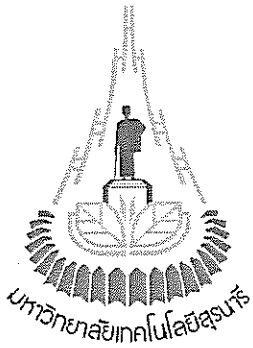


การจำลองการเคลื่อนที่ของโหนดในโครงข่ายเคลื่อนที่แบบแอดฮอคด้วยภาพกราฟฟิก

(GUI for Studying Random Way Point Mobility in MANETs)

โดย

นางสาวชลธิดา สุธรรม รหัส B4601760

นางสาวนาฏนิภา ชูจิตารมย์ รหัส B4604365

นายพงษ์นรินทร์ ศรีพลอย รหัส B4605614

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา 427499 โครงงานวิศวกรรมโทรคมนาคม

ประจำภาคการศึกษาที่ 3 ปีการศึกษา 2549

หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม-2546

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

โครงงาน	การจำลองการเคลื่อนที่ของโหนดในโครงข่ายเคลื่อนที่แบบแอดฮอคด้วยภาพกราฟฟิก (GUI for Studying Random Way Point Mobility in MANETs)
โดย	นางสาวชลธิดา สุธรรม นางสาวนาฏนิภา ชูจิตารมย์ นายพงษ์นรินทร์ ศรีพลอย
อาจารย์ที่ปรึกษา	อาจารย์ ดร.วิภาวี หัตถกรรม
สาขาวิชา	วิศวกรรมโทรคมนาคม
ภาคการศึกษาที่	3/ 2549

บทคัดย่อ

ปัจจุบันการเชื่อมต่อไร้สายมีความสำคัญมากต่อการดำรงชีวิตในยุคโลกาภิวัฒน์ ไม่ว่าจะเป็นการรับส่งข้อมูลกันในระยะใกล้หรือไกล การเชื่อมต่อไร้สายสามารถทำให้การรับส่งข้อมูลนั้นมีความสะดวกและรวดเร็วยิ่งขึ้น โดยโครงข่ายเคลื่อนที่แบบแอดฮอค (Mobile Ad Hoc Networks : MANETs) เป็นระบบการสื่อสารไร้สายในอนาคตซึ่งแต่ละโหนดสามารถติดต่อสื่อสารกันในขณะที่เคลื่อนที่ได้ โดยไม่จำเป็นต้องพึ่งพาสถานีฐาน (Base Station) ในการพัฒนาระบบโครงข่ายเคลื่อนที่แบบแอดฮอคจากระบบจริงด้วยสถานการณ์จริงนั้นค่อนข้างยุ่งยาก ทั้งนี้เพราะว่าการศึกษาจำลองแบบโครงข่ายเคลื่อนที่แบบแอดฮอคประสบปัญหาเกี่ยวกับรูปร่างโครงข่ายซึ่งไม่สามารถแสดงออกมาเป็นภาพจำลองได้ ส่งผลให้ยากต่อการศึกษาระบบการทำงานในโครงข่ายที่มีผู้ใช้จำนวนมากและมีหลายเส้นทางในการเชื่อมต่อระหว่างโหนดต้นทางและโหนดปลายทาง ดังนั้นคณะผู้จัดทำจึงได้ทำการออกแบบโปรแกรมจำลองการเคลื่อนที่ของโหนดในแบบสุ่มสำหรับโครงข่ายเคลื่อนที่แบบแอดฮอค (Random Way Point Model)เพื่อแสดงเส้นทางการติดต่อกันระหว่างโหนดต้นทางและโหนดปลายทางในรูปแบบของจียูไอ (Graphic User Interface : GUI) ซึ่งจะแสดงเส้นทางการติดต่อที่สั้นที่สุด (Shortest Path Routing) โดยโปรแกรมนี้จะเป็นประโยชน์แก่ผู้ที่สนใจศึกษาเกี่ยวกับการเชื่อมต่อในโครงข่ายเคลื่อนที่แบบแอดฮ

อคในอนาคตข้างหน้า

กิตติกรรมประกาศ

โครงงานนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี
เนื่องจากได้รับการสนับสนุนจากบุคคลหลายฝ่ายดังต่อไปนี้

- อาจารย์ ดร.วิภาวี หัตถกรรม อาจารย์ที่ปรึกษาโครงงาน ผู้ให้คำปรึกษาและช่วยเหลือเป็นอย่างดี
- คณาจารย์ประจำสาขาวิศวกรรมโทรคมนาคม มหาวิทยาลัยเทคโนโลยีสุรนารี
- คุณวิภาดา นฤพิพัฒน์ นักศึกษาปริญญาโท ผู้ให้คำปรึกษาและคำแนะนำ
- คุณประพล จาระตะคุ ผู้สนับสนุนในจัดหาอุปกรณ์ในการทำโครงงาน
- คุณมณีรัตน์ ทุมพงษ์ เลขานุการประจำสาขาวิศวกรรมโทรคมนาคม ที่อำนวยความสะดวกเรื่องต่างๆ

ขอขอบคุณเพื่อนๆ สาขาวิศวกรรมโทรคมนาคม ที่คอยให้กำลังใจเสมอมา และขอบคุณเพื่อนสาขาวิศวกรรมคอมพิวเตอร์ ที่ให้คำปรึกษาเกี่ยวกับการศึกษาโปรแกรมทำให้การทำงานสำเร็จได้รวดเร็วยิ่งขึ้น

ท้ายสุด ขอกราบขอบพระคุณ บิดามารดาและครอบครัวของข้าพเจ้า ที่คอยให้กำลังใจและสนับสนุนส่งเสริมการศึกษามาโดยตลอด

หากเกิดข้อผิดพลาดประการใด คณะผู้จัดทำขอกราบอภัยมา ณ ที่นี้ด้วย

นางสาวชลธิดา สุธรรม รหัส B4601760
นางสาวนาฏนิภา ชูจิตารมย์ รหัส B4604365
นายพงษ์นรินทร์ ศรีพลอย รหัส B4605614

สารบัญ

หน้า

หน้า

สารบัญรูป

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงการ

โครงข่ายเคลื่อนที่แบบแอดฮอค (Mobile Adhoc Networks : MANETs) เป็นระบบการสื่อสารไร้สายในอนาคต ซึ่งแต่ละโหนด สามารถติดต่อสื่อสารกันในขณะที่เคลื่อนที่ได้โดยไม่จำเป็นต้องพึ่งพาสถานีฐาน (Base Station) หรือ หน่วยงานกลาง (Central Administration) ในการพัฒนาระบบ โครงข่ายเคลื่อนที่แบบแอดฮอค จากระบบจริงด้วยสถานการณ์จริงนั้น ค่อนข้างยุ่งยาก ทั้งนี้เพราะว่าการศึกษาจำลองแบบโครงข่ายเคลื่อนที่แบบแอดฮอค ประสบปัญหาเกี่ยวกับรูปร่างโครงข่ายซึ่งไม่สามารถแสดงออกมาเป็นภาพจำลองได้ ส่งผลให้ยากต่อการศึกษาระบบการทำงานในโครงข่ายที่มีผู้ใช้จำนวนมาก และมีหลายเส้นทางในการเชื่อมต่อระหว่างโหนดต้นทางและโหนดปลายทางโดยระยะทางในการสื่อสารนั้นมีผลต่อความเร็วในการส่งผ่านข้อมูล เนื่องจากระยะทางที่สั้นกว่าจะส่งผ่านข้อมูลได้รวดเร็วกว่า

ดังนั้นเราจึงออกแบบโปรแกรมจำลองการเคลื่อนที่ของโหนดในโครงข่ายเคลื่อนที่แบบแอดฮอคด้วยภาพกราฟฟิก (GUI for Studying Random Way Point Mobility in MANETs) เพื่อแสดงเส้นทางที่สั้นที่สุดที่สามารถติดต่อสื่อสารได้ในโครงข่ายเคลื่อนที่แบบแอดฮอค

1.2 วัตถุประสงค์

1. เพื่อศึกษาโครงข่ายเคลื่อนที่แบบแอดฮอค
2. เพื่อออกแบบการเขียนโปรแกรมแสดงการเคลื่อนที่ของโหนดที่มีการเคลื่อนที่แบบสุ่มของโครงข่ายเคลื่อนที่แบบแอดฮอค
3. เพื่อศึกษาขั้นตอนวิธีการหาเส้นทางที่สั้นที่สุด

1.3 ขอบเขตงาน

1. ศึกษาลักษณะการเชื่อมต่อไร้สายแบบแอดฮอค
2. ศึกษาการใช้โปรแกรมเดฟซีพลัสพลัส (Dev C++)
3. ออกแบบโปรแกรมแสดงจียูไอ (Graphic User Interface : GUI) ของโครงข่ายเคลื่อนที่แบบแอดฮอค

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาค้นคว้าหาข้อมูลเกี่ยวกับโครงข่ายเคลื่อนที่แบบแอดฮอค
2. ศึกษาโปรแกรม Dev C++
3. เขียนโครงการและเสนอกับอาจารย์ที่ปรึกษา
4. เขียนโปรแกรมและแสดงการเคลื่อนที่ของโหนด ที่มีการเคลื่อนที่แบบสุ่มของโครงข่าย เคลื่อนที่แบบแอดฮอค
5. นำเสนออาจารย์เพื่อปรับปรุงและแก้ไขงาน
6. สรุปผลการดำเนินงาน
7. เขียนรายงานผลการดำเนินงาน
8. นำเสนอโครงงานแก่อาจารย์

บทที่ 2
ความรู้เบื้องต้นของการเชื่อมต่อแบบไร้สาย IEEE 802.11 และโครงข่ายเคลื่อนที่แบบแอดฮอค

2.1 ความรู้เบื้องต้นเกี่ยวกับมาตรฐาน IEEE 802.11

มาตรฐาน IEEE 802.11 ซึ่งได้รับการตีพิมพ์ครั้งแรกเมื่อปีพ.ศ.2540 โดย IEEE (The Institute of Electronics and Electrical Engineers) และเป็นเทคโนโลยีสำหรับเครือข่ายแลนแบบไร้สาย (Wireless LAN : WLAN) ที่นิยมใช้กันอย่างแพร่หลายมากที่สุด คือข้อกำหนด (Specification) สำหรับอุปกรณ์ WLAN ในส่วนของฟิซิคัลเลเยอร์ (Physical Layer) และมีเดียแอกเซสคอนโทรลเลเยอร์ (Media Access Control Layer : MAC Layer) โดยในส่วนของ Physical Layer มาตรฐาน IEEE 802.11 ได้กำหนดให้อุปกรณ์มีความสามารถในการรับส่งข้อมูลด้วยความเร็ว 1, 2, 5.5, 11 และ 54 Mbps โดยมีสื่อ 3 ประเภทให้เลือกใช้ได้แก่ คลื่นวิทยุที่ความถี่สาธารณะ 2.4 และ 5 GHz, และ อินฟราเรด (Infrared) (1 และ 2 Mbps เท่านั้น) สำหรับในส่วนของ MAC Layer มาตรฐาน IEEE 802.11 ได้กำหนดให้มีกลไกการทำงาน ที่เรียกว่า ซีเอสเอ็มเอซีเอ (Carrier Sense Multiple Access/Collision Avoidance : CSMA/CA) ซึ่งมีความคล้ายคลึงกับหลักการซีเอสเอ็มเอซีดี (Carrier Sense Multiple Access/Collision Detection : CSMA/CD) ของมาตรฐาน IEEE 802.3 Ethernet ซึ่งเป็นที่นิยมใช้กันทั่วไปในเครือข่ายแลน (LAN) แบบใช้สายนำสัญญาณ นอกจากนี้ในมาตรฐาน IEEE802.11 ยังกำหนดให้มีทางเลือกสำหรับสร้างความปลอดภัยให้กับเครือข่าย IEEE 802.11 WLAN โดยกลไกการเข้ารหัสข้อมูล (Encryption) และการตรวจสอบผู้ใช้ (Authentication) ที่มีชื่อเรียกว่า WEP (Wired Equivalent Privacy) ด้วย

วิวัฒนาการของมาตรฐาน IEEE 802.11

มาตรฐาน IEEE 802.11 ได้รับการตีพิมพ์ครั้งแรกในปี พ.ศ. 2540 ซึ่งอุปกรณ์ตามมาตรฐานดังกล่าวจะมีความสามารถในการรับส่งข้อมูลด้วยความเร็ว 1 และ 2 Mbps ด้วยสื่อ อินฟราเรด (Infrared) หรือคลื่นวิทยุที่ความถี่ 2.4 GHz และมีกลไก WEP ซึ่งเป็นทางเลือกสำหรับสร้างความปลอดภัยให้กับเครือข่าย WLAN ได้ในระดับหนึ่ง เนื่องจากมาตรฐาน IEEE 802.11 เวอร์ชันแรกเริ่มมีประสิทธิภาพค่อนข้างต่ำและไม่มีการรองรับคุณภาพของการให้บริการ (Quality of

Service : QoS) ซึ่งเป็นที่ต้องการของตลาด อีกทั้งกลไกรักษาความปลอดภัยที่ใช้ยังมีช่องโหว่อยู่มาก IEEE จึงได้จัดตั้งคณะทำงาน (Task Group) ขึ้นมาหลายชุดด้วยกันเพื่อทำการปรับปรุงเพิ่มเติมมาตรฐานให้มีศักยภาพสูงขึ้น โดยคณะทำงานกลุ่มที่มีผลงานที่น่าสนใจและเป็นที่รู้จักกันดีได้แก่ IEEE 802.11a, IEEE 802.11b, IEEE 802.11e, IEEE 802.11g, และ IEEE 802.11i

- **IEEE 802.11b**

คณะทำงานชุด IEEE 802.11b ได้ตีพิมพ์มาตรฐานเพิ่มเติมนี้เมื่อปี พ.ศ. 2542 ซึ่งเป็นที่รู้จักกันดีและใช้งานกันอย่างแพร่หลายมากที่สุด มาตรฐาน IEEE 802.11b ใช้เทคโนโลยีที่เรียกว่า ซีซีเค (Complimentary Code Keying : CCK) ผนวกกับดีทริปเปิ้ลเอส (Direct Sequence Spread Spectrum : DSSS) เพื่อปรับปรุงความสามารถของอุปกรณ์ให้รับส่งข้อมูลได้ด้วยความเร็วสูงสุดที่ 11 Mbps ผ่านคลื่นวิทยุความถี่ 2.4 GHz ซึ่งเป็นย่านความถี่ที่เรียกว่า ไอเอสเอ็ม (Industrial, Scientific and Medical : ISM) ซึ่งถูกจัดสรรไว้อย่างสากลสำหรับการใช้งานอย่างสาธารณะด้านวิทยาศาสตร์ อุตสาหกรรม และการแพทย์ โดยอุปกรณ์ที่ใช้ความถี่ย่านนี้ เช่น IEEE 802.11, บลูทูธ (Bluetooth), โทรศัพท์ไร้สาย, และเตาไมโครเวฟ ส่วนใหญ่แล้วอุปกรณ์ IEEE 802.11 WLAN ที่ใช้กันอยู่ในปัจจุบันจะเป็นอุปกรณ์ตามมาตรฐาน IEEE 802.11b นี้และใช้เครื่องหมายการค้าที่รู้จักกันดีในนามไวไฟ (Wi-Fi) ซึ่งเครื่องหมายการค้าดังกล่าวถูกกำหนดขึ้นโดยสมาคมดับเบิ้ลยูอีซีเอ (Wireless Ethernet Compatibility Alliance : WECA) โดยอุปกรณ์ที่ได้รับเครื่องหมายการค้าดังกล่าวได้ผ่านการตรวจสอบแล้วว่าเป็นไปตามมาตรฐาน IEEE 802.11b และสามารถนำไปใช้งานร่วมกับอุปกรณ์ยี่ห้ออื่นๆ ที่ได้รับเครื่องหมาย Wi-Fi ได้

- **IEEE 802.11a**

คณะทำงานชุด IEEE 802.11a ได้ตีพิมพ์มาตรฐานเพิ่มเติมนี้เมื่อปี พ.ศ. 2542 มาตรฐาน IEEE 802.11a ใช้เทคโนโลยีที่เรียกว่าโอเอฟดีเอ็ม (Orthogonal Frequency Division Multiplexing : OFDM) เพื่อปรับปรุงความสามารถของอุปกรณ์ให้รับส่งข้อมูลได้ด้วยความเร็วสูงสุดที่ 54 Mbps แต่จะใช้คลื่นวิทยุที่ความถี่ 5 GHz ซึ่งเป็นย่านความถี่สาธารณะสำหรับใช้งานในประเทศสหรัฐอเมริกาที่มีสัญญาณรบกวนจากอุปกรณ์อื่น

น้อยกว่าในย่านความถี่ 2.4 GHz อย่างไรก็ตามข้อเสียหนึ่งของมาตรฐาน IEEE 802.11a ที่ใช้คลื่นวิทยุที่ความถี่ 5 GHz คือในบางประเทศย่านความถี่ดังกล่าวไม่สามารถนำมาใช้งานได้อย่างสาธารณะ ตัวอย่างเช่น ประเทศไทยไม่อนุญาตให้มีการใช้งานอุปกรณ์ IEEE 802.11a เนื่องจากความถี่ย่าน 5 GHz ได้ถูกจัดสรรสำหรับกิจการอื่นอยู่ก่อนแล้ว นอกจากนี้ข้อเสียอีกอย่างหนึ่งของอุปกรณ์ IEEE 802.11a WLAN ก็คือรัศมีของสัญญาณมีขนาดค่อนข้างสั้น (ประมาณ 30 เมตร ซึ่งสั้นกว่ารัศมีสัญญาณของอุปกรณ์ IEEE 802.11b WLAN ที่มีขนาดประมาณ 100 เมตร สำหรับการใช้งานภายในอาคาร) อีกทั้งอุปกรณ์ IEEE 802.11a WLAN ยังมีราคาสูงกว่า IEEE 802.11b WLAN ด้วย ดังนั้นอุปกรณ์ IEEE 802.11a WLAN จึงได้รับความนิยมน้อยกว่า IEEE 802.11b WLAN มาก

- **IEEE 802.11g**

คณะทำงานชุด IEEE 802.11g ได้ใช้นำเทคโนโลยี OFDM มาประยุกต์ใช้ในช่องสัญญาณวิทยุความถี่ 2.4 GHz ซึ่งอุปกรณ์ IEEE 802.11g WLAN มีความสามารถในการรับส่งข้อมูลด้วยความเร็วสูงสุดที่ 54 Mbps ส่วนรัศมีสัญญาณของอุปกรณ์ IEEE 802.11g WLAN จะอยู่ระหว่างรัศมีสัญญาณของอุปกรณ์ IEEE 802.11a และ IEEE 802.11b เนื่องจากความถี่ 2.4 GHz เป็นย่านความถี่สาธารณะสากล อีกทั้งอุปกรณ์ IEEE 802.11g WLAN สามารถทำงานร่วมกับอุปกรณ์ IEEE 802.11b WLAN ได้แบกเวิร์ดคอมพาทิเบิ้ล (Backward Compatible) ดังนั้นจึงมีแนวโน้มสูงว่าอุปกรณ์ IEEE 802.11g WLAN จะได้รับความนิยมอย่างแพร่หลายหากมีราคาไม่แพงจนเกินไปและน่าจะมาแทนที่ IEEE 802.11b ในที่สุด ตามแผนการแล้วมาตรฐาน IEEE 802.11g จะได้รับการตีพิมพ์ประมาณช่วงกลางปี พ.ศ. 2546

- **IEEE 802.11e**

คณะทำงานชุดนี้ได้รับมอบหมายให้ปรับปรุง MAC Layer ของ IEEE 802.11 เพื่อให้สามารถรองรับการใช้งานหลักการ QoS สำหรับการประยุกต์ใช้งาน (Application) เกี่ยวกับมัลติมีเดีย (Multimedia) เนื่องจาก IEEE 802.11e เป็นการปรับปรุง MAC Layer ดังนั้นมาตรฐานเพิ่มเติมนี้จึงสามารถนำไปใช้กับอุปกรณ์ IEEE 802.11 WLAN ทุกเวอร์ชันได้ แต่อย่างไรก็ตามการทำงานของคณะทำงานชุดนี้ยังไม่แล้วเสร็จในขณะนี้ (พฤษภาคม พ.ศ. 2546)

- **IEEE 802.11i**

คณะทำงานชุดนี้ได้รับมอบหมายให้ปรับปรุง MAC Layer ของ IEEE 802.11 ในด้านความปลอดภัย เนื่องจากเครือข่าย IEEE 802.11 WLAN มีช่องโหว่อยู่มากโดยเฉพาะอย่างยิ่งการเข้ารหัสข้อมูล (Encryption) ด้วยคีย์ (Key) ที่ไม่มีการเปลี่ยนแปลง คณะทำงานชุด IEEE 802.11i จะนำเอาเทคนิคขั้นสูงมาใช้ในการเข้ารหัสข้อมูลด้วย Key ที่มีการเปลี่ยนค่าอยู่เสมอและการตรวจสอบผู้ใช้ที่มีความปลอดภัยสูง มาตรฐานเพิ่มเติมนี้จึงสามารถนำไปใช้กับอุปกรณ์ IEEE 802.11 WLAN ทุกเวอร์ชันได้ แต่อย่างไรก็ตามการทำงานของคณะทำงานชุดนี้ยังไม่แล้วเสร็จในขณะนี้ (พฤษภาคม พ.ศ. 2546)

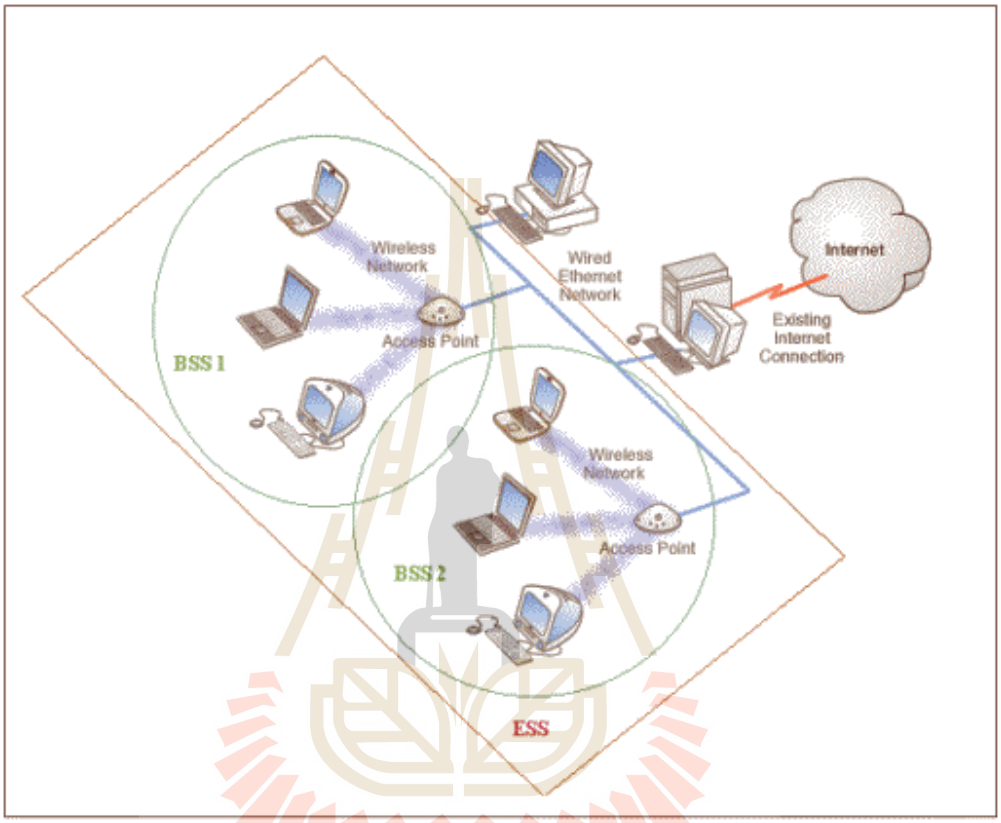
ลักษณะการเชื่อมต่อของอุปกรณ์ IEEE 802.11 WLAN

มาตรฐาน IEEE 802.11 ได้กำหนดลักษณะการเชื่อมต่อของอุปกรณ์ภายในเครือข่าย WLAN ไว้ 2 ลักษณะคือโหมดอินฟราสตรักเชอร์ (Infrastructure) และโหมดแอดฮอค (Ad Hoc) หรือเพียร์ทูเพียร์ (Peer-to-Peer)

1. **โหมดอินฟราสตรักเชอร์ (Infrastructure)**

โดยทั่วไปแล้วอุปกรณ์ในเครือข่าย IEEE 802.11 WLAN จะเชื่อมต่อกันในลักษณะของโหมด Infrastructure ซึ่งเป็นโหมดที่อนุญาตให้อุปกรณ์ภายใน WLAN สามารถเชื่อมต่อกับเครือข่ายอื่นได้ ในโหมด Infrastructure นี้เครือข่าย IEEE 802.11 WLAN จะประกอบไปด้วยอุปกรณ์ 2 ประเภท ได้แก่ สถานีผู้ใช้ (Client Station) ซึ่งก็คืออุปกรณ์คอมพิวเตอร์ (เดสท็อป (Desktop), แลพท็อป (Laptop), หรือพีดีเอ (PDA) ต่างๆ) ที่มีอุปกรณ์ (Client Adapter) เพื่อใช้รับส่งข้อมูลผ่าน IEEE 802.11 WLAN และสถานีแม่ข่าย (Access Point) ซึ่งทำหน้าที่ต่อเชื่อมสถานีผู้ใช้เข้ากับเครือข่ายอื่น (ซึ่งโดยปกติจะเป็นเครือข่าย IEEE 802.3 Ethernet LAN) การทำงานในโหมด Infrastructure มีพื้นฐานมาจากระบบเครือข่ายโทรศัพท์มือถือ กล่าวคือสถานีผู้ใช้จะสามารถรับส่งข้อมูลโดยตรงกับสถานีแม่ข่ายที่ให้บริการแก่สถานีผู้ใช้นั้นอยู่เท่านั้น ส่วนสถานีแม่ข่ายจะทำหน้าที่ส่งต่อ (Forward)

ข้อมูลที่ได้รับจากสถานีผู้ใช้ไปยังจุดหมายปลายทางหรือส่งต่อข้อมูลที่ได้รับจากเครือข่ายอื่นมายังสถานีผู้ใช้ เพื่อให้เข้าใจลักษณะการเชื่อมต่อของอุปกรณ์ในโหมดนี้ดูรูปที่ 1



รูปที่ 1 แสดงบีเอสเอส (Basic Service Set : BSS) และอีเอสเอส (Extended Service Set : ESS)

บีเอสเอส (Basic Service Set : BSS)

Basic Service Set (BSS) หมายถึง บริเวณของเครือข่าย IEEE 802.11 WLAN ที่มีสถานีแม่ข่าย 1 สถานี ซึ่งสถานีผู้ใช้ภายในขอบเขตของ BSS นี้ทุกสถานีจะต้องสื่อสารข้อมูลผ่านสถานีแม่ข่ายดังกล่าวเท่านั้น

อีเอสเอส (Extended Service Set : ESS)

Extended Service Set (ESS) หมายถึง บริเวณของเครือข่าย IEEE 802.11 WLAN ที่ประกอบด้วย BSS มากกว่า 1 BSS ซึ่งได้รับการเชื่อมต่อเข้าด้วยกัน สถานีผู้ใช้สามารถเคลื่อนย้ายจาก BSS หนึ่งไปอยู่ในอีก BSS หนึ่งได้โดย BSS เหล่านี้จะทำการ Roaming หรือติดต่อสื่อสารกันเพื่อทำการโอนย้ายการให้บริการสำหรับสถานีผู้ใช้ดังกล่าว

2. โหมดแอดฮอค (Ad Hoc) หรือเพียร์ทูเพียร์ (Peer-to-Peer)

เครือข่าย IEEE 802.11 WLAN ในโหมด Ad Hoc หรือ Peer-to-Peer เป็นเครือข่ายที่ปิดคือไม่มีสถานีแม่ข่ายและไม่มีการเชื่อมต่อกับเครือข่ายอื่น บริเวณของเครือข่าย IEEE 802.11 WLAN ในโหมด Ad Hoc จะถูกเรียกว่าไอบีเอสเอส (Independent Basic Service Set : IBSS) ซึ่งสถานีผู้ใช้หนึ่งสามารถติดต่อสื่อสารข้อมูลกับสถานีผู้ใช้อื่นๆในเขต IBSS เดียวกันได้โดยตรงโดยไม่ต้องผ่านสถานีแม่ข่าย แต่สถานีผู้ใช้จะไม่สามารถรับส่งข้อมูลกับเครือข่ายอื่นๆได้

รูปที่ 2 แสดงการทำงานในโหมดแอดฮอค (Ad Hoc) หรือเพียร์ทูเพียร์ (Peer-to-Peer)

การเข้าใช้ช่องสัญญาณด้วยกลไกซีเอสเอ็มเอซีเอ (Carrier Sense Multiple Access/Collision Avoidance : CSMA/CA)

บทบาทหนึ่งของ MAC Layer ในมาตรฐาน IEEE 802.11 คือการจัดสรรการเข้าใช้ช่องสัญญาณซึ่งแต่ละสถานีใน BSS หรือ IBSS จะต้องแบ่งกันใช้ช่องสัญญาณที่ถูกกำหนดมาสำหรับใช้งานร่วมกันอย่างเป็นธรรม มาตรฐาน IEEE 802.11 ได้กำหนดให้ใช้กลไก CSMA/CA เพื่อจัดสรรการใช้ช่องสัญญาณร่วมกันดังกล่าว

- **ซีเอสเอ็มเอวิดธ์แรนดอมแบ็กออฟ (CSMA with Random Back-Off)**

กลไก CSMA (Carrier Sense Multiple Access) with Random Back-Off เป็นเทคนิคอย่างง่ายสำหรับจัดสรรการเข้าใช้ช่องสัญญาณของผู้ใช้แต่ละคน (ซึ่งต้องแบ่งกันใช้ช่องสัญญาณร่วมนี้) อย่างยุติธรรม กลไกนี้เป็นที่ยอมรับและนิยมใช้กันอย่างแพร่หลาย เช่น ในมาตรฐาน IEEE 802.3 Ethernet LAN หลักการทำงานของกลไก CSMA คือ เมื่อสถานีหนึ่งต้องการเข้าใช้ช่องสัญญาณ สถานีดังกล่าวจะต้องตรวจสอบช่องสัญญาณก่อนว่ามีสถานีอื่นทำการรับส่งสัญญาณข้อมูลอยู่หรือไม่และรอจนกว่าช่องสัญญาณจะว่าง เมื่อช่องสัญญาณว่างแล้วสถานีที่ต้องการเข้าใช้ช่องสัญญาณจะต้องรอต่อไปอีกระยะหนึ่ง (Random Back-Off)
ซึ่งแต่ละสถานีได้กำหนดระยะเวลาในการรอดังกล่าวไว้แล้วด้วยการสุ่มค่าหลังจากเสร็จการใช้ช่องสัญญาณครั้งก่อน สถานีที่สุ่มได้ค่าระยะเวลาในการรอน้อยกว่าก็จะมีสิทธิในการเข้าใช้ช่องสัญญาณก่อน แต่อย่างไรก็ตามในบางกรณีกลไกดังกล่าวอาจจะกำหนดให้สถานีมากกว่าหนึ่งสถานีส่งข้อมูลในเวลาพร้อมๆ กันซึ่งจะทำให้เกิดการชนกันของสัญญาณได้ ซึ่งหากเกิดการชนกันของสัญญาณขึ้นจะต้องมีการส่งสัญญาณข้อมูลเดิมซ้ำอีกครั้งด้วยกลไกที่กล่าวมาแล้วข้างต้น

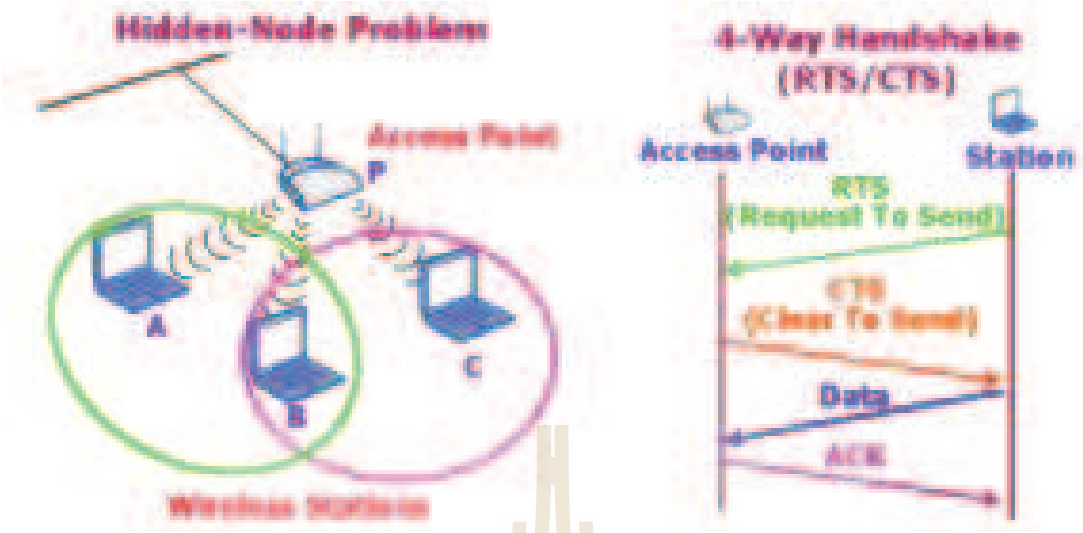
- **ซีเอสเอ็มเอซีดี (Carrier Sense Multiple Access/Collision Detection : CSMA/CD)**

กลไก CSMA/CD เป็นเทคนิคที่รู้จักกันดีซึ่งถูกนำมาใช้ในมาตรฐาน IEEE 802.3 Ethernet LAN ซึ่งการทำงานกลไก CSMA/CD โดยหลักแล้วเป็นเช่นเดียวกับที่กล่าวไว้ในส่วนของ CSMA with Random Back-Off แต่จะมีรายละเอียดเพิ่มเติมเกี่ยวกับการตรวจสอบว่าเกิดการชนกันของสัญญาณหรือไม่ ในกรณีนี้สถานีที่กำลังทำการส่งสัญญาณข้อมูลอยู่จะต้องคอยตรวจสอบด้วยว่ามีการชนกันของสัญญาณเกิดขึ้นหรือไม่ (ในขณะเดียวกันกับที่ทำการส่งสัญญาณข้อมูล) โดยการตรวจวัดระดับแรงดัน

(Voltage) ของสัญญาณในสายสัญญาณว่ามีค่าสูงกว่าปกติหรือไม่ ซึ่งหากระดับ Voltage ของสัญญาณในสายสัญญาณในสายสัญญาณมีค่าสูงกว่าค่าที่กำหนดแสดงว่าเกิดการชนกันของสัญญาณขึ้น ในกรณีดังกล่าวสถานีที่กำลังส่งสัญญาณข้อมูลอยู่จะต้องยกเลิกการส่งสัญญาณทันทีและปฏิบัติตามกลไกที่กล่าวมาแล้วข้างต้นเพื่อทำการส่งข้อมูลเดิมซ้ำอีกต่อไป

- **ซีเอสเอ็มเอซีเอวิดธ์แอคโนเลดจ์เม็นท์ (CSMA/CA with Acknowledgement)**

เป็นที่ควรสังเกตว่าเทคนิค CSMA/CD ไม่สามารถนำมาใช้กับ WLAN ซึ่งใช้การสื่อสารแบบไร้สายได้ สาเหตุหลักคือการตรวจสอบการชนกันของสัญญาณในระหว่างที่ทำการส่งสัญญาณจะต้องใช้อุปกรณ์รับส่งคลื่นวิทยุที่สามารถรับและส่งสัญญาณในเวลาเดียวกันได้ (Full Duplex) ซึ่งจะมีราคาแพงกว่าอุปกรณ์รับส่งคลื่นวิทยุที่ไม่สามารถรับและส่งสัญญาณในเวลาเดียวกัน นอกจากนี้แต่ละสถานีใน BSS หรือ IBSS อาจไม่ได้ยินสัญญาณจากสถานีอื่นทุกสถานี หรือปัญหาที่เรียกว่า ฮิดเดนโหนดพรอบเบลม (Hidden Node Problem) (ดังในรูปที่ 3: สถานี A ได้ยินสัญญาณจากสถานีแม่ข่าย (Access Point) แต่ไม่ได้ยินสัญญาณจากสถานี C และในทางกลับกันสถานี C ไม่ได้ยินสัญญาณจากสถานี A แต่ได้ยินสัญญาณจากสถานีแม่ข่าย ซึ่งสถานการณ์ดังกล่าวนี้เป็นสถานการณ์เกิดขึ้นใน WLAN โดยทั่วไป) ดังนั้น การตรวจสอบการชนกันของสัญญาณโดยตรงเป็นไปได้ยากหรือเป็นไปไม่ได้เลย มาตรฐาน IEEE 802.11 จึงได้กำหนดให้ใช้เทคนิค CSMA/CA with Acknowledgement สำหรับการจัดสรรการเข้าใช้ช่องสัญญาณของแต่ละสถานีเพื่อแก้ไขปัญหาเหล่านี้ ซึ่งการทำงานของกลไก CSMA/CA โดยหลักแล้วเป็นเช่นเดียวกับที่กล่าวไว้ในส่วนของ CSMA with Random Back-Off แต่จะมีรายละเอียดเพิ่มเติมเกี่ยวกับการหลีกเลี่ยงไม่ให้เกิดการชนกันของสัญญาณและเทคนิคสำหรับการตรวจสอบว่าเกิดการชนของสัญญาณ โดยสถานีผู้ส่งสัญญาณข้อมูลจะต้องรอรับแอกโนเลดจ์เมนต์ (Acknowledgement) จากสถานีที่ส่งข้อมูลไปให้ หากไม่ได้รับ Acknowledgement กลับมาภายในเวลาที่กำหนดจะถือว่าเกิดการชนของสัญญาณขึ้นและต้องทำการส่งข้อมูลเดิมซ้ำอีกต่อไป



รูปที่ 3 แสดงฮิดเดนโหนดพรอบเบลม (Hidden Node Problem)
และกลไกอาร์ทีเอสซีทีเอสแฮนด์เชค (RTS/CTS Handshake)

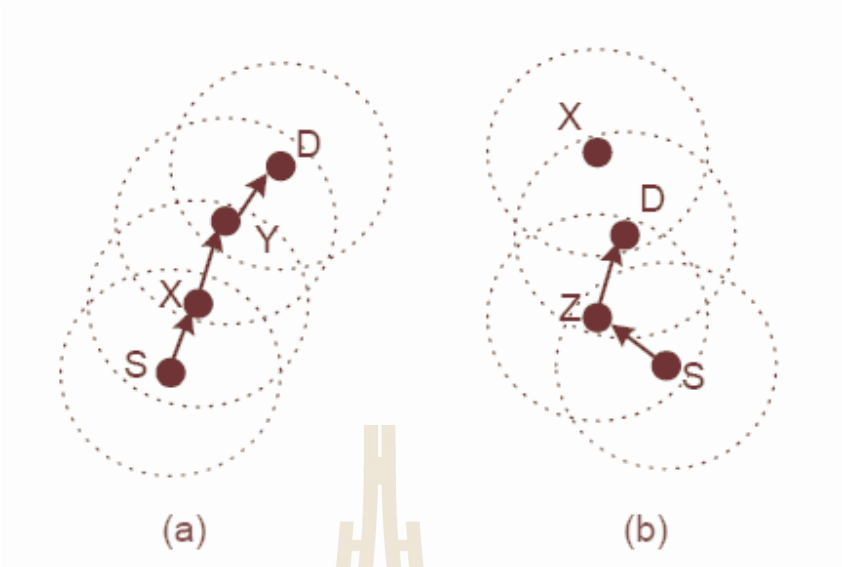
สำหรับการหลีกเลี่ยงไม่ให้เกิดการชนกันของสัญญาณนั้น มาตรฐาน IEEE 802.11 ได้ใช้กลไกที่เรียกว่าเวอร์ชวลแคเรียร์เซ็นส์ (Virtual Carrier Sense) เพื่อแก้ไขปัญหาที่แต่ละสถานีใน BSS หรือ IBSS อาจไม่ได้ยินสัญญาณจากสถานีอื่นบางสถานี (Hidden Node Problem) กลไกดังกล่าวมีการทำงานดังนี้ เมื่อสถานีที่ต้องการจะส่งแพ็กเก็ตข้อมูลได้รับสิทธิในการเข้าใช้ช่องสัญญาณแล้วจะทำการส่งแพ็กเก็ตสั้นๆ ที่เรียกว่าอาร์ทีเอส (Request To Send : RTS) เพื่อเป็นการจองช่องสัญญาณก่อนที่จะส่งแพ็กเก็ตข้อมูลจริง ซึ่งแพ็กเก็ต RTS ประกอบไปด้วยระยะเวลาที่คาดว่าใช้ช่องสัญญาณจนแล้วเสร็จ (Duration ID) รวมถึงที่อยู่ (Address) ของสถานีผู้ส่งและผู้รับ เมื่อสถานีผู้รับได้ยินสัญญาณ RTS ก็จะตอบรับกลับมาด้วยการส่งสัญญาณซีทีเอส (Clear To Send : CTS) ซึ่งจะบ่งบอกข้อมูลระยะเวลาที่คาดว่าสถานีที่กำลังจะทำการส่งข้อมูลนั้นจะใช้ช่องสัญญาณจนแล้วเสร็จ หลักการก็คือทุกๆสถานีใน BSS หรือ IBSS ควรจะได้ยินสัญญาณ RTS หรือไม่ก็ CTS อย่างใดอย่างหนึ่งหรือทั้งสองอย่าง เมื่อได้รับ RTS หรือ CTS ทุกๆสถานีจะทราบถึงว่าช่วงเวลาที่ระบุไว้ใน Duration ID ซึ่งช่องสัญญาณจะถูกใช้และทุกสถานีที่ยังไม่ได้รับสิทธิในการเข้าใช้ช่องสัญญาณจะตั้งค่าเอ็นเอวี (Network Allocation Vector : NAV) ให้เท่ากับ Duration ID

ซึ่งแสดงถึงช่วงเวลาที่ยังไม่สามารถเข้าใช้ช่องสัญญาณได้ ทุกๆสถานีจะใช้กลไก Virtual Carrier Sense ดังกล่าวผนวกกับการฟังสัญญาณในช่องสัญญาณจริงๆ ในการตรวจสอบว่าช่องสัญญาณว่างหรือไม่

2.2 โครงข่ายเคลื่อนที่แบบแอดฮอค (Mobile Ad Hoc Networks : MANETs)

โครงข่ายเคลื่อนที่แบบแอดฮอค (Mobile Ad hoc Networks) หรือเขียนสั้น ๆ ว่า MANETs เริ่มมีการศึกษาคิดค้นตั้งแต่ช่วงกลางทศวรรษที่ 1960 ซึ่งในเริ่มแรกใช้ชื่อว่าแพ็กเก็ตเรดิโอเนตเวิร์ก (Packet Radio Network) แต่ได้หยุดการพัฒนาไปในช่วงทศวรรษที่ 1980 เนื่องจากเทคโนโลยีด้านหน่วยประมวลผลและหน่วยความจำยังมีราคาแพงมาก ปัจจุบันปัญหาที่ว่านั้นได้หมดไป แนวความคิดเรื่อง MANETs กลับมาได้รับความสนใจและมีการพัฒนาอีกครั้งอย่างต่อเนื่องตั้งแต่ปี 1995 MANETs คือ กลุ่มของโหนดเคลื่อนที่ไร้สาย (Wireless Mobile Nodes) รวมกันก่อให้เกิดเป็นโครงข่ายอิสระที่มีโครงสร้างการเชื่อมโยงแบบไร้สายทั้งหมด และมีลักษณะการทำงานแบบพลวัตร (Dynamic Autonomous Network) โหนดต่าง ๆ จะทำการติดต่อสื่อสารระหว่างกันได้โดยไม่ต้องผ่านการเชื่อมโยงที่สถานีฐาน (Base Stations) หรือแอกเซสพอยต์ (Access Points) ซึ่งก็หมายความว่า โหนดแต่ละโหนดจะเป็นได้ทั้งเราท์เตอร์ (Router) และโฮสต์ (Host) และเนื่องจากโหนดเคลื่อนที่ไร้สายมีข้อจำกัดในการส่งสัญญาณติดต่อกันภายในระยะทางช่วงหนึ่ง ดังนั้นการติดต่อสื่อสารระหว่างโหนดต่าง ๆ ที่อยู่ในโครงข่ายจะมีลักษณะที่เรียกว่า มัลติเปิ้ลฮอพส์ (Multiple Hops) ในบางครั้งเราจะพบคำว่าโครงข่าย Multiple Hops (Multi Hop Networks) ซึ่งก็คือ MANETs นั่นเอง

จากรูปจะเห็นว่าภายใน MANETs มีการติดต่อสื่อสารระหว่างโหนด S และโหนด D โดยโหนด S ส่งข้อมูลไปยังโหนด D การเชื่อมโยงระหว่างโหนด เป็นดังนี้ S →X→Y →D (ดังรูป a) และในเวลาต่อมา โหนดต่าง ๆ มีการเคลื่อนที่การเชื่อมโยงระหว่างโหนดเปลี่ยนเป็น S →Z → D (ดังรูป b)



รูปที่ 4 การเชื่อมโยงระหว่างโหนดในโครงข่ายเคลื่อนที่แบบแอดฮอค
(Mobile Ad Hoc Networks : MANETs)

นอกจากนี้ MANETs มีลักษณะการติดต่อสื่อสารแบบ Peer-to-Peer คือ การติดต่อสัญญาณข้อมูลจากโหนดต้นทางไปยังโหนดปลายทางมีการติดต่อกันระหว่างชั้นโพรโทคอลที่อยู่ในระดับเดียวกัน ข้อจำกัดที่สำคัญของ MANETs ได้แก่

• **โทโพโลยีแบบพลวัตร** (Dynamic Topology) โหนดต่าง ๆ เคลื่อนที่ได้และสามารถติดต่อกับโหนดใดๆ ก็ได้ในโครงข่าย ดังนั้น การเชื่อมโยงโหนดต่าง ๆ ในโครงข่ายจึงมีการเปลี่ยนแปลงไปตามเวลาโดยขึ้นอยู่กับตำแหน่งของโหนดต่าง ๆ ณ เวลาขณะใดขณะหนึ่ง ทำให้การสื่อสารระหว่างโหนดต้นทางและโหนดปลายทางในบางครั้งเชื่อมต่อไม่ถึงกัน เนื่องจากจุดเชื่อมต่อระหว่างโหนดมีการเปลี่ยนแปลงจึงเกิดหัวข้อในการแก้ปัญหาเกี่ยวกับการสื่อสารที่ไม่เชื่อมต่อกันเมื่อโหนดมีการเคลื่อนที่

• **ข้อจำกัดการใช้แถบความถี่** ความจุในการส่งสัญญาณข้อมูลทางการเชื่อมโยงไร้สายมีน้อยกว่าการเชื่อมโยงโดยใช้สายเป็นอย่างมาก และการเชื่อมโยงไร้สายยังมีปัญหาในเรื่องของสัญญาณรบกวน ทำให้สัญญาณข้อมูลด้านรับมีคุณภาพลดลง ซึ่งมีอัตราบิตผิดพลาด (Bit Error Rate) อยู่ในช่วง 10-4-10-5

• **ข้อจำกัดด้านพลังงาน** พลังงานที่ใช้ในโหนดเคลื่อนที่ถูกจำกัดด้วยระดับแบตเตอรีของโหนด ดังนั้น การออกแบบระบบ MANETs จะต้องเกี่ยวข้องกับเทคโนโลยีด้านการอนุรักษ์พลังงานด้วย

• **ข้อจำกัดทางด้านการรักษาความปลอดภัย** โครงข่ายที่โหนดมีการเคลื่อนที่และรูปแบบของโครงข่ายมีการเปลี่ยนแปลงจึงมีความเสี่ยงในด้านความปลอดภัยของข้อมูลมากกว่าโครงข่ายแบบถาวร เนื่องจากอุปกรณ์สื่อสารที่โหนดหรือของผู้ใช้อาจถูกขโมยได้ หรือสัญญาณข้อมูลอาจถูกส่งไปในการเชื่อมโยงไร้สายที่ไม่ปลอดภัย เป็นต้น ดังนั้นจะต้องมีการป้องกันปัญหาในเรื่องการลักลอบหรือดักสัญญาณ
ข้อมูล และปัญหาด้านความปลอดภัยของโครงข่าย

• **โครงข่ายอิสระ** คือ ไม่มีการควบคุมสั่งการ หรือการจัดการจากโครงข่ายที่เป็นส่วนกลาง MANETs สามารถดำเนินการต่าง ๆ ด้วยตัวเองได้ ดังนั้น การออกแบบระบบจะต้องมีความซับซ้อนมากขึ้นด้วย

MANETs มีข้อได้เปรียบว่าโครงข่ายไร้สายแบบอื่น ๆ ที่มีอยู่เดิมหลายประการ เนื่องจาก MANETs สามารถสร้างขึ้นได้ทันที โดยไม่ต้องมีโครงสร้างของสถานีฐานและไม่ต้องมีผู้ดูแลระบบโครงข่าย จึงไม่ยุ่งยากซับซ้อน ลดการก่อสร้างในส่วนของโครงข่ายที่ต้องติดตั้งอยู่กับที่ทำให้งบประมาณการสร้างโครงข่ายลดลงและยังทำให้การใช้พลังงานลดลงด้วย

MANETs สามารถนำไปประยุกต์ใช้กับระบบต่าง ๆ ได้อย่างเหมาะสม ได้แก่ ใช้ติดต่อส่วนบุคคล ซึ่งอุปกรณ์ที่ใช้ติดต่อ เช่น เครื่องโทรศัพท์มือถือ คอมพิวเตอร์พกพา หรือเครื่อง PDA ใช้ติดต่อภายในกลุ่ม เช่น สร้างโครงข่ายติดต่อสื่อสารภายในงานแสดงนิทรรศการ งานสัมมนาต่างๆ
ใช้ในงานด้านการทหาร ใช้ในกรณีฉุกเฉินต่างๆ เช่น กรณีที่เกิดภัยพิบัติ หรือใช้เป็นโครงข่ายสื่อสารภายในเมือง เป็นต้น

บทที่ 3
ขั้นตอนวิธีสำหรับเลือกทางเดินข้อมูล

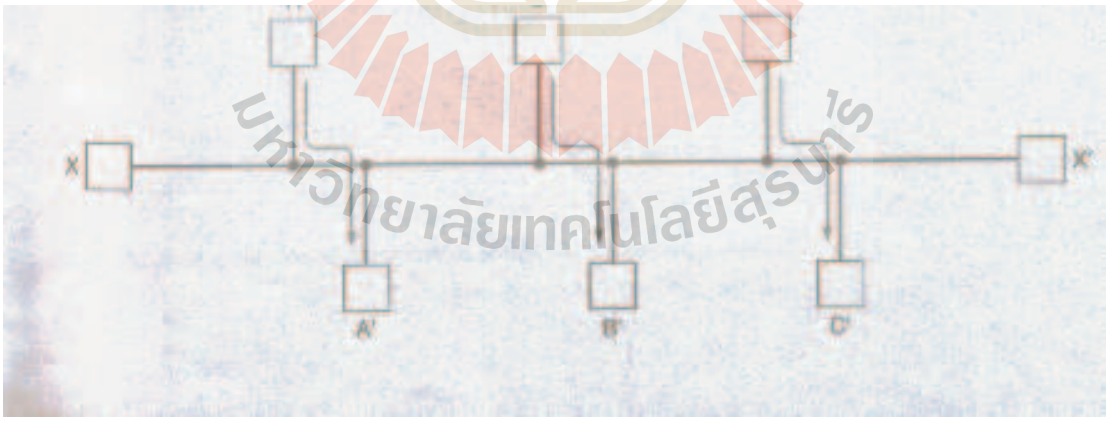
หน้าที่หลักของชั้นควบคุมเครือข่ายคือการจัดส่งแพ็กเก็ตจากเครื่องผู้ส่งไปยังเครื่องผู้รับ ในระบบเครือข่ายย่อยส่วนมากแพ็กเก็ตจะต้องถูก “รับ-แล้ว-ส่งต่อ (Hop)” หลายครั้งจนถึงจุดหมายปลายทาง ยกเว้นชั้นควบคุมเครือข่ายของระบบที่ใช้การส่งข้อมูลแบบกระจายที่ไม่ต้องจัดการปัญหานี้ แต่ก็ยังคงเกิดขึ้นได้ถ้าผู้ส่งและผู้รับข้อมูลไม่ได้อยู่ในเครือข่ายเดียวกัน ดังนั้น ขั้นตอนวิธี (Algorithm) และโครงสร้างข้อมูลที่ทำหน้าที่เลือกเส้นทางเดินข้อมูล จึงเป็นส่วนประกอบหลักในการออกแบบชั้นควบคุมเครือข่าย

ขั้นตอนวิธีการเลือกเส้นทางของตัวเราท์เตอร์ (Router) เมื่อแพ็กเก็ตที่รับเข้ามาควรตัดสินใจว่าจะส่งต่อไปให้เราท์เตอร์ตัวใดเป็นลำดับต่อไป ถ้าเครือข่ายย่อยใช้ดาต้าแกรมในการส่งข้อมูลภายในแล้ว การตัดสินใจของแต่ละเราท์เตอร์จะเกิดขึ้นทุกครั้งที่รับดาต้าแกรมตัวใหม่เข้ามาเพราะทางเลือกที่ดีที่สุด (สำหรับเป้าหมายเดิม) อาจเปลี่ยนแปลงไปแล้ว ถ้าเครือข่ายย่อยใช้วงจรเสมือนในการส่งข้อมูลภายใน การเลือกเส้นทางตลอดทั้งเครือข่ายจะเกิดขึ้นเพียงครั้งเดียวในช่วงการจัดตั้งช่องสื่อสาร แพ็กเก็ตที่ส่งมาในลำดับหลัง จะถูกส่งไปยังเราท์เตอร์ที่แพ็กเก็ตแรกๆถูกส่งไปเสมอ วิธีการเลือกเส้นทางเดินแบบนี้บางครั้งเรียกว่า “เซสชั่นเราติ้ง (Session Routing)” เนื่องจากเส้นทางเดินจะคงเดิมตลอดช่วงเวลาสื่อสารที่กำหนดขึ้น

ไม่ว่าเส้นทางเดินข้อมูล จะได้รับการกำหนดขึ้นมาด้วยอัลกอริทึมใดก็ตาม คุณลักษณะที่พึงควรมี 6 ประการ คือ ความถูกต้อง (Correctness), ความง่าย (Simplicity), ความคงทน (Robustness), ความแน่นอน (Stability), ความเป็นธรรม (Fairness), และความเหมาะสม (Optimality) ความถูกต้องและความง่าย เป็นคุณสมบัติพื้นฐานที่สำคัญและชัดเจน 2 ข้อแรกที่ทุกอัลกอริทึมจะต้องมี ความคงทนของอัลกอริทึมสามารถพิจารณาได้จากการที่เครือข่ายใหม่ที่พึ่งได้รับการจัดตั้งขึ้นมาใช้งานย่อมมีความต้องการที่ให้บริการคงอยู่ โดยไม่ประสบปัญหาความล้มเหลวในระดับเครือข่าย แต่ความล้มเหลวของอุปกรณ์หรือโปรแกรมบางส่วนเป็นเหตุการณ์ที่เกิดขึ้นได้ตลอดเวลาสำหรับทุกระบบ และไม่สามารถหลีกเลี่ยงได้ เช่น โฮสต์ เราท์เตอร์ และการสื่อสารอาจเกิดการเสียหายเมื่อใดก็ได้ ซึ่งจะต้องได้รับการซ่อมแซมอยู่เสมอ หรือ การจัดโครงข่าย (Network Topology)

อาจมีการเปลี่ยนแปลง ได้เช่นกัน อัลกอริทึมสำหรับเลือกทางเดินข้อมูลจำเป็นจะต้องแก้ไขหรือสามารถปรับตัวให้เข้ากับสภาพปัญหาและการเปลี่ยนแปลงเหล่านี้ได้ เมื่อเกิดปัญหาขึ้น โฮสต์ส่วนใหญ่จะต้องสามารถทำงานต่อไปได้ โดยไม่ต้องเข้ามามีส่วนร่วมในการแก้ไข หรือจัดการการเปลี่ยนแปลงที่เกิดขึ้น

ความแน่นอนเป็นส่วนที่สำคัญมากเช่นกัน อัลกอริทึมสำหรับเลือกทางเดินข้อมูลจำนวนไม่น้อยที่ไม่สามารถจัดการให้เครือข่ายเข้าสู่สถานะความสมดุลในการกระจายปริมาณข้อมูล ทำให้ระบบอยู่ในสภาวะที่เกิดความไม่แน่นอนในการสื่อสารข้อมูลในระดับสูง ซึ่งเป็นสิ่งที่ไม่ต้องการ ความเป็นธรรมและความเหมาะสมเป็นสิ่งที่ผู้ใช้ในทุกระบบต้องการ อย่างไรก็ตาม ในทางปฏิบัติจะต้องมีผู้ที่ไม่ได้รับความเป็นธรรมด้วยเหตุผลต่างๆกัน ในรูปที่ 5 แสดงระบบเครือข่ายหนึ่งที่สมมติให้ปริมาณข้อมูลที่ไหลจาก A ไป A' จาก B ไป B' และจาก C ไป C' มีปริมาณมากพอ ที่จะทำให้การส่งข้อมูลตามแนวราบอยู่ในสภาวะอิ่มตัว ถ้าต้องการให้ระบบนี้มีประสิทธิภาพโดยรวมสูงสุดแล้ว การสื่อสารระหว่าง X และ Y' ควรจะต้องถูกระงับไว้ก่อน มิฉะนั้น จะทำให้การสื่อสารในส่วนที่เหลือทั้งหมดต้องหยุดชะงัก ผลคือทำให้ประสิทธิภาพโดยรวมของระบบต่ำลงไป การตัดสินใจเช่นนี้เป็นการขัดต่อหลักความเป็นธรรมที่จะพึงให้แก่สมาชิกทุกคนของระบบอย่างเท่าเทียมกัน ดังนั้น จึงต้องมีการประนีประนอมระหว่างความเหมาะสมกับความเป็นธรรม



รูปที่ 5 ข้อขัดแย้งระหว่างความเป็นธรรมกับการจัดการให้มีประสิทธิภาพสูงสุด

ความเหมาะสมที่ทุกระบบต้องการคือ การลดเวลารอคอยโดยเฉลี่ยของแพ็กเก็ต ในขณะที่การเพิ่มผลสัมฤทธิ์ (Throughput) ของระบบเครือข่ายก็เป็นสิ่งที่ควรพิจารณาควบคู่ไปด้วย ความต้องการทั้ง 2 ข้อนี้ขัดแย้งซึ่งกันและกัน เนื่องจากการเพิ่มประสิทธิผลของระบบต้องการให้แพ็กเก็ตข้อมูลมารอคอยอยู่ในเราท์เตอร์ให้มากที่สุด เพื่อจะได้รักษาความต่อเนื่องในการส่งข้อมูลให้อยู่ในระดับสูงสุดตลอดเวลา แต่การทำงานนี้ทำให้เวลารอคอยโดยเฉลี่ยของแต่ละแพ็กเก็ตสูงขึ้น เพื่อลดความขัดแย้งดังกล่าวเครือข่ายหลายระบบได้พยายามลดจำนวนเราท์เตอร์ที่ใช้เป็นตัวกลางลงไปให้มากที่สุด ทั้งนี้ จะทำให้เวลารอคอยของแพ็กเก็ตลดลง ในขณะที่ช่วยเพิ่มประสิทธิผลของระบบไปในเวลาเดียวกัน

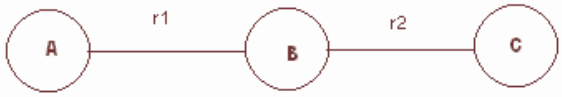
อัลกอริทึมสำหรับเลือกทางเดินข้อมูล แบ่งออกเป็น 2 ประเภทคือ พวกที่มีการปรับตัว (Adaptive Algorithms) และ พวกที่ไม่มีการปรับตัว (Nonadaptive Algorithms) พวกที่ไม่มีการปรับตัวจะไม่นำสภาวะของเครือข่าย เช่น ปริมาณข้อมูลในระบบ หรือ รูปแบบโครงสร้างเครือข่าย เข้ามาใช้ในการพิจารณาเลือกทางเดินข้อมูล ทางเดินข้อมูลทุกเส้นทางจะมีการวางแผนไว้ล่วงหน้าและจัดส่งแผนนี้ไปให้กับเราท์เตอร์ทุกตัวในระบบ บางครั้งเรียกว่าการเลือกทางเดินแบบสถิตย์ (Static Router Algorithms)

การเลือกทางเดินแบบมีการปรับตัว จะเปลี่ยนแปลงการตัดสินใจเพื่อให้เหมาะสมกับสภาวะแวดล้อมของระบบในขณะที่จะส่งข้อมูลอัลกอริทึมประเภทนี้ จะแตกต่างกันตรงวิธีการให้ได้มาซึ่งข้อมูลเมื่อมีการเปลี่ยนแปลงเส้นทาง และกรรมวิธีที่ใช้ในการวัดความเหมาะสมของแต่ละเส้นทาง บางครั้งเรียกว่าการเลือกทางเดินแบบพลวัตร (Dynamic Routing Algorithms)

3.1 หลักการหาเส้นทางที่เหมาะสมที่สุด (Optimality Principle)

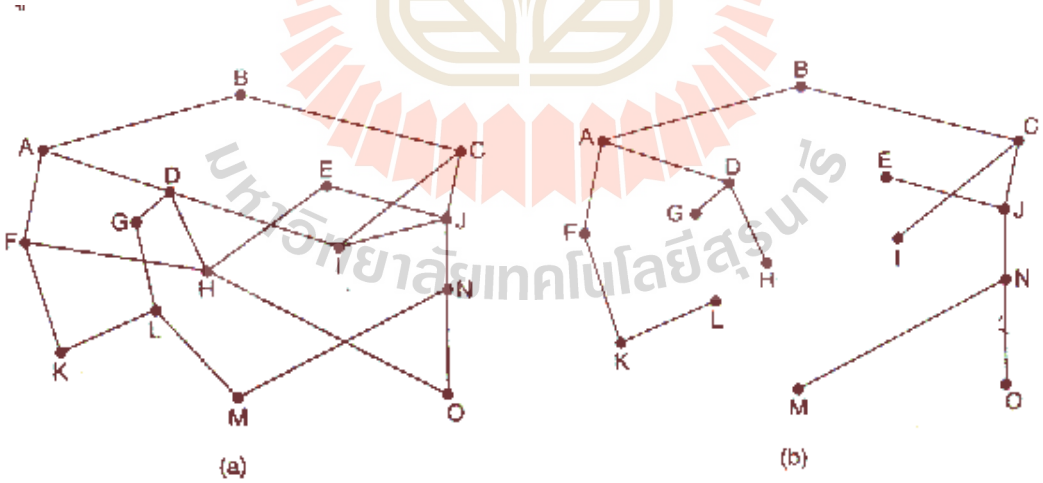
หลักการพื้นฐานของการหาเส้นทางที่เหมาะสมที่สุด (Optimality Principle) ที่ไม่ขึ้นอยู่กับโครงสร้างของระบบเครือข่ายแบบใดจากรูปที่ 6 กล่าวว่าถ้าเราท์เตอร์หนึ่ง (จุด B) อยู่บนเส้นทางที่เหมาะสมที่สุดระหว่างผู้ส่ง (จุด A) และผู้รับข้อมูล (จุด C) แล้ว เส้นทางนั้นจะเป็นเส้นทางที่เหมาะสมที่สุดระหว่างเราท์เตอร์นั้น (จุด B) กับผู้รับข้อมูล (จุด C) ด้วย ข้อความนี้สามารถพิสูจน์ได้ โดยใช้หลักการตรรกศาสตร์ธรรมดา คือ สมมติให้ r1-r2 คือเส้นทางที่ดีที่สุดระหว่างจุด A กับจุด C ถ้า r1 คือเส้นทางที่เหมาะสมที่สุดระหว่างจุด A กับจุด B แล้วให้ r2 เป็นเส้นทางระหว่างจุด B กับจุด C ถ้ามีเส้นทางอื่นที่เหมาะสมกว่าเส้นทาง r2

ก็ควรจะนำมาใช้เป็นเส้นทางที่ดีที่สุดระหว่างจุด A กับจุด C ซึ่งขัดกับข้อสมมติฐานที่ว่า r1-r2 คือเส้นทางที่ดีที่สุด ดังนั้นเส้นทาง r2 จึงเป็นเส้นทางที่ดีที่สุดระหว่างจุด B กับจุด C ด้วย



รูปที่ 6 แสดงเส้นทางที่เหมาะสมที่สุดระหว่างจุด A และจุด C

เมื่อนำหลักการพื้นฐานของการหาเส้นทางที่เหมาะสมที่สุด มาใช้พิจารณาหาเส้นทางที่ดีที่สุดจากเราท์เตอร์กลุ่มหนึ่งกับเราท์เตอร์สมมติตัวหนึ่งมาเขียนเป็นรูปทรี (Tree) ที่มีเราท์เตอร์สมมติตัวนั้นเป็นโหนดราก จะเรียกทรีนี้ว่า “ซิงค์ทรี (Sink Tree)” ดังที่แสดงในรูปที่ 7 ซึ่งสมมติให้ใช้จำนวนครั้งในการส่งแพ็กเก็ตผ่านเราท์เตอร์เป็นมาตรฐานที่ใช้ในการเลือกเส้นทาง การสร้างซิงค์ทรีของเราท์เตอร์ตัวหนึ่งไม่จำเป็นว่าจะมีได้เพียงรูปแบบเดียว วัตถุประสงค์ของอัลกอริทึม สำหรับการเลือกเส้นทางเดินข้อมูล คือการหาซิงค์ทรีให้พบ และใช้เป็นเส้นทางในการส่งข้อมูล สำหรับทุกเราท์เตอร์ที่มีอยู่ในระบบเครือข่ายหนึ่งๆ



รูปที่ 7 (a) ระบบเครือข่ายย่อย (b) ซิงค์ทรีจากเราท์เตอร์ B

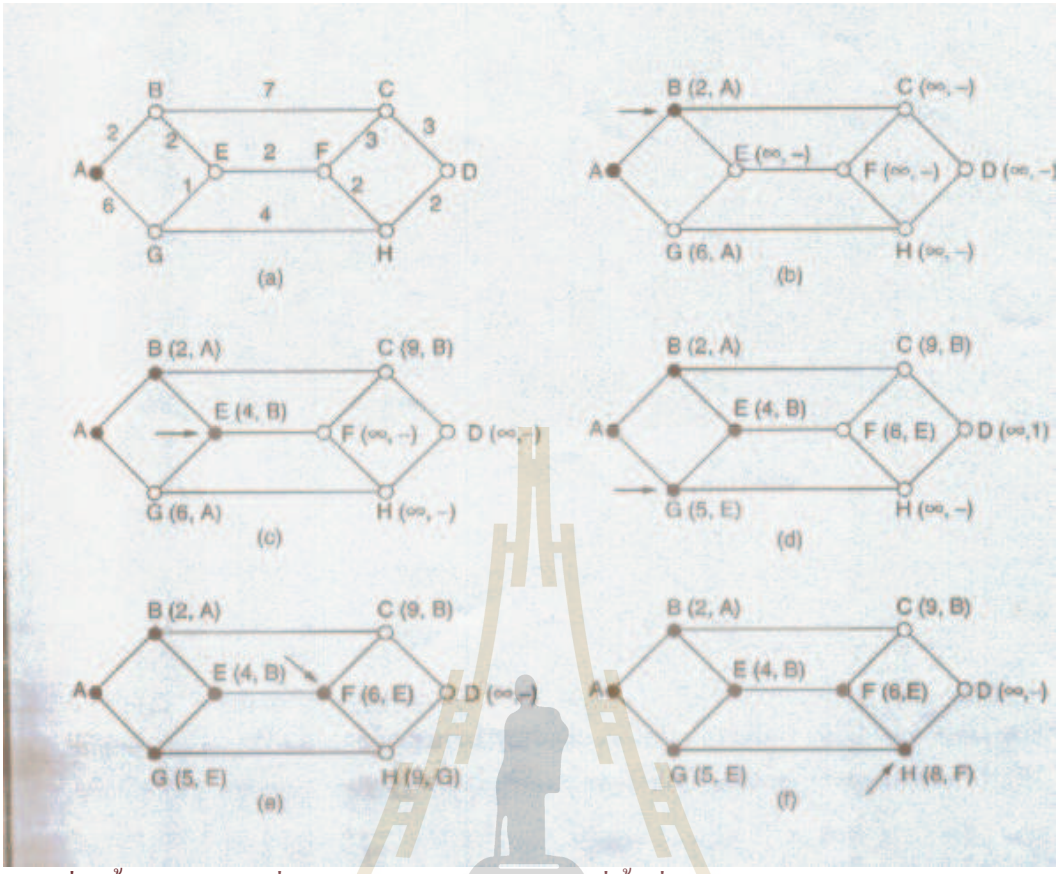
เนื่องจากซิงค์ทรีเป็นทรีชนิดหนึ่งจึงไม่มีลักษณะของวังวน (Loop) อยู่ภายในการส่งข้อมูลจากเราท์เตอร์ใดๆ ไปยังเราท์เตอร์ที่เป็นโหนดรากจะสามารถทราบจำนวนครั้งการ “รับเพื่อส่งต่อ” ได้แน่นอน แต่ในความเป็นจริงนั้นอาจเป็นไปได้ค่อนข้างยาก เพราะสายสื่อสารและเราท์เตอร์อาจจะเสียหายหรือถูกยกเลิกการใช้งานชั่วคราว และอาจจะกลับมาให้บริการอีกก็เมื่อใดก็ได้ ดังนั้นเราท์เตอร์จะเห็นโครงสร้างของเครือข่ายมีการเปลี่ยนแปลงอยู่ตลอดเวลา การนับจำนวนครั้งที่กล่าวถึงจึงไม่สามารถนำไปใช้งานได้โดยตรง อย่างไรก็ตามซิงค์ทรีได้รับความนิยมในการนำมาใช้เป็นมาตรฐานในการเปรียบเทียบประสิทธิภาพของอัลกอริทึมแบบต่างๆ ส่วนวิธีการที่เราท์เตอร์ใช้ในการหาข้อมูลของเครือข่าย รวมทั้งรายละเอียดที่เกี่ยวข้องจะได้นำมากล่าวในหัวข้อต่อไป

3.2 การเลือกเส้นทางที่สั้นที่สุด (Shortest Path Routing)

การเลือกเส้นทางที่สั้นที่สุด เป็นวิธีการที่ถูกนำไปใช้มากที่สุดแบบหนึ่ง หลักการทำงานเริ่มต้นด้วยการสร้างรูปกราฟของระบบเครือข่ายย่อย โดยให้แต่ละโหนดในรูปกราฟแทนเราท์เตอร์แต่ละตัวในเครือข่าย และให้เส้นเชื่อมโหนด (Arc) แทนสายสื่อสารที่เชื่อมต่อระหว่างเราท์เตอร์ การเลือกเส้นทางเดินระหว่างเราท์เตอร์คู่หนึ่งทำได้โดยการค้นหาเส้นทางที่สั้นที่สุดในรูปกราฟ

นิยามของคำว่าเส้นทางที่สั้นที่สุดอาจมีได้หลายความหมาย เช่น การใช้จำนวนครั้งของการรับ-ส่งข้อมูล หรือจำนวนเส้นเชื่อมในรูปกราฟเป็นหลักพิจารณา เส้นทาง ABCF และ ABEF ในรูปที่ 8 จะถือว่ายาวเท่ากัน แต่ถ้าใช้ระยะทาง (เช่น เป็นกิโลเมตร) มาพิจารณาแล้ว เส้นทาง ABEF ย่อมสั้นกว่า

อีกวิธีการหนึ่ง ที่สามารถนำมาใช้แทนการนับจำนวนเส้นเชื่อม หรือการวัดระยะทาง คือการกำหนดให้ตัวเลขบนเส้นเชื่อมแต่ละเส้นเป็นตัวเลขที่บอกจำนวนแพ็กเก็ตที่รอการจัดส่งและเวลารอคอยโดยเฉลี่ย ที่คำนวณมาจากการรับส่งแพ็กเก็ตมาตรฐาน ด้วยวิธีการนี้เส้นทางที่สั้นที่สุดจึงหมายถึงเส้นทางที่ส่งข้อมูลได้เร็วที่สุด



รูปที่ 8 ขั้นตอน(a)-(f) ที่ใช้ในการคำนวณหาเส้นทางที่สั้นที่สุดเราท์เตอร์ A ไปยังเราท์เตอร์ D (ลูกศรชี้ตำแหน่งที่กำลังทำการคำนวณอยู่)

วิธีการที่นำมาใช้กับกรณีทั่วไปนั้น จะกำหนดให้ตัวเลขบนเส้นเชื่อมคือผลลัพธ์ที่ได้จากการคำนวณโดยมีระยะทาง ความเร็วในการส่งข้อมูล ปริมาณข้อมูลโดยเฉลี่ย ค่าใช้จ่าย ค่าเฉลี่ยมาตรฐานของจำนวนแพ็กเก็ตที่รอการจัดส่ง ระยะเวลารอคอย และอื่นๆเป็นตัวประกอบการเปลี่ยนค่าความสำคัญของตัวประกอบเหล่านี้ทำให้แต่ละเราท์เตอร์สามารถใช้อัลกอริทึมเดียวกันในการคำนวณแต่ให้ความสำคัญตัวประกอบไม่เหมือนกันได้

Dijkstra (1959) ได้นำเสนออัลกอริทึมสำหรับการค้นหาเส้นทางที่สั้นที่สุดระหว่างจุด 2 จุดในรูปกราฟ ดังตัวอย่างในรูปที่ 8 ต้องการหาเส้นทางที่สั้นที่สุดจากจุด A ไปยังจุด D เริ่มด้วยการระบายสีทึบที่จุด A (เพื่อบอกให้ทราบว่าได้พิจารณาจุดนี้แล้ว) หาระยะทางของแต่ละจุดที่มีเส้นเชื่อมมาที่จุด A แล้วใส่ป้ายบอกระยะทางและโหนดกำกับเส้นเชื่อมเหล่านั้น คือ ใส่ (2,A) ไว้ที่จุด B และใส่ (6,A) ไว้ที่จุด G แล้วเลือกเส้นทางที่สั้นที่สุดจากตัวเลขทั้งหมดที่หาได้ ซึ่งก็คือจุด B ให้ระบายสีทึบที่จุด B

แล้วใช้วิธีการเดิม จะได้ (4,B) ที่จุด E, (9,B) ที่จุด C,และของเดิม (6,A) ที่จุด G เลือกจุด E เป็นจุดต่อไปเพราะมีค่าต่ำที่สุด ระบายสีทึบที่จุด E แล้วใช้วิธีการเดิม จะได้ (5,E) ที่จุด G ซึ่งใช้แทน (6,A) เพราะมีค่าต่ำกว่า, (6,E) ที่จุด F, และของเดิม (9,B) ที่จุด C เลือกจุด G เป็นจุดต่อไป ระบายสีทึบที่จุด G แล้วใช้วิธีการเดิม จะได้ (9,G) ที่จุด H, ของเดิม (6,E) ที่จุด F, และของเดิม (9,B) ที่จุด C เลือกจุด F เป็นจุดต่อไป ระบายสีทึบที่จุด F แล้วใช้วิธีการเดิม จะได้ (8,F) ที่จุด H ซึ่งใช้แทน (9,G) เพราะมีค่าต่ำกว่า, และของเดิม (9,B) ที่จุด C เลือกจุด H เป็นจุดต่อไป ให้ระบายสีทึบที่จุด H แล้วใช้วิธีการเดิม จะได้ (10,H) ที่จุด D, และของเดิม (9,B) ที่จุด C แม้ว่าจะได้เส้นทางมาถึงจุดปลายทางแล้วก็ตามแต่ค่า (9,B) ที่จุด C มีค่าต่ำกว่าจึงต้องคำนวณต่อไปโดยใช้จุด C เป็นจุดต่อไป ระบายสีทึบที่จุด C แล้วใช้วิธีการเดิม จะได้ (12,C) ที่จุด D แต่ค่าเดิมคือ (10,H) มีค่าต่ำกว่าจึงใช้ค่าเดิม ผลลัพธ์ที่ได้คือเส้นทาง ABEFHD มีความยาว 10 หน่วย เป็นเส้นทางที่สั้นที่สุด

3.3 การเลือกเส้นทางเดินแบบฟลัดดิ้ง (Flooding)

วิธีการเลือกเส้นทางเดินข้อมูลแบบฟลัดดิ้ง (Flooding) เป็นวิธีการแบบที่ไม่ปรับตัวเข้ากับสภาพแวดล้อมของระบบ เราท์เตอร์จะส่งทุกแพ็กเก็ตที่รับเข้ามาออกไปทุกทิศทางที่มีการเชื่อมต่อกับเราท์เตอร์ตัวอื่นยกเว้นเราท์เตอร์ที่เป็นผู้ส่งเข้ามา วิธีนี้จะเพิ่มปริมาณข้อมูลในเครือข่ายขึ้นมาอย่างมากมายซึ่งส่วนมากจะเป็นข้อมูลที่ซ้ำกัน ถ้าไม่ใช้กรรมวิธีอื่นเข้ามาช่วยแล้วจะทำให้เกิดข้อมูลจำนวนมหาศาลเลยทีเดียว หนทางหนึ่งที่นำมาใช้ได้คือ การใส่ตัวเลขเพื่อนับจำนวนเราท์เตอร์เข้าไปในข้อมูลส่วนหัวของแต่ละแพ็กเก็ต ทุกครั้งที่เราท์เตอร์รับแพ็กเก็ตเข้ามาก็จะตรวจสอบค่าตัวเลขนี้ ถ้าหากมีค่าเป็นศูนย์ก็จะลบแพ็กเก็ตทิ้ง มิฉะนั้น ก็จะลดค่าจำนวนนับลงไปหนึ่งหน่วยก่อนที่จะส่งต่อไป เลขนับจำนวนเราท์เตอร์นี้ควรจะมีค่าเท่ากับจำนวนเราท์เตอร์ที่จะต้องเดินทางผ่านจากผู้ส่งไปยังผู้รับ ถ้าผู้ส่งไม่ทราบจำนวนที่แน่นอนก็จะใช้ค่าสมมติซึ่งเป็นค่าสูงสุดของเครือข่ายนั้นๆแทน

อีกวิธีหนึ่งที่เป็นไปได้ คือการจัดทำบันทึกแพ็กเก็ตที่ได้ส่งออกไปแล้วเพื่อที่จะได้ไม่ต้องส่งออกไปอีก ทำได้โดยการให้เราท์เตอร์ที่ให้กำเนิดแพ็กเก็ตนั้นกำหนดหมายเลขลำดับและใส่ลงไปในทุกแพ็กเก็ตเราท์เตอร์แต่ละตัว

จะต้องทำตารางข้อมูลสำหรับบันทึกหมายเลขลำดับของเราท์เตอร์อื่นทุกตัว ถ้าแพ็กเก็ตที่รับเข้ามามีหมายเลขที่ยังไม่ได้บันทึกเอาไว้ ก็แสดงว่าเป็นแพ็กเก็ตตัวใหม่ เราท์เตอร์ก็จะบันทึกหมายเลขไว้พร้อมกับส่งแพ็กเก็ตนั้นออกไป มิฉะนั้นก็จะลบแพ็กเก็ตทิ้งไป เพื่อเป็นการป้องกันตารางบันทึกข้อมูลไม่ให้โตขึ้นแบบไม่มีขีดจำกัดการจัดเก็บหมายเลขอาจเก็บในลักษณะของช่วงตัวเลขเช่น มีหมายเลขฐานคือ 1000000 และมีความกว้างของช่วงเท่ากับ 55 แปลว่าเราท์เตอร์ตัวนี้ได้จัดการส่งแพ็กเก็ตหมายเลขระหว่าง 1000000 ถึง 1000055 ไปเรียบร้อยแล้ว

วิธีการแบบฟลัดดิ้ง สามารถปรับปรุงให้มีประสิทธิภาพดีขึ้นเรียกว่า วิธีฟลัดดิ้งบางทิศทาง นั่นคือแทนที่จะส่งข้อมูลออกไปทุกทิศทาง เราท์เตอร์จะเลือกทิศทางที่เหมาะสมในการส่งข้อมูลต่อไป เช่น ต้องการส่งข้อมูลไปยังจุดหมายทางทิศตะวันออก เราท์เตอร์แต่ละตัวก็ควรจะส่งแพ็กเก็ตออกไปยัง
เราท์เตอร์ที่อยู่ทิศทางนั้น วิธีการนี้ใช้ได้ผลดีกับเครือข่ายที่มีโครงสร้างเป็นรูปร่างทรงเรขาคณิตเท่านั้น

วิธีการฟลัดดิ้งเหมาะกับงานบางประเภทเช่น ประเภทแรก นำไปใช้ในกิจการทางทหาร ซึ่งในยามสงครามเราท์เตอร์แต่ละตัวอาจถูกข้าศึกทำลายได้ทุกเมื่อ ดังนั้นการที่มีสำเนาของข้อมูลค้างอยู่ในระบบเป็นจำนวนมาก จะทำให้มีความมั่นใจได้ว่าข่าวสารจะเดินทางไปถึงผู้รับ ประการต่อมาในระบบฐานข้อมูลแบบกระจายนั้น สามารถใช้วิธีการฟลัดดิ้งเข้ามาช่วยในการปรับปรุงข้อมูลในทุกฐานข้อมูลพร้อมกัน ประการที่สามคือการใช้ฟลัดดิ้งเพื่อการเปรียบเทียบอัลกอริทึมแบบต่างๆเนื่องจากวิธีฟลัดดิ้งจะส่งข้อมูลออกไปทุกทิศทางทำให้สามารถหาเส้นทางที่สั้นที่สุดได้เสมอ ยิ่งกว่านั้นไม่มีอัลกอริทึมใดที่สามารถส่งข้อมูลโดยใช้เวลาในการรอคอยต่ำกว่านี้ได้

3.4 การเลือกเส้นทางแบบกระจาย (Broadcasting Routing)

โปรแกรมประยุกต์ส่วนหนึ่งมีความต้องการให้โฮสต์สามารถส่งข่าวสารชุดเดียวกันไปยังโฮสต์อื่นๆได้หลายตัวหรือทุกตัวที่มีอยู่ในระบบ ตัวอย่างเช่น การให้บริการข่าวพยากรณ์อากาศ บริการข่าวตลาดหุ้นหรือการกระจายสัญญาณวิทยุบริการเหล่านี้ต้องการส่งข้อมูลให้แก่เครื่องทุกเครื่องที่เชื่อมต่อกับเครือข่าย การส่งข้อมูลวิธีนี้เรียกว่า “การกระจายข่าว (Broadcasting Routing)”

วิธีการกระจายข่าวแบบแรก เป็นการส่งข้อมูลโดยตรงซึ่งไม่ต้องการคุณสมบัติพิเศษใดๆจากระบบเครือข่ายย่อย คือการให้ผู้ส่งข้อมูลสร้างแพ็กเก็ตข้อมูลขึ้นมาเท่ากับจำนวนโหนดทั้งหมดที่มีอยู่ในเครือข่าย แต่ละแพ็กเก็ตจะระบบที่อยู่ของแต่ละโหนดไว้ แล้วจึงส่งแพ็กเก็ตทั้งหมดออกไป วิธีการนี้นอกจากจะเพิ่มปริมาณข้อมูลในระบบอย่างมากแล้ว ผู้กระจายข่าวยังมีความจำเป็นต้องเก็บรายชื่อ(ที่อยู่ประจำตัว) ของทุกโหนดไว้ด้วย ในทางปฏิบัติวิธีการนี้อาจเป็นวิธีการเดียวที่สามารถทำได้ แต่ก็มักจะถูกกำหนดให้เป็นหนทางเลือกสุดท้ายเสมอ

วิธีการฟลัดดิ้ง(Flooding) เป็นทางเลือกอีกทางหนึ่งที่มีความเป็นไปได้สูง แม้ว่าจะไม่เหมาะสมกับการสื่อสารแบบจุด-ต่อ-จุด แต่ก็มีความเหมาะสมเป็นอย่างมากสำหรับการกระจายข่าว โดยเฉพาะอย่างยิ่งถ้าวิธีการอื่นๆที่จะกล่าวถึงต่อไปนี้ไม่สามารถนำไปใช้งานได้ ปัญหาหลักของวิธีฟลัดดิ้งก็คือ ปัญหาการสร้างสำเนาข้อมูลจำนวนมากซึ่งจะทำให้ประสิทธิภาพโดยรวมของระบบลดลง

วิธีการที่ 3 เรียกว่า การใช้แพ็กเก็ตแบบหลายจุดหมาย (Multidestination Routing) แต่ละแพ็กเก็ตจะต้องบรรจุรายการที่อยู่ของจุดหมายปลายทางที่ต้องการทั้งหมดเอาไว้ ซึ่งอาจใช้วิธีการเก็บที่อยู่ประจำตัวของแต่ละจุดหมาย หรือใช้เทคนิคที่เรียกว่า “บิทแมบ (Bit Map)” ก็ได้ เมื่อแพ็กเก็ตเดินทางมาถึงเราท์เตอร์จะตรวจสอบรายการที่อยู่ของผู้รับทั้งหมดเพื่อเลือกสายสื่อสารที่จะต้องใช้ แล้วจึงสร้างสำเนาแพ็กเก็ตให้เท่ากับจำนวนสายสื่อสารที่เลือกไว้ จากนั้นจึงปรับปรุงรายการที่อยู่ผู้รับของแต่ละ
แพ็กเก็ตให้ถูกต้อง (มีเฉพาะที่อยู่ผู้รับที่อยู่ในเส้นทางนั้นๆ) ก่อนที่จะส่งออกไป หลังจากทำการกระจายข้อมูลไปพอสมควร แต่ละแพ็กเก็ตที่ยังไม่ถึงผู้รับจะกลายสภาพเป็นแพ็กเก็ตธรรมดา คือมีผู้รับเพียงโหนดเดียว การส่งแพ็กเก็ตข้อมูลแบบหลายจุดหมาย จึงมีลักษณะเหมือนกับการส่งข้อมูลแบบโดยตรง เพียงแต่เป็นการรวมกลุ่มแพ็กเก็ตที่ต้องเดินทางผ่านสายสื่อสารเส้นเดียวกันเข้าด้วยกัน

วิธีการที่ 4 นำหลักการ ซิงค์ทรี (Sink Tree) หรือ สแปนนิ่งทรี (Spanning Tree) มาใช้ในการส่งข้อมูลโดยกำหนดให้ผู้ส่งข้อมูลเป็นโหนดราก สแปนนิ่งทรีคือส่วนประกอบส่วนหนึ่งของรูปโครงสร้างเครือข่ายย่อยที่มีเราท์เตอร์ครบทุกตัว แต่อาจมีเส้นทางสื่อสารเพียงบางส่วนซึ่งจะต้องมากพอที่จะเชื่อมเราท์เตอร์ทุกตัวเข้าด้วยกัน

และมีเส้นทางเดิน (Path) จากโหนดรากไปยังโหนดใดๆ ได้เพียงเส้นทางเดียวเท่านั้น ถ้าแต่ละโหนด (เราท์เตอร์) ทราบว่าสายสื่อสารที่เชื่อมต่อกับตนเองเส้นใดบ้างเป็นเส้นทางที่อยู่ในสแปนนิ่งทรี ก็จะสร้างสำเนาแพ็กเก็ตเพื่อส่งไปตามเส้นทางเหล่านั้นทุกเส้น ยกเว้นเส้นที่รับแพ็กเก็ตเข้ามา วิธีการนี้ทำให้มีจำนวนแพ็กเก็ตอยู่ในระบบน้อยที่สุดเท่าที่จะทำให้งานสำเร็จได้ และยังทำให้การใช้งานเครือข่ายเป็นไปอย่างมีประสิทธิภาพ ส่วนปัญหาที่จะเกิดขึ้นก็คือ ทุกเราท์เตอร์จะต้องมีข้อมูลสแปนนิ่งทรีที่ทันสมัยอยู่ตลอดเวลา (ซึ่งอาจทำได้โดยการใช้แพ็กเก็ตบอกสถานะภาพการเชื่อมต่อ) แต่ในบางครั้งก็ไม่สามารถกระทำได้ (เช่น การใช้ตารางบอกระยะทาง)

ตัวอย่างสุดท้าย ของอัลกอริทึมการกระจายข่าวได้แก่ วิธีการสังเกตพฤติกรรมของการกระจายข่าว แม้ว่าเราท์เตอร์จะไม่มีข้อมูลเกี่ยวกับสแปนนิ่งทรีเลยก็ตาม แต่เมื่อแพ็กเก็ตแบบกระจายข่าวเดินทางมาถึงเราท์เตอร์จะสามารถตรวจสอบได้ว่าแพ็กเก็ตเดินทางมาโดยใช้สายสื่อสารที่โดยปกติแล้วใช้เป็นทางในการส่งข้อมูลไปยังแหล่งกำเนิดข้อมูลหรือไม่ ถ้าใช่ก็แสดงว่าแพ็กเก็ตนั้นมีโอกาสเป็นไปได้มากที่จะเป็นแพ็กเก็ตใหม่ที่เพิ่งส่งมาจากแหล่งข้อมูล จึงสร้างสำเนาข้อมูลแล้วส่งออกไปทุกเส้นทางยกเว้นเส้นทางที่รับเข้ามา แต่ถ้าไม่ใช่แสดงว่าแพ็กเก็ตนั้นอาจเป็นข้อมูลซ้ำจึงต้องลบทิ้งไป

ตัวอย่างของการใช้อัลกอริทึมได้แก่ วิธีการที่เรียกว่า รีเวอร์สพาร์ธฟอร์เวิร์ดดิ้ง (Reverse Path Forwarding) ดังแสดงในรูปที่ 9 ในส่วน(a) แสดงโครงสร้างเครือข่ายย่อย ส่วน(b) แสดงซิงค์ทรีสำหรับเราท์เตอร์ I และในส่วน (c)

รูปที่ 9 รีเวอร์สพาร์ธฟอร์เวิร์ดดิ้ง (Reverse Path Forwarding)

(a) ระบบเครือข่ายย่อย (b) ซิงค์ทรี (c) ทรีที่สร้างขึ้นมาด้วยวิธี Reverse Path Forwarding

แสดงวิธีการทำงานของ Reverse Path Forwarding ในการส่งข้อมูลรอบแรก โหนด I ส่งแพ็กเก็ตไปโหนด F, H, J,และ N ดังแสดงไว้ในโหนดระดับที่สองในรูป แต่ละแพ็กเก็ตถูกส่งมาตามเส้นทางที่เหมาะสมที่สุดจากโหนด I (บางส่วนของเส้นทางที่เหมาะสมเหล่านี้อยู่ในเส้นทางของซิงค์ทรี ซึ่งแสดงให้เห็นโดยการใช้วงกลมล้อมรอบ) ในการส่งผ่านรอบที่ 2 จะมีแพ็กเก็ตจำนวน 8 ตัวที่ถูกส่งต่อไป(ตามรูป) ซึ่งจะส่งไปยังโหนดที่ไม่เคยส่งมาก่อนเลย เส้นทาง 5 จาก 8 เส้นทางเป็นเส้นทางที่อยู่ในซิงค์ทรี การส่งผ่านข้อมูลรอบที่ 3 แพ็กเก็ต 3 ตัวจากจำนวนทั้งหมด 6 ตัวจะส่งผ่านเส้นทางที่อยู่ในซิงค์ทรี (C, E, และ K) นอกนั้นเป็นข้อมูลซ้ำ (G, D, และ N) หลังจากส่งผ่านข้อมูล 5 รอบจะมีจำนวนแพ็กเก็ตทั้งสิ้น 23 ตัว และการกระจายข้อมูลจึงยุติเพราะได้ส่งข้อมูลไปครบทุกโหนดแล้ว ถ้าใช้การส่งข้อมูลตามเส้นทางซิงค์ทรีอย่างเคร่งครัด จะเกิดการส่งข้อมูลเพียง 4 รอบ ด้วยแพ็กเก็ตเพียง 14 ตัว

ข้อเด่นของการใช้วิธี Reverse Path Forwarding คือเป็นวิธีการที่มีประสิทธิภาพและง่ายแก่การนำไปใช้จริง เราท์เตอร์ไม่จำเป็นต้องมีข้อมูลของสแปนนิ่งทรี หรือข้อมูลรายการที่อยู่ประจำตัวของเราท์เตอร์ทั้งหมดในระบบดังที่การกระจายแบบหลายจุดหมายต้องใช้ และไม่ต้องการวิธีการยุติการส่งข้อมูล เหมือนอย่างวิธีฟลัดดิ้งใช้

3.5 การเลือกเส้นทางแบบกระจายหลายจุด (Multicasting Routing)

โปรแกรมประยุกต์บางอย่างสามารถที่จะกระจายการประมวลผลออกไปยังเครื่องคอมพิวเตอร์เครื่องอื่นภายในกลุ่มของตนเพื่อช่วยกันทำงานได้ เช่น การประมวลผลของระบบฐานข้อมูลแบบกระจาย (Distributed Database System) ซึ่งจะต้องมีตัวจัดการอย่างน้อย 1 ตัวที่คอยส่งข้อความต่างๆไปให้สมาชิกภายในกลุ่มได้รับทราบข่าวสารแบบจุดต่อจุด (Point To Point) แต่ถ้าเป็นกลุ่มขนาดใหญ่ วิธีนี้จะไม่มีประสิทธิภาพเพียงพอ จึงมักจะใช้วิธีการแพร่กระจายข้อมูล (Broadcasting) แต่ก็จะเกิดการใช้งานที่ไม่มีประสิทธิภาพ ถ้าต้องแพร่กระจายข่าวสารไปยังเครื่องคอมพิวเตอร์พันๆเครื่องบนระบบเครือข่ายที่ประกอบด้วยโหนดหลายล้านโหนด

เพราะว่าผู้รับข่าวสารส่วนใหญ่จะไม่สนใจในข่าวสารที่แพร่ออกมา หรือในทางกลับกัน การแพร่ข่าวสารในเครือข่ายควรกำหนดให้นำส่งเฉพาะผู้ที่ควรจะได้รับข่าวสาร ดังนั้นจึงต้องมีวิธีการส่งข่าวสารไปยังสมาชิกของกลุ่มที่มีจำนวนมากพอสมควร แต่ก็ยังจัดว่าเป็นจำนวนน้อยเมื่อเทียบกับจำนวนโหนดทั้งหมดในเครือข่าย

การส่งข่าวสารไปยังสมาชิกในกลุ่มของตนเองบนเครือข่าย เรียกว่า “Multicasting” และวิธีการจัดหาเส้นทางเพื่อส่งข่าวสารไปยังสมาชิกในกลุ่มของตนเองบนเครือข่าย เรียกว่า “Multicasting”

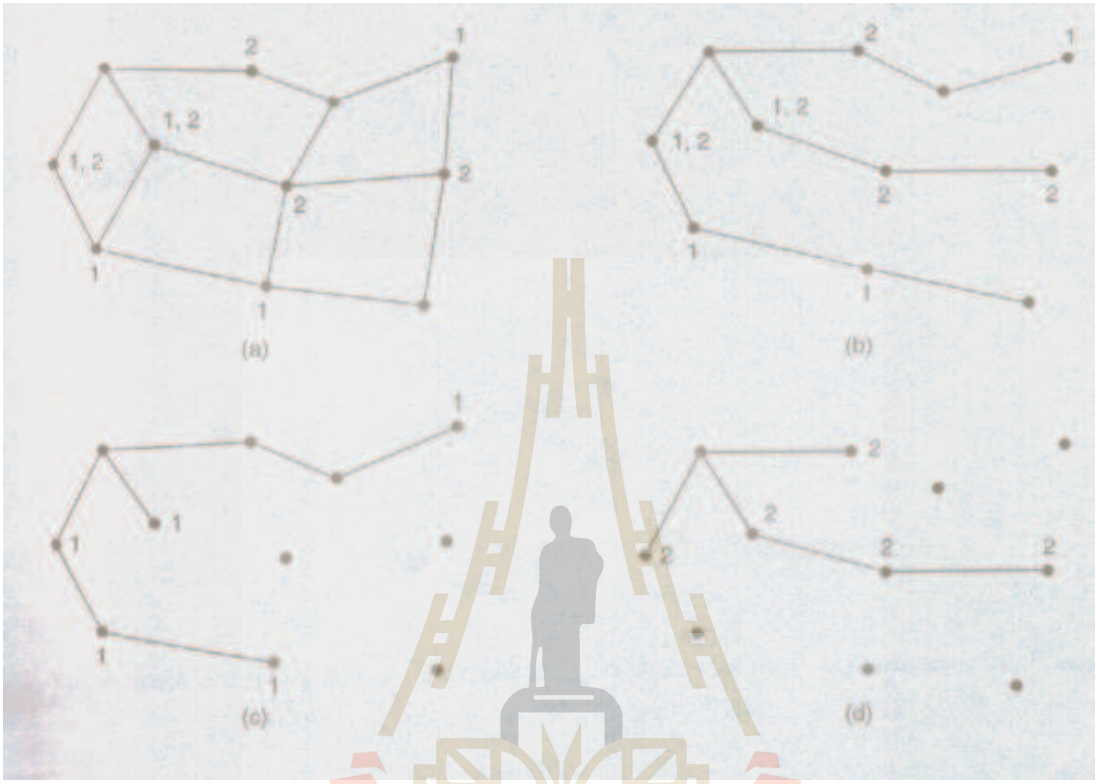
และวิธีการการจัดการหาเส้นทางเพื่อส่งข่าวสารเรียกว่า“การเลือกเส้นทางแบบกระจายหลายจุด (Multicasting Routing) ” ในหัวข้อนี้จะกล่าวถึงวิธีการหนึ่งที่ใช้เลือกเส้นทางที่จะทำการแพร่กระจายข่าวสาร

ในการแพร่กระจายข่าวสารแบบหลายจุดนั้น เริ่มต้นด้วยการจัดตั้งกลุ่ม ซึ่งจะต้องมีวิธีการสร้างหรือทำลายกลุ่ม วิธีการเข้าร่วมเป็นสมาชิกในกลุ่ม และวิธีการลาออกจากกลุ่ม อย่างไรก็ตามกระบวนการทำงานต่างๆ ที่กล่าวถึงนี้ไม่ได้อยู่ในกระบวนการเลือกเส้นทางส่งข้อมูลเลย แต่ที่เกี่ยวกับการหาเส้นทางส่งข้อมูลคือ เมื่อโปรเซสเข้ามาเป็นสมาชิกในกลุ่ม จะต้องทำการแจ้งบอกไปยังเครื่องโฮสต์ประจำเครือข่ายย่อย และโฮสต์จะต้องแจ้งต่อไปยังเราท์เตอร์ เพราะเราท์เตอร์จำเป็นต้องทราบว่าบรรดาโฮสต์ที่มีนั้นแต่ละโฮสต์มีใครบ้างเป็นสมาชิกในกลุ่มใด ข้อมูลส่วนนี้อาจเป็นหน้าที่ของโฮสต์ที่จะต้องเป็นผู้แจ้งให้เราท์เตอร์ทราบ หรืออีกทางหนึ่งคือเราท์เตอร์จะต้องคอยสอบถามไปยังบรรดาโฮสต์ทั้งหลายว่ามีการเปลี่ยนแปลงเกี่ยวกับสมาชิกบ้างหรือไม่ ท้ายที่สุดเราท์เตอร์แต่ละตัวจะต้องเรียนรู้ข้อมูลเหล่านี้และแลกเปลี่ยนข้อมูลกับเราท์เตอร์อื่นๆ เพื่อให้ทั้งเครือข่ายย่อยนั้นทราบข้อมูลได้ทั่วกัน

ในการเลือกเส้นทางในการส่งข่าวสารนั้น เราท์เตอร์แต่ละตัวจะทำการสร้างสแปนนิ่งทรี (Spanning Tree) ของตัวเองซึ่งจะต้องครอบคลุมทุกๆ เราท์เตอร์ทั่วทั้งเครือข่ายย่อย ดังตัวอย่างในรูปที่ 10 (a)ในเครือข่ายย่อยประกอบด้วย 2 กลุ่มตามที่ปรากฏส่วนรูปที่ 10 (b) แสดงภาพของสแปนนิ่งทรีของเราท์เตอร์ที่อยู่ด้านซ้ายสุดและเป็นตัวบนสุด

เมื่อโปรเซสทำการแพร่กระจายแพ็กเก็ตข่าวสารออกไปยังกลุ่มของตน เราท์เตอร์ตัวแรกที่รับแพ็กเก็ตได้จะเป็นตัวตรวจสอบสแปนนิ่งทรีอย่างทั่วถึง เส้นทางใดไม่สามารถติดต่อกับโฮสต์ที่เป็นสมาชิกในกลุ่มได้ก็จะตัดเส้นทางนั้นออกไป ดังตัวอย่างในรูป 10 (c) แสดงในเห็นถึงผลการสร้างเส้นทางสำหรับสมาชิกกลุ่มที่ 1 ตัวอย่างในรูป 10

(d) แสดงให้เห็นถึง ผลการสร้างเส้นทางสำหรับสมาชิกกลุ่มที่ 2 แพ็กเก็ตที่ถูกส่งด้วยวิธีการแจ้งข่าวแบบกระจายนี้ จะถูกผ่านไปตามเส้นทางของกลุ่มเท่านั้น

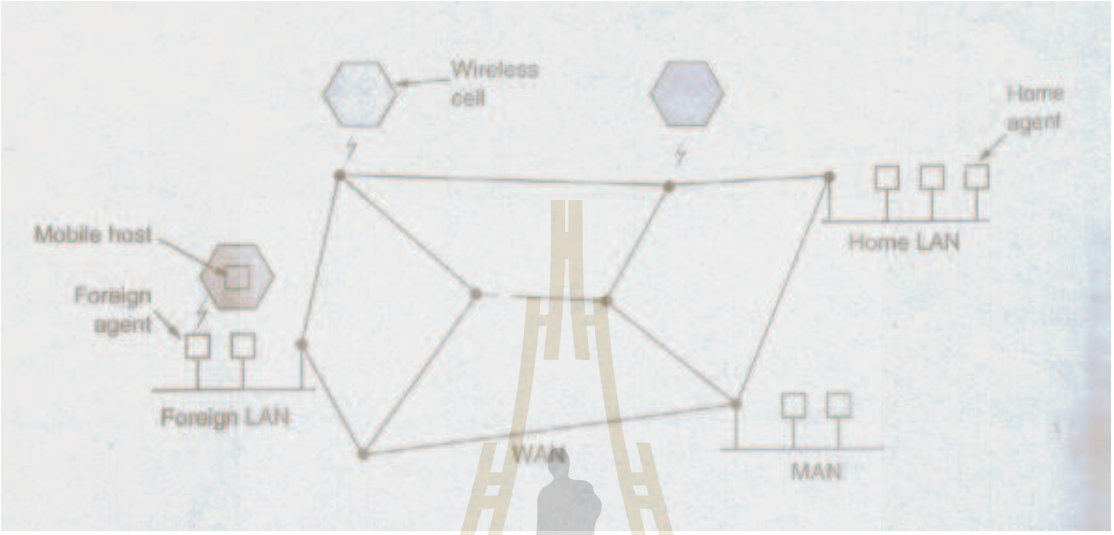


รูปที่ 10 (a) ระบบเครือข่าย (b) สแปนนิ่งทรีแบบเลฟท์โมสท์ (Leftmost)
(c) มัลติแคสท์ทรี (Multicast Tree) สำหรับกลุ่ม 1 (d) มัลติแคสท์ทรี Multicast Tree สำหรับกลุ่ม 2

3.6 การเลือกเส้นทางสำหรับแม่ข่ายเคลื่อนที่ (Mobile Network Routing)

ปัจจุบันมีผู้คนหลายล้านคนทั่วโลกที่นิยมใช้เครื่องคอมพิวเตอร์แบบนำพาไปได้ (Portable Computers) ผู้คนเหล่านี้ต้องการใช้บริการจดหมายอิเล็กทรอนิกส์และใช้บริการระบบแฟ้มข้อมูลที่สำนักงานในทุกสถานที่ที่เขาเดินทางไปทำธุรกิจ การใช้งานเครื่องคอมพิวเตอร์ในระหว่างการเดินทางจำเป็นต้องอาศัยการติดต่อสื่อสารแบบไร้สาย (Wireless Communication) ซึ่งจะติดต่อกับระบบแม่ข่ายสื่อสารที่สามารถให้บริการแก่ลูกข่ายที่กำลังเคลื่อนที่ได้

รูปโครงสร้างของระบบเครือข่ายสื่อสารแบบไร้สายแสดงไว้ในรูปที่ 11 ระบบนี้ประกอบด้วยระบบเครือข่ายวงกว้างที่มีทั้งเราท์เตอร์และโฮสต์คอมพิวเตอร์ นอกจากนั้นยังมีระบบเครือข่ายเฉพาะที่ ระบบเครือข่ายในเขตเมือง และโหนดสื่อสารไร้สาย (Cells)



รูปที่ 11 ระบบเครือข่าย WAN ที่มีระบบ LAN, MAN และเซลไร้สายเชื่อมต่ออยู่ด้วย

ผู้ใช้ที่ติดต่อสื่อสารผ่านระบบเครือข่าย สามารถแบ่งออกได้เป็น 3 ประเภทคือ ประเภทแรกเป็นผู้ใช้ที่นั่งทำงานประจำที่ (Stationary Users) จะติดต่อกับระบบเครือข่ายผ่านสายสื่อสารแบบต่างๆ ได้แก่ ผู้ใช้คอมพิวเตอร์สำหรับงานทั่วไป ประเภทที่ 2 คือผู้ใช้ที่ย้ายที่ทำงานอยู่เสมอคือจะมีสถานที่ทำงานอยู่หลายแห่ง (Migratory Users) ก็จะติดต่อกับระบบเครือข่ายผ่านสื่อประเภทต่างๆ ได้แก่ ผู้บริหารระดับสูงที่ต้องคอยดูแลกิจการหลายแห่ง ประเภทที่ 3 เป็นผู้ใช้ ที่ติดต่อรับ-ส่งข้อมูลผ่านระบบเครือข่ายในขณะที่ตนเองกำลังเคลื่อนที่ (Roaming Users) เช่น ผู้ใช้เครื่องแล็บท็อปคอมพิวเตอร์ที่กำลังนั่งทำงานอยู่บนเครื่องบินโดยมีการรับ-ส่งข้อมูลกับเครื่องคอมพิวเตอร์เครื่องอื่นผ่านเครือข่ายสื่อสาร
ไร้สาย ในหัวข้อนี้จะใช้คำว่า “ผู้ใช้สัญจร (Mobile Users)” แทนผู้ใช้ 2 ประเภทหลัง คือเป็นผู้ใช้ที่ไม่ได้นั่งทำงานประจำที่

ผู้ใช้สัญจรทุกคนต้องมีเครื่องคอมพิวเตอร์ในสำนักงานที่มีการเชื่อมต่อกับระบบเครือข่ายอย่างถาวรหรืออย่างน้อยที่สุดก็จะต้องมีที่อยู่ประจำตัว (Home Address)

ซึ่งเป็นที่อยู่บนระบบเครือข่าย (Network Address) เป็นของตนเอง และที่สำคัญที่สุดคือจะต้องจดทะเบียนที่อยู่ประจำตัวไว้ ณ เครื่องโฮสต์ของระบบเครือข่ายย่อยแห่งใดแห่งหนึ่ง วัตถุประสงค์ของระบบที่ให้บริการแก่ผู้ใช้สัญจรคือจะต้องทำหน้าที่เป็นตัวกลางในการรับและส่งข้อมูลให้แก่ผู้ใช้สัญจรคือจะต้องทำหน้าที่เป็นตัวกลางในการรับและส่งข้อมูลให้แก่ผู้ใช้สัญจรซึ่งอาจจะอยู่ที่ไหนก็ได้ ดังนั้นการค้นหาตัว (เครื่องคอมพิวเตอร์) ผู้ใช้สัญจรจึงเป็นปัญหาแรกที่ต้องแก้ไขให้ได้

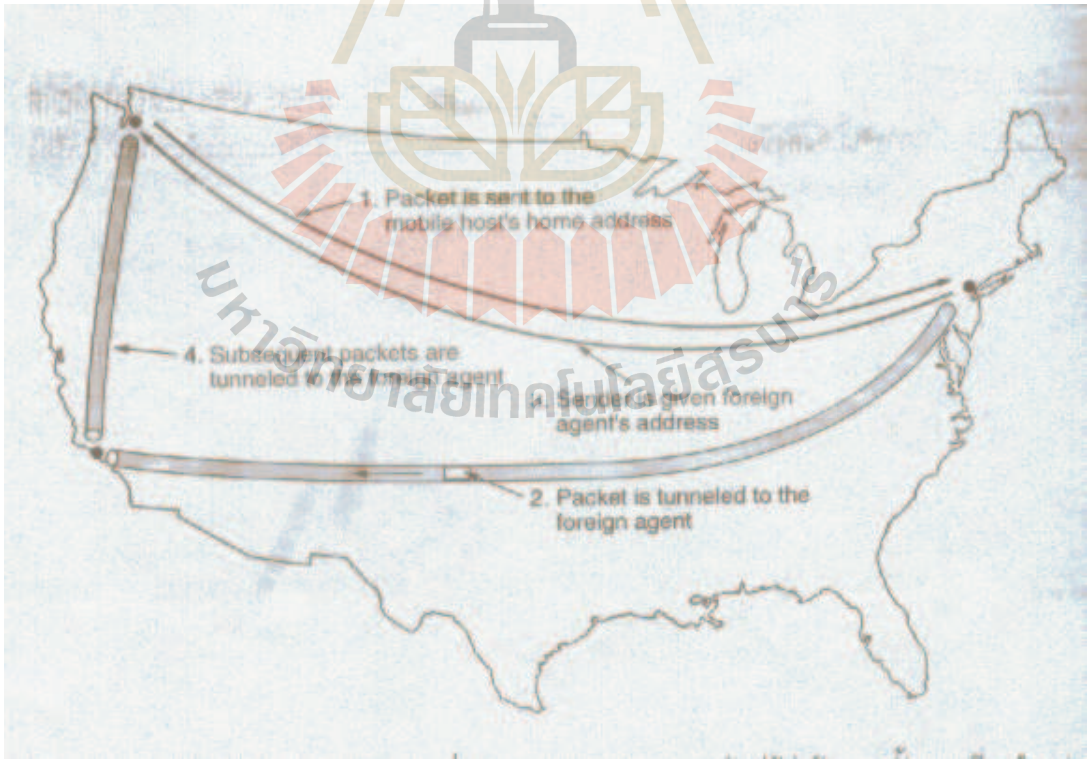
จากรูปที่ 11 ขอบเขตการให้บริการถูกแบ่งออกเป็นส่วนย่อยๆ เรียกว่า “พื้นที่ (Region)” ซึ่งโดยปกติจะเป็นระบบเครือข่ายเฉพาะที่ (LAN) หรือโหนดสื่อสารไร้สาย (Cell) แต่ละพื้นที่จะมี “หน่วยเฝ้าตรวจ (Foreign Agent) ” ซึ่งทำหน้าที่ในการตรวจตราและบันทึกข้อมูลของผู้ใช้สัญจรทุกคนที่กำลังอยู่ในพื้นที่นั้นๆ และจะมี “หน่วยบ้าน (Home Agent)” ทำหน้าที่ในการตรวจตรา และบันทึกข้อมูลของผู้ใช้สัญจรทุกคนที่จดทะเบียนไว้ในเขตพื้นที่นั้นแต่กำลังอยู่ในพื้นที่อื่น

เมื่อผู้ใช้สัญจรเริ่มต้นทำงานในเขตพื้นที่หนึ่ง ไม่ว่าจะเพิ่งเริ่มการติดต่อหรืออาจกำลังติดต่ออยู่แล้วได้เดินทางออกนอกพื้นที่เดิมเข้าสู่เขตพื้นที่ใหม่ ผู้ใช้สัญจร (เครื่องคอมพิวเตอร์) จะต้องทำการจดทะเบียนกับหน่วยเฝ้าตรวจในเขตพื้นที่นั้นทันที ขั้นตอนในการลงทะเบียนมีรายละเอียดคือ

1. ในทุกระยะเวลาที่กำหนด หน่วยเฝ้าตรวจของแต่ละพื้นที่จะต้องส่งแพ็กเก็ตข้อมูลแบบกระจายข่าวเพื่อประกาศที่อยู่ของตนเองให้แก่ผู้ใช้สัญจรที่เพิ่งเข้ามาในเขตความรับผิดชอบ หรือในบางครั้งผู้ใช้สัญจรอาจเป็นผู้ส่งแพ็กเก็ตถามหาแพ็กเก็ตข้อมูลจากหน่วยเฝ้าตรวจเองก็ได้
2. ผู้ใช้สัญจรจะต้องลงทะเบียนกับหน่วยเฝ้าตรวจในพื้นที่นั้นเพื่อแจ้งให้ทราบข้อมูลเกี่ยวกับที่อยู่ประจำตัว วิธีการติดต่อในชั้นสื่อสารควบคุมเครือข่าย และรหัสประจำตัว (เพื่อประโยชน์ในการรักษาความปลอดภัย)
3. หน่วยเฝ้าตรวจจะต้องติดต่อกับเครื่องโฮสต์ของผู้ใช้สัญจรเพื่อแจ้งให้ทราบว่าขณะนี้ผู้ใช้สัญจรได้เข้ามาอยู่ในเขตพื้นที่ของตนเองแล้ว โดยจะแจ้งที่อยู่เครือข่ายของตนเองให้ทราบพร้อมทั้งแจ้งรหัสประจำตัวของผู้ใช้สัญจรเพื่อเป็นการยืนยันความถูกต้อง

4. เครื่องโฮสต์จะตรวจสอบความถูกต้องของรหัสประจำตัวของผู้ใช้สัญจร ถ้าเป็นรหัสฯที่ถูกต้องตามที่ปรากฏในทะเบียนโฮสต์ก็จะตอบยืนยันกลับไปยังหน่วยเฝ้าตรวจ มิฉะนั้นจะตอบปฏิเสธ
5. เมื่อหน่วยเฝ้าตรวจได้รับการยืนยันแล้วจึงจะแจ้งให้ผู้ใช้สัญจรทราบ กระบวนการลงทะเบียนก็เสร็จสิ้นเพียงเท่านี้

หลังจากการลงทะเบียนเสร็จสิ้นเรียบร้อยแล้ว จะเกิดกระบวนการทำงาน 4 ขั้นตอนเพื่อจัดส่งแพ็กเก็ตข้อมูลให้แก่ผู้ใช้สัญจรดังแสดงในรูปที่ 12 ขั้นตอนแรก แพ็กเก็ตจะถูกส่งมายังที่อยู่ประจำตัวของผู้ใช้สัญจรนั้นด้วยวิธีการใดๆก็ได้ตามปกติ ก่อนที่แพ็กเก็ตนี้จะถูกส่งต่อไปถึงเครื่องโฮสต์ของผู้ใช้สัญจร หน่วยบ้านในเขตพื้นที่นั้น (อาจเป็นโฮสต์หรือเราท์เตอร์ก็ได้) จะกักข้อมูลเอาไว้แล้วค้นหาที่อยู่ของหน่วยเฝ้าตรวจที่กำลังดูแลผู้ใช้สัญจร (ขั้นตอนที่ 1 ในรูปที่ 12) จากนั้นจึงบรรจุแพ็กเก็ตข้อมูลไว้ในแพ็กเก็ตพิเศษสำหรับการส่งข้อมูลระหว่างโฮสต์แล้วส่งไปให้หน่วยเฝ้าตรวจ (ขั้นตอนที่ 2 ในรูปที่ 12) วิธีการนี้เรียกว่า จูนเนลลิ่ง (Tunneling) ซึ่งจะกล่าวถึงรายละเอียดในภายหลัง หน่วยเฝ้าตรวจจะดึงแพ็กเก็ตข้อมูลจริงออกมาแล้วส่งไปให้ผู้ใช้สัญจรในลักษณะของแพ็กเก็ตระดับชั้นสื่อสารเชื่อมต่อข้อมูล



รูปที่ 12 การค้นหาเส้นทางเดินแพ็กเก็ตสำหรับผู้ใช้สัญจร

ในขณะเดียวกัน หน่วยบ้านจะแจ้งที่อยู่ของหน่วยเฝ้าตรวจให้แก่ผู้ส่งข้อมูล (ขั้นตอนที่ 3 ในรูปที่ 12) หลังจากนี้ผู้ส่งข้อมูลก็จะสามารถส่งข้อมูลไปยังผู้เฝ้าตรวจได้โดยตรง (ขั้นตอนที่ 4 ในรูปที่ 12) โดยไม่ต้องส่งมาที่หน่วยบ้านอีกต่อไป

ยังมีวิธีอื่นอีกมากมาย ที่นำมาใช้ในการค้นหาและติดต่อกับผู้ใช้สัญจร ซึ่งมีความแตกต่างกันหลายประการ คือ ประการแรกโปรโตคอลจะต้องแบ่งการทำงานระหว่างเราท์เตอร์ และเครื่องโฮสต์ให้เหมาะสม โดยเฉพาะที่เครื่องโฮสต์จะต้องกำหนดว่าการติดต่อนั้นอยู่ในความรับผิดชอบของชั้นสื่อสารใด ประการที่สองบางวิธีได้กำหนดให้เราท์เตอร์ตัวอื่นที่อยู่ในเส้นทางการส่งผ่านข้อมูลมีความสามารถในการบันทึกที่อยู่หน่วยเฝ้าตรวจของผู้ใช้สัญจรได้ และอนุญาตให้เราท์เตอร์ดักจับแพ็กเก็ตเพื่อส่งไปให้หน่วยเฝ้าตรวจได้ในทันทีโดยไม่จำเป็นต้องรอให้หน่วยบ้านหรือโฮสต์ของผู้ใช้สัญจรเป็นผู้ทำงานในขั้นตอนนี้ ผลที่ได้รับคือทำให้แพ็กเก็ตเดินทางสั้นลงอย่างเห็นได้ชัดเจน ประการที่สามบางวิธีจะให้หน่วยเฝ้าตรวจของพื้นที่นั้นๆ เป็นผู้กำหนดหมายเลขที่อยู่ชั่วคราว ซึ่งไม่ซ้ำกับใครให้แก่ผู้ใช้สัญจรเมื่อเข้าไปในเขตพื้นที่ให้บริการของตน ในบางกรณีที่อยู่ชั่วคราวจะหมายถึงที่อยู่ของหน่วยพิเศษที่คอยให้บริการแก่ผู้สัญจรทั้งหมดแต่ละพื้นที่

ประการที่สี่วิธีการให้บริการจะแตกต่างกันในกระบวนการจัดส่งแพ็กเก็ตจากจุดหมายที่ระบุไว้ไปยังจุดหมายอื่น หมายถึงการทำ Tunneling, ขั้นตอนที่ 2 ในรูปที่ 12 หนทางแรกคือการเปลี่ยนที่อยู่ผู้รับก่อนที่จะส่งแพ็กเก็ตออกไป แม้ว่าจะเป็นวิธีการตรงไปตรงมา แต่มักจะเกิดปัญหาเมื่อต้องส่งแพ็กเก็ตผ่านระบบเครือข่ายต่างชนิดกัน อีกหนทางหนึ่งคือการนำแพ็กเก็ตไปใส่ไว้ในแพ็กเก็ตตัวใหม่ เรียกว่า การเอนแคพซูเลท (Encapsulate) ข้อมูลที่ระบุที่อยู่ผู้รับได้ตามต้องการได้โดยไม่ต้องแก้ไขแพ็กเก็ตเดิม ท้ายที่สุดวิธีการให้บริการจะแตกต่างกันในเรื่องของการรักษาความปลอดภัย โดยทั่วไป การที่ผู้ส่งข้อมูลได้รับแจ้งจากโหนดที่ผู้ส่งอาจไม่รู้จักให้ส่งข้อมูลไปยังที่อยู่ใหม่ที่ไม่ใช่ที่อยู่ของผู้รับข้อมูลก็ย่อมจะต้องเกิดความสงสัยว่าทำไมต้องทำเช่นนั้น การใช้รหัสประจำตัวของผู้รับข้อมูลจึงถูกนำมาใช้ในการขจัดความสงสัยนี้

3.7 การค้นหาเส้นทางเดินข้อมูลในระบบเครือข่ายแอดฮอค (Routing in Ad Hoc Networks)

ในหัวข้อที่ผ่านมาได้กล่าวถึงวิธีการค้นหาเส้นทางเดินข้อมูลสำหรับผู้ใช้สัญจรที่เคลื่อนที่อยู่เสมอแต่ตำแหน่งของเราท์เตอร์นั้นนิ่งอยู่กับที่ ในสถานการณ์ที่มีความสับสนวุ่นวายกว่านี้ คือการที่เราท์เตอร์นั้นเคลื่อนที่ด้วย สถานการณ์ที่อาจเป็นไปได้คือ

1. ยานพาหนะของหน่วยทหารที่เคลื่อนที่อยู่ภายในสนามรบซึ่งไม่มีโครงข่ายการสื่อสารที่คงที่
2. กองเรือที่เคลื่อนที่อยู่ในทะเล
3. ผู้ทำงานฉุกเฉินอยู่ในพื้นที่ที่เกิดแผ่นดินไหวซึ่งโครงข่ายการสื่อสารข้อมูลได้ถูกทำลายไปแล้ว
4. กลุ่มคนทำงานที่มีเครื่องคอมพิวเตอร์โน๊ตบุ๊คเป็นของตนเองซึ่งไม่มีโปรโตคอล 802.11 ใช้งาน

ในทุกกรณีที่กล่าวถึงนี้และในกรณีอื่นๆ แต่ละโหนดประกอบด้วยเราท์เตอร์และโฮสต์ซึ่งจะอยู่ในเครื่องคอมพิวเตอร์เครื่องเดียวกัน ระบบเครือข่ายของโหนดที่ตั้งอยู่ใกล้เคียงกัน โดยไม่ได้วางแผนล่วงหน้า นั้นเรียกว่า “ระบบเครือข่ายแอดฮอค (Ad Hoc Networks)” หรือ MANETs (Mobile Ad Hoc Networks)

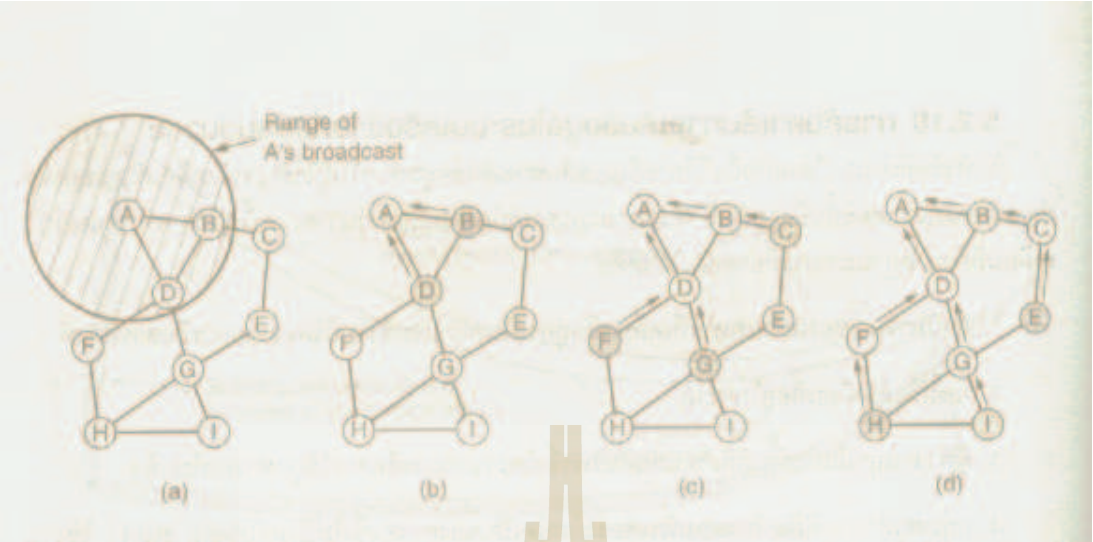
ระบบเครือข่ายแบบแอดฮอคนี้ มีความแตกต่างจากระบบเครือข่ายคงที่แบบใช้สายสื่อสารตรงที่กฎเกณฑ์ต่างๆ ที่เกี่ยวข้องกับรูปทรงระบบเครือข่าย โหนดข้างเคียง ความสัมพันธ์ระหว่างหมายเลขไอพี (IP) และตำแหน่งที่อยู่ และอื่นๆ นั้นไม่สามารถใช้งานได้เลย เราท์เตอร์อาจจะมีอาการ “เดี๋ยวก็อยู่เดี๋ยวก็ไป” หรือย้ายตำแหน่งไปที่อยู่อื่นได้ตลอดเวลา ในระบบเครือข่ายแบบคงที่ที่ใช้สายสื่อสารนั้น ถ้าเราท์เตอร์ทราบเส้นทางไปยังโหนดใดก็ตาม เส้นทางนั้นจะอยู่คงที่ตลอดไป ยกเว้นในกรณีที่สายสื่อสารถูกตัดขาดหรือเราท์เตอร์เสีย) ในระบบเครือข่ายแบบแอดฮอคอาจมีรูปทรง (Topology) ที่เปลี่ยนแปลงอยู่ตลอดเวลาทำให้เส้นทางเดินข้อมูลเปลี่ยนแปลงตามไปด้วย

โดยไม่จำเป็นจะต้องมีการเตือนให้ทราบล่วงหน้า สถานการณ์เช่นนี้ทำให้การค้นหาเส้นทางเดินข้อมูลในระบบเครือข่ายแบบแอดฮอคนั้นแตกต่างไปจากแบบที่ใช้ในระบบเครือข่ายคงที่

วิธีการค้นหาเส้นทางเดินข้อมูลในระบบเครือข่ายแบบแอดฮอคนั้นได้รับการพัฒนาขึ้นมาหลายวิธี วิธีการหนึ่งเรียกว่า เอโอดีวี (Ad Hoc On-Demand Distance Vector : AODV) ซึ่งเป็นวิธีการที่ใกล้เคียงกับวิธีของเบลแมนฟอร์ดดิสแทนซ์เวกเตอร์ (Bellman-Ford Distance Vector) แต่ได้รับปรับปรุงให้สามารถนำมาใช้ในสถานการณ์ของผู้ใช้สัญจรได้ และยังได้พิจารณาถึงขนาดความกว้างของช่องสัญญาณที่ค่อนข้างจำกัด รวมทั้งพลังงานในแบตเตอร์รี่ที่อาจกำลังจะหมดลง ขีดความสามารถที่ไม่ธรรมดาของวิธีการนี้คือเป็นอัลกอลิทึมแบบออนดีมาน (On-Demand) ซึ่งจะทำการค้นหาเส้นทางก็ต่อเมื่อมีความต้องการที่จะส่งข้อมูลเท่านั้น

การค้นหาเส้นทางเดินข้อมูล

ในเวลาใดก็ตาม ระบบเครือข่ายแอดฮอค สามารถอธิบายได้โดยใช้รูปกราฟของโหนดต่างๆ (เราท์เตอร์ และโฮสต์) โหนด 2 โหนดจะเชื่อมต่อกัน (มีเส้นเชื่อมถึงกันในรูปกราฟ) เมื่อทั้ง 2 โหนดนั้นสามารถสื่อสารถึงกันได้โดยใช้สัญญาณวิทยุของตนเอง เนื่องจาก โหนด 1 ใน 2 โหนดนั้นอาจมีขีดความสามารถในการส่งวิทยุได้ไกลกว่าอีกโหนดหนึ่ง จึงเป็นไปได้ว่าโหนด A จะสามารถเชื่อมต่อไปยังโหนด B โดยที่โหนด B ไม่สามารถเชื่อมต่อไปยังโหนด A อย่างไรก็ตาม เพื่อความง่ายจึงถือว่าการสื่อสารนั้นจะเป็นแบบ 2 ทางหรือไม่มีการสื่อสารถึงกันเลย นอกจากนี้ยังมีข้อสังเกตอีกว่า ถ้าโหนด 2 โหนดอยู่ภายในระยะการส่งสัญญาณวิทยุของกันและกันแล้ว โหนดทั้ง 2 อาจไม่มีการเชื่อมต่อระหว่างกันเลยก็ได้ แลภายในเขตการสื่อสารอาจมีอาคารเนินเขาหรืออุปสรรคอื่นๆ ที่ขัดขวางการสื่อสารระหว่างโหนดต่างๆ ก็ได้



รูปที่ 13 (a) ขอบเขตการส่งสัญญาณวิทยุของโหนด A

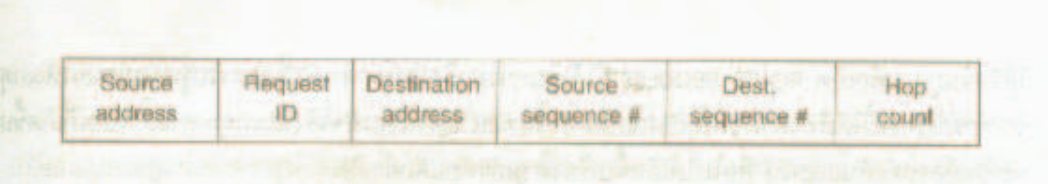
(b) ภายหลังจากที่โหนด A โหนด B และโหนดD ได้รับสัญญาณวิทยุจากโหนดA

(c) ภายหลังจากที่ C, F, และ G ได้รับข้อมูลจากโหนด A

(d) ภายหลังจากที่โหนด E, H, และ I ได้รับข่างสารจากโหนด A ส่วน ที่แรเงาหมายถึง รับข้อมูลใหม่ลูกศรแสดงให้เห็นทิศทางที่เป็นไปได้ในการส่งข้อมูลย้ายกลับ

พิจารณารูประบบเครือข่ายแอดฮอค ในรูปที่ 13 โปรเซสที่โหนด A ต้องการส่งแพ็กเก็ตไปยังโหนด I อัลกอลิทึม AODV จะมีตารางข้อมูลอยู่ในแต่ละโหนดที่เรียงลำดับโดยโหนดเป้าหมาย ตารางนี้จะให้ข้อมูลเกี่ยวกับโหนดเป้าหมายรวมทั้งชื่อโหนดที่จะต้องส่งแพ็กเก็ตออกไป เพื่อให้ไปถึงเป้าหมายที่ต้องการ สมมุติว่าโหนด A ค้นหาตารางข้อมูลของตนเองแต่ไม่พบโหนดเป้าหมาย I โหนด A จึงต้องเรียกใช้อัลกอลิทึมนี้ (ถ้ามีโหนดเป้าหมายอยู่ในตารางก็จะไม่เรียกใช้)

ในการค้นหาโหนด I โหนด A จะสร้างแพ็กเก็ตพิเศษที่เรียกว่า เราท์รีเควส (Route Request)และจัดการกระจายแพ็กเก็ตนี้ออกไป ในรูปที่ 13(a) แพ็กเก็ตจะเดินทางไปยังโหนด B และ D เนื่องจากโหนดทั้งสองนี้อยู่ภายในขอบเขตการส่งสัญญาณวิทยุจากโหนด A แต่โหนด F จะไม่รวมอยู่ในนี้เพราะอยู่นอกขอบเขตการกระจายสัญญาณวิทยุของโหนด A



รูปที่ 14 โครงสร้างของแพ็กเก็ตเราท์รีเควสท์ (Route Request)

รูปที่ 14 แสดงโครงสร้างของแพ็กเก็ต Route Request ซึ่งประกอบด้วยหมายเลขที่อยู่ของผู้รับและผู้ส่งข้อมูลโดยปกติจะใช้หมายเลขไอพี (IP) ซึ่งบอกให้ทราบว่าใครกำลังมองหาใคร เขตข้อมูลรีเควสท์ไอดี (Request ID) เป็นเลขจะนวนนับของแพ็กเก็ตสำหรับโหนดแต่ละโหนดที่เพิ่มค่าทีละ “1” เมื่อโหนดนั้นๆ ได้กระจายแพ็กเก็ตนี้ไปยังโหนดอื่น ทั้งที่อยู่ผู้ส่ง (Source Address) และ Request ID จะถูกนำมาใช้เป็นตัวบ่งชี้ Route Request แต่ละตัวออกจากกันทำให้สามารถกำจัดแพ็กเก็ตที่ถูกสร้างขึ้นมาซ้ำซ้อนกันออกไปจากระบบเครือข่ายได้

นอกเหนือจากจำนวนนับ Request ID แล้ว แต่ละโหนดยังมีหมายเลขลำดับของตนเองที่จะถูกเพิ่มค่าขึ้นทุกครั้งที่แพ็กเก็ต Route Request ถูกส่งออกไปหรือใช้ในการตอบรับแพ็กเก็ต Route Request ของโหนดอื่นเป็นตัวนับ (Counter) ที่ทำหน้าที่คล้ายกับนาฬิกาที่ใช้บอกว่าเป็นเส้นทางใหม่หรือเก่าเขตข้อมูลที่ 4 คือ หมายเลขลำดับของเจ้าของแพ็กเก็ต (Source Sequence Number) ในที่นี้คือโหนด A และขอบเขตข้อมูลที่ 5 คือ ค่าของหมายเลขลำดับล่าสุดของโหนดผู้รับ (Destination Sequence Number) ในที่นี้คือโหนด I ที่โหนด A ทราบ (ใช้ค่า “0” ถ้าไม่ทราบค่า) การใช้งานเขตข้อมูลนี้จะกล่าวถึงในลำดับต่อไป เขตข้อมูลท้ายสุดคือ บอกจำนวนครั้งที่แพ็กเก็ตนี้ถูกส่งผ่านโหนดต่างๆ (Hop Count) ถูกกำหนดค่าเริ่มต้นเป็น “0”

เมื่อแพ็กเก็ต Route Request เดินทางมาถึงโหนด B และโหนด D จะเกิดการทำงานดังนี้

1. ข้อมูล Source Address และ Request ID จะถูกนำมาตรวจสอบในตารางข้อมูลส่วนตัวของแต่ละโหนด เพื่อตรวจดูว่าเป็นแพ็กเก็ตที่ได้รับการประมวลผลไปแล้วหรือไม่ ถ้าได้ทำไปแล้วก็จะลบแพ็กเก็ตนั้นทิ้งไป แต่ถ้าไม่ใช่ก็จะบันทึกหมายเลขนี้ไว้ในตารางข้อมูลของตนเอง และทำการประมวลผลในข้อ 2 ต่อไป

2. โหนดผู้รับจะตรวจดูหมายเลขเป้าหมายกับข้อมูลในตารางเส้นทางการเดินข้อมูลของตนเอง ถ้าเส้นทางใหม่ที่ไปยังโหนดเป้าหมายปรากฏอยู่ (ดูได้จากการส่งแพ็กเก็ตเราท์รีพลาย (Route Reply) กลับไปยังโหนดเป้าหมายที่ส่งแพ็กเก็ต Route Request ออกมาเพื่อบอกให้เส้นทางทราบว่าเส้นทางเดินของข้อมูลไปทางใด) แสดงว่าสามารถนำค่าที่เก็บอยู่นั้นไปใช้ได้ คำว่า “เส้นทางใหม่” หมายความว่า หมายเลข Destination Sequence Number ที่เก็บอยู่ในตารางเส้นทางเดินข้อมูลมีค่ามากกว่าหรือเท่ากับค่าของ Destination Sequence Number ที่เก็บอยู่ใน Route Request แต่ถ้าค่ามีน้อยกว่าแสดงว่าค่าที่เก็บอยู่นั้นเก่ากว่าจึงไม่สามารถนำไปใช้งานได้ ให้ทำในข้อ 3 ต่อไป
3. เนื่องจากผู้รับไม่ทราบเส้นทางใหม่ที่จะส่งแพ็กเก็ตไปยังเป้าหมายจึงทำการเพิ่มค่า Hop Count และทำการส่งแบบกระจาย (Broadcast) แพ็กเก็ต Route Request นั้นไปยังโหนดข้างเคียงทั้งหมด รวมทั้งดึงข้อมูลจากแพ็กเก็ตเข้าไปเก็บไว้ในตารางข้อมูลของตนเอง (Reverse Route Table) ข้อมูลนี้จะถูกนำไปใช้ในการสร้างรีเวอร์สเราท์ (Reverse Route) เพื่อให้แพ็กเก็ตตอบรับ (Reply Request) สามารถเดินทางมายังโหนดต้นตอของ Route Request ได้ เส้นทางลูกศรที่ปรากฏอยู่ในรูปที่ 13 คือข้อมูลที่ใช้ในการสร้าง Reverse Route สำหรับข้อมูลใหม่ที่ใส่ไว้ในตาราง Reverse Route Table ทุกตัวจะถูกบันทึกไว้เพื่อใช้ในการตรวจสอบว่าถ้าข้อมูลนั้นถูกบันทึกนานเกินไปโดยไม่มีการทำ Reverse Route ก็จะถูกลบทิ้งไป

สมมุติว่าทั้งโหนด B และโหนด D ไม่ทราบว่าโหนด I อยู่ที่ใด (ข้อมูลที่เก็บอยู่นั้นเก่ากว่าหรือไม่มีอยู่เลย) ดังนั้นโหนดทั้ง 2 จึงสร้างข้อมูลขึ้นในตาราง Reverse Route Table ที่ชี้กลับไปยังโหนด A ดังที่แสดงด้วยลูกศรในรูปที่12 แล้วเพิ่มค่า Hop Count เป็น “1” พร้อมกับส่งแบบกระจายแพ็กเก็ตนั้นออกไปยังโหนดข้างเคียงทุกโหนดของตนเอง แพ็กเก็ตจากโหนด B จะถูกส่งกลับไปยังโหนด C และโหนด D โหนด C จะสร้างข้อมูลขึ้นในตาราง Reverse Route Table เพิ่มค่า Hop Count แล้วส่งแบบกระจายแพ็กเก็ตนั้นออกไป ในทางกลับกัน ที่โหนด D จะพบว่าแพ็กเก็ตนี้เป็นแพ็กเก็ตที่ซ้ำซ้อนและจัดการลบทิ้งไป

ในทำนองเดียวกันแพ็กเก็ตที่ส่งจากโหนด D ไปยังโหนด B จะถูกลบทิ้งแต่จะได้รับการประมวลผลต่อไปที่โหนด F และG ดังที่แสดงในรูปที่ 13(c) ภายหลังจากที่โหนด E, H, และ I ได้รับข้อมูล แพ็กเก็ต Route Request ก็ได้เดินทางมาถึงเป้าหมาย ดังที่แสดงในรูปที่ 13(d) สังเกตว่า การอธิบายนี้ได้เรียงลำดับเหตุการณ์เป็น 3 ขั้นตอน แต่ในข้อเท็จจริงนั้นเหตุการณ์ทั้งหมดที่เกิดขึ้นจะเป็นอิสระต่อกันและกัน คือไม่ได้เกิดขึ้นเรียงตามลำดับแต่อย่างใด

โหนด I จะตอบสนองแพ็กเก็ต Route Request โดยการสร้างแพ็กเก็ต Route Reply ขึ้นมา ดังมีโครงสร้างในรูปที่ 15 เขตข้อมูลที่อยู่ผู้ส่ง (Source Address),ที่อยู่ผู้รับ (Destination Address) และจำนวนครั้งที่แพ็กเก็ตถูกส่งผ่านโหนด (Hop Count) จะถูกทำสำเนาข้อมูลมาจากแพ็กเก็ต Route Request ค่าของ Destination Sequence Number จะถูกทำสำเนามาจากตัวนับ (Counter) ของโหนด I เขตข้อมูล Hop Count ถูกกำหนดให้เป็น “0” เขตข้อมูลไลฟ์ทาม (Lifetime) เป็นตัวกำหนดอายุของเส้นทางนี้ จากนั้นแพ็กเก็ตจะถูกส่งกลับไปตามเส้นทางที่กำหนดไว้ใน Reverse Route Table ของแต่ละโหนด ในแต่ละโหนดที่ส่งต่อแพ็กเก็ตจะทำการเพิ่มค่าให้แก่ Hop Count ทำให้โหนดต้นทางคือโหนด A ทราบว่าโหนด I อยู่ห่างออกไปเท่าใด

แต่ละโหนดที่อยู่ระหว่างทางจากโหนด I กลับไปยังโหนด A จะทำการตรวจสอบข้อมูลในแพ็กเก็ต Route Reply ข้อมูลจะถูกส่งเข้าไปในตารางเส้นทางเดินข้อมูลของแต่ละโหนดถ้าเป็นไปตามเงื่อนไขข้อใดข้อหนึ่งต่อไปนี้

1. ไม่มีข้อมูลของโหนด I อยู่ในตาราง
2. หมายเลข Sequence Number สำหรับโหนด I ในแพ็กเก็ต Route Reply นั้นมีค่ามากกว่าค่าที่เก็บไว้ในตารางข้อมูล
3. หมายเลข Sequence Number นั้นมีค่าเท่ากันแต่เส้นทางใหม่นั้นสั้นกว่า (ดูจาก Hop Count)

ด้วยวิธีการนี้ทุกโหนดที่อยู่ในเส้นทาง Reverse Route จะสามารถเรียนรู้ข้อมูลใหม่ได้อัตโนมัติ ส่วนโหนดที่ได้รับแพ็กเก็ต Route Request แต่ไม่อยู่ในเส้นทาง Reverse Route (ในที่นี้คือโหนด B, C, E, F, และ H) จะมาปรับข้อมูลในตารางจะลบทิ้งข้อมูลนี้ไปเมื่อหมดระยะเวลารอคอยที่กำหนดไว้ล่วงหน้า

ในระบบเครือข่ายขนาดใหญ่ อัลกอลิทึมนี้จะสร้างสำเนาแพ็กเก็ต Route Request ขึ้นมามากมายแม้ว่าโหนดเป้าหมายจะอยู่ใกล้ๆ ก็ตาม

จำนวนสำเนาแพ็กเก็ตสามารถลดลงได้ด้วยวิธีการดังนี้ เขตข้อมูลทามทูลิฟ (Time To Live) นี้จะถูกลดลง “1” ทุกครั้งที่สร้างสำเนา และส่งต่อไปยังโหนดอื่น เมื่อโหนดรับเห็นว่าค่านี้เป็น “0” ก็จะไม่ส่งต่อไปและกำจัดแพ็กเก็ตนั้นทิ้ง

กระบวนการค้นหาโหนดเป้าหมายจึงถูกดัดแปลงเป็นดังนี้ ครั้งแรกจะกำหนดค่า Time To Live ไว้เป็น “1” และส่งแพ็กเก็ต Route Request ออกไป (ซึ่งจะไปถึงโหนดข้างเคียงเท่านั้น) ถ้าไม่พบเป้าหมายก็จะเปลี่ยนค่าทามทูไลฟ์ (Time To Life) เป็น “2” แล้วส่งออกไปใหม่ ถ้ายังไม่พบก็จะค่อยๆ เพิ่มค่า Time To Live ไปเรื่อยๆ จนกว่าจะพบโหนดเป้าหมาย หรือถึงขีดจำกัดอันหนึ่ง ก็จะสรุปว่า ไม่สามารถหาโหนดเป้าหมายได้

รูปที่ 15 โครงสร้างของแพ็กเก็ตเราท์รีพลาย (Route Reply)

การบำรุงรักษาเส้นทางเดินข้อมูล

เนื่องจากโหนดอาจจะเคลื่อนที่หรือถูกปิดลงเมื่อใดก็ได้จึงทำให้รูปทรงของระบบเครือข่ายนั้นเปลี่ยนแปลงอยู่เสมอ จากรูปที่ 13 ถ้าโหนด G ถูกปิด โหนด A จะไม่ทราบว่าเส้นทางไปยังโหนด I (ADGI) นั้นไม่สามารถใช้งานได้แล้ว อัลกอลิทึมนำมาใช้จึงต้องมีขีดความสามารถในกาแก้ปัญหานี้ได้ โหนดต่างๆ จะทำการส่งแบบกระจายแพ็กเก็ตฮัลโลแพ็กเก็ต (Hello Packet) ไปยังโหนดข้างเคียงเสมอโดยคาดหวังว่าโหนดข้างเคียงจะส่งแพ็กเก็ตตอบรับกลับมา ถ้าไม่มีการตอบกลับมา (ภายในเวลาที่กำหนด) ผู้ที่ส่งแพ็กเก็ตไปทักทายก็จะทราบได้ว่าไม่สามารถติดต่อกับโหนดนั้นได้อีกต่อไป

ข้อมูลนี้ถูกนำไปใช้ในการลบข้อมูลออกจากตารางข้อมูลของแต่ละโหนด (สำหรับเส้นทางที่ใช้ไม่ได้อีกต่อไป) สำหรับแต่ละเป้าหมาย โหนดจะเก็บรายชื่อของแต่ละโหนดข้างเคียงที่ได้ส่งข้อมูลบอกว่าสามารถไปยังเป้าหมายนั้นๆ ได้ภายในระยะเวลา T วินาทีซึ่งเรียกว่า โหนดข้างเคียงที่ยังทำงานอยู่ (Active Neighbors) ของโหนดเป้าหมายนั้น แต่ละโหนดจะมีตารางเส้นทางเดินข้อมูลไปยังทุกเป้าหมาย (Destination) ที่รู้จักพร้อมกับบอกชื่อโหนดที่ต้องส่งแพ็กเก็ตออกไป (Out Going Node Or Next Hop)

เพื่อไปยังเป้าหมายนั้น รวมทั้งจำนวนโหนดที่ต้องเดินทาง (Hop Count) หมายเลข Destination Sequence Number ล่าสุดและรายชื่อโหนดข้างเคียงที่ยังทำงานอยู่ของโหนดเป้าหมายนั้นๆรูปที่ 16(a) แสดงตารางเส้นทางเดินข้อมูลสำหรับโหนด D

เมื่อโหนดข้างเคียงของโหนด N (โหนดใดๆ) เกิดไม่ทำงาน โหนด N จะตรวจสอบตารางเส้นทางเดินข้อมูลเพื่อดูว่าเป้าหมายใดบ้างที่ใช้เส้นทางผ่านโหนดที่ไม่ได้ทำงานนั้น สำหรับแต่ละโหนดข้างเคียงที่ยังทำงานอยู่บนเส้นทางเหล่านี้จะได้รับแจ้งให้ทราบว่าเส้นทางผ่านโหนด N ไปยังเป้าหมายนั้นไม่สามารถใช้งานได้อีกต่อไป และจะต้องลบออกไปจากตารางเส้นทางเดินข้อมูล โหนดข้างเคียงที่ยังทำงานอยู่ก็จะทำงานในลักษณะเดียวกันนี้ต่อๆ กันไปจนกระทั่งโหนดทุกโหนดในระบบเครือข่ายที่เกี่ยวข้องกับโหนดที่หยุดทำงานได้รับทราบข้อมูลนี้ทั้งหมดแล้ว

ตัวอย่าง การบำรุงรักษาตารางเส้นทางเดินข้อมูล ขอให้พิจารณาในตัวอย่างที่แล้ว แต่ในขณะนี้โหนด G หยุดทำงาน รูปทรงของระบบเครือข่ายได้เปลี่ยนไปดังแสดงในรูปที่ 16(b) เมื่อโหนด D พบว่าโหนด G หยุดทำงาน โหนด D จะตรวจข้อมูลในตารางเส้นทางเดินข้อมูลของตนเองและพบว่าโหนด G ถูกนำไปใช้ในเส้นทางไปยังโหนด E, G, และ I โหนดข้างเคียงที่ยังทำงานอยู่ของทุกเส้นทางรวมกันคือ {A, B}

Dest.	Next hop	Distance	Active neighbors	Other fields
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

(a)

(b)

รูปที่ 16 (a) ตารางเส้นทางเดินข้อมูล
(b) รูปกราฟของระบบเครือข่าย

ซึ่งหมายความว่าโหนด A และโหนด B นั้นขึ้นอยู่กับโหนด G เพราะจะต้องใช้เป็นเส้นทางผ่านไปยังโหนดอื่นๆ ดังนั้นจึงจะต้องแจ้งให้ทราบว่าโหนด G นั้นหยุดทำงานแล้ว (ทำให้เส้นทางเหล่านั้นก็ใช้งานไม่ได้) โหนด D จึงส่งข่าวสารไปบอกโหนด A และโหนด B ซึ่งทั้งสองโหนดก็จะปรับปรุงข้อมูลในตารางเส้นทางเดินข้อมูลของตนเองในทำนองเดียวกันนี้ และโหนด D ก็จะลบข้อมูลเส้นทางไปยังโหนด E, G, และ I ทิ้งไป

บทที่ 4
การจำลองการเคลื่อนที่ของโหนดในโครงข่ายเคลื่อนที่แบบแอดฮอคด้วยภาพกราฟฟิก
(GUI for studying random way point mobility in MANETs)

4.1 การทำงานของโปรแกรม

จาก Source code ได้อธิบายการทำงานของโปรแกรมได้อธิบายการทำงานของโปรแกรมด้วยการแสดงคอมเมนต์ (ที่มีเครื่องหมาย // หรือ /*…..*/)และแบ่งอธิบายการทำงานออกเป็นส่วนๆ ดังต่อไปนี้

Source Code

1. ประกาศค่าคงที่และเรียกใช้ไลเบอร์รี่ต่างๆ

```
#include <graphics.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>

#define MAX_NODES 20
#define INFINITY 1000000
#define LEFT 0
#define TOP 0
#define RIGHT 800
#define BOTTOM 600
#define RADIUS 120

double randunif01;
```

2. สร้างคลาสของตัวแปรที่ใช้ในการระบุตำแหน่งและการเคลื่อนที่ของโหนด


```
class my_circle{
    private:
        int x,y,dx,dy;
    public:
        void setX(int setx){
            x = setx;
        }
        void setY(int sety){
            y = sety;
```

```
        }
        void setDx(int setdx){
            dx = setdx;
        }
        void setDy(int setdy){
            dy = setdy;
        }
        int getX(){
           return x;
        }
        int getY(){
           return y;
        }
        int getDx(){
           return dx;
        }
        int getDy(){
           return dy;
        }
};
```

3. ประกาศฟังก์ชันและตัวแปรต่างๆที่จำเป็นต้องใช้ในโปรแกรม

```
double Randunif01();
 void initcir(my_circle[],int);          /* initial position of circle*/
 void initdist(int);                     /*  loop initial of circle topology (=0) */
 void drawcircle(int,int);               /* Draw circle */
 void delcircle(int,int);                /* Delete circle  */
```

```
void delpixel(int,int);                 /* Delete center point */
void delline(int,int,int,int);          /* Delete Line  */
int lengthline(int,int,int,int);        /* Check line between circles  */
void showcircle(my_circle[],int);       /* show all circle and set cost to topology=0 */
void hidecircle(my_circle[],int);       /* Delete circle*/
void movecircle(my_circle[],int);       /* Bounching of circles */
void animation(my_circle[],int,int);    /* Moving of circles */
void shotparh(my_circle[],int);         /* Shortest path Dijkstra algorlithm */
void graph(my_circle[],int,int[],int);  /* Draw line that out of short path */
int dist[MAX_NODES][MAX_NODES];         //matrix of circle topology
```

4. ส่วนหลักของโปรแกรม

```
Int main(){
    my_circle cir[MAX_NODES];
    int input=0 ;
    printf("Enter number of circle: ");   // input number of circle
    scanf("%d",&input);
    int seed=0;                           // input seed of random
    printf("Enter seed of random: ");
    scanf("%d",&seed);
    srand(seed);                          // seed of random
    int delayt=0;                         // input delay
```

```
    printf("Enter time delay (0-250): ");
    scanf("%d",&delayt);
    initcir(cir,input);                  // call function initcir
    initdist(input);                     // call function initdist
    int i,j;
    for(i=0;i<input;i++){                // show initial number of circle
      for(j=0;j<input;j++){
         printf("%5d",dist[i][j]);
      }
      printf("\n");
    }
    initwindow(RIGHT,BOTTOM);            // draw window
    animation(cir,input,delayt);         // call animation function
    getch();
    closegraph();
    return 0;                            // End
}
```

5. ฟังก์ชันสุ่ม

```
double Randunif01()
  {   randunif01 = ((double) rand())/RAND_MAX;
   return( randunif01);
   }
```

6. ฟังก์ชันสร้างตำแหน่งเริ่มต้นของโหนดแบบสุ่ม

```
void initcir(my_circle cir[],int input){
     int x[20];
     int y[20];
     int i=0,j=0;
     for(i=0;i<input;i++)
     {
      x[i] = int (Randunif01()*(800));
      y[i] = int (Randunif01()*(600));
      printf("%d\t",x[i]);
        }

    int dx[]={1,-1,-1,1,1,-1,-1,-1,-1,-1
            ,1,-1,1,-1,-1,-1,1,-1,-1,1};          // direction of circle x-axis
    int dy[]={1,-1,1,-1,-1,-1,1,1,-1,-1,1
            ,1,-1,-1,1,1,-1,-1,-1,-1,-1};         // direction of circle y-axis
     i=0,j=0;
    while(i<=(input-1)){                          // input to class my_circle
        cir[i].setX(x[i]);
        cir[i].setY(y[i]);
        cir[i].setDx(dx[i]);
        cir[i].setDy(dy[i]);
        i++;
     }
  }
```

7. ฟังก์ชันกำหนดค่าเริ่มต้นของเมทริกซ์ของรูปทรง (Topology)ให้เท่ากับศูนย์

```
void initdist(int input){
    int i,j;
    for(i=0;i<input;i++){
      for(j=0;j<input;j++){
         dist[i][j] = 0;
      }
    }
 }
```

8. ฟังก์ชันวาดโหนด

```
Void drawcircle(int x,int y,int radius){
        circle(x,y,RADIUS);
    }
```

9. ฟังก์ชันลบโหนด

```
Void delcircle(int x,int y,int radius){
        setcolor(BLACK);
        circle(x,y,RADIUS);   }
```

10. ลบจุดศูนย์กลางของโหนด

```
Void delpixel(int x,int y){
        putpixel(x,y,BLACK);
    }
```

11. ฟังก์ชันลบเส้น

```
Void delline(int x1,int y1,int x2,int y2){
        setcolor(BLACK);
        line(x1,y1,x2,y2);
    }
```

12. ฟังก์ชันตรวจสอบว่าโหนดได้ซ้อนทับกันอยู่หรือไม่

```
int lengthline(int x1,int y1,int x2,int y2){
        int x=0;
        int y=0;
        int length=0;

        x = x2-x1;
        y = y2-y1;
        length = sqrt((x*x)+(y*y));
        return length;
}
```

13. แสดงโหนดและกำหนดให้ค่าใช้จ่ายของเส้นทางให้เท่ากับศูนย์

```
Void showcircle(my_circle cir[],int input){
    int k=0,length=0;
    int count=1,l=0,m=0;
    while(k<=(input-1)){
        if(k==0 || k==(input-1)) setcolor(YELLOW);
        else setcolor(WHITE);
```

```
            drawcircle(cir[k].getX(),cir[k].getY(),RADIUS);
            putpixel(cir[k].getX(),cir[k].getY(),YELLOW);
            count=1;l=0;
            m=count;
            while(m<input){
               length = lengthline(cir[l].getX(),cir[l].getY(),cir[m].getX(),cir[m].getY());

               if(length <= RADIUS*2){

                  dist[l][m] = length;
                  dist[m][l] = length;

               }else{
                  dist[l][m] = 0;
                  dist[m][l] = 0;
               }
               if(m==(input-1)){
                  m=++count;
                  l++;
               }
               else m++;
            }
            k++;
      }
   }
```

14. ฟังก์ชันส่วนของการเคลื่อนที่ของโหนดเมื่อโหนดวิ่งไปชนขอบ

```
Void movecircle(my_circle cir[],int i){
      int nx,ny;
      if(cir[i].getX()>=RIGHT || cir[i].getX()<=LEFT) cir[i].setDx(-cir[i].getDx());
      if(cir[i].getY()>=BOTTOM || cir[i].getY()<=TOP) cir[i].setDy(-cir[i].getDy());
      nx = cir[i].getX()+cir[i].getDx();
      ny = cir[i].getY()+cir[i].getDy();
      cir[i].setX(nx);
      cir[i].setY(ny);
  }
```

15. ฟังก์ชันส่วนการเคลื่อนที่ของวงกลม

```
void animation(my_circle cir[],int input,int delayt){
      int i=0;
      while(i<input){

        showcircle(cir,input);



        delay(delayt);                  //delay of circle moving
        if(kbhit()) {                   //stop moving
          int i,j;
          for(i=0;i<input;i++){
            for(j=0;j<input;j++){
              printf("%10d",dist[i][j]);
            }
            printf("\n");
          }
          printf("\n");
```

```
            shotparh(cir,input);
            continue;
        }
        hidecircle(cir,input);
        movecircle(cir,i);

        i++;
        if(i>(input-1)) i=0;
    }
}
```

16. ฟังก์ชันส่วนการหาเส้นทางที่สั้นที่สุดโดยไดจค์สตราอัลกอลิทึม (Dijkstra's Algorithm)

```
                                    void shotparh(my_circle cir[],int input){
    struct state{                   // creat structer
     int predecessor;               // Before node
     int length;                    // intial length = infinity
     int label;                     // check algorlithm
}state[MAX_NODES];

int n;
n=input;

int i,k,min,t,s=0,path[100];        // min = shortpath
struct state *p;                    // pointer to structer state
t=n-1;
for(p=&state[0];p<&state[n];p++){
      p->predecessor = -1;
      p->length = INFINITY;
```

```
      p->label = 1;
}
state[t].length=0;
state[t].label = 0;
k=t;
do{
   for(i=0;i<n;i++){
      if(dist[k][i]!=0 && state[i].label==1){
         if(state[k].length + dist[k][i] < state[i].length){
            state[i].predecessor = k;
            state[i].length = state[k].length + dist[k][i];
         }
      }
   }

   k=0;min = INFINITY;
   for(i=0;i<n;i++){                        // check for finding shotpath
      if(state[i].label==1 && state[i].length<min){
         min = state[i].length;
         k = i;
      }
   }
   state[k].label = 0;                      // label = 0 not check
}while(k!=s);

printf("\n");
i=0;k=s;
do{
  path[i] = k;
```

```
    k = state[k].predecessor;
    printf("%d ",path[i]);
    i++;
  }while(k>=0);                                  // input cost of path
  printf("\n");
  int x;
  if(i > 1){
  for(x=0;x<i;x++){
   if((x+1)<i){
     setcolor(YELLOW);
     line(cir[path[x]].getX(),cir[path[x]].getY(),cir[path[x+1]].getX(),cir[path[x+1]].getY());
     dist[path[x]][path[x+1]] = 0;
     dist[path[x+1]][path[x]] = 0;
   }
  }
  }
  graph(cir,input,path,i);
 }
```

17. ฟังก์ชันแสดงเส้นทางที่นอกเหนือจากเส้นทางที่สั้นที่สุด

```
void graph(my_circle cir[],int input,int path[],int pathsize){
     int i,j,x;
    for(i=0;i<input;i++){
       for(j=0;j<input;j++){
         if(dist[i][j]!=0){
            setcolor(WHITE);
```

```
line(cir[i].getX(),cir[i].getY(),cir[j].getX(),cir[j].getY()); }}}}
```


บทที่ 5
การทดสอบหาค่าพารามิเตอร์ที่ส่งผลต่อการหาเส้นทางในโครงข่าย

กล่าวนำ

จากการศึกษาโปรแกรมแบบจำลองการเคลื่อนที่ของโหนดในโครงข่ายเคลื่อนที่แบบแอดฮอคด้วยภาพกราฟฟิก (GUI for studying random way point mobility in MANETs) ที่สร้างขึ้นมานี้ทำให้ทราบว่าในขั้นตอนการค้นหาเส้นทางของเครือข่ายนั้น

มีปัจจัยหลายอย่างที่ส่งผลต่อการหาเส้นทางของเครือข่าย
จึงได้ทำการทดลองเพื่อทดสอบว่าปัจจัยเหล่านี้จะส่งผลต่อการหาเส้นทางของเครือข่ายมากน้อยเพียงใด ตามที่ได้ตั้งสมมติฐานเอาไว้

1. ค่าพารามิเตอร์ที่ต้องการทดสอบมีทั้งหมด 4 ประเภทคือ

- รัศมีของโหนด (Radius)
- ความหน่วงในการเคลื่อนที่ของโหนด (Delay)
- จำนวนโหนดต่อหนึ่งหน่วยพื้นที่ (Density)
- ช่วงเวลาการเฝ้าสังเกตการณ์ (Observation Time)

โดยกำหนดค่าพารามิเตอร์คงที่ดังนี้

- ขนาดของพื้นที่ทั้งหมดเท่ากับ 800x600 pixels (กำหนดให้ 1 pixel : 1 เมตร)

2. ในการวัดหาค่าพารามิเตอร์ว่ามีผลต่อการหาเส้นทางหรือไม่จะทดสอบโดยการหาค่า

- ค่าเฉลี่ยของจำนวนเส้นทางที่หาได้ (Average Path Cost)
- อัตราความสำเร็จในการหาเส้นทาง (Success Ratio)

3. การทดลอง

3.1 รัศมีของโหนด (Radius)

- สมมติฐาน

การเพิ่มขึ้นของรัศมีของโหนดในพื้นที่เท่าเดิม
จะทำให้การเชื่อมต่อกันของแต่ละโหนดมีโอกาสมากขึ้น ส่งผลให้ค่าเฉลี่ยของจำนวนเส้นทางที่หาได้ (Average Path Cost) และอัตราความสำเร็จในการหาเส้นทาง (Success Ratio) มีค่าเพิ่มขึ้น

- ตัวแปรต้น คือ รัศมีของโหนด (Radius) หน่วย pixel โดยเปลี่ยนค่า 50,60,70,…,140 pixels

- ตัวแปรต้นที่กำหนดค่าคงที่ ได้แก่

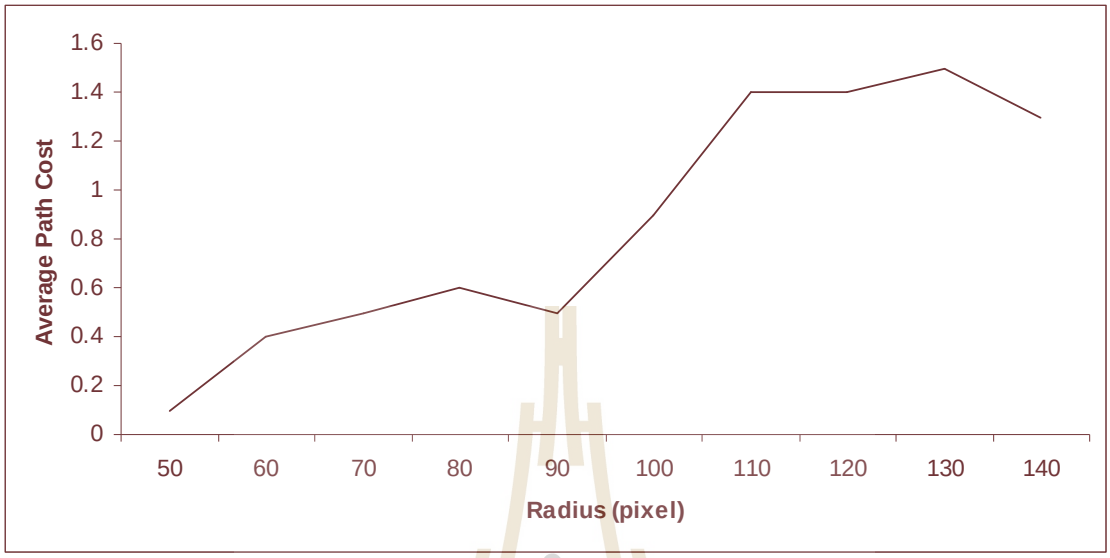
ความหน่วงในการเคลื่อนที่ของโหนด (Delay) = 0 msec
จำนวนโหนดต่อหนึ่งหน่วยพื้นที่ (Density) = 10 nodes
ขนาดของพื้นที่ทั้งหมด = 800 x 600 pixels
ช่วงเวลาการเฝ้าสังเกตการณ์ (Time) = 20 sec

- ผลการทดลอง

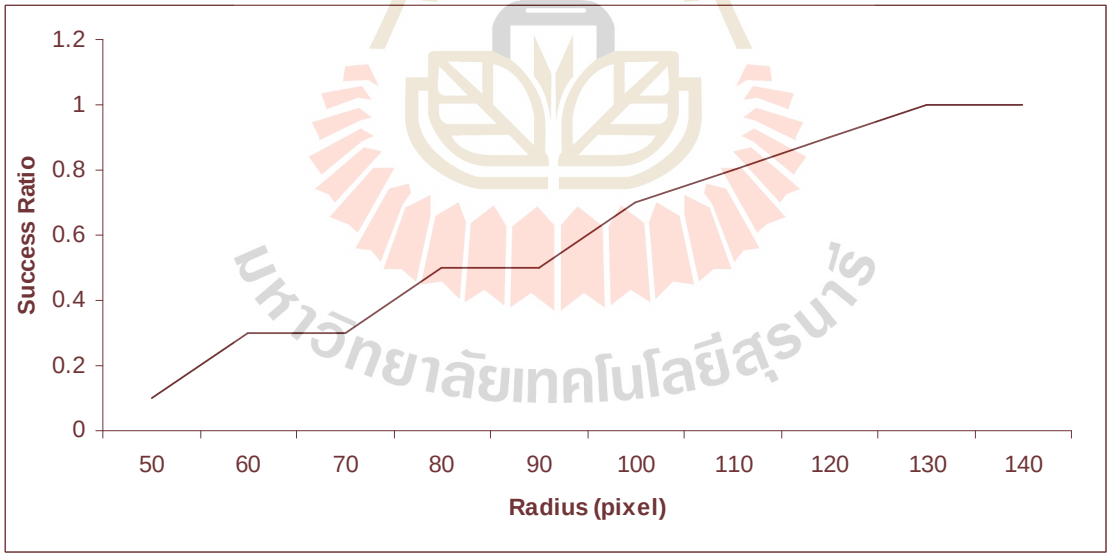
ตารางแสดงความสัมพันธ์ระหว่าง Radius, Average Path Cost และSuccess Ratio

Radius (pixel)	50	60	70	80	90	100	110	120	130	140
Average Path Cost	1	4	5	6	5	9	14	14	15	13
	0.1	0.4	0.5	0.6	0.5	0.9	1.4	1.4	1.5	1.3
Success Ratio	1	3	3	5	5	7	8	9	10	10
	0.1	0.3	0.3	0.5	0.5	0.7	0.8	0.9	1	1

กราฟแสดงความสัมพันธ์ระหว่าง Radius และ Average Path Cost



กราฟแสดงความสัมพันธ์ระหว่าง Radius และ Success Ratio



จากกราฟสรุปได้ว่า

เมื่อโหนดมีรัศมีการส่งมากขึ้นทำให้เกิดการแผ่ของสัญญาณกว้างขึ้น ซึ่งเป็นผลให้มีโอกาสในการหาการเชื่อมต่อของโหนดได้ง่ายขึ้น การเพิ่มรัศมีในพื้นที่เท่าเดิมนั้นจะมีค่าเท่ากับการคงรัศมีแต่ลดพื้นที่ ซึ่งมีผลให้พื้นที่ว่างนอกเครือข่ายน้อยลง (ความหนาแน่นมากขึ้น)

3.2 ความหน่วงในการเคลื่อนที่ของโหนด (Delay)

- สมมติฐาน

เมื่อโหนดมีการเคลื่อนที่ช้าลง(เพิ่ม delay มากขึ้น) จะทำให้ลักษณะทางกายภาพ (topology) ของโครงข่ายเกิดการเปลี่ยนแปลงน้อยส่งผลให้ค่าเฉลี่ยของจำนวนเส้นทางที่หาได้ (Average Path Cost) และอัตราความสำเร็จในการหาเส้นทาง (Success Ratio) มีค่าคงที่

- ตัวแปรต้น คือ ความหน่วงในการเคลื่อนที่ของโหนด (Delay) หน่วย msec โดยเปลี่ยนค่า 0,10,20,30,…,200 msec

- ตัวแปรต้นที่กำหนดค่าคงที่ ได้แก่

รัศมีของโหนด (Radius) = 120 pixels

จำนวนโหนดต่อหนึ่งหน่วยพื้นที่ (Density) = 10 nodes

ขนาดของพื้นที่ทั้งหมด = 800x600 pixels

ช่วงเวลาการเฝ้าสังเกตการณ์ (Time) = 20 sec

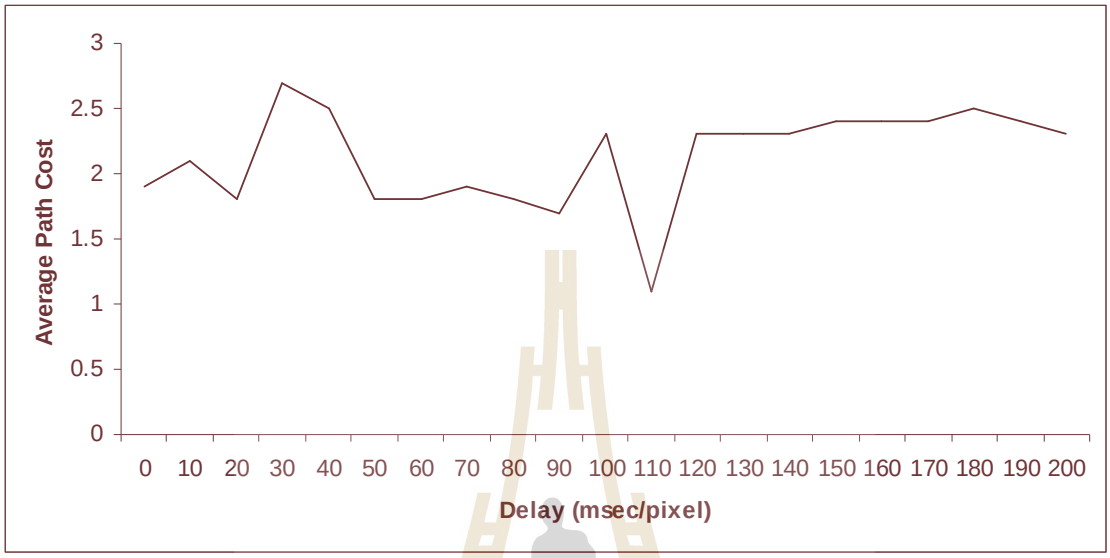
- ผลการทดลอง

ตารางแสดงความสัมพันธ์ระหว่าง Delay, Average Path Cost และ Success Ratio

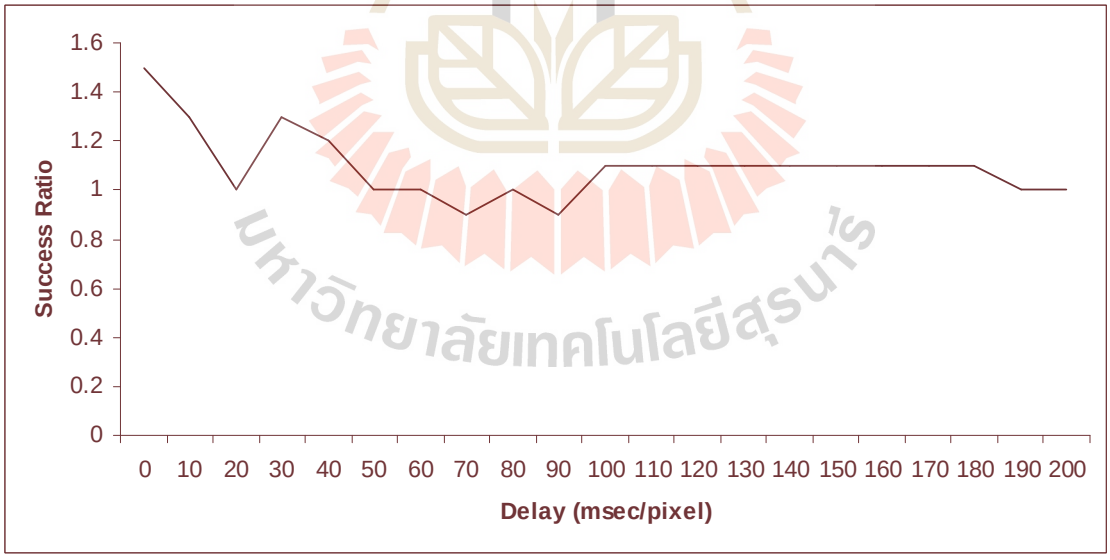
Delay (msec/pixel)	0	10	20	30	40	50	60	70	80	90
Average Path Cost	19	21	18	27	25	18	18	19	18	17
	1.9	2.1	1.8	2.7	2.5	1.8	1.8	1.9	1.8	1.7
Success Ratio	15	13	10	13	12	10	10	9	10	9
	1.5	1.3	1	1.3	1.2	1	1	0.9	1	0.9

Delay (msec/pixel)	100	110	120	130	140	150	160	170	180	190	200
Average Path Cost	23	11	23	23	23	24	24	24	25	24	23
	2.3	1.1	2.3	2.3	2.3	2.4	2.4	2.4	2.5	2.4	2.3
Success Ratio	11	11	11	11	11	11	11	11	11	10	10
	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1.1	1	1

กราฟแสดงความสัมพันธ์ระหว่าง Delay และAverage Path Cost



กราฟแสดงความสัมพันธ์ระหว่าง Delay และSuccess Ratio



จากกราฟสรุปได้ว่า

เมื่อเพิ่ม Delay หรือลดเวลาในการเคลื่อนที่ของโหนดจะส่งผลต่อการเปลี่ยนแปลง topology ของโครงข่ายน้อยมาก นั่นก็คือ ผลของการค้นหาเส้นทางทั้งค่าเฉลี่ยของจำนวนเส้นทางที่หาได้ (Average Path Cost) และอัตราความสำเร็จในการหาเส้นทาง (Success Ratio) มีค่าคงที่

3.3 จำนวนโหนดต่อหนึ่งหน่วยพื้นที่ (Density)

- สมมติฐาน

การเพิ่มขึ้นของจำนวนโหนด จะส่งผลให้ในการเชื่อมต่อในโครงข่ายมีจำนวนโหนดมากขึ้น ซึ่งส่งผลให้ค่าเฉลี่ยของจำนวนเส้นทางที่หาได้ (Average Path Cost) และอัตราความสำเร็จในการหาเส้นทาง (Success Ratio) มีค่าเพิ่มขึ้น

- ตัวแปรต้น คือ จำนวนโหนดต่อหนึ่งหน่วยพื้นที่ (Density) หน่วย node โดยเปลี่ยนค่า 5,6,7,…,15 nodes

- ตัวแปรต้นที่กำหนดค่าคงที่ ได้แก่

รัศมีของโหนด (Radius) = 120 pixels

ความหน่วงในการเคลื่อนที่ของโหนด (Delay) = 0 msec

ขนาดของพื้นที่ทั้งหมด = 800x600 pixels

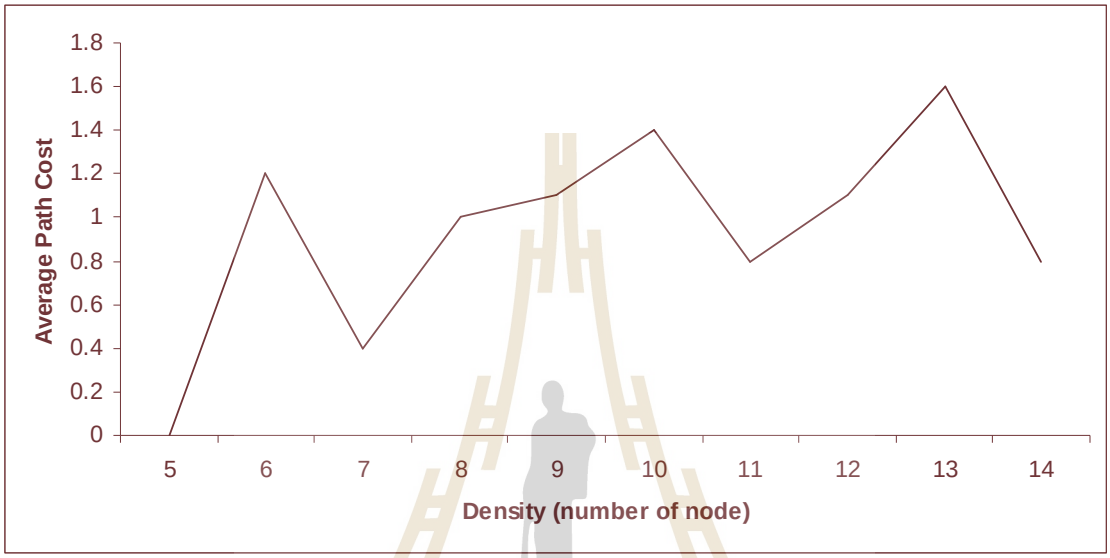
ช่วงเวลาการเฝ้าสังเกตการณ์ (Time) = 20 sec

- ผลการทดลอง

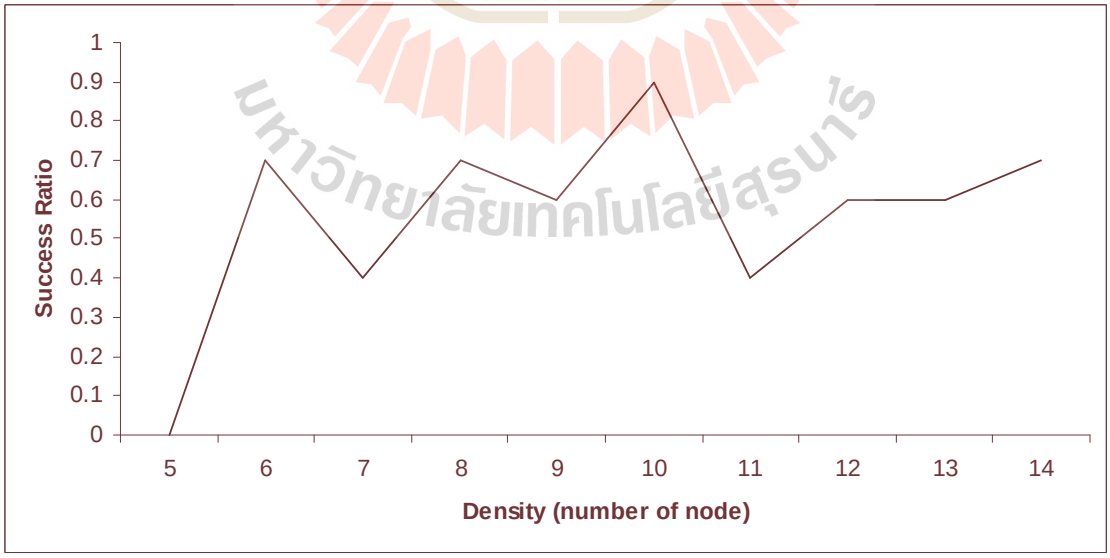
ตารางแสดงความสัมพันธ์ระหว่าง Density, Average Path Cost และ Success Ratio

Density(number of node)	5	6	7	8	9	10	11	12	13	14
Average Path Cost	0	12	4	10	11	14	8	11	16	8
	0	1.2	0.4	1	1.1	1.4	0.8	1.1	1.6	0.8
Success Ratio	0	7	4	7	6	9	4	6	6	7
	0	0.7	0.4	0.7	0.6	0.9	0.4	0.6	0.6	0.7

กราฟแสดงความสัมพันธ์ระหว่าง DensityและAverage Path Cost



กราฟแสดงความสัมพันธ์ระหว่าง Density และSuccesss Ratio



จากกราฟสรุปได้ว่า

เมื่อมีจำนวนโหนดเพิ่มขึ้นจะทำให้โครงข่ายมีความหนาแน่นมากขึ้น ดังนั้นโอกาสในการเกิดเส้นทางจึงมีมากขึ้น เป็นผลให้ค่าเฉลี่ยของจำนวนเส้นทางที่หาได้ (Average Path Cost) และอัตราความสำเร็จในการหาเส้นทาง (Success Ratio) มีค่าเพิ่มขึ้น

3.4 ช่วงเวลาการเฝ้าสังเกตการณ์ (Observation Time)

- สมมติฐาน

เมื่อเพิ่มเวลาในการสังเกตการณ์ จะไม่ส่งผลต่อค่าเฉลี่ยของจำนวนเส้นทางที่หาได้ (Average Path Cost) และอัตราความสำเร็จในการหาเส้นทาง (Success Ratio) นั่นคือ การเชื่อมต่อของโหนดไม่ขึ้นอยู่กับเวลาสังเกตการณ์ที่เพิ่มขึ้น

- ตัวแปรต้น คือ ช่วงเวลาการเฝ้าสังเกตการณ์ (Observation Time) หน่วย sec โดยเปลี่ยนค่า 10,20,30,…,200 secs

- ตัวแปรต้นที่กำหนดค่าคงที่ ได้แก่

รัศมีของโหนด (Radius) = 120 pixels

ความหน่วงในการเคลื่อนที่ของโหนด (Delay) = 0 msec

ขนาดของพื้นที่ทั้งหมด = 800x600 pixels

จำนวนโหนดต่อหนึ่งหน่วยพื้นที่ (Density) = 10 nodes

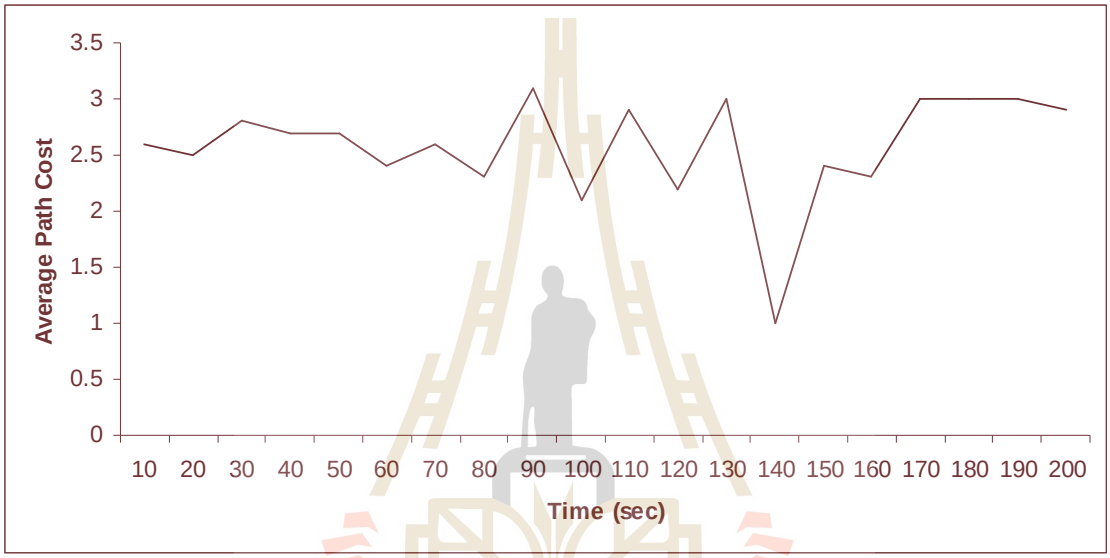
- ผลการทดลอง

ตารางแสดงความสัมพันธ์ระหว่าง Observation Time, Average Path Cost และSuccess Ratio

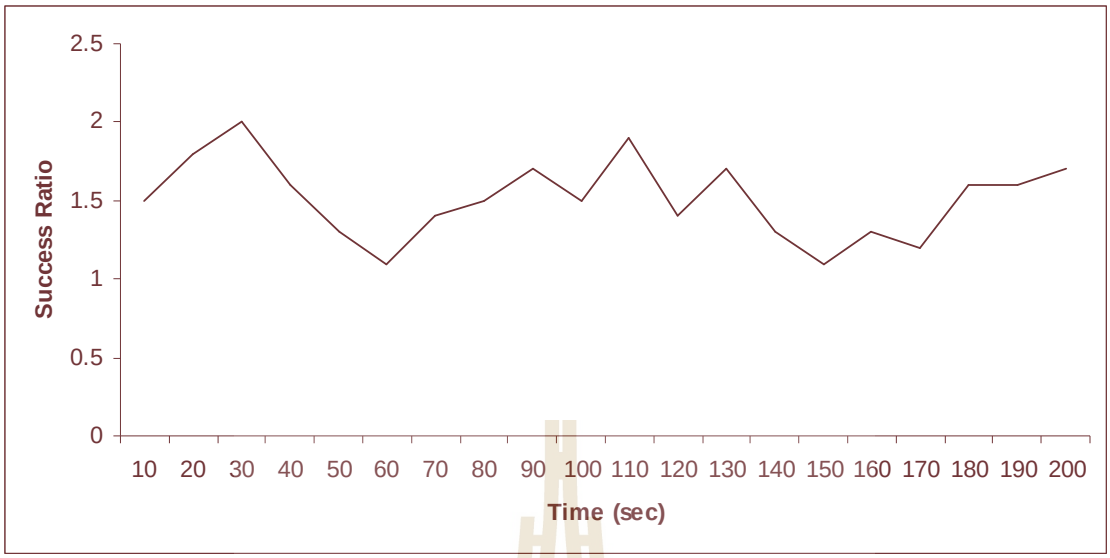
Observation Time (sec)	10	20	30	40	50	60	70	80	90	100
Average Path Cost	26	25	28	27	27	24	26	23	31	21
	2.6	2.5	2.8	2.7	2.7	2.4	2.6	2.3	3.1	2.1
Success Ratio	15	18	20	16	13	11	14	15	17	15
	1.5	1.8	2	1.6	1.3	1.1	1.4	1.5	1.7	1.5

Observation Time (sec)	110	120	130	140	150	160	170	180	190	200
Average Path Cost	29	22	30	41	24	23	30	30	30	29
	2.9	2.2	3	1	2.4	2.3	3	3	3	2.9
Success Ratio	19	14	17	13	11	13	12	16	16	17
	1.9	1.4	1.7	1.3	1.1	1.3	1.2	1.6	1.6	1.7

กราฟแสดงความสัมพันธ์ระหว่าง Observation Time และAverage Path Cost



กราฟแสดงความสัมพันธ์ระหว่าง Observation Time และSuccess Ratio



จากกราฟสรุปได้ว่า

เวลาที่สังเกตการณ์ไม่ส่งผลต่อการหาเส้นทาง เนื่องจากโปรแกรมได้สุ่ม topology ที่เกิดขึ้นในแต่ละครั้งดังนั้นจึงได้รูปร่าง topology ที่แตกต่างกัน เส้นทางในการเชื่อมต่อที่โปรแกรมสามารถหาได้จึงแตกต่างกัน ดังนั้นเวลาสังเกตการณ์จึงไม่ส่งผลต่อค่าเฉลี่ยของจำนวนเส้นทางที่หาได้ (Average Path Cost) และอัตราความสำเร็จในการหาเส้นทาง (Success Ratio)

บทที่ 6
สรุปและข้อเสนอแนะ

6.1 สรุป

ปัจจุบันการติดต่อสื่อสารแบบไร้สายมีบทบาทกับชีวิตประจำวันมากและมีแนวโน้มเพิ่มขึ้นมากเรื่อยๆ เทคโนโลยีไร้สายจึงถูกพัฒนาอย่างรวดเร็วเพื่อรองรับกับความต้องการของผู้บริโภค เทคโนโลยีโครงข่ายแอดฮอค (Ad Hoc Network) เป็นอีกทางเลือกหนึ่งที่น่าสนใจ เนื่องมาจากเป็นการสื่อสารไร้สายโดยที่ไม่ต้องใช้สถานีฐาน สามารถส่งข้อมูลจากต้นทางไปยังปลายทางได้ทันที โดยระยะทางมีผลต่อความเร็วในการส่งข้อมูล ดังนั้นโครงงานนี้จึงได้จัดทำขึ้นเพื่อศึกษาเส้นทางการติดต่อกันระหว่างโหนดต้นทางและโหนดปลายทางในรูปแบบของจียูไอ (Graphic User Interface :GUI) ซึ่งเป็นการแสดงการเชื่อมต่อกันด้วยเส้นทางที่สั้นที่สุด จากการทำโครงงานสามารถสรุปผลได้ดังนี้

1. สามารถศึกษาการเคลื่อนที่ของโหนดในโครงข่ายแบบแอดฮอคได้จากโปรแกรมการจำลองการเคลื่อนที่ของโหนดในโครงข่ายเคลื่อนที่แบบแอดฮอคด้วยภาพกราฟฟิก (GUI for Studying Random Way Point Mobility in MANETs) ที่ได้จัดทำขึ้นนี้ เพื่อประกอบการศึกษาซึ่งทำให้เข้าใจการเชื่อมต่อภายในโครงข่ายแบบแอดฮอคได้ง่ายขึ้น

2. สามารถศึกษากระบวนการหาเส้นทางที่สั้นที่สุดในโครงข่ายเคลื่อนที่แบบแอดฮอคได้จากโปรแกรมการจำลองการเคลื่อนที่ของโหนดในโครงข่ายเคลื่อนที่แบบแอดฮอคด้วยภาพกราฟฟิก

3. สามารถนำโปรแกรมมาทำการทดลองเพื่อศึกษาหาค่าพารามิเตอร์ที่มีผลต่อการทำงานของตัวโปรแกรม

6.2 ปัญหาและอุปสรรค

1. คณะผู้จัดทำโครงงานมีความรู้พื้นฐานในการเขียนโปรแกรมไม่มากนัก ทำให้ใช้เวลาในการศึกษาการเขียนโปรแกรมภาษาซีพลัสพลัส (C++) เป็นอย่างมาก โดยได้แก้ไขและศึกษาโปรแกรมจากหนังสือต่างๆ รวมทั้งขอคำปรึกษาจากผู้ที่มีความรู้ในการเขียนโปรแกรม

2. เนื่องจากโปรแกรมภาษา C++ เป็นโปรแกรมที่ได้รับการพัฒนามาในลำดับต้นๆจึงค่อนข้างมีปัญหาในด้านการพัฒนาทางกราฟฟิก

จึงเป็นการยากในการปรับปรุงโปรแกรมการจำลองการเคลื่อนที่ของโหนดในโครงข่ายเคลื่อนที่แบบแอดฮอคด้วยภาพกราฟฟิก

3. ในระหว่างการทดลองหาค่าพารามิเตอร์ที่มีผลต่อการเชื่อมต่อของเส้นทางในโปรแกรมการจำลองการเคลื่อนที่ของโหนดในโครงข่ายเคลื่อนที่แบบแอดฮอคด้วยภาพกราฟฟิกนั้นต้องมีการจับเวลาตลอดการทดลอง ดังนั้น เนื่องจากใช้เวลาในการจับเวลาค่อนข้างนานและบ่อยครั้ง จึงอาจทำให้คลาดเคลื่อนบ้างแต่ไม่มีผลมากนัก

6.3 ขีดจำกัดของโครงงาน

1. ตัวโปรแกรมใช้ออกแบบในการหาเส้นทางการเชื่อมต่อที่สั้นที่สุดระหว่างโหนดต้นทางและปลายทางของโครงข่ายเคลื่อนที่แบบแอดฮอค เพื่อทำการหาเส้นทางที่สั้นที่สุดในรูปทรง (Topology) หนึ่งๆ นั้นจะต้องทำการหยุดการเคลื่อนที่ แต่โปรแกรมจะไม่สามารถให้โครงข่ายมีการเคลื่อนที่ต่อเนื่องจากเดิมที่หยุดไว้เพื่อหาเส้นทางเชื่อมต่อที่สั้นที่สุดอีกครั้งในโครงข่ายเดิมได้ โดยจะต้องทำการรันโปรแกรมใหม่ทุกครั้งที่ต้องการหาเส้นทางการเชื่อมต่อที่สั้นที่สุด นั่นคือการสุ่มลักษณะ Topology ของโครงข่ายใหม่นั่นเอง

2. ตัวโปรแกรมมีความเร็วในการแสดงผลค่อนข้างช้า อีกทั้งสร้างแบบจำลองจำนวนโหนดได้ไม่มากนัก อันเนื่องมาจากความซับซ้อนของตัวโปรแกรมจึงจำเป็นต้องหาอัลกอริทึมใหม่ๆเพื่อลดความซับซ้อนของตัวโปรแกรม

3. ตัวโปรแกรมนี้ใช้รันกับโปรแกรมเดฟซีพลัสพลัส (Dev-C++) ซึ่งอาจเป็นโปรแกรมที่ไม่เป็นที่แพร่หลายมากนักดังนั้นอาจมีการพัฒนาตัวโปรแกรมนี้ให้สามารถรันได้กับโปรแกรมอื่นๆ ซึ่งเป็นที่นิยมมากขึ้น

6.4 ข้อเสนอแนะ

ผู้ที่สนใจเกี่ยวกับการทำภาพกราฟฟิกสามารถใช้โปรแกรมนี้เพื่อการศึกษาในเบื้องต้นแล้วทำการปรับปรุงความสามารถของโปรแกรม เช่น

ความสามารถในการรองรับการสร้างแบบจำลองของจำนวนโหนด พัฒนารูปแบบการนำเสนอของหน้าต่างกราฟฟิกให้มีความดึงดูดมากขึ้นและทำการปรับปรุงความสามารถของโปรแกรมให้สามารถทำงานต่อเนื่องในโครงข่ายเดิม เช่นไม่ต้องทำการรันโปรแกรมใหม่หลายๆ ครั้ง ผลกระทบของการแสดงผลเมื่อเพิ่มจำนวนโหนดมากขึ้นแล้วทำให้การแสดงผลช้าลง เป็นต้น

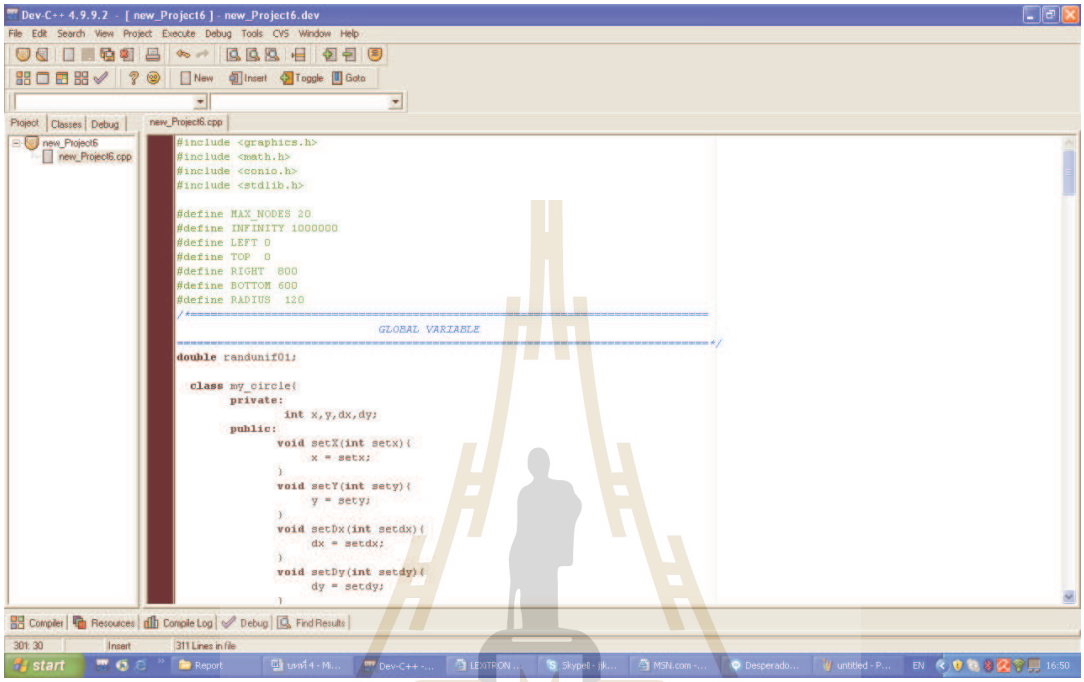
บรรณานุกรม

- R. D. Vines, Wireless Security Essential, Wiley Publishing, 2002
- C. Peikari and S. Fogie, Wireless Maximum Security, SAMS, 2002
- M. Maxim and D. Pollino, Wireless Security, RSA Press, 2002
- C. Barnes, T. Bautts, D. Lloyd, E. Ouellet, J. Posluns, D. M. Zendzian, N. O'Farrell, Hack Proofing Your Wireless Network, Syngress, 2002
- Wireless LAN security, ISS Technical White Paper, http://documents.iss.net/whitepapers/wireless_LAN_security.pdf
- Security and the 802.11b Wireless LAN, S. Griffin, September 2001, http://www.sans.org/rr/wireless/80211b.php
- Wireless LANs, Trinity Security Services, April 2002, http://www.itsecurity.com/papers/trinity6.htm
- Hardening IEEE 802.11 Wireless Networks, T. Macaulay, Febuary 2002, http://www.ewa-canada.com/Papers/Hardening_802.11.pdf
- Wireless Security Black Paper, T. A. Dismukes, July 2002, http://www.arstechnica.com/paedia/w/wireless/security-1.html
- Security Practicum: Essential Home Wireless Security Practices, K. C. Fisher, November 2002, http://www.arstechnica.com/paedia/w/wireless-security-howto/home-802.11b-1.html
- IEEE 802.11b High Rate Wireless Local Area Network, K. Sarinnapakorn, March 2001, http://alpha.fdu.edu/~kanoksri/IEEE80211b.html
- Cisco Aironet Wireless LAN Security Overview, Cisco, http://www.cisco.com/warp/public/cc/pd/witc/ao350ap/prodlit/a350w_ov.pdf

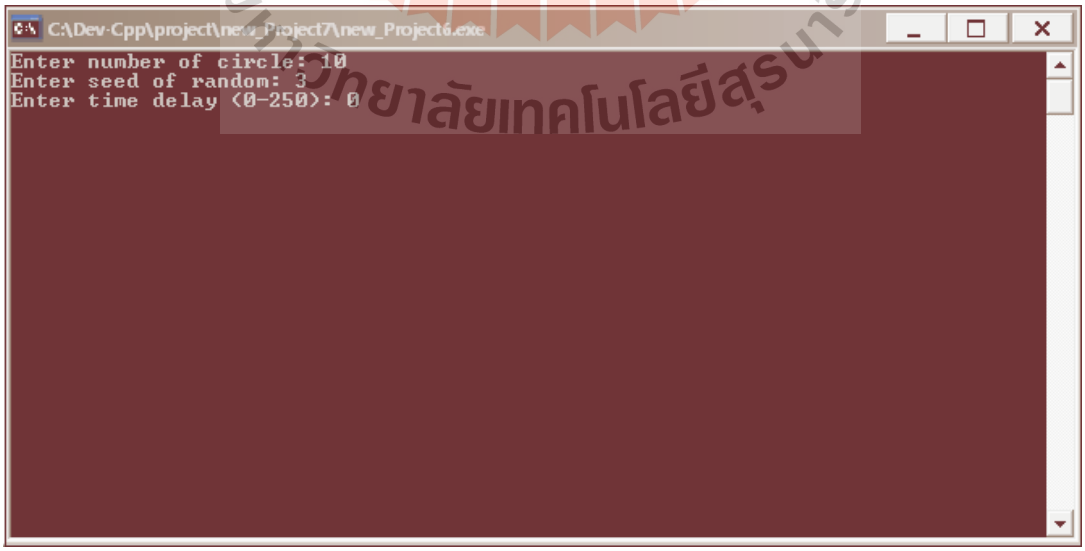
ภาคผนวก

ตัวอย่างการใช้งานโปรแกรม

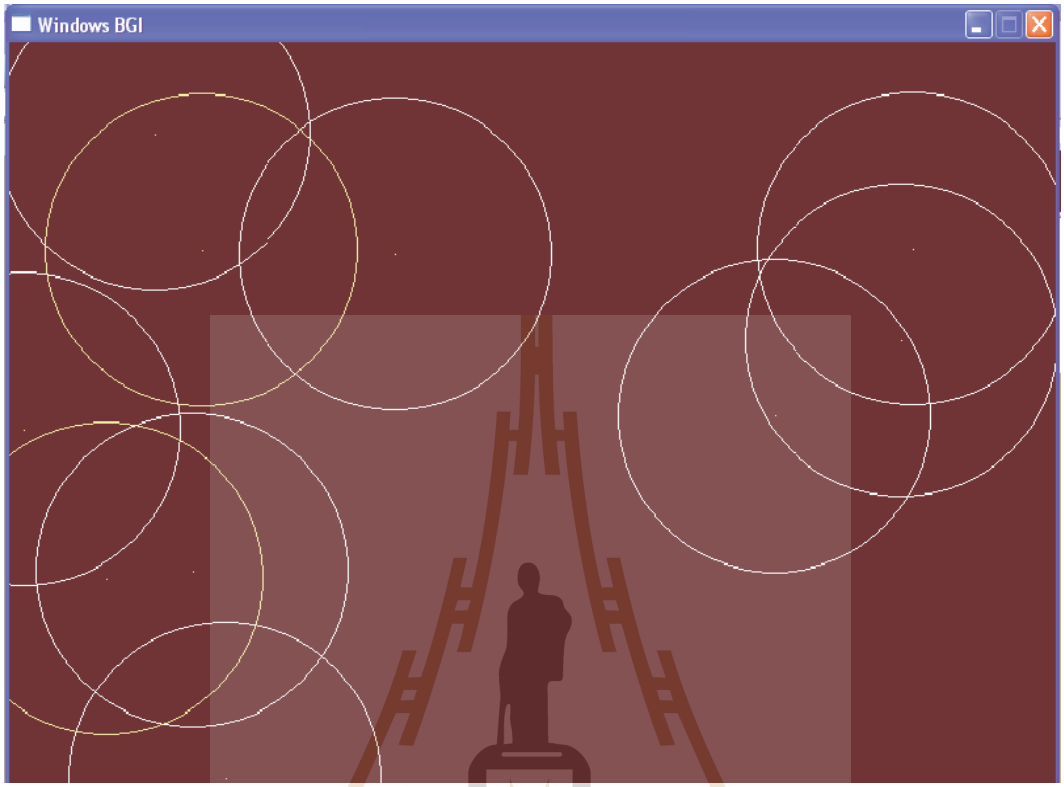
1.) ตรวจสอบความผิดพลาด (Error) ของโปรแกรม โดยใช้คำสั่ง Execute--> Compile หรือ Ctrl+F9



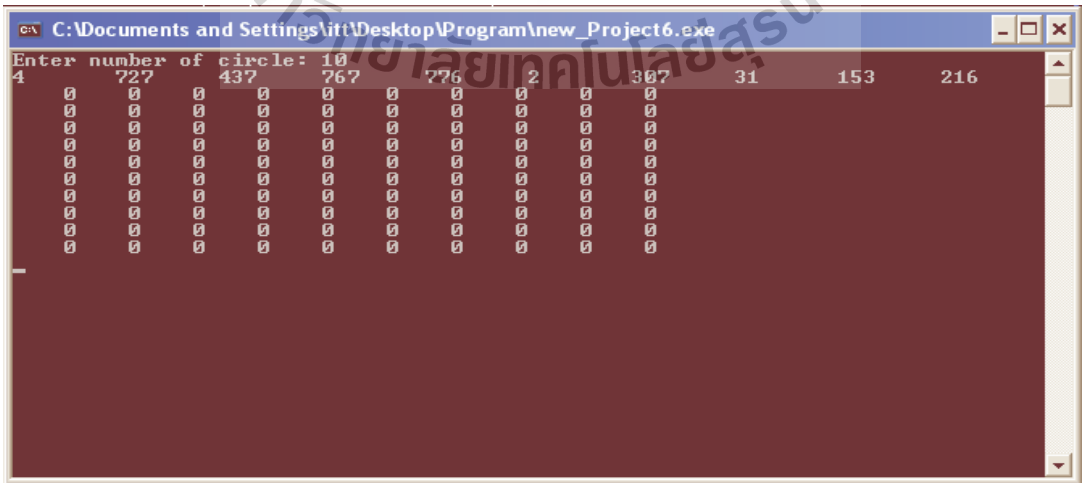
2.) ทำการประมวลผลเพื่อแสดงผลทางหน้าจอ โดยใช้คำสั่ง Execute--> Run หรือ Ctrl+F10 จะปรากฏหน้าต่างนี้ขึ้นมา เพื่อให้เราป้อนจำนวนโหนดลงไป



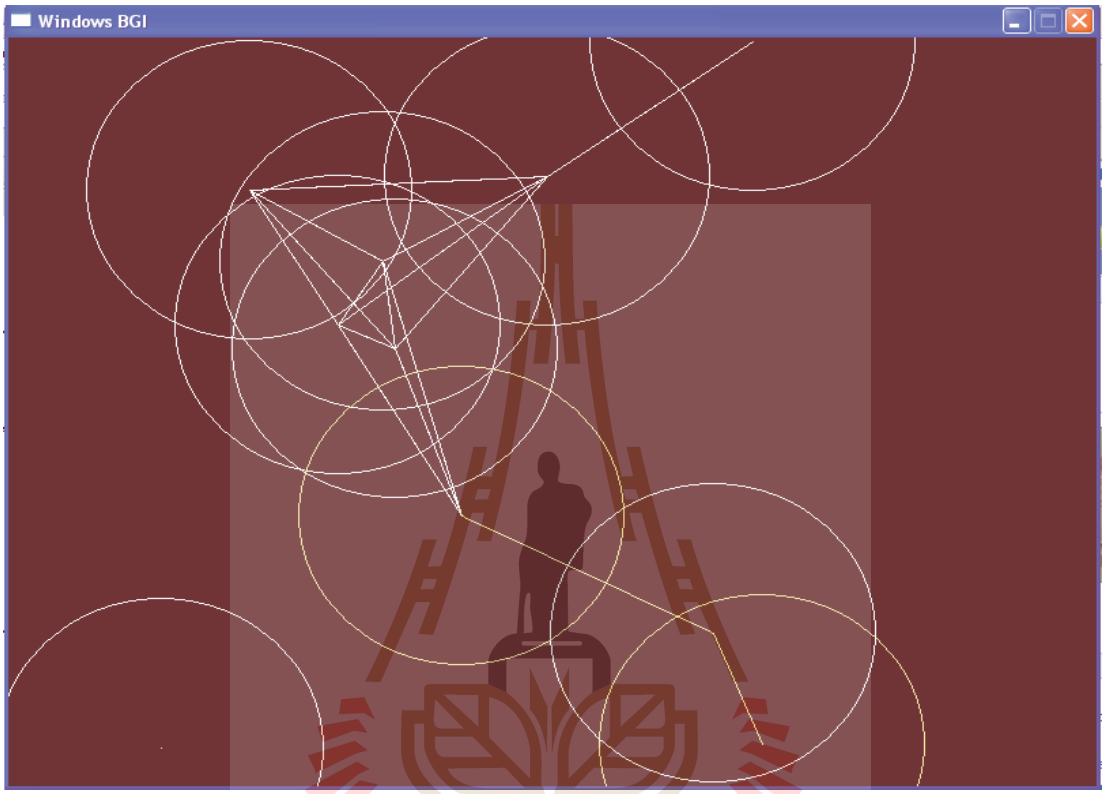
3.) จะปรากฏหน้าต่างแสดงภาพเคลื่อนไหวของวงกลมซึ่งแทนโหนดของเครือข่าย



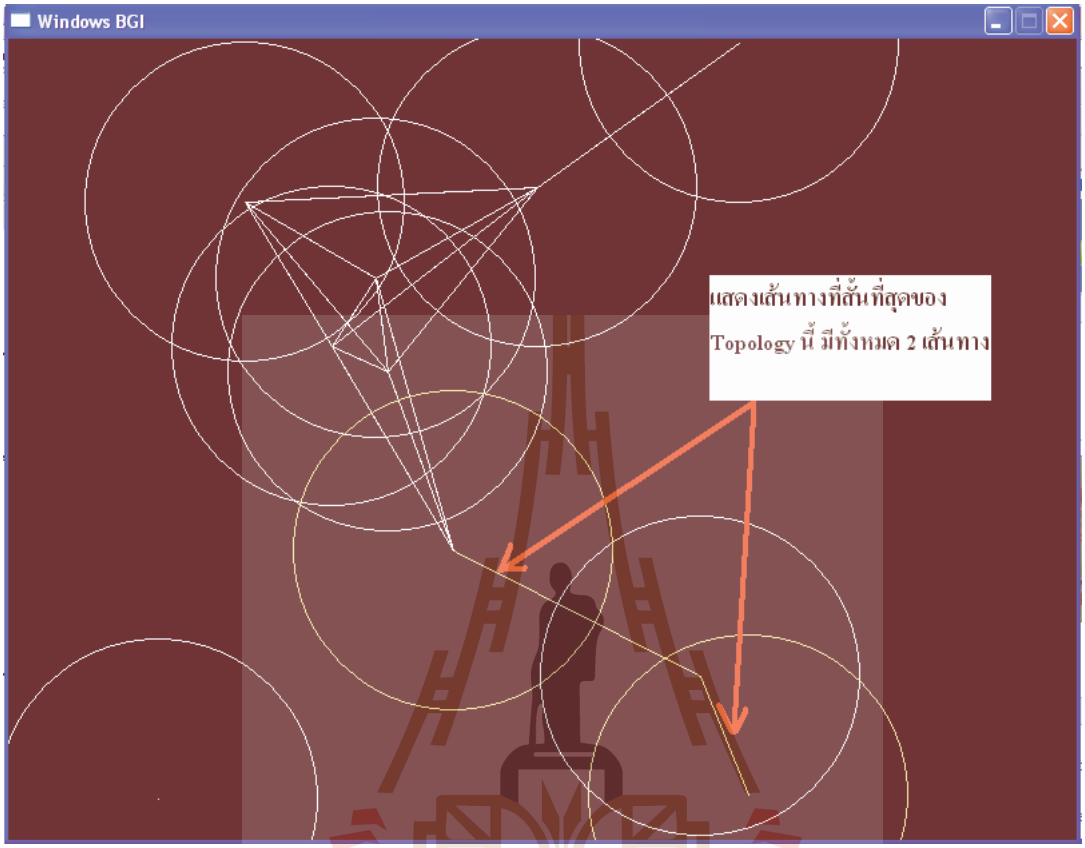
4.) พร้อมกันนี้หน้าต่างนี้จะปรากฏขึ้นพร้อมกับหน้าต่างที่แสดงภาพเคลื่อนไหว หมายเลขในบรรทัดที่ 2 แสดงถึงตำแหน่งสุ่ม (Random) ตำแหน่งที่เกิด จำนวนหมายเลขจะเท่ากับจำนวนโหนดที่เราป้อนลงไป



5.) ขณะที่หน้าต่างภาพเคลื่อนไหวนี้มีการแสดงผลอยู่นั้น เมื่อเรากดปุ่มใดปุ่มหนึ่งในแป้นคีย์บอร์ด ภาพจะหยุดเคลื่อนไหวแสดงถึงลักษณะรูปทรง (Topology) ที่ปรากฏขึ้น



6.) เส้นสีเหลืองแสดงเส้นทางที่สั้นที่สุด (Shortest Path) ของ Topology นี้



ประวัติผู้เขียน

นางสาวชลธิดา สุธรรม เกิดเมื่อวันที่ 7 เดือนพฤศจิกายน พ.ศ.2527 ภูมิลำเนาเดิมอยู่ที่ตำบลไชยบุรี อำเภอท่าอุเทน จังหวัดนครพนม สำเร็จการศึกษาระดับมัธยมศึกษาจากโรงเรียนวัดป่าประดู่ อำเภอเมือง จังหวัดระยอง เมื่อปี พ.ศ. 2545 ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

นางสาวนาฏนิภา ชูจิตารมย์ เกิดเมื่อวันที่ 1 เดือนมิถุนายน พ.ศ. 2527 ภูมิลำเนาเดิมอยู่ที่ตำบลในเมือง อำเภอเมือง จังหวัดนครราชสีมา สำเร็จการศึกษาระดับมัธยมศึกษาจากโรงเรียนสุรนารีวิทยา อำเภอเมือง จังหวัดนครราชสีมา เมื่อปี พ.ศ. 2545 ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

นายพงษ์นรินทร์ ศรีพลอย เกิดเมื่อวันที่ 30 เดือนมีนาคม พ.ศ. 2528 ภูมิลำเนาเดิมอยู่ที่ตำบลพิชัย อำเภอเมือง จังหวัดลำปาง สำเร็จการศึกษาระดับมัธยมศึกษาจากโรงเรียนบุญวาทย์วิทยาลัย อำเภอเมือง จังหวัดลำปาง เมื่อปี พ.ศ. 2545 ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี