บทที่ 3
ระเบียบวิธีวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อศึกษาและเปรียบเทียบการทำงานของโพรโทคอล AODV สำหรับการสื่อสารข้อมูลภายในเครือข่ายเซนเซอร์ไร้สายตามมาตรฐาน IEEE 802.15.4 ระหว่าง Router Node กับ Coordinator Node ซึ่งตัว Coordinator Node จะทำหน้าที่เป็น Gateway ในการติดต่อสื่อสารกับเครือข่ายอื่นๆ โดยเปรียบเทียบกรณีที่มี Coordinator Node 1 ตัว กับ กรณีที่มี Coordinator Node 2 ตัว โดยงานวิจัยนี้จะแสดงให้เห็นถึงประโยชน์ของการมี Coordinator Node ที่มากกว่า 1 ตัว

3.1 แนวทางการวิจัยและพัฒนา

1) ศึกษาค้นคว้าทฤษฎีและงานวิจัยที่เกี่ยวข้อง โดยจะเริ่มจากเครือข่ายเซนเซอร์ไร้สายมาตรฐาน IEEE802.15.4 มีการอธิบายโครงสร้างและหลักการทำงานของอุปกรณ์ในเครือข่ายในชั้นกายภาพ (Physical layers) ชั้นการควบเข้าถึงสื่อกลางการสื่อสารข้อมูล (Medium Access Control) จากนั้นจะเป็นส่วนของชั้นเครือข่าย (Network layers) ซึ่งเป็นชั้นที่พูดถึงการเดินทางของข้อมูล และจะเป็นส่วนที่สำคัญในงานวิจัยนี้ เพราะจะต้องมีการพัฒนาและออกแบบโพรโทคอลเส้นทางเพื่อใช้งานในชั้นนี้ ต่อมาจะศึกษาเกี่ยวกับโพรโทคอลเส้นทาง AODV ที่ใช้ในเครือข่ายเซนเซอร์ไร้สาย โดยดูเรื่องการทำงานของโพรโทคอล ข้อดีและข้อเสียของตัวโพรโทคอลนี้ เพื่อที่จะนำมาเป็นแนวทางในการพัฒนาโพรโทคอลใหม่ขึ้น สุดท้ายเป็นการศึกษาการใช้งานโปรแกรมจำลองการทำงานเครือข่าย (Network Simulator) ในงานวิจัยนี้จะใช้โปรแกรม NS2 ช่วยในการสร้างโหนดเซนเซอร์ไร้สายและโพรโทคอลเส้นทาง เพื่อที่จะเปรียบเทียบการทำงานของโพรโทคอล AODV ในกรณีที่มี Coordinator Node 1 ตัว กับ กรณีที่มี Coordinator Node 2 ตัว

2) ออกแบบโปรแกรมเพื่อใช้ในการจำลองการทำงานของเครือข่ายเซนเซอร์ไร้สายแบบ Multiple gateways โดยใช้ภาษา OTcL ซึ่งเป็นภาษาสคิป ในการสร้างลำดับขั้นตอนการทำงานของเครือข่าย แล้วจึงใช้โปรแกรม Network Simulator 2 หรือ NS2 ในการจำลองการทำงานของเครือข่ายเซนเซอร์ไร้สาย โดยใช้โปรโคอล AODV ในการค้นหาและสร้างเส้นทางจาก Router

Node ไปยัง Coordinator Node โดยมุ่งเน้นการออกแบบและพัฒนาโปรแกรมให้เป็นไปตามที่ระบุไว้ในวัตถุประสงค์ของวิจัย

3) สร้างโมเดลจำลองการทำงานของ IEEE802.15.4 โดยกำหนดให้มีโหนดต้นทางจำนวน 1 โหนด และโหนดปลายทางจำนวน 2 โหนด โดยโหนดทั้งหมดเป็นแบบ FFD (Full Function Device) และเป็นโหนดแบบไม่มีการเคลื่อนที่

4) ทดสอบโมเดลจำลอง จะทำโดยการใช้โปรแกรม NS2 ในการจำลองการทำงานของเครือข่ายเซนเซอร์ไร้สาย เพื่อเปรียบเทียบปริมาณ Throughput, Delay ของเครือข่าย ในกรณีที่มี Gateway 1, 2 ตัว โดยใช้โปรโคอลเส้นทาง ระหว่างโปรโคอล AODV

5) เปรียบเทียบ วิเคราะห์ผลที่ได้จากการทดสอล และสรุปการทดสอบ เพื่อที่จะสรุปผลการจำลองระบบว่าประสิทธิภาพของระบบนั้นเป็นอย่างไร

6) รวบรวมข้อมูลที่ได้ทั้งหมด เพื่อจัดทำวิทยานิพนธ์ โดยแผนการดำเนินงานที่ได้วางแผนไว้แสดงอยู่ในตารางที่ 3.1

ตารางที่ 3.1 แผนการดำเนินงาน

	พ.ย. - ธ.ค. 55	**ม.ค. - มี.ค. 56**	**เม.ย. - มิ.ย. 56**	**ก.ค. - ก.ย. 56**	**ต.ค. - ธ.ค. 56**	**ม.ค. - เม.ย. 57**	**พ.ค. - ก.ค. 57**
ศึกษาข้อมูลและงานวิจัยที่เกี่ยวข้อง							
ศึกษาการทำงานของเครือข่าย WSN และโพรโทคอล AODV							
ศึกษาการใช้งานโปรแกรม NS2							
วิเคราะห์และศึกษาแนวทางการพัฒนา ทำการเปรียบเทียบผลจากการจำลองระบบ							
รวบรวมข้อมูลและวิเคราะห์ผลจากการจำลองระบบ แล้วสรุปผล							
รวบรวมข้อมูลที่ได้ทั้งหมดจัดทำวิทยานิพนธ์							

3.2 เครื่องมือที่ใช้ในงานวิจัย

3.2.1 เครื่องคอมพิวเตอร์แบบพกพาจำนวน 1 เครื่อง ซึ่งมีคุณสมบัติดังนี้

1) CPU Intel® Core™ i3-3227U @ 1.90GHz
2) VGA Intel® HD Graphics 4000
3) RAM DDR3 4GB
4) Hard Disk, SATA 500GB
5) Network Adapter: Wireless LAN 802.11n
6) OS Microsoft Windows 8, 64-bit

3.2.2 ซอฟแวร์ที่ใช้ในการพัฒนาและวิจัย มีรายละเอียดดังนี้

1) VMWare Workstation Ver.9.01
   a. CPU 2 Processor
   b. Memory 2 GB
   c. Hard Disk (SCSI) 20GB
   d. Network Adapter: NAT
2) OS Ubuntu Desktop 12.04 LTS
3) Network Simulator 2.34 (NS2)
4) C++
5) OTcL Script

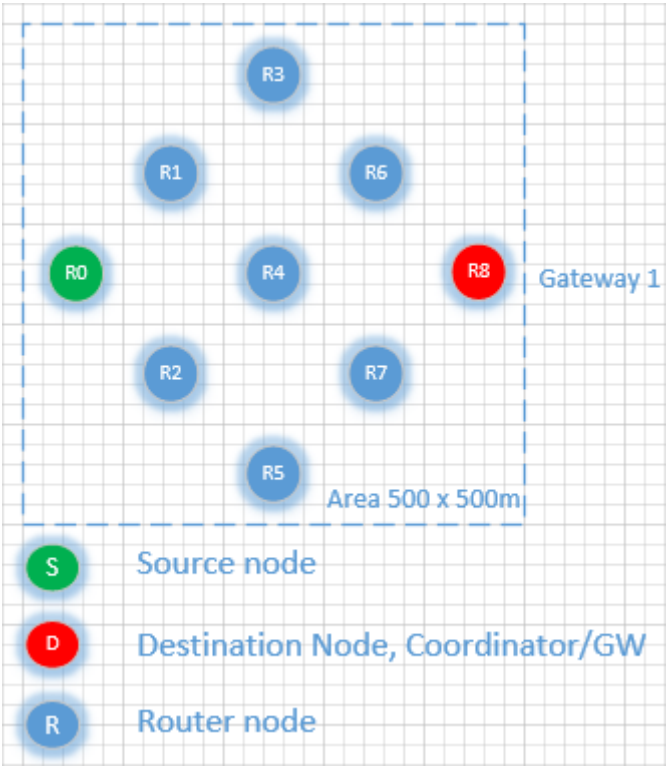
3.3 ขั้นตอนการดำเนินงานวิจัย

3.3.1 สร้างโมเดลโครงข่าย WSN

สร้างโมเดลโครงข่ายเพื่อใช้ในการจำลองการทำงานของโพรโทคอล AODV โดยมีการกำหนดโหนดที่เป็น Source และ Destination ในการสร้างโมเดลโครงข่ายจำลองของเครือข่ายเซนเซอร์ไร้สายเพื่อใช้งานงานวิจัยนี้ จะแบ่งเป็น 2 โมเดล โดยมีการอ้างอิงโครงข่ายต้นแบบของงานวิจัย Node disjoint multi-path routing for ZigBee cluster-tree Wireless Sensor Network ซึ่งเป็น Circular Topology โดยมีการปรับขนาดของจำนวนโหนดให้เล็กลง และเพิ่มโหนดที่เป็น Coordinator Node ดังนี้

1) แบบที่ 1 Coordinator Node (1 Gateway) จะประกอบไปด้วยโหนดที่เป็น FFD โหนด จำนวน 9 โหนด มีโหนดที่ทำหน้าที่เป็น Source จำนวน 1 โหนด และโหนดที่ทำหน้าที่เป็นตัว Destination จำนวน 1 โหนด โดยจะมีการส่งข้อมูลที่เป็น CBR Packet จาก Source โหนด ไปยัง

Destination โหนด กำหนดขนาดของทอพอโลยี จะเป็น 500 x 500 เมตร จากรูปข้างล่าง จะประกอบไปด้วย โหนด R0 ถึง R8 รวมเป็นทั้งหมด 9 โหนด โดยโหนด R0 จะทำหน้าที่เป็น Source โหนด และโหนด R8 จะทำหน้าที่เป็น Destination โหนด ในที่นี้จะทำงานเป็น PAN Coordinator และ Gateway ในตัว นอกนั้นโหนดที่เหลือ R1 ถึง R7 จำนวน 7 โหนด จะทำหน้าที่เป็นโหนด Router โดยมีการใช้โพรโทคอล AODV ในการให้เส้นทางภายในเครือข่าย



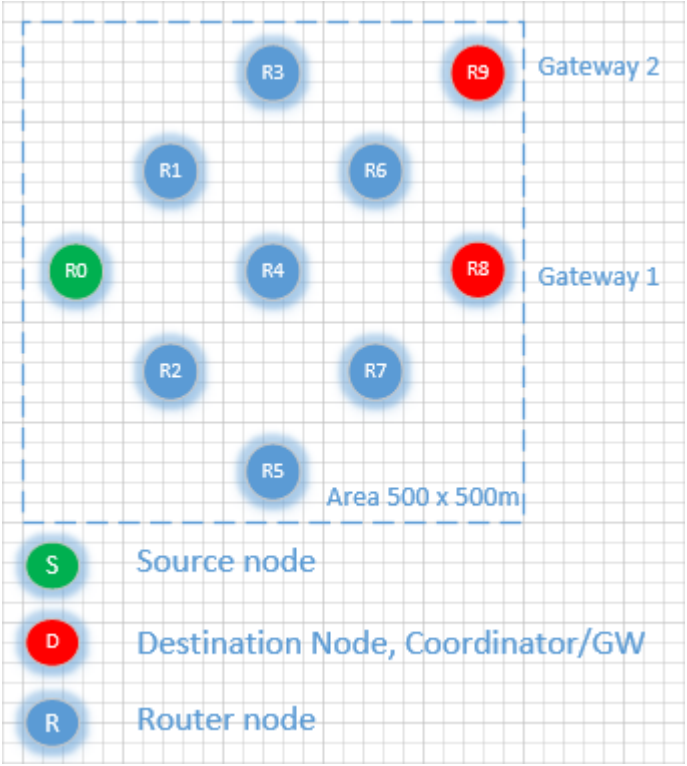
ภาพที่ 3.1 โครงสร้างของเครือข่ายที่จะใช้ในการจำลอง แบบมี 1 Gateway

เมื่อมีการกำหนดตัวโครงสร้างของเครือข่ายที่จะใช้ในการจำลอง แบบ 1 Gateway แล้ว ได้ทำการกำหนดค่าคุณสมบัติต่างๆ ในโหนด และเครือข่ายที่จะทำการจำลองโดยมีรายละเอียดตามตารางข้างล่าง

ตารางที่ 3.2 การตั้งค่าต่างๆ บนทอพอโลยี แบบ 1 Coordinator/Gateway

พารามิเตอร์	ค่าที่กำหนด
Number of Nodes	9
Number of FFD	9
Circle Radius R (m)	10
Number of Sources	1
Number of Sink	1 (PAN Coordinator)
Sink Position	Right of the area
Node Mobility	None
MAC Protocol	IEEE 802.15.4 (Non-Beacon mode with unslotted CSMA/CA mechanism)
Propagation Model	Two Ray Ground Model
Queue Size	50
Data Transfer Model	Direct Data Transmission
Traffic Model	Poisson traffic
Packet Type	CBR
Packet Size (bytes)	100
Simulation Area (m*m)	500*500
Packet Quantity	5000, 10000, 15000, 20000, 25000
Simulation Time (sec)	45

2) แบบ 2 PAN Coordinator จะประกอบไปด้วยโหนดที่เป็น FFD โหนด จำนวน 10 โหนด มีโหนดที่ทำหน้าที่เป็น Source จำนวน 1 โหนด และโหนดที่ทำหน้าที่เป็นตัว Sink จำนวน 2 โหนด โดยจะมีการส่งข้อมูลที่เป็น CBR Packet จาก Source โหนด ไปยัง Sink โหนด กำหนดขนาดของทอพอโลยี จะเป็น 500 x 500 เมตร จากรูปข้างล่าง จะประกอบไปด้วย โหนด R0 ถึง R9 รวมเป็นทั้งหมด 10 โหนด โดยโหนด R0 จะทำหน้าที่เป็น Source โหนด และโหนด R8, R9 จะทำหน้าที่เป็น Destination โหนด ในที่นี้จะทำงานเป็น PAN Coordinator และ Gateway ในตัว นอกนั้นโหนดที่เหลือ R1 ถึง R7 จำนวน 7 โหนด จะทำหน้าที่เป็นโหนด Router โดยมีการใช้โพรโทคอล AODV ในการให้เส้นทางภายในเครือข่าย



ภาพที่ 3.2 โครงสร้างของเครือข่ายที่จะใช้ในการจำลอง แบบมี 2 Gateways

เมื่อมีการกำหนดตัวโครงสร้างของเครือข่ายที่จะใช้ในการจำลอง แบบ 1 Gateway แล้ว ได้ทำการกำหนดค่าคุณสมบัติต่างๆ ในโหนด และเครือข่ายที่จะทำการจำลองโดยมีรายละเอียดตามตารางข้างล่าง

ตารางที่ 3.3 การตั้งค่าต่างๆ บนทอพอโลยี แบบ 2 Coordinator/Gateway

พารามิเตอร์	ค่าที่กำหนด
Number of Nodes	10
Number of FFD	10
Circle Radius R (m)	10
Number of Sources	1

ตารางที่ 3.3 (ต่อ)

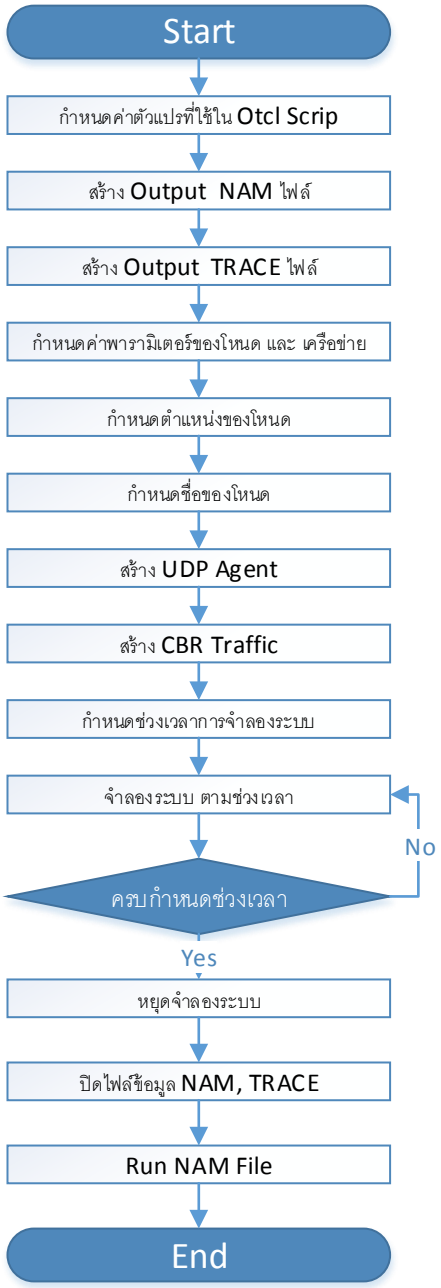
พารามิเตอร์	ค่าที่กำหนด
Number of Sink	2 (PAN Coordinator)
Sink Position	Right of the area
Node Mobility	None
MAC Protocol	IEEE 802.15.4 (Non-Beacon mode with unslotted CSMA/CA mechanism)
Propagation Model	Two Ray Ground Model
Queue Size	50
Data Transfer Model	Direct Data Transmission
Traffic Model	Poisson traffic
Packet Type	CBR
Packet Size (bytes)	100
Simulation Area (m*m)	500*500
Packet Quantity	5000, 10000, 15000, 20000, 25000
Simulation Time (sec)	45

3.3.2 เริ่มกำหนดค่าพารามิเตอร์ในโปรแกรม NS2

กำหนดค่าพารามิเตอร์ที่จะใช้ในตัวโปรแกรม Network Simulation 2 (NS-2) ตามตารางที่ 3.2 และ 3.3 โดยจะสามารถกำหนดได้โดยการเขียนโปรแกรมผ่าน Otcl Script โดยมีลำดับการกำหนดค่าพารามิเตอร์ต่างๆ ตามแผนผังการกำหนดค่าในโปรแกรมภาษา Otcl Script โดยตัวโปรแกรมที่เขียนขึ้นจะใช้โปรแกรม editor ชื่อ gedit ในการสร้างส่วนของ Otcl Scrip โดยใช้นามสกุล tcl ซึ่งเป็นโปรแกรมที่ทำงานอยู่บน Ubuntu 12.04 และใช้ตัวโปรแกรม Network Simulator 2.35 ในการจำลองการทำงาน โดยใช้คำสั่งตามรูปด้านล่างในการเริ่มโปรแกรม

```
chanadej@ubuntu:~/thesis$ ns wsn2gw.tcl
```

ภาพที่ 3.3 คำสั่งการใช้โปรแกรม Network Simulator 2.35 เพื่อสั่งงาน Otcl Script



ภาพที่ 3.4 แผนผังการกำหนดค่าในโปรแกรมภาษา Otcl Script สำหรับโมเดลจำลอง

โดยลำดับขั้นตอนการทำงานของแผนผังมีดังนี้

1) กำหนดค่าตัวแปรที่ใช้ใน Otcl Scrip ในงานวิจัยนี้มีการกำหนดตัวแปรต่างๆ ดังนี้ เพื่อใช้เก็บค่าพารามิเตอร์ต่างๆ ที่จะใช้ในการกำหนดโหนด และเครือข่าย

ตารางที่ 3.4 ชื่อและค่าตัวแปรที่กำหนด

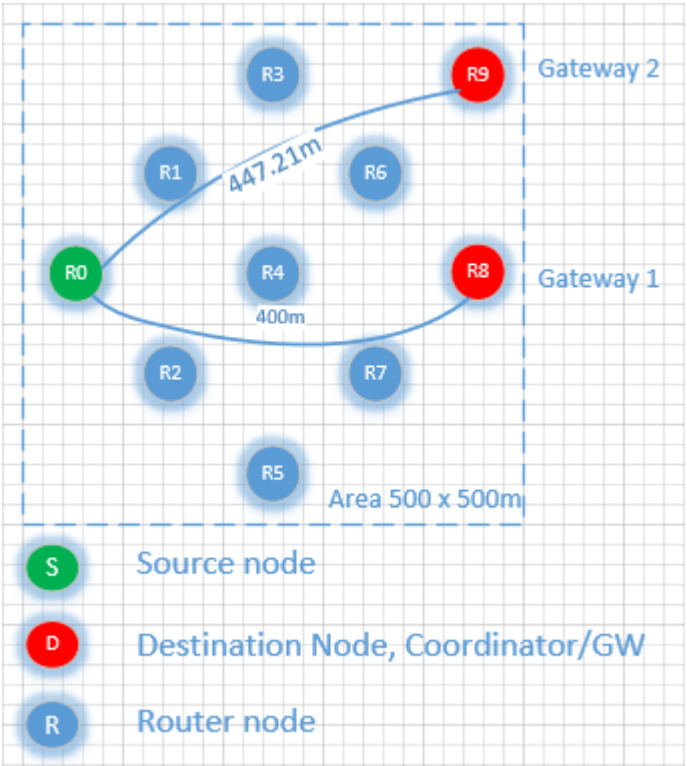
ชื่อตัวแปร	ค่าตัวแปร	การใช้งาน
Chan	WirelessChannel	การกำหนดประเภทของช่องการสื่อสารเป็นแบบไร้สาย
Prop	TwoRayGround	การกำหนดค่าชนิดของ radio-propagation
netif	802_15_4	การกำหนดค่าชนิดของ Network interface
mac	802_15_4	การกำหนดค่าชนิดของสื่อกลางที่ใช้ในการเข้าถึงช่องการสื่อสาร
ifq	PriQueue	การกำหนดค่าชนิดของการ Queue
ant	OmniAmtenna	การกำหนดค่าชนิดของสายอากาศเป็นแบบ Omni
ifqlen	50	การกำหนดค่าขนาดของ Buffer สูงสุดในแต่ละโหนด
Nn	9	การกำหนดค่าจำนวนโหนดทั้งหมดของเครือข่าย เป็น 9 โหนด
rp	AODV	การกำหนดค่าชนิดของ Rounting Protocol เป็น AODV
x	500	การกำหนดค่าขนาดของเครือข่ายในแนวนอนเป็น 500 เมตร
y	500	การกำหนดค่าขนาดของเครือข่ายในแนวตั้งเป็น 500 เมตร
stop	50	การกำหนดค่าช่วงเวลาทั้งหมดของการทำงานในโปรแกรม เป็น 50 วินาที
Energymodel	EnergyModel	การกำหนดค่าชนิดของโมเดลพลังงานที่ใช้
Initialenergy	100	การกำหนดค่าเริ่มต้นของพลังงานที่ใช้

2) สร้าง Output NAM ไฟล์ เพื่อใช้ในการแสดงผลบนโปรแกรม Network Animation โดยในงานวิจัยนี้มีการสร้างไฟล์ wsn.nam

3) สร้าง Output Trace ไฟล์ เพื่อใช้ในการเก็บผลการจำลองการทำงานที่เหตุการณ์ต่างๆ โดยในงานวิจัยนี้มีการสร้างไฟล์ wsn.tr

4) สร้างโหนด FFD และกำหนดค่าโหนด และค่าเริ่มต้นเครือข่าย โดยการเอาค่าที่ใส่ไว้ในตัวแปรมาใส่ในแต่ละโหนด

5) กำหนดตำแหน่งของโหนดต่างๆ บนเครือข่าย โดยโหนด Source มีระยะห่างจากโหนด Destination 1 (Gateway 1) อยู่ 400 เมตร และมีระยะห่างจากโหนด Destination 2 (Gateway 2) อยู่ 447.21 เมตร โดยมีลักษณะเป็น Circular Topology ตามรูปด้านล่าง



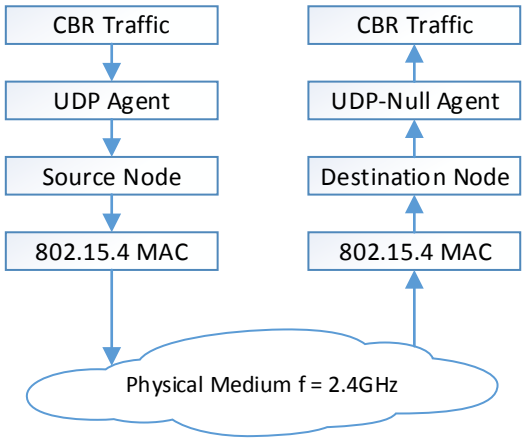
ภาพที่ 3.5 โครงสร้าง WSN และระยะห่างระหว่างโหนดต้นทาง กับโหนดปลายทาง

6) กำหนดชื่อโหนดเพื่อให้แสดงผลบนโปรแกรม Network Animation

7) สร้าง UDP Agent โดยจะมีการกำหนดส่วนของตัวที่เป็น Source Node ให้เป็น UDP และตัวที่เป็น Destination Node ให้เป็นแบบ Null เพื่อคอยรับข้อมูล

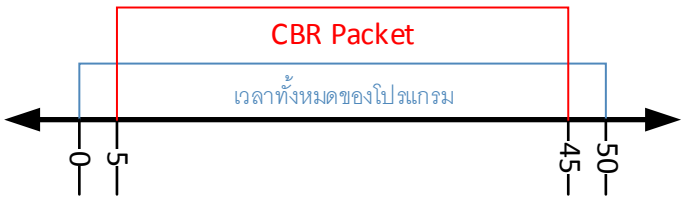
8) สร้าง Packet Traffic ที่เป็นแบบ CBR เพื่อส่งเข้าไปในตัว Source Node ผ่าน UDP Agent เมื่อ CBR Packet ถูกส่งเข้ามาในตัว UDP Agent ก็จะมีการเชื่อมต่อการส่งข้อมูลไปยังตัว Source node และส่งผ่านไปยัง MAC ที่เป็น 802.15.4 จากนั้นก็ส่งออกไปยัง Physical Layer ที่เป็น

แบบไร้สายโดยอาศัยคลื่นความถี่ 2.4GHz เป็นสื่อกลาง เพื่อส่งต่อไปยัง Destination Node ผ่านทาง 802.15.4 MAC ของโหนดปลายทางและส่งขึ้นไปยังตัว Agent เป็นการจบการส่ง CRB Packet



ภาพที่ 3.6 การส่งข้อมูลระหว่างโหนด Source กับโหนด Destination ใน NS2

9) กำหนดช่วงเวลาในการทดสอบแบบจำลอง โดยมีการกำหนดช่วงเวลาในการทดสอบ 50 วินาที ดังนี้



ภาพที่ 3.7 ช่วงเวลาในส่ง CBR Packet กรณีที่มี 1 Gateway

10) เริ่มทำงานโปรแกรมจำลองในระบบ โดยมีช่วงการทำงาน 50 วินาที และตัว CBR Packet เริ่มส่งไปยัง Destination Node ที่เวลา 5 วินาที และหยุดส่งที่เวลา 45 วินาที และโปรแกรมหยุดทำงานที่ 50 วินาที โดยมีช่วงเวลาการส่ง CBR Packet อยู่ที่ 40 วินาที

11) เมื่อโปรแกรมทำงานเสร็จก็จะทำการปิดไฟล์ wsn.nam กับ wsn.tr โดยจะบันทึกไว้ในที่อยู่เดียวกันกับ wsn.tcl ไฟล์

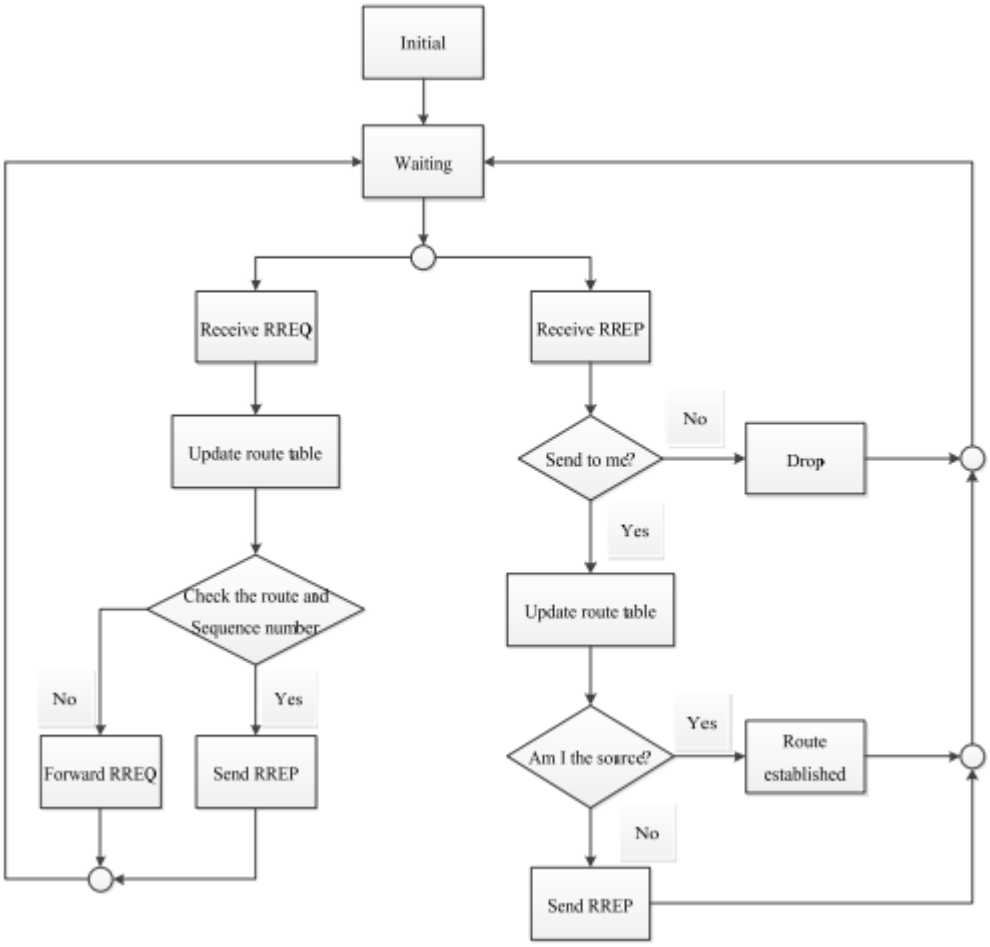
12) เรียกโปรแกรม Network Animation ขึ้นมาเพื่อแสดงการจำลองการทำงานของเครือข่ายในรูปของกราฟฟิคภาพเคลื่อนไหว

13) จบการทำงานของโปรแกรม นำผลของการจำลองเครือข่ายเซนเซอร์ไร้สายที่ได้ในรูปของ trace file ที่อยู่ในไฟล์ wsn.tr มาหาค่าที่ต้องการ

3.3.3 จำลองการทำงานของเครือข่ายในรูปแบบภาพเคลื่อนไหว

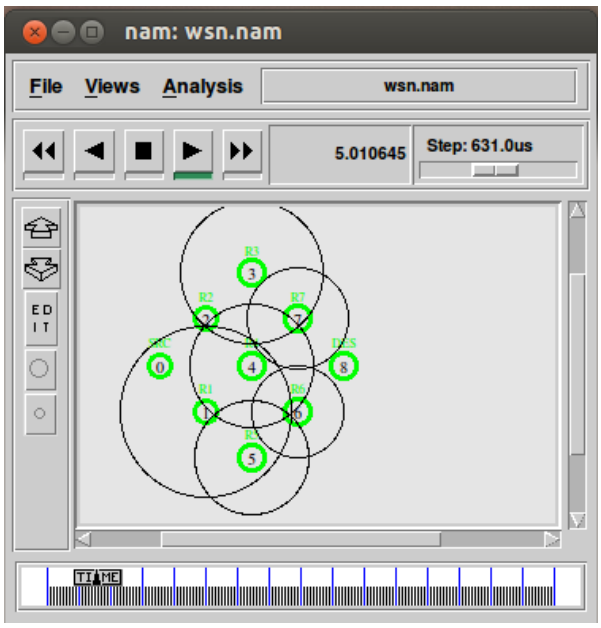
การจำลองการทำงานของเครือข่ายในรูปแบบภาพเคลื่อนไหว เพื่อให้เห็นเส้นทางการเคลื่อนที่ของข้อมูล CBR packets โดยเมื่อตัวโปรแกรม Network Simulator ทำงาน เสร็จแล้วก็จะทำการสั่งให้มีการเรียกตัวโปรแกรม Network Animation เพื่อมาแสดงผลไฟล์ที่มีนามสกุล nam ในที่นี้ใช้ชื่อ wsn.nam โดยในการแสดงผลนั้นจะสามารถสั่งให้ตัวโปรแกรมแสดงเหตุการณ์ที่เกิดขึ้นต่างๆ ในเครือข่ายตามเวลาที่ได้กำหนดไว้ตั้งแต่เริ่มต้น ในงานวิจัยนี้ใช้เวลาทั้งหมด 50 วินาที โดยบนหน้าจอของตัวโปรแกรม Network Simulator จะแสดงรูปของโหนดที่สร้างขึ้น ในที่นี้จะมีโหนดทั้งหมด 9 และ 10 โหนด โดยกรณีที่มี 9 โหนด หมายถึงกรณีที่มีโหนดที่เป็นต้นทาง Source โหนดจำนวน 1 โหนด คือ R0 และมี Gateway จำนวน 1 โหนด คือ R8 และที่เหลืออีก 7 โหนด จะเป็นโหนด Router คือ R1 – R7 ส่วนกรณีที่มี 10 โหนด หมายถึงกรณีที่มีโหนดที่เป็นต้นทาง Source โหนด จำนวน 1 โหนด คือ R0 และมี Gateway จำนวน 2 โหนด คือ R8, R9 และที่เหลืออีก 7 โหนด คือ R1 – R7 จะเป็นโหนด Router

เมื่อมีการกดปุ่มเริ่มทำงาน ตัวโหนดที่ทำหน้าที่เป็น Source “R0” ก็จะเริ่มทำการ Broadcast packet “RREQ” ไปยังโหนดข้างเคียง “R1”, “R2”, “R4” เมื่อโหนดข้างเคียงได้รับ packet “RREQ” ก็จะตรวจสอบว่าตัวมันเองเป็นโหนดปลายทางหรือไม่ ถ้าไม่ใช้ก็จะส่งต่อ packet “RREQ” ไปยังโหนดข้างเคียงต่อไป “R3”,”R7”,”R6”,”R5”,”R8” เมื่อโหนด “R8” ได้รับ packet “RREQ” และตรวจสอบแล้วว่า ตัวเองเป็นโหนด Destination ก็จะสร้าง packet “RREP” ตอบกลับไปยังเส้นทางที่ส่ง packet “RREQ” ที่มีหมายเลขลำดับนี้มา คือโหนด “R4” จากนั้นเมื่อโหนด “R8” ได้รับ packet “RREQ” ที่มีหมายเลขลำดับเดียวกัน ที่มาจากโหนดอื่นๆ ก็จะ Drop packet “RREQ” นั้นทิ้งไป เพื่อป้องกันการเกิดลูปในเครือข่าย เมื่อ packet “RREP” ที่ตอบกลับ ไปถึงตัวโหนด Source “R0“ ตัวโหนด Source “R0” ก็จะส่ง packet “CBR” ไปยังเส้นทางที่รับมา ในกรณีที่เส้นทางจากโหนด “R4” ที่ไปยังโหนด “R8” ไม่สามารถใช้งานได้ ตัวโหนด “R4” ก็จะกระจาย Broadcast packet “RRER” กลับไปยังโหนดต้นทาง ก็คือโหนด “R0” เพื่อให้เริ่มกระบวนการค้นหาเส้นทางใหม่

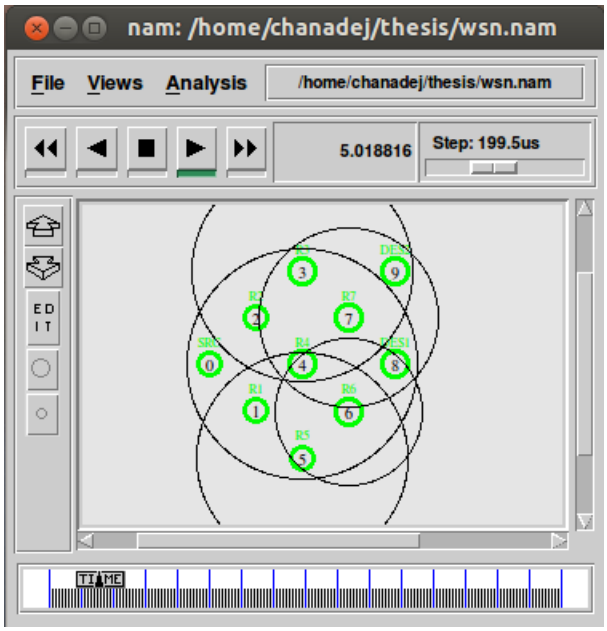


ภาพที่ 3.8 ลำดับขั้นตอนการส่งข้อมูลของตัวโพรโทคอล AODV

จากภาพที่ 3.8 ตัวโปรโตตคอล AODV ในสภาวะที่กำลังรอที่จะส่งข้อมูล เมื่อโหนดได้รับการร้องขอเส้นทาง “RREQ” จะมีการปรับปรุงเส้นทางในตารางเส้นทาง โดยการตรวจสอบเส้นทางและหมายเลขลำดับ ถ้ามีโหนดปลายทางในตารางเส้นทาง และหมายเลขลำดับเป็นปัจจุบันก็จะส่ง “RREP” กลับไปยังโหนดต้นทางตามเส้นทางเดิมที่รับ “RREP” มา แต่ถ้าไม่มีโหนดปลายทางในตารางเส้นทาง ก็จะทำการส่งต่อ “Forward RREQ” ไปยังโหนดข้างเคียง เมื่อโหนดใดๆ ได้รับ “RREP” ก็จะตรวจสอบว่าเป็นของตัวเองหรือไม่ ถ้าไม่ใช่ก็จะ Drop “RREP” นั้นทิ้งไป แต่ถ้าใช่ ก็จะปรับปรุงเส้นทางในตารางเส้นทางของตัวเอง แล้วดูว่าตัวเองเป็นโหนดต้นทางหรือไม่ ถ้าไม่ใช่ก็จะส่ง “RREP” ต่อไป ถ้าใช่ ก็จะสร้างเส้นทาง เพื่อให้ UDP ส่งข้อมูลไปยังโหนดปลายทาง



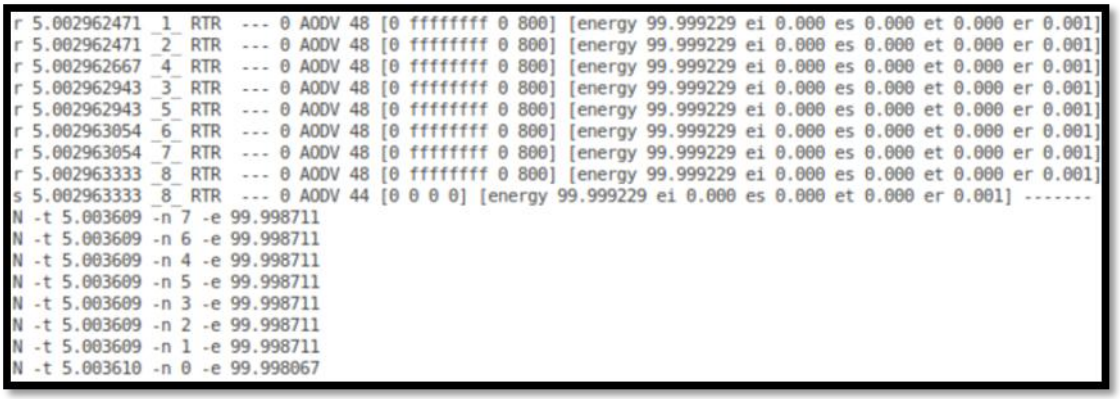
ภาพที่ 3.9 โปรแกรม Network Animation ขณะจำลองการทำงานของ WSN 1 Gateway



ภาพที่ 3.10 โปรแกรม Network Animation ขณะจำลองการทำงาน WSN แบบ 2 Gateways

3.3.4 วิเคราะห์ผลการจำลองการทำงานของ WSN แบบ 1 Gateway

นำผลการจำลองการทำงานของเครือข่ายที่อยู่ใน trace file มาทำการวิเคราะห์ในโปรแกรม LibreOffice3 ซึ่งเป็นโปรแกรมที่ทำงานบนระบบปฏิบัติการ Ubuntu โดยรายละเอียดของ Trace ไฟล์ที่ถูกสร้างขึ้น จะมีตัวอย่างรายละเอียดตามภาพที่ 3.8 ด้านล่าง



```
r 5.002962471 _1_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] [energy 99.999229 ei 0.000 es 0.000 et 0.000 er 0.001]
r 5.002962471 _2_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] [energy 99.999229 ei 0.000 es 0.000 et 0.000 er 0.001]
r 5.002962667 _4_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] [energy 99.999229 ei 0.000 es 0.000 et 0.000 er 0.001]
r 5.002962943 _3_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] [energy 99.999229 ei 0.000 es 0.000 et 0.000 er 0.001]
r 5.002962943 _5_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] [energy 99.999229 ei 0.000 es 0.000 et 0.000 er 0.001]
r 5.002963054 _6_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] [energy 99.999229 ei 0.000 es 0.000 et 0.000 er 0.001]
r 5.002963054 _7_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] [energy 99.999229 ei 0.000 es 0.000 et 0.000 er 0.001]
r 5.002963333 _8_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] [energy 99.999229 ei 0.000 es 0.000 et 0.000 er 0.001]
s 5.002963333 _8_ RTR  --- 0 AODV 44 [0 0 0 0] [energy 99.999229 ei 0.000 es 0.000 et 0.000 er 0.001] -------
N -t 5.003609 -n 7 -e 99.998711
N -t 5.003609 -n 6 -e 99.998711
N -t 5.003609 -n 4 -e 99.998711
N -t 5.003609 -n 5 -e 99.998711
N -t 5.003609 -n 3 -e 99.998711
N -t 5.003609 -n 2 -e 99.998711
N -t 5.003609 -n 1 -e 99.998711
N -t 5.003610 -n 0 -e 99.998067
```

ภาพที่ 3.11 โครงสร้างของ Trace ไฟล์ ที่สร้างขึ้น

โดยใน Otcl script ที่เขียนขึ้นจะกำหนดให้มีการสร้าง Trace ไฟล์ ชื่อ wsn.tr ที่จะบันทึกเหตุการณ์ที่เกิดขึ้น ที่ส่วนของ Router และ Agent เพื่อดูเหตุการณ์ที่เกิดขึ้นในแต่ละ packet โดยมีโครงสร้างของ Trace ไฟล์จะมีรายละเอียดของแต่ละคอลัมน์ดังนี้

รายละเอียดคอลัมน์ที่ 1-7

Event	Time	Node	Trace Level	Flags	SEQNO	Packet Type

รายละเอียดคอลัมน์ที่ 8-13

Packet Size	MAC [a b c d]	Energy ei es et er	Flags	Network	Transport	Message

คอลัมน์ที่ 1 (Event) แสดงรายละเอียดของเหตุการณ์ที่เกิดขึ้นนะเวลานั้นๆ โดยมีความหมายของแต่ละ Event ดังนี้

1) s: มีการส่งข้อมูล
2) r: มีการรับข้อมูล
3) D: ข้อมูลถูก Drop

คอลัมน์ที่ 2 (Time) แสดงเวลาที่เกิดขึ้นที่เหตุการณ์นั้นๆ มีหน่วยเป็นวินาที

คอลัมน์ที่ 3 (Node) แสดงหมายเลขโหนดที่เกิดเหตุการณ์นั้นๆ

คอลัมน์ที่ 4 (Trace Level) บอกถึงส่วนที่มีการเก็บข้อมูล แต่ละชื่อมีความหมายดังนี้

1) RTR: เกี่ยวกับเส้นทาง

2) AGT: เกี่ยวกับตัวโปรแกรมประยุกต์ เช่น UDP

คอลัมน์ที่ 5 (Flags) เป็นการแสดงเครื่องหมายต่างๆ

คอลัมน์ที่ 6 (SEQNO) บอกถึงหมายเลขลำดับของเพ็คเก็จ

คอลัมน์ที่ 7 (Packet Type) บอกถึงชนิดของเพ็คเก็จที่เกิดขึ้น เช่น AODV, CBR

คอลัมน์ที่ 8 (Packet Size) บอกถึงขนาดของเพ็คเก็จมีหน่วยเป็นไบต์ (Byte)

คอลัมน์ที่ 9 (MAC [a b c d]) มีความหมายแต่ละตำแหน่งดังนี้

1) ตำแหน่ง a -- the packet duration in mac layer header

2) ตำแหน่ง b -- the mac address of destination

3) ตำแหน่ง c -- the mac address of source

4) ตำแหน่ง d -- the mac type of the packet body

คอลัมน์ที่ 10 (Energy ei es et er) บอกถึงระดับพลังงานในโหนดนั้นๆ โดยมีรายละเอียดดังนี้

1) ตำแหน่ง Energy – ระดับพลังงานที่เหลืออยู่ในโหนดนั้นๆ

2) ตำแหน่ง ei – คือ Idle Power เป็นค่าพลังงานที่ถูกใช้ไปในขณะ idle มีหน่วยเป็นวัตต์

3) ตำแหน่ง es – คือ Sleep Power เป็นค่าพลังงานทีถูกใช้ไปในขณะที่โหนดหลับอยู่ มีหน่วยเป็นวัตต์

4) ตำแหน่ง et – คือ Transmitting Power เป็นค่าพลังงานทีถูกใช้ไปในขณะที่มีการส่งข้อมูล มีหน่วยเป็นวัตต์

5) ตำแหน่ง er -- คือ Receiving Power เป็นค่าพลังงานทีถูกใช้ไปในขณะที่มีการรับข้อมูล มีหน่วยเป็นวัตต์

คอลัมน์ที่ 11 (Flags) เป็นการแสดงเครื่องหมายต่างๆ

คอลัมน์ที่ 12 (Network) เป็นการแสดงค่าในส่วนของเครือข่ายประกอบไปด้วย

1) src_addr, dst_addr, TTL, next_hop_addr

คอลัมน์ที่ 12 (Transport) เป็นการแสดงค่าในส่วนของชั้น Transport ประกอบไปด้วย

1) seq_no, ack_no, no_of_fwds, opt_no_of_fwds

คอลัมน์ที่ 13 (Message) บอกประเภทของข้อมูลที่โต้ตอบกัน เช่น Request, Reply

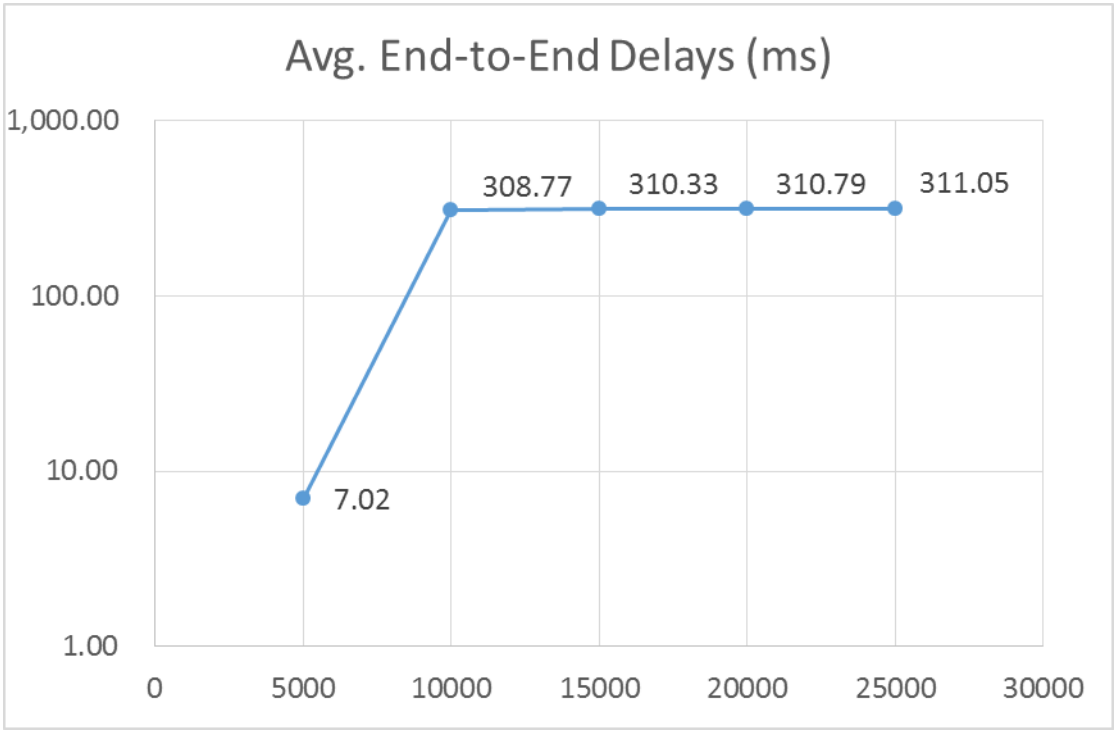
ตัวอย่างข้อมูลใน Trace ไฟล์ที่เขียนขึ้น เช่น s 5.000000000 _0_ AGT --- 18455 cbr 100 [0 0 0 0] [energy 91.093916 ei 0.000 es 0.000 et 0.438 er 8.468] ------- [0:255 -1:255 30 0] หมายความว่า มีการส่งข้อมูลที่เวลา 5.0 วินาที และเกิดขึ้นที่โหนด “0” โดยมีชนิดของข้อมูลเป็นแบบ “cbr” และหมายเลขลำดับของแพ็คเก็จเป็น “18455” มีขนาดข้อมูลเป็น 100 Byte มีระดับพลังงานที่เหลืออยู่ 91.093916 วัตต์ พลังงานที่ใช้ในสภาวะ Idle = 0 วัตต์ พลังงานที่ใช้ในสภาวะ Sleep = 0 วัตต์ พลังงานที่ใช้ในการส่งข้อมูล = 0.438 วัตต์ พลังงานที่ใช้ในการรับข้อมูล = 8.468 วัตต์ มีโหนดต้นทาง source address “0” port “255” มีโหนดปลายทาง destination address “-1” คือการกระจาย Broadcast port “255” มี TTL “30” และมีโหนดถัดไปจำนวน “0”

การเปิดไฟล์ wsn.tr พื่อทำการวิเคาะห์ข้อมูลที่ได้จากการจำลอง ในงานวิจัยนี้ได้ใช้โปรแกรม LibreOffice 3 เนื่องจากเป็นโปรแกรม Freeware ที่มีหาให้ในตัว Ubuntu ที่เป็นโปรแกรม Freeware GNU ด้วยคุณสมบัติของตัวโปรแกรม LibreOffice 3 ที่สามารถใช้ฟังก์ชั่นในการคำนวณทางสถิติ และการกำหนดเงื่อนไขได้ เช่น ฟังก์ชัน SUM, MAX, MIN, AVERAGE, IF ทำให้สามารถหาค่าผลลัพธ์ต่างๆ ได้

เมื่อนำข้อมูลที่ได้มาวิเคราะห์แล้วสามารถทำการสร้างกราฟแสดงผลการทดลองโดยใช้โปรแกรม LibreOffice 3 โดยในภาพที่ 3.12 แสดงตัวอย่างตารางข้อมูลได้จากการวิเคราะห์ข้อมูลใน wsn.tr โดย เมื่อได้ข้อมูลที่ต้องการทั้งหมดแล้ว จะทำการสร้างกราฟแสดงผลข้อมูลที่ได้ดังภาพที่ 3.13

Parameter	AODV				
CBR Packet Size (byte)	100	100	100	100	100
CBR Packet Rate (Mb)	0.1	0.2	0.3	0.4	0.5
CBR Interval	0.008000	0.004000	0.002667	0.002000	0.001600
Gateway	1	1	1	1	1
Start Time (sec)	5.00	5.00	5.00	5.00	5.00
Stop Time (sec)	45.00	45.00	45.00	45.00	45.00
Data send (Packets)	5,000.00	10001	15000	20000	25000
Data Receive (Packets)	5,000.00	6443	6442	6443	6443
Avg. Throughput (Kbps) Source Node	12.50	24.81	37.22	49.61	62.02
Avg. Throughput (Kbps) Destination Node	15.00	19.18	19.18	19.18	19.18
Avg. Throughput Ratio	120%	77%	52%	39%	31%
Avg. Delay (ms)	7.02	308.77	310.33	310.79	311.05
Packet Delivery Ratio (%)	100%	64%	43%	32%	26%
Packet Drops (Packets)	-	3558	8558	13557	18557

ภาพที่ 3.12 ข้อมูลใน trace file หลังจากแบ่งข้อมูลแต่ละ column แล้ว



ภาพที่ 3.13 กราฟของค่าเฉลี่ยการหน่วงเวลาแบบจุดต่อจุด

จากภาพที่ 3.13 เป็นกราฟที่แสดงค่าเฉลี่ยการหน่วงเวลาแบบจุดต่อจุด โดยเริ่มจากโหนดต้นทาง R0 ไปยังโหนดปลายทาง R8 โดยมีการเปลี่ยนแปลงปริมาณ packet CBR ที่ส่งเข้ามาในเครือข่ายเซนเซอร์ไร้สายที่มีตัว Coordinator/Gateway Node เพียง 1 ตัว ในแกนนอนแสดงให้เห็นถึงปริมาณ ความหนาแน่นของ packet CBR ที่เวลาเท่ากันคือ 40 วินาที เมื่อความหนาแน่นของ packet CBR มีปริมาณสูงขึ้น จะทำให้เกิดค่าเฉลี่ยการหน่วงเวลาแบบจุดต่อจุด มากขึ้นตาม

3.3.5 สร้างทอพอโลยี ของ WSN แบบ 2 gateways

เพิ่มตัว Gateway ตัวที่ 2 เข้าไปใน WSN เพื่อให้เห็นถึงความแตกต่าง จึงทำการจำลองตัวเครือข่ายเซนเซอร์ไร้สายที่มี Gateway 1 ตัว โดยใช้ตัวโพรโทคอล AODV แล้วใส่รูปแบบของข้อมูลที่เป็น CBR (constant bit rate) โดยจะทำการปรับเปลี่ยนปริมาณความหนาแน่นของข้อมูลที่ส่งเข้าไปยังตัวโหนดต้นทาง R0 โดยมีปริมาณความหนาแน่นของข้อมูลต่อเวลาดังนี้

ตารางที่ 3.5 ค่าปริมาณความหนาแน่นของข้อมูลต่อเวลา

ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ครั้งที่ 4	ครั้งที่ 5
5000 packets	10000 packets	15000 packets	20000 packets	25000 packets
125 packets/sec	250 packets/sec	375 packets/sec	500 packets/sec	625 packets/sec

ในการกำหนดค่าของปริมาณความหนาแน่นของข้อมูลในส่วนของโปรแกรมจะสามารถทำได้โดยกำหนด ค่าของ CBR Packet Interval ในส่วนของ Otcl script ตอนกำหนดรายละเอียดของข้อมูล CBR โดยมีรายละเอียดดังนี้

ตารางที่ 3.6 ค่าที่ใช้ในการกำหนดค่า Packet Interval

CBR Interval 1	CBR Interval 2	CBR Interval 3	CBR Interval 4	CBR Interval 5
0.008	0.004	0.002667	0.002	0.0016

ตัวอย่างคำสั่งที่ใช้ในการกำหนดค่าชนิดข้อมูลที่เป็น CBR ในการกำหนดค่านั้นจะต้องมีการกำหนดค่าของ Packet Size : 100 และค่า Packet Interval เมื่อกำหนดเรียบร้อยแล้วก็ทำการเชื่อมต่อ CBR packet เข้ากับตัว Agent “udp_src”

```
#Setup CBR traffic
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp_src
$cbr set packetSize_ 100 #(CBR Packet is 100 byte)
$cbr set interval_ 0.008 #ใช้ในการกำหนดค่า Packet Interval
```

ภาพที่ 3.14 ตัวอย่างการกำหนดค่าของ CBR Packet

3.3.6 ประเมินผลการจำลองเบื้องต้นใน WSN

ในประเมินผลการวิจัยนี้ จะต้องหาตัวแปรต่างๆ โดยค่าตัวแปรที่ต้องการหาหลังจากจำลองการทำงานของเครือข่ายเซนเซอร์ไร้สาย จะประกอบไปด้วยตัวแปรต่างๆ ดังนี้

1) จำนวน CBR Packet ที่โหนดต้นทาง ในการจำลองเครือข่ายนี้คือ R0 โดยสามารถเขียนเป็นซูโดโค้ด (Pseudo Code) ได้ดังนี้

```
ALGORITHM SOURCE_CBR (NODE, TYPE, TOTAL_DATA)
INPUT NODE          #NODE NUMBER
INPUT TYPE          #TYPE OF DATA
INPUT TOTAL_DATA    #NUMBER OF LINE
OUTPUT SOURCE_CBR   #TOTAL CBR_PACKET AT SOURCE NODE
SOURCE_CBR = 0
FOR I=0, I=TOTAL_DATA, I++
    IF NODE = R0 AND TYPE = CBR THEN
            SOURCE_CBR = SOURCE_CBR + 1
            NEXT
    ELSE NEXT
END FOR
RETURN (SOURCE_CBR)
```

ภาพที่ 3.15 ซูโดโค้ด (Pseudo Code) การหาจำนวน CBR Packet ที่โหนดต้นทาง

2) จำนวน CBR Packet ที่โหนดปลายทาง ในการจำลองเครือข่ายนี้คือ R8 โดยสามารถเขียนเป็นซูโดโค้ด (Pseudo Code) ได้ดังนี้

```
ALGORITHM DESTINATION_CBR (NODE, TYPE, TOTAL_DATA)
INPUT NODE              #NODE NUMBER
INPUT TYPE              #TYPE OF DATA
INPUT TOTAL_DATA        #NUMBER OF LINE
OUTPUT DESTINATION_CBR  #TOTAL CBR_PACKET AT DESTINATION NODE
DESTINATION_CBR = 0
FOR I=0, I=TOTAL_DATA, I++
    IF NODE = R8 AND TYPE = CBR THEN
            DESTINATION_CBR = DESTINATION_CBR + 1
            NEXT
    ELSE NEXT
END FOR
RETURN (DESTINATION_CBR)
```

ภาพที่ 3.16 ซูโดโค้ด (Pseudo Code) การหาจำนวน CBR Packet ที่โหนดปลายทาง

3) ค่าเฉลี่ย CBR Packet ต่อเวลา ที่โหนดต้นทาง ในการจำลองเครือข่ายนี้คือ R0 โดยสามารถเขียนเป็นซูโดโค้ด (Pseudo Code) ได้ดังนี้

```
ALGORITHM SOURCE_AVG_CBR (NODE[],TYPE[],TOTAL_DATA[],TIME,TRACE_LEVEL[],PACKET_SIZE[])
INPUT NODE              #NODE NUMBER
INPUT TYPE              #TYPE OF DATA
INPUT TOTAL_DATA        #NUMBER OF LINE
INPUT TIME              #TOTAL TIME "SEC"
INPUT TRACE_LEVEL       #POINT OF MEASUREMENT "AGT"
INPUT PACKET_SIZE       #PACKET_SIZE "BYTE"
OUTPUT SOURCE_AVG_CBR   # AVERAGE CBR_PACKET AT SOURCE NODE
SOURCE_AVG_CBR = 0
FOR I=0, I=TOTAL_DATA, I++
    IF NODE[I] = R0 AND TYPE[I] = CBR AND TRACE_LEVEL = AGT THEN
            SOURCE_AVG_CBR = SOURCE_AVG_CBR + PACKET_SIZE[I]
            NEXT
    ELSE NEXT
END FOR
SOURCE_AVG_CBR = SOURCE_AVG_CBR/TIME
RETURN (SOURCE_AVG_CBR)
```

ภาพที่ 3.17 ซูโดโค้ด (Pseudo Code) การหาค่าเฉลี่ย CBR Packet ต่อเวลา ที่โหนดต้นทาง

4) ค่าเฉลี่ย CBR Packet ต่อเวลา ที่โหนดปลายทาง ในการจำลองเครือข่ายนี้คือ R8 โดยสามารถเขียนเป็นซูโดโค้ด (Pseudo Code) ได้ดังนี้

```
ALGORITHM DESTINATION_AVG_CBR (NODE[],TYPE[],TOTAL_DATA[],TIME,TRACE_LEVEL[],PACKET_SIZE[])
INPUT NODE              #NODE NUMBER
INPUT TYPE              #TYPE OF DATA
INPUT TOTAL_DATA        #NUMBER OF LINE
INPUT TIME              #TOTAL TIME "SEC"
INPUT TRACE_LEVEL       #POINT OF MEASUREMENT "AGT"
INPUT PACKET_SIZE       #PACKET_SIZE "BYTE"
OUTPUT DESTINATION_AVG_CBR      # AVERAGE CBR_PACKET AT DESTINATION NODE
DESTINATION_AVG_CBR = 0
FOR I=0, I=TOTAL_DATA, I++
    IF NODE[I] = R8 AND TYPE[I] = CBR AND TRACE_LEVEL = AGT THEN
            DESTINATION_AVG_CBR = DESTINATION_AVG_CBR + PACKET_SIZE[I]
            NEXT
    ELSE NEXT
END FOR
DESTINATION_AVG_CBR = DESTINATION_AVG_CBR/TIME
RETURN (DESTINATION_AVG_CBR)
```

ภาพที่ 3.18 ซูโดโค้ด (Pseudo Code) การหาค่าเฉลี่ย CBR Packet ต่อเวลา ที่โหนดปลายทาง

5) ค่าสัดส่วนของค่าเฉลี่ย CBR Packet ต่อเวลาระหว่างที่โหนดปลายทาง ในการจำลองเครือข่ายนี้คือ R8 เทียบกับ โหนดต้นทาง ในการจำลองเครือข่ายนี้คือ R0

```
ALGORITHM THROUGHPUT_RATIO (SOURCE_AVG_CBR, DESTINATION_AVG_CBR)
INPUT SOURCE_AVG_CBR            # AVERAGE CBR_PACKET AT SOURCE NODE
INPUT DESTINATION_AVG_CBR       # AVERAGE CBR_PACKET AT DESTINATION NODE
OUTPUT THROUGHPUT_RATIO         # AVERAGE CBR_PACKET AT DESTINATION NODE
THROUGHPUT_RATIO = 0
      THROUGHPUT_RATIO = (DESTINATION_AVG_CBR/SOURCE_AVG_CBR)*100
RETURN (THROUGHPUT_RATIO)
```

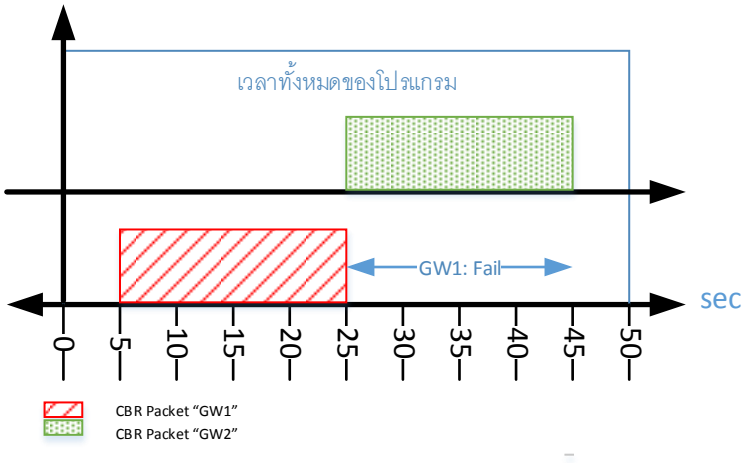
ภาพที่ 3.19 ซูโดโค้ด (Pseudo Code) การหาค่าค่าสัดส่วนของค่าเฉลี่ย CBR Packet ต่อเวลาระหว่างโหนดต้นทาง กับโหนดปลายทาง

6) จำนวน CBR Packet ที่ถูก Drop ที่โหนดปลายทาง ในการจำลองเครือข่ายนี้คือ R8

7) ค่าเฉลี่ยการหน่วงเวลาแบบ End-to-End delay ของ CBR Packet ระหว่างโหนดต้นทาง ในการจำลองเครือข่ายนี้คือ R0 กับโหนดปลายทาง ในการจำลองเครือข่ายนี้คือ R8

8) ค่าสัดส่วนของการส่งข้อมูลสำเร็จ PDR% ระหว่างโหนดต้นทาง ในการจำลองเครือข่ายนี้คือ R0 กับโหนดปลายทาง ในการจำลองเครือข่ายนี้คือ R8

เมื่อทำการจำลองการทำงานของ WSN แบบ 1 Gateway และได้ค่าตัวแปรที่ต้องการทราบค่าทั้งหมดแล้ว ก็จะทำการจำลองการทำงานของ WSN แบบ 2 Gateway โดยมีการปรับค่าในส่วนของเวลาที่ส่งข้อมูล CBR packet ไปยังโหนดปลายทาง ในการจำลองเครือข่ายนี้คือ R8 จากเดิมจะเริ่มส่งที่เวลา 5 วินาที และหยุดส่งที่เวลา 45 วินาที รวมเป็น 40 วินาที ให้มาเป็น ส่งเริ่มที่เวลา 5 วินาที และหยุดส่งที่เวลา 25 วินาที รวมเป็น 20 วินาที จากนั้นให้หยุดการทำงานของโหนดปลายทาง ในการจำลองเครือข่ายนี้คือ R8 และเปลี่ยนโหนดปลายทางใหม่ ในการจำลองเครือข่ายนี้คือ R9 และเริ่มส่งข้อมูล CBR packet ไปยังโหนดปลายทาง R9 ใหม่ที่เวลา 25 วินาที และหยุดส่งที่เวลา 45 วินาที รวมเป็นเวลา 20 วินาที เมื่อนำเวลาที่ส่งข้อมูลไปที่โหนด R8 และ R9 รวมกัน จะได้ 40 วินาที ซึ่งเท่ากับตอนที่มี โหนดปลายทางเดียว โดยการโหนดปลายทางนี้ เพื่อเป็นการจำลองในการทำงานของเครือข่ายในกรณีที่ มี Gateway อยู่ 2 ตัว เมื่อทำงานได้สักระยะแล้ว ตัว Gateway 1 มีปัญหาไม่สามารถรับข้อมูลได้ จึงจำเป็นต้องย้าย Gateway จาก Gateway 1 ไปเป็น Gateway 2 เพื่อส่งข้อมูลออกนอกเครือข่ายแทน รายละเอียดช่วงเวลาในการทำงานสามารถแสดงได้ดังภาพที่ 3.20



ภาพที่ 3.20 ช่วงเวลาในส่ง CBR Packet กรณีที่มี 2 Gateways

```
#Setup CBR1 for 1 Gateways
set cbr1 [new Application/Traffic/CBR]
$ns at $val(cbr_start) "$cbr1 attach-agent $udp_src1"
$cbr1 set packetSize_ 100 #(CBR Packet is 100 byte)
$cbr1 set rate_ 0.1Mb # (Unit bit) (1-100000: No drop) Step 1=0.1Mb, Step 2=0.2Mb
$cbr1 set random_ false

#Setup CBR2 for 2 Gateways
set cbr2 [new Application/Traffic/CBR]
$ns at $val(cbr_start) "$cbr2 attach-agent $udp_src2"
$cbr2 set packetSize_ 100 #(CBR Packet is 100 byte)
$cbr2 set rate_ 0.1Mb # (Unit bit) (1-100000: No drop) Step 1=0.1Mb, Step 2=0.2Mb
$cbr2 set random_ false

#Schedule Time to sent CBR packets for 2 Gateways
$ns at 5.0 "$cbr1 start"
$ns at 15.0 "$cbr1 stop"

$ns at 15.0 "$cbr2 start"
$ns at 25.0 "$cbr2 stop"

$ns at 25.0 "$cbr1 start"
$ns at 35.0 "$cbr1 stop"

$ns at 35.0 "$cbr2 start"
$ns at 45.0 "$cbr2 stop"
```

ภาพที่ 3.21 การปรับปรุงโปรแกรมกรณีที่มี 2 Gateways

ในภาพที่ 3.21 แสดงส่วนโปรแกรมที่ทำการปรับปรุง เพื่อให้สามารถส่งข้อมูล packets CBR ไปยัง Gateways ทั้ง 2 ตัว ตามช่วงเวลาที่กำหนดในภาพที่ 3.20 เพื่อจำลองให้เห็นความแตกต่างของ WSN ในกรณีที่มี 1 Gateway กับ 2 Gateway โดยอาศัยโพรโทคอล AODV

จากตารางที่ 3.7 ค่า Throughput และค่า PDR จะเห็นได้ว่าเมื่อจำนวน CBR packets มีปริมาณมากขึ้นจะทำให้ Throughput และค่า PDR มีค่าที่น้อยลง เนื่องจากตัวโหนดไม่สามารถรองรับปริมาณข้อมูลที่มีจำนวนมากขึ้น ดังจะเห็นได้จากข้อมูลที่รับได้ในโหนด R8 ขณะที่มีจำนวน CBR packets ตั้งแต่ 7000 ขึ้นไปมีค่าเท่าเดิม และเมื่อพิจารณาในส่วนของจำนวน Packets Drops จะเห็นว่ามีจำนวนมากขึ้นเรื่อยๆ อันเป็นผลมาจากการที่โหนด R8 หยุดทำงาน

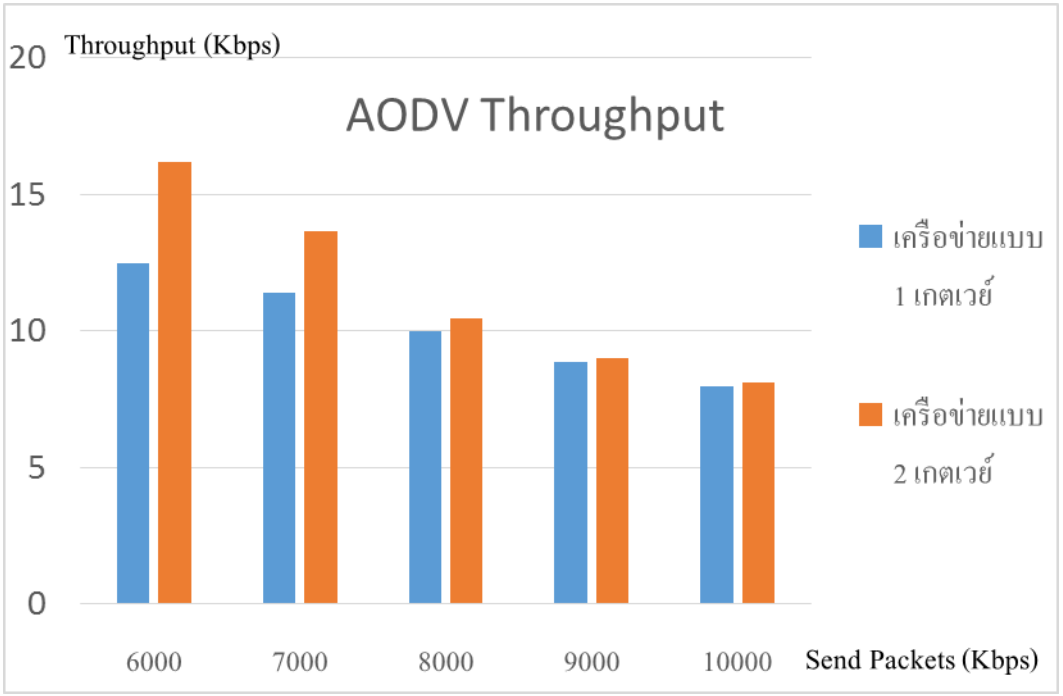
ตารางที่ 3.7 ผลการทดลองเบื้องต้นใน WSN แบบ 1 เกตเวย์

Parameter	Sending Data (Packets)				
	6000	7000	8000	9000	10000
Simulate time (s)	40	40	40	40	40
Data Receive R8 (Packets)	2999	3192	3192	3192	3192
Avg. Throughput R8 (Kbps)	12.50	11.40	9.98	8.87	7.98
Packet Delivery Ratio (R8) (%)	49.98	45.60	39.90	35.47	31.92
Packet Drops (R8) (Packets)	3001	3808	4808	5808	6808

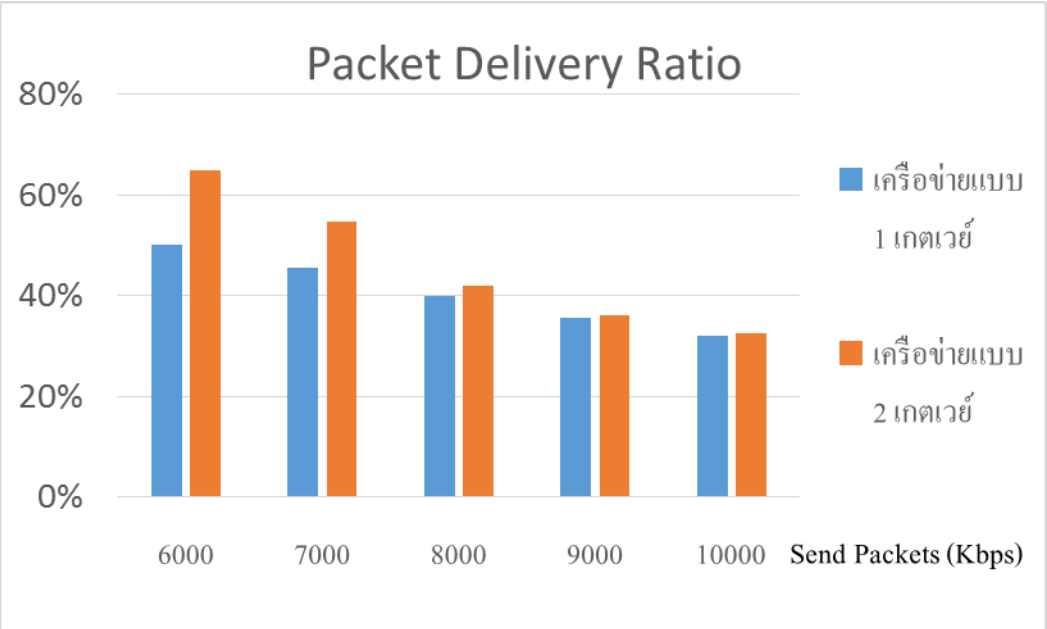
จากตารางที่ 3.8 ผลการทดลองเบื้องต้นใน WSN แบบ 2 เกตเวย์ โดยผลที่ได้เมื่อมีการหาค่าปริมาณ Throughput และค่า PDR จะเห็นได้ว่าที่จำนวนปริมาณ CBR packets ต่างๆ จะได้ปริมาณค่าของ Throughput และค่า PDR ของ WSN แบบ 2 เกตเวย์ มีค่ามากกว่าเมื่อเทียบกับแบบเครือข่ายที่มี 1 เกตเวย์ เนื่องจากตัวโหนด R9 ที่ทำหน้าที่เป็นเกตเวย์ 2 สามารถช่วยรองรับปริมาณข้อมูลที่มีจำนวนมากขึ้นได้

ตารางที่ 3.8 ผลการทดลองเบื้องต้นใน WSN แบบ 2 เกตเวย์

Parameter	Sending Data (Packets)				
	6000	7000	8000	9000	10000
Simulate time (s)	40	40	40	40	40
Data Received R8 (Packets)	2999	3192	3192	3192	3192
Data Received R9 (Packets)	887	634	159	54	54
Avg. Throughput R8+R9 (Kbps)	16.19	13.66	10.47	9.02	8.12
Packet Delivery Ratio R8+R9 (%)	64.77	54.66	41.89	36.07	32.46
Packet Drops R8+R9 (Packets)	2114	3174	4649	5754	6754



ภาพที่ 3.22 ค่าเปรียบเทียบ Throughput ระหว่าง WSN แบบ 1 เกตเวย์ กับ 2 เกตเวย์



ภาพที่ 3.23 ค่า PDR ระหว่าง WSN แบบ 1 เกตเวย์ กับ 2 เกตเวย์

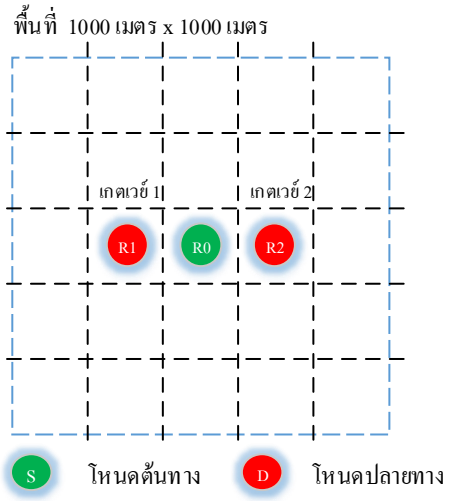
จากกราฟในภาพที่ 3.22 และ 3.23 แสดงการเปรียบเทียบค่า Throughput และค่า PDR ของเครือข่ายแบบ 1 และ 2 เกตเวย์ ซึ่งจะเห็นได้ว่าเครือข่ายแบบ 2 เกตเวย์ สามารถทำได้สูงกว่าแบบเครือข่าย 1 เกตเวย์ ซึ่งปริมาณ Throughput ที่การส่ง Packets 6000 มีค่ามากกว่าที่ส่ง Packets 10000 นั้น เนื่องจากถ้าดูตารางที่ 3.8 พบว่าจำนวน Packets Drop ของเกตเวย์ 2 มีค่ามากขึ้นเมื่อมีการส่งข้อมูลมากขึ้น เนื่องจากเกตเวย์ 2 อยู่ไกลสุด ทำให้ Buffer ที่โหนดระหว่างทางเต็ม มีผลทำให้ Packets ที่ถูกส่งมาใหม่ถูก Drop ไป แต่โดยรวมของการจำลองการส่ง CBR packets ที่ค่าต่างๆ พบว่าในเครือข่าย 2 เกตเวย์ สามารถทำได้สูงกว่าแบบเครือข่าย 1 เกตเวย์ โดยการใช้โพรโทคอลเส้นทาง AODV และโหนดภายในเครือข่ายไม่มีการเคลื่อนที่ แต่ในกรณีมีความหนาแน่นของข้อมูลในเครือข่ายจำนวนมาก ส่วนต่างของค่า Throughput ของเครือข่าย และค่า PDR นั้นจะมีค่าไม่สูงมาก โดยพบว่าค่า Throughput มีค่าสูงขึ้นเฉลี่ย 1.35 Kbps และค่า Packet Delivery Ratio (PDR) ของ WSN แบบ 2 เกตเวย์ มีค่าสูงขึ้นเฉลี่ย 5.39% เมื่อเทียบกับ WSN แบบเดิมที่มี 1 เกตเวย์

จากการทดลองเบื้องต้นจะเห็นได้ว่า เครือข่าย WSN แบบ 2 เกตเวย์ สามารถเพิ่มปริมาณ Throughput ของเครือข่ายได้ เมื่อเทียบกับ WSN แบบ 1 เกตเวย์ หลังจากการทดลองเบื้องต้นแล้วได้มีการกำหนดเงื่อนไขในการทดลองเพิ่มเติมและมีการปรับปรุงตัวโพรโตคอล AODV เพิ่มเติมเพื่อให้สามารถทำงานบน WSN แบบเกตเวย์หลายตัวได้

หลังจากได้ผลการทดลองเบื้องต้นแล้ว ได้ทำการปรับปรุงการทดลองเครือข่ายเซนเซอร์ไร้สาย Wireless Sensor Network (WSN) แบบเกตเวย์หลายตัว เพื่อให้ครอบคลุมรูปแบบการใช้งานจริงของ WSN และลดปัญหาการเกิดการ drop ของข้อมูลระหว่างทาง โดยมุ่งเน้นการทดลองเพื่อหาค่าพลังงานที่ใช้ในเกตเวย์สำรอง และเวลาที่ใช้ในการเปลี่ยนเกตเวย์ รวมถึงแสดงให้เห็นถึงประสิทธิภาพจากการเปรียบเทียบการทำงานของ WSN แบบ 2 เกตเวย์ กับ แบบ 1 เกตเวย์

3.3.7 กำหนดรูปแบบเครือข่าย WSN ใหม่ที่ใช้ในการทดลองโพรโตคอล AODV

เครือข่าย WSN ที่ใช้ในการทดลองนี้ประกอบไปด้วยโหนดเกตเวย์ 2 โหนด ซึ่งมีโหนด R0 ทำหน้าที่เป็นโหนดต้นทาง โหนด R1 ทำหน้าที่เป็นเป็นเกตเวย์หลัก โหนด R2 ทำหน้าที่เป็นเกตเวย์สำรอง รายละเอียดแสดงให้เห็นดังรูปที่ 3.24



ภาพที่ 3.24 เครือข่ายเซนเซอร์ไร้สาย แบบ 2 เกตเวย์ ที่ใช้ในการทดลอง

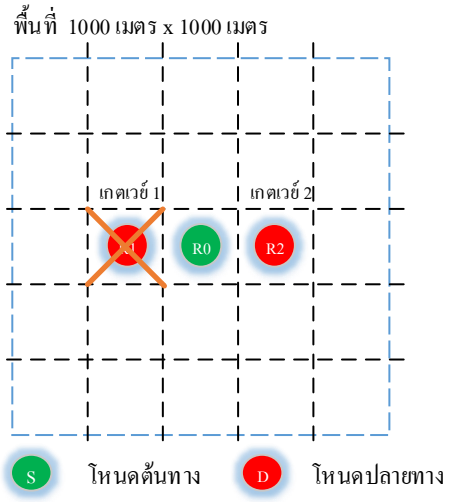
ในการทดลองมีการส่งข้อมูลจากโหนดต้นทางไปยังเกตเวย์ โดยใช้ชนิดของข้อมูลแบบ CBR ขนาด 100 Byte และมีการปรับเปลี่ยนปริมาณของข้อมูลที่ส่งเข้าไป ในช่วงเวลาที่กำหนด โดยมีรายละเอียดของลำดับข้อมูลที่ใช้ในการทดลองตามตารางที่ 3.9

ตารางที่ 3.9 จำนวน packets ที่ใช้ในการทดลอง

Time (sec)	40	40	40	40	40	40
Packets	3000	4000	5000	6000	7000	8000

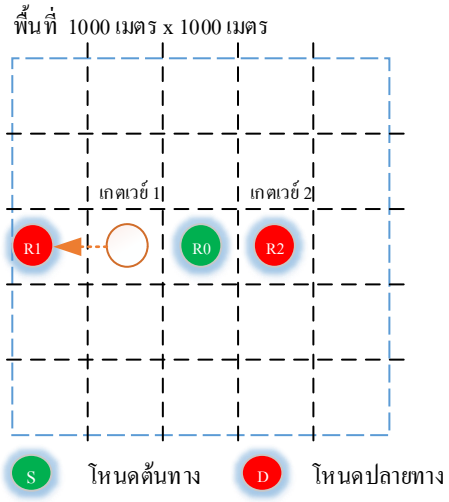
เพื่อให้การทดลองครอบคลุมรูปแบบการใช้งานจริงของเครือข่าย WSN ที่จะเกิดปัญหาที่ตัวเกตเวย์ จึงมีการกำหนดรูปแบบของการเกิดปัญหาที่ตัวเกตเวย์เดิมไว้ 2 แบบ ดังนี้

1. เมื่อเกตเวย์ 1 ไม่สามารถติดต่อได้ เนื่องจากรับความเสียหาย แบบทันทีทันใด



ภาพที่ 3.25 เครือข่ายที่ใช้ทดลองแบบเกตเวย์ 1 ไม่สามารถติดต่อได้ เนื่องจากรับความเสียหาย

2. เมื่อเกตเวย์ 1 ไม่สามารถติดต่อได้ เนื่องจากมีการเคลื่อนที่ออกจากเครือข่าย



ภาพที่ 3.26 เครือข่ายที่ใช้ทดลองแบบเกตเวย์ 1 ไม่สามารถติดต่อได้ เนื่องจากเคลื่อนที่ออกจากเครือข่าย

ในการทดลองจะมีการกำหนดเวลาที่เกิดการเสียหายที่ตัวเกตเวย์ _1_ ไว้ที่ วินาทีที่ 25 และมีการเคลื่อนที่ของเกตเวย์ _1_ ออกจากเครือข่าย จากตำแหน่ง Y: 250m, X: 250m มาเป็น Y: 0.1m, X: 250m ด้วยความเร็ว 100m/s หลังการกำหนดเวลาในการทดสอบแล้ว จะทำการทดสอบโดยเริ่มส่ง packets ที่เวลา 5 วินาที และหยุดที่เวลา 45 วินาที

ในการทดลองนี้ได้มีการปรับปรุงตัวโพรโทคอล AODV ให้มีการแสดงเวลาที่โหนดต้นทางได้รับ Message RERR เมื่อโหนดปลายทางเสียหาย เพื่อจะใช้เป็นเวลาที่จะเปลี่ยนตัวเกตเวย์จากตัวที่ 1 เป็นตัวที่ 2 โดยมีส่วนที่ปรับเปลี่ยนใน AODV.CC เพื่อตรวจสอบ Message RERR เมื่อเกตเวย์ปลายทาง ที่โหนดต้นทางส่งข้อมูลไป ไม่สามารถรับข้อมูล และทำการเพิ่ม Time Stamp และ Error Status เพื่อให้ตัวโหนดต้นทางรับรู้ เพื่อจะได้ทำการเปลี่ยนเกตเวย์ได้ โดยมีปรับเปลี่ยนตามภาพที่ 3.27

```
1238  void
1239 AODV::sendError(Packet *p, bool jitter) {
1240   gw_ = gw_ + 1; //cnd
1241   gw_chgtime_ = CURRENT_TIME; //cnd
1242
1243   printf("Send error counting: %d \n",gw_);
1244   printf("Current time to send error: %f \n",CURRENT_TIME);
```

ภาพที่ 3.27 ส่วนที่ปรับปรุงเพิ่มเติมใน AODV.CC

ในส่วนของการทำงานของ NS2 จะมีการทำงานของภาษา C++ ภาษา OCTL และภาษา TCL ที่เป็นเช่นนี้เนื่องจาก ตัวโครงสร้างของภาษา C++ จะทำงานได้เร็ว จึงถูกใช้ในส่วนของการทำงานของโพรโทคอล ที่มีเรื่องของเวลาเข้ามาเกี่ยวข้อง แต่ในส่วนของการปรับเปลี่ยนค่าพารามิเตอร์ต่างๆ ในการควบคุมการทำงานของตัวโหนดนั้น จะใช้ส่วนของตัว OTCL และ TCL ในการส่งข้อมูลให้เนื่องจากง่ายต่อการใช้งาน ดังนั้นในส่วนนี้จึงจำเป็นต้องสร้างตัวแปรที่ใช้ในการเชื่อมโยง แลกเปลี่ยนข้อมูล ระหว่าง ภาษา C++ ภาษา OCTL และTCL ซึ่งในที่นี้จะเป็นการสร้างตัวแปร “gw_chgtime”_ และ “gw_” ซึ่งเป็นตัวแปรประเภททศนิยม กับตัวเลข ตามลำดับ เพื่อส่งข้อมูลให้กับตัวโหนดต้นทาง ในการตัดสินใจในการเปลี่ยนเกตเวย์ ซึ่งส่วนที่เป็นการปรับปรุงใน AODV.CC เพื่อเชื่อมโยงข้อมูลมีรายละเอียดตาม ภาพที่ 3.28

```
142  AODV::AODV(nsaddr_t id) : Agent(PT_AODV),
143                            btimer(this), htimer(this), ntimer(this),
144                            rtimer(this), lrtimer(this), rqueue() {
145
146    bind("gw_", &gw_); //cnd
147    bind("gw_chgtime_", &gw_chgtime_); //cnd
```

ภาพที่ 3.28 ส่วนที่ปรับปรุงเพิ่มเติมใน AODV.CC เพื่อเชื่อมโยงข้อมูล

ในการทดลองนี้ได้มีการปรับปรุงตัวโพรโทคอล AODV ให้มีการแสดงเวลาที่โหนดต้นทางได้รับ Message RERR เมื่อโหนดปลายทางเสียหาย เพื่อจะใช้เป็นเวลาที่จะเปลี่ยนตัวเกตเวย์จากตัวที่ 1 เป็นตัวที่ 2 โดยมีส่วนที่ปรับเปลี่ยนใน AODV.CC เพื่อตรวจสอบ Message RERR เมื่อเกตเวย์ปลายทาง ที่โหนดต้นทางส่งข้อมูลไป ไม่สามารถรับข้อมูล และทำการเพิ่ม Time Stamp และ Error Status เพื่อให้ตัวโหนดต้นทางรับรู้ เพื่อจะได้ทำการเปลี่ยนเกตเวย์ได้ โดยมีปรับเปลี่ยนตามภาพที่ 3.29

```
1238  void
1239  AODV::sendError(Packet *p, bool jitter) {
1240    gw_ = gw_ + 1; //cnd
1241    gw_chgtime_ = CURRENT_TIME; //cnd
1242
1243    printf("Send error counting: %d \n",gw_);
1244    printf("Current time to send error: %f \n",CURRENT_TIME);
```

ภาพที่ 3.29 ส่วนที่ปรับปรุงเพิ่มเติมใน AODV.CC

ในส่วนของการทำงานของ NS2 จะมีการทำงานของภาษา C++ ภาษา OCTL และภาษา TCL ที่เป็นเช่นนี้เนื่องจาก ตัวโครงสร้างของภาษา C++ จะทำงานได้เร็ว จึงถูกใช้ในส่วนของการทำงานของโพรโทคอล ที่มีเรื่องของเวลาเข้ามาเกี่ยวข้อง แต่ในส่วนของการปรับเปลี่ยนค่าพารามิเตอร์ต่างๆ ในการควบคุมการทำงานของตัวโหนดนั้น จะใช้ส่วนของตัว OTCL และ TCL ในการส่งข้อมูลให้เนื่องจากง่ายต่อการใช้งาน ดังนั้นในส่วนนี้จึงจำเป็นต้องสร้างตัวแปรที่ใช้ในการเชื่อมโยง แลกเปลี่ยนข้อมูล ระหว่าง ภาษา C++ ภาษา OCTL และTCL ซึ่งในที่นี้จะเป็นการสร้างตัวแปร “gw_chgtime”_ และ “gw_” ซึ่งเป็นตัวแปรประเภททศนิยม กับตัวเลข ตามลำดับ เพื่อส่งข้อมูล

ให้กับตัวโหนดต้นทาง ในการตัดสินใจในการเปลี่ยนเกตเวย์ ซึ่งส่วนที่เป็นการปรับปรุงใน AODV.CC เพื่อเชื่อมโยงข้อมูลมีรายละเอียดตาม ภาพที่ 3.30

```
142  AODV::AODV(nsaddr_t id) : Agent(PT_AODV),
143                btimer(this), htimer(this), ntimer(this),
144                rtimer(this), lrtimer(this), rqueue() {
145
146    bind("gw_", &gw_); //cnd
147    bind("gw_chgtime_", &gw_chgtime_); //cnd
```

ภาพที่ 3.30 ส่วนที่ปรับปรุงเพิ่มเติมใน AODV.CC เพื่อเชื่อมโยงข้อมูล

จากภาพที่ 3.29 การเพิ่มและเชื่อมโยงตัวแปร “gw_chgtime”_ และ “gw_” เมื่อโปรแกรมเริ่มทำงาน ในกรณีที่โหนดมีการส่ง Message RERR ก็จะเข้ามาทำงานในส่วนของโปรแกรมตามภาพที่ 3.25 ซึ่งจะทำการเพิ่มค่า “gw_” ขึ้น และทำการเก็บค่าเวลาขณะนั้น เพื่อใช้เป็นเวลาที่จะเริ่มทำการเปลี่ยนเกตเวย์ในส่วนของ TCL ซึ่งผลจากการทดลอง ได้มีการเก็บค่าเวลาที่จะทำเริ่มทำการเปลี่ยนเกตเวย์ ที่ปริมาณข้อมูล ตามตารางที่ 3.9

ในส่วนของการทำงานของตัวโปรแกรมที่ใช้ภาษา TCL เพื่อกำหนดการทำงานและเงื่อนไขต่างๆ บนเครือข่ายนั้น จะมีการกำหนดเงื่อนไขในการเปลี่ยนเกตเวย์ในเครือข่าย โดยการพัฒนาโปรแกรมตามรายละเอียดในภาพที่ 3.31 และ 3.32

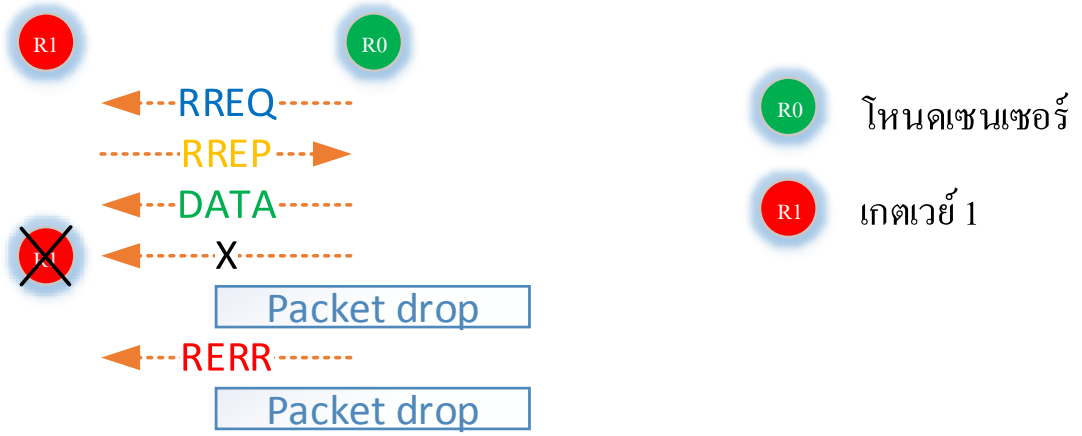
```
 98 #Setup CBR1
 99 set cbr1 [new Application/Traffic/CBR]
100 $ns at 5.0 "$cbr1 attach-agent $udp_src1"
101 $cbr1 set packetSize_ 100 #(CBR Packet is 100 byte)
102 $cbr1 set rate_ 0.06Mb # (Unit bit)
103 $cbr1 set random_ false
104 $ns at 5.0 "$cbr1 start"
105 $ns at 45.0 "$cbr1 stop"
106
107 #Schedule Time to take action
108 $ns at $val(gw1_action_) "$n(1) setdest 0.1 250.0 100"
109 $ns at $val(gw1_action_) "$n(1) node-down"
```

ภาพที่ 3.31 ส่วนของโปรแกรมที่พัฒนาโดยภาษา TCL เพื่อสร้าง packets CBR

```
121 for {set i 1} {$i < $val(stop)} { incr i } {
122 if { $val(gw_) == 1 } {
123         $ns at $val(gw_chgtime_) "$ns attach-agent $n(2) $udp_des2"
124         $ns at $val(gw_chgtime_) "$ns connect $udp_src1 $udp_des2"
125 }
126 }
```

ภาพที่ 3.32 ส่วนของโปรแกรมที่พัฒนาโดยภาษา TCL เพื่อกำหนดเงื่อนไขในการเปลี่ยนเกตเวย์

ในส่วนของการเปลี่ยนเกตเวย์จะอาศัยเงื่อนไขเมื่อเกตเวย์หลักมีการส่ง Message RERR ออกไป เมื่อเกตเวย์หลักไม่สามารถทำงานได้ โดยแสดงในภาพที่ 3.33

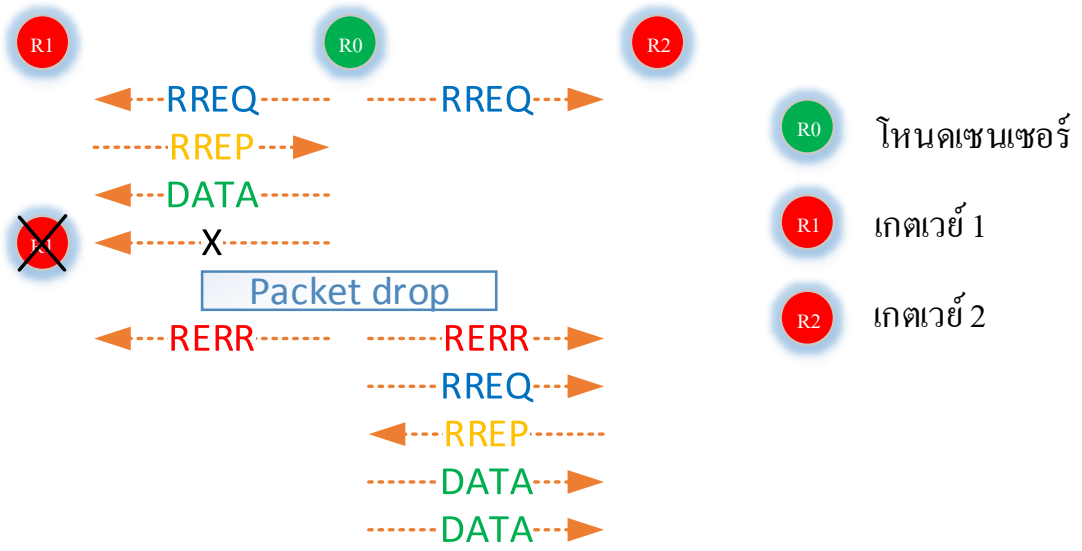


ภาพที่ 3.33 การส่ง Message RERR ออกไป เมื่อเกตเวย์หลักไม่สามารถทำงานได้

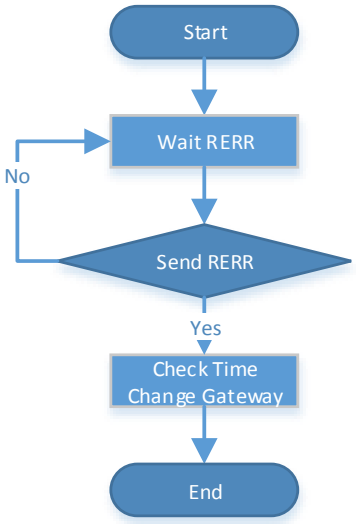
ในภาพที่ 3.33 เป็นการทำงานของโพรโทคอล AODV แบบเดิม ซึ่งเมื่อโหนดต้นทางต้องการส่งข้อมูลไปให้เกตเวย์หลัก ก็จะมีการส่ง Message RREQ ออกไปเพื่อร้องขอเส้นทาง เมื่อเกตเวย์หลักได้รับ Message RREQ แล้วก็จะมีการตอบกลับ Message RREP ไปยังเส้นทางที่ดีที่สุด เมื่อโหนดต้นทางได้รับ Message RREP ก็จะเริ่มกระบวนการส่งข้อมูล เมื่อเกตเวย์หลักไม่สามารถทำงานได้หรือเส้นทางไปยังเกตเวย์หลักไม่สามารถใช้งานได้ ตัวโหนดต้นทางก็จะส่ง Message RERR ออกไปยังโหนดข้างเคียงเพื่อร้องขอเส้นทางใหม่ที่ไปยังเกตเวย์หลักอีกครั้ง ซึ่งในกรณีนี้จะไม่สามารถร้องขอเส้นทางใหม่ได้ เนื่องจากเกตเวย์หลักเสียหาย ดังนั้นข้อมูลที่เหลือก็จะถูก drop ไป

ในภาพที่ 3.34 เป็นการทำงานของโพรโทคอล AODV แบบใหม่ที่ใช้กับ WSN แบบเกตเวย์หลายตัว ซึ่งจะอาศัยจังหวะที่โหนดต้นทางส่ง Message RERR ออกไป โดยโหนดต้นทางจะมี

เกตเวย์สำรองอยู่ในรายการแล้ว ก็จะเปลี่ยนไปใช้เกตเวย์สำรองทันทีโดยเริ่มกระบวนการ RREQ ใหม่ แต่ใช้โหนดปลายทางเป็นเกตเวย์สำรองแทน โดยภาพที่ 3.35 เป็นลำดับขั้นตอนการทำงานของโพรโทคอล AODV แบบใหม่ที่ใช้กับ WSN แบบเกตเวย์หลายตัว



ภาพที่ 3.34 การส่ง Message RERR ออกไป เมื่อเกตเวย์หลักไม่สามารถทำงานได้



ภาพที่ 3.35 ขั้นตอนการรอ Message RERR เพื่อเปลี่ยนเกตเวย์