล DECA ที่พัฒนาบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 กับ เวอร์ชั่น 3

การพัฒนาโพรโทคอลระหว่างโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 กับเวอร์ชั่น 3 นั้นมีความแตกต่างกันอย่างมาก โดยมีส่วนความแตกต่างหลัก ๆ ดังนี้

1. การเรียกใช้ Packet Header

<u>โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2</u>

การเรียกใช้ Packet Header นั้นจะเรียกใช้ได้ผ่าน Struct ของ header ดังนี้

- hdr_cmn
- hdr_ip
- hdr_deca_broadcast

ซึ่งสามารถที่จะตั้งค่าผ่านตัวแปรที่สร้างขึ้นมาได้เลย เพื่อที่จะเพิ่ม header ให้กับ packet ที่จะส่ง มีตัวอย่างดังนี้

```
struct hdr_cmn *ch = HDR_CMN(p);
struct hdr_ip *ih = HDR_IP(p);
struct hdr_deca_broadcast *br = HDR_DECA_BROADCAST(p);
//Fill out the broadcast packet
  ch->ptype() = PT_DECA;
  ch->size() = IP_HDR_LEN + br->size();
  ch->direction()=hdr_cmn::DOWN;
  ch->iface() = -2;
  ch->error() = 0;
  ch->addr_type() = NS_AF_NONE;
  ch->prev_hop_ = index;

  ih->saddr() = index;
  ih->daddr() = IP_BROADCAST;
  ih->sport() = RT_PORT;
  ih->dport() = RT_PORT;
  ih->ttl_ = 3;

  // Fill up DECA packet fields.
  br->br_type=DECATYPE_BR;
```

```
 br->prefer = nbtable.prefer_nbtable();
 br->src = index;
 br->seq = seqno;
 br->precursor = index;
 br->time_stamp = CURRENT_TIME;
 br->ttl = MAX_TTL;
```

โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3

การเรียกใช้ Packet header ของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 นั้นแตกต่างจากเวอร์ชั่น 2 อย่างมาก เพราะจะสร้าง packet ออกมาในเชิงวัตถุมากกว่า และเรียกใช้ object ของ header เพื่อทำการปรับค่า หลังจากนั้นก็เรียกใช้ object ของ packet เพื่อเพิ่มหรือลบ header มีตัวอย่างดังนี้

```
//create BCAST header
 BcastHeader bcastHeader;

 bcastHeader.SetSrc (t.GetSrc ());
 bcastHeader.SetSeqno (t.GetSeqNo ());
 bcastHeader.SetPrefer (t.GetPrefer ());
 bcastHeader.SetPrecursor (t.GetPrecursor ());
Ptr<Socket> socket = j->first;
                Ipv4InterfaceAddress iface = j->second;
                Ptr<Packet> packet = Create<Packet> ();
                packet->AddHeader (bcastHeader);
                TypeHeader tHeader (DECATYPE_BCAST);
                packet->AddHeader (tHeader);
```

เมื่อเปรียบเทียบกันระหว่างสองเวอร์ชั่นแล้วจะเห็นได้ว่าการพัฒนาโพรโทคอล DECA บนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 จะทำได้ง่ายกว่าเพราะออกแบบเป็นเชิงวัตถุมากขึ้น

2. ส่วนเชื่อมต่อระหว่างรหัสที่เขียนกับโปรแกรม

โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2

การเขียนโพรโทคอลที่ทำงานบนโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 นั้นต้องใช้ภาษาในการเขียนทั้ง C++ และ Tcl script ทำให้ต้องมีการเขียนส่วนเชื่อมต่อในการทำงานระหว่างรหัสที่เขียนกับโปรแกรมด้วยการเขียน Tcl เชื่อมต่อกับ C++ มีตัวอย่างดังนี้

```
//=============================================================
======================
//TCL Hooks
//-------------------------------------------------------------
----------------------
int hdr_deca::offset_;
```

```
static class DECAHeaderClass : public PacketHeaderClass {
public:
        DECAHeaderClass()                                               :
PacketHeaderClass("PacketHeader/DECA",sizeof(hdr_all_deca)) {
         bind_offset(&hdr_deca::offset_);
  }
} class_rtProtoDECA_hdr;

static class DECAclass : public TclClass {
public:
        DECAclass() : TclClass("Agent/DECA") {
         //printf("TclBinding called\n");
  }
        TclObject* create(int argc, const char*const* argv) {
         //printf("build tcl object called\n");
                return           (new           DECA((nsaddr_t)
Address::instance().str2addr(argv[4])));
        }
} class_rtProtoDECA;

int
DECA::command(int argc, const char*const* argv){
  if(argc == 2) {
         Tcl& tcl = Tcl::instance();

         if(strncasecmp(argv[1], "id", 2) == 0) {
                tcl.resultf("%d", index);
                return TCL_OK;
         }

         if(strncasecmp(argv[1], "start", 2) == 0) {
                btimer.expire((Event*) 0);
                return TCL_OK;
         }
  }
  else if(argc == 3) {
         if(strcmp(argv[1], "index") == 0) {
                index = atoi(argv[2]);
                return TCL_OK;
         }
         else   if(strcmp(argv[1],   "log-target")   ==   0   ||
strcmp(argv[1], "tracetarget") == 0) {
                logtarget               =               (Trace*)
TclObject::lookup(argv[2]);
                if(logtarget == 0)
                        return TCL_ERROR;
                return TCL_OK;
         }
         else if(strcmp(argv[1], "if-queue") == 0) {
```

```
                ifqueue              =              (PriQueue*)
TclObject::lookup(argv[2]);
                if(ifqueue == 0)
                        return TCL_ERROR;
                return TCL_OK;
        }
        else if (strcmp(argv[1], "port-dmux") == 0) {
                dmux_              =             (PortClassifier
*)TclObject::lookup(argv[2]);
                if (dmux_ == 0) {
                        fprintf (stderr, "%s: %s lookup of %s
failed\n", __FILE__,argv[1], argv[2]);
                        return TCL ERROR;
                }
                return TCL OK;
        }

  }
  return Agent::command(argc, argv);

}
```

โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3

การเขียนส่วนเชื่อมต่อระหว่างรหัสของโปรแกรมนั้นไม่จำเป็น เพราะว่าการทำงานของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 นั้นรองรับ C++ เท่านั้น

3. การสร้างส่วนแสดงการจำลองเครือข่าย

โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2

การสร้างส่วนแสดงการจำลองเครือข่ายของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 2 นั้นมีความซับซ้อนเพราะว่าต้องกำหนดลำดับชั้นต่าง ๆ ของเครือข่ายอย่างละเอียด ดังนี้

```
# Initialize Global Variables
set ns_          [new Simulator]
set tracefd     [open wireless_mitf.tr w]
$ns_ trace-all $tracefd
set namtrace [open wireless_mitf.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
# set up topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
# Create God
create-god $val(nn)
```

```
# New API to config node:
# 1. Create channel (or multiple-channels);
# 2. Specify channel in node-config (instead of channelType);
# 3. Create nodes for simulations.
puts "check point 1"
# Create channel #1 and #2
set chan_1_ [new $val(chan)]
set chan_2_ [new $val(chan)]
set chan_3_ [new $val(chan)]
set chan_4_ [new $val(chan)]
set chan_5_ [new $val(chan)]
puts "check point 1.1"
# Create node(0) "attached" to channel #1
# configure node, please note the change below.
$ns  node-config -adhocRouting $val(rp) \
         -llType $val(ll) \
         -macType $val(mac) \
         -ifqType $val(ifq) \
         -ifqLen $val(ifqlen) \
         -antType $val(ant) \
         -propType $val(prop) \
         -phyType $val(netif) \
         -topoInstance $topo \
         -agentTrace ON \
         -routerTrace ON \
         -macTrace ON \
         -movementTrace OFF \
         -channel $chan_1_
puts "check point 1.2"
for {set i 0} {$i < $val(nn)} {incr i} {
set node_($i) [$ns_ node]
}
# node_(1) can also be created with the same configuration, or
with a different
# channel specified.
#  Uncomment  below  two  lines  will  create  node_(1)  with  a
different channel.
#  $ns_ node-config \
#         -channel $chan_2_
puts "check point 2"
source $val(sc)
puts "check point 3"
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 10
$cbr_(0) set random_ 0
$cbr_(0) set maxpkts_ 10000
```

```
$cbr_(0) attach-agent $udp_(0)
$ns_ at 140.0 "$cbr_(0) start"
puts "check point 4"
#
# Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at 190.0 "$node_($i) reset";
}
$ns_ at 190.0 "stop"
$ns_ at 190.01 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
    global ns  tracefd
    $ns_ flush-trace
    close $tracefd
}

puts "Starting Simulation..."
$ns_ run
```

โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3

การสร้างส่วนแสดงการจำลองเครือข่ายนั้นสามารถที่จะเขียนด้วย C++ อย่างเดียวเท่านั้น และมีความเป็นวัตถุมากกว่า ดังนั้นทำให้สามารถที่จะกำหนดการทำงานได้ง่ายกว่าในแต่ละลำดับชั้น โดยเรียกใช้ object ที่เป็น helper ของแต่ละลำดับชั้น ดังนี้

```
class DecaExample
{
public:
  DecaExample ();
  /// Configure script parameters, \return true on successful
configuration
         bool Configure (int argc, char **argv);
         /// Run simulation
         void Run ();
         /// Report results
         void Report (std::ostream & os);

private:
  ///\name parameters
         //\{
         /// Number of nodes
  uint32_t size;
  /// Distance between nodes, meters
         double step;
  /// Simulation time, seconds
  double totalTime;
```

```
  /// Write per-device PCAP traces if true
  bool pcap;
  //\}

  ///\name network
  //\{
  NodeContainer nodes;
  NetDeviceContainer devices;
  Ipv4InterfaceContainer interfaces;
  //\}
private:
  void CreateNodes ();
  void CreateDevices ();
  void InstallInternetStack ();
  void InstallApplications ();

};

int main (int argc, char **argv)
{

  DecaExample test;
  if     (!     test.Configure(argc,     argv))     NS_FATAL_ERROR
("Configuration failed. Aborted.");



  test.Run ();
  test.Report (std::cout);

  //*(char *)0 = 0;
  return 0;
}
DecaExample::DecaExample ():
  size (4),
  step (50),
  totalTime (10),
  pcap (true)
{
}
bool
DecaExample::Configure (int argc, char **argv)
{
  // Enable DECA logs by default. Comment this if too noisy
  // LogComponentEnable("DecaProtocol", LOG_LEVEL_ALL);

  SeedManager::SetSeed(12345);
  CommandLine cmd;
```

```
  cmd.AddValue ("pcap", "Write PCAP traces.", pcap);
  cmd.AddValue ("size", "Number of nodes", size);
  cmd.AddValue ("time", "Simulation time, s.", totalTime);
  cmd.AddValue ("step", "Grid step, m", step);

  cmd.Parse (argc, argv);
  return true;
}
void
DecaExample::Run()
{
  CreateNodes ();
  CreateDevices ();
  InstallInternetStack ();
  InstallApplications ();

  std::cout << "Starting simulation for " << totalTime << " s
...\n";

  Simulator::Stop (Seconds (totalTime));
  Simulator::Run ();
  Simulator::Destroy ();
}
void
DecaExample::Report (std::ostream &)
{
}
void
DecaExample::CreateNodes ()
{
  std::cout << "Creating " << (unsigned)size << " nodes " <<
step << " m apart.\n";
  nodes.Create (size);
  // Name nodes
  for (uint32_t i = 0; i < size; ++i)
     {
       std::ostringstream os;
       os << "node-" << i;
       Names::Add (os.str (), nodes.Get (i));
     }
}
void
DecaExample::CreateDevices ()
{
  NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
  wifiMac.SetType ("ns3::AdhocWifiMac");
  YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
  YansWifiChannelHelper              wifiChannel              =
YansWifiChannelHelper::Default ();
```

```
  wifiPhy.SetChannel (wifiChannel.Create ());
  WifiHelper wifi = WifiHelper::Default ();
  wifi.SetRemoteStationManager  ("ns3::ConstantRateWifiManager",
"DataMode",  StringValue  ("OfdmRate6Mbps"),  "RtsCtsThreshold",
UintegerValue (0));
  devices = wifi.Install (wifiPhy, wifiMac, nodes);

  if (pcap)
    {
      wifiPhy.EnablePcapAll (std::string ("deca"));
    }
}
void
DecaExample::InstallInternetStack ()
{
  DecaHelper deca;
  InternetStackHelper stack;
  stack.SetRoutingHelper (deca);
  stack.Install (nodes);
  Ipv4AddressHelper address;
  address.SetBase ("10.0.0.0", "255.0.0.0");
  interfaces = address.Assign (devices);
}
void
DecaExample::InstallApplications ()
{
  V4PingHelper ping (interfaces.GetAddress (1));
  ping.SetAttribute ("Verbose", BooleanValue (true));

  ApplicationContainer p = ping.Install (nodes.Get (0));
  p.Start (Seconds (0));
  p.Stop (Seconds (totalTime) - Seconds(0.001));
}
```

4. การเรียกใช้ Timer

<u>โปรแกรมจำลองเครือข่ายเวอร์ชั่น 2</u>

การเรียกใช้ timer เมื่อกำหนดเวลาในการทำงานของแต่ละฟังก์ชันนั้นสามารถที่จะเรียกใช้ผ่าน agent ที่สร้างขึ้นมา และ event ซึ่งสามารถกำหนดได้ง่าย ตัวอย่างดังนี้

```
void
PacketLifeTimer_deca::expire(Event*) {
  agent->packet_expire_chk(CURRENT_TIME);
  if(agent->next_packet_expire_time()>0)resched(agent-
>next_packet_expire_time());
}
```

```
void
BeaconMsgTimer_deca::expire(Event*) {

  agent->beacon();
  resched(agent->beacon_interval_cal());
}

void
RebroadcastTimer_deca::expire(Event*) {
  agent->timeout_ttable(CURRENT_TIME);
  if(agent->next_rebroadcast_time()>0)resched(agent-
>next rebroadcast time());
}
```

โปรแกรมจำลองเครือข่ายเวอร์ชั่น 3

การเรียกใช้ time ของโปรแกรมจำลองเครือข่ายเวอร์ชั่น 3 นั้นสามารถเรียกใช้ผ่าน class Simulator กับ class Timer ได้ ซึ่งมีการกำหนดค่าว่าต้องการทำฟังก์ชั่นไหนก่อน แล้วก็ทำการตั้งค่าเวลาที่ต้องการ ตัวอย่างดังนี้

```
// timer beacon
  m_btimer.SetFunction (&DecaProtocol::BeaconTimerExpire, this);
  m_btimer.Schedule (MilliSeconds (UniformVariable ().GetInteger
(0, 100)));
```

ประวัติผู้เขียนวิทยานิพนธ์

นายณัฐวิทย์ กมลธรรม เกิดเมื่อวันที่ 29 กันยายน พ.ศ. 2531 ที่จังหวัดสมุทรสงคราม สำเร็จการศึกษาระดับประถมศึกษาจากโรงเรียนดรุณานุกูล จังหวัดสมุทรสงคราม สำเร็จการศึกษาระดับมัธยมศึกษาจากโรงเรียนสาธิตมหาวิทยาลัยศรีนครินทรวิโรฒ ปทุมวัน กรุงเทพมหานคร สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2553 (เกียรตินิยมอันดับ 2) และเข้าศึกษาในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2554