กระบวนการจัดลำดับความสำคัญของข้อความเพื่อให้ได้ความเหมาะสมที่สุดในเครือข่ายที่มีความหน่วงสูง

ALGORITHM FOR PRIORITIZING MESSAGES IN DELAY-TOLERANT NETWORK

กฤษกร์ เอกวรรณัง

วิศรุต ศีรอรวรรณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 2 ปีการศึกษา 2558

กระบวนการจัดลำดับความสำคัญของข้อความเพื่อให้ได้ความเหมาะสมที่สุดในเครือข่ายที่มีความทนทานต่อความหน่วงสูง

ALGORITHM FOR PRIORITIZING MESSAGES IN DELAY-TOLERANT NETWORK

โดย

กฤษกรช์ เอกวรรณัง

KRITSAKORN AKWANNANG

วิศรุต ตีรณะวรวรรณ

VISARUT TIRATAWORAWAN

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ ดร. โชติพัชร์ ภรณวลัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 2 ปีการศึกษา 2558

กระบวนการจัดลำดับความสำคัญของข้อความเพื่อให้ได้ความเหมาะสมที่สุดในเครือข่ายที่มีความทนทานต่อความหน่วงสูง

ALGORITHM FOR PRIORITIZING MESSAGES IN DELAY-TOLERANT NETWORK

โดย

กฤษกรช์ เอกวรรณัง

KRITSAKORN AKWANNANG

วิศรุต ตีรณะวรวรรณ

VISARUT TIRATAWORAWAN

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ ดร. โชติพัชร์ ภรณวลัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 2 ปีการศึกษา 2558

ALGORITHM FOR PRIORITIZING MESSAGES IN DELAY-TOLERANT NETWORK

KRITSAKORN AKWANNANG

VISARUT TIRATAWORAWAN

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2/2015

ใบรับรองปริญญานิพนธ์ ประจำปีการศึกษา 2558

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง กระบวนการจัดลำดับความสำคัญของข้อความเพื่อให้ได้ความเหมาะสมที่สุดในเครือข่ายที่มีความทนทานต่อความหน่วงสูง

ALGORITHM FOR PRIORITIZING MESSAGES IN DELAY-TOLERANT NETWORK

ผู้จัดทำ

1.	**นายกฤษกรช์ เอกวรรณัง**	**รหัสนักศึกษา**	**55070002**
2.	**นายวิศรุต ตีรณะวรวรรณ**	**รหัสนักศึกษา**	**55070110**

.....อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร. โชติพัชร์ ภรณวลัย)

หัวข้อโครงงาน	กระบวนการจัดลำดับความสำคัญของข้อความเพื่อให้ได้ความเหมาะสมที่สุดในเครือข่ายที่มีความทนทานต่อความหน่วงสูง
นักศึกษา	นายกฤษกรช์ เอกวรรณัง รหัสนักศึกษา 55070002
	นายวิศรุต ตีรถะวรวรรณ รหัสนักศึกษา 55070110
ปริญญา	วิทยาศาสตรบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
ปีการศึกษา	2558
อาจารย์ที่ปรึกษา	รองศาสตราจารย์ ดร. โชติพัชร์ ภรณวลัย

บทคัดย่อ

เครือข่ายที่มีความทนทานต่อความหน่วงสูงเป็นเครือข่ายที่มีความหนาแน่นของโหนดที่น้อย ไม่สามารถระบุเส้นทางการเชื่อมต่อที่แน่นอนระหว่างคู่โหนดได้ มักถูกใช้เป็นเครือข่ายในพื้นที่ทุรกันดารที่ไม่มีอินเตอร์เน็ตในการส่งข้อมูลหากันหรือแม้แต่ในขณะเกิดภัยธรรมชาติที่ทำให้เครือข่ายอินเตอร์เน็ตเดิมไม่สามารถใช้งานได้ โหนดมีระยะเวลาที่อยู่ในระยะที่สามารถส่งข้อมูลกันได้น้อย สามารถส่งแลกเปลี่ยนข้อความกันได้ในจำนวนจำกัดต่อการเจอกันหนึ่งครั้ง ทำให้มีดีเลย์ในการส่งข้อความระหว่างโหนดสูงกว่าระบบเครือข่ายทั่วไป จึงเรียกเครือข่ายประเภทนี้ว่า

เครื่อข่ายที่มีความทนทานต่อความหน่วงสูง การตัดสินใจว่าควรจะจัดลำดับข้อความเพื่อที่จะแลกเปลี่ยนกันในการเจอกันครั้งถัดไปนั้นจึงเป็นกระบวนการที่สำคัญเพื่อที่จะทำให้แน่ใจว่าข้อความสามารถส่งไปถึงเป้าหมายได้ โดยใช้ระยะเวลาที่น้อยที่สุด ไม่มีการสูญหายในเครือข่าย

Project Title	Algorithm for prioritizing messages in delay-tolerant network	
Student	Mr. Kritsakorn Akwannang	Student ID 55070002
	Mr. Visarut Tirataworawan	Student ID 55070110
Degree	Bachelor of Science	
Program	Information Technology	
Academic Year	2015	
Advisor	Assoc. Prof. Dr. Chotipat Pornavalai	

ABSTRACT

Delay-Tolerant Network (DTN) is a sparse network of mobile wireless nodes where that has no end-to-end connectivity between nodes. DTN usually used to provide a network in extreme-terrestrial environments, mobile environments, or even when there is a disaster happened. In DTN, contact time between nodes is usually low, so the amount of message exchanges between two nodes is limited causes the network to have high message transfer delay, hence the name Delay-Tolerant Network. To prioritizing a message to transfer in the next contact-time is crucial to be able to ensure that the network can delivers a message efficiently, low overhead, with maximum delivery rate, low delay and no loss in the network

กิตติกรรมประกาศ

ปริญญานิพนธ์เรื่อง กระบวนการจัดลำดับความสำคัญของข้อความเพื่อให้ได้ความเหมาะสมที่สุดในเครือข่ายที่มีความทนทานต่อความหน่วงสูง ฉบับนี้สำเร็จลุล่วงได้จากคำแนะนำและการให้คำปรึกษาของรองศาสตราจารย์ ดร. โชติพัชร์ ภรณวลัย ผู้เป็นอาจารย์ผู้ควบคุมปริญญานิพนธ์ กลุ่มผู้วิจัยรู้สึกทราบซึ้งในความเสียสละ รวมถึงความอนุเคราะห์จากท่านอาจารย์และขอกล่าวขอบพระคุณท่านเป็นอย่างยิ่ง

ขอกราบพระคุณคณาจารย์คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุก ๆท่านที่ได้ให้คำปรึกษา ตลอดจนประสิทธิ์ประสาทวิชาให้กับกลุ่มผู้จัดทำ

ขอกล่าวขอบคุณห้องวิจัยและปฏิบัติการ Intelligent Internet and Informatics Laboratory คณะเทคโนโลยีสารสนเทศ หนังสือทุกเล่ม ตลอดจนข้อมูลทุกอย่างที่ใช้ในการทำปริญญานิพนธ์ฉบับนี้

ขอกล่าวขอบคุณ พี่ๆ เพื่อนๆ น้องๆ ทุกคนทั้งในและนอกคณะที่คอยให้คำปรึกษาและให้กำลังใจเสมอมา

สุดท้ายนี้กลุ่มผู้จัดขอกราบขอบพระคุณบิดา มารดา และครอบครัวของกลุ่มผู้วิจัยที่เป็นกำลังใจและให้การสนับสนุนในทุกๆเรื่อง ซึ่งทั้งหมดล้วนแล้วแต่มีส่วนให้ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์พึงมาจากปริญญานิพนธ์ฉบับนี้กลุ่มผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

สารบัญ

หน้า

บทคัดย่อ I

ABSTRACT II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญตาราง VII

สารบัญรูป VIII

บทที่ 1 บทนำ 1

1.1 ความเป็นมาและความสำคัญ 1

1.2 วัตถุประสงค์ 2

1.3 สมมติฐานของการศึกษา 2

1.4 ขอบเขตของงานวิจัย 3

1.5 ขั้นตอนการศึกษา 3

บทที่ 2 การทบทวนวรรณกรรมที่เกี่ยวข้อง 4

2.1 Routing Protocols ในเครือข่ายที่มีความทนทานต่อความหน่วงสูง 4

2.1.1 Epidemic-Based Routing Protocols 4

2.1.2 Two-Hop Routing Protocols 7

2.2 Mobility Model 9

2.2.1 Random Waypoint 9

2.2.2 Random Walk 9

บทที่ 3 กระบวนการจัดลำดับความสำคัญของข้อความในเครือข่าย
ที่มีความทนทานต่อความหน่วงสูง 10

3.1 ปัจจัยที่มีผลต่อประสิทธิภาพของโปรโตคอลในด้านอัตราการส่ง
ข้อความสำเร็จ (Delivery Probability) ค่าความหน่วง (Delivery Delay) และค่า
Overhead 10

สารบัญ (ต่อ)

3.1.1 ความหนาแน่นของโหนดในเครือข่าย 10
3.1.2 จำนวนสำเนาของข้อความ 10
3.1.3 ขนาดของบัฟเฟอร์ของโหนด 10
3.1.4 โปรโตคอลที่ใช้ในการส่งข้อความ 11
3.1.5 จำนวนและการกระจายตัวของข้อความในเครือข่าย 11
3.2 ผลของโมเดลการเคลื่อนที่ (Mobility Model) ต่อจำนวนการพบกันของโหนด 11
3.3 สมมติฐานที่จะนำไปพัฒนาเป็นโปรโตคอลในเครือข่าย DTN 11
3.3.1 หากสามารถจัดลำดับของข้อความที่จะส่งในแต่ละ Contact Time ได้ดียิ่งขึ้น จะทำให้โปรโตคอลมีประสิทธิภาพที่ดีขึ้น(FMS 11
3.3.2 หากโหนดในโปรโตคอล FMS มี Dropping Policy ที่เหมาะสม จะทำให้โปรโตคอลมีประสิทธิภาพโดยรวมที่ดี 17
3.3.2.1 หลักการทำงานของ Dropping Policy 18
3.3.3 หากโปรโตคอลมี Fuzzy Rule ที่ทำหน้าที่เฉพาะในการคิดค่า Dropping Policy ที่ดีขึ้นจะทำให้โปรโตคอล FMSD มีประสิทธิภาพ ที่ดีมากขึ้นทั้งในด้าน Delivery Probability, Overhead, และ Delivery Delay 21
3.3.3.1 Fuzzy Logic Controller 22
3.3.3.1.1. Input ของ Dropping Fuzzy Rule 23
3.3.3.1.2. Output ของ Dropping Fuzzy Rule 24
3.3.3.2 Dropping Policy 25
3.3.3.2.1. หลักการทำงานของ Dropping Policy 25

บทที่ 4 ผลการทดลอง 27
4.1 ตัวแปรที่ใช้วัดประสิทธิภาพของโปรโตคอลในเครือข่าย DTN 27
4.1.1 Delivery Probability 27
4.1.2 Overhead Ratio 27
4.1.3 Delay 27
4.2 สภาพแวดล้อมที่ใช้ในการทดลอง 27
4.3 ผลการทดลอง 28

สารบัญ (ต่อ)

4.3.1 ผลการทดลองของสมมติฐานที่ 3.3.1 28
4.3.2 ผลการทดลองของสมมติฐานที่ 3.3.2 31
4.3.3 ผลการทดลองของสมมติฐานที่ 3.3.3 34
4.3.4 ผลการทดลองเมื่อนำ FMSD2 มาเปรียบเทียบกับโปรโตคอลอื่นๆ 35

บทที่ 5 สรุปผลการทดลอง....41
5.1 สรุปผล 41

บรรณานุกรม....42

ประวัติผู้เขียน....44

สารบัญตาราง

ตารางที่ **หน้า**

3.1 แสดงค่า Buffer Section ของ FMSD2 ในการคิด Sending Priority ทั้ง 9 section.....16
3.2 แสดงค่า Dropping Section ของ FMSD2 กระบวนการ Dropping policy ทั้ง 9 section.....25
4.1 ผลการทดลองที่ขนาดบัฟเฟอร์ 3MB ของ FMS เปรียบเทียบกับ FuzzySpray 29
4.2 ผลการทดลองที่ขนาดบัฟเฟอร์ 12MB ของ FMS เปรียบเทียบกับ FuzzySpray 29
4.3 ผลการทดลองที่บัฟเฟอร์ขนาด 3MB ของ FMSD เปรียบเทียบกับโปรโตคอลอื่นๆ 32
4.4 ผลการทดลองที่บัฟเฟอร์ขนาด 12MB ของ FMSD เปรียบเทียบกับโปรโตคอลอื่นๆ 32
4.5 ผลการทดลองที่บัฟเฟอร์ขนาด 3MB ของ FMSD2 เปรียบเทียบกับโปรโตคอลอื่นๆ 34
4.6 แสดงผลการทดลองของ FMSD2 เมื่อเทียบกับโปรโตคอลต่างๆ ที่ 3MB.....36
4.7 แสดงผลการทดลองของ FMSD2 เมื่อเทียบกับโปรโตคอลต่างๆ ที่ 6MB.....36
4.8 แสดงผลการทดลองของ FMSD2 เมื่อเทียบกับโปรโตคอลต่างๆ ที่ 9MB.....38
4.9 แสดงผลการทดลองของ FMSD2 เมื่อเทียบกับโปรโตคอลต่างๆ ที่ 12MB.....38

สารบัญรูป

รูปที่ **หน้า**

1.1 การทำ Store and Forward ในเครือข่าย DTN 2
3.1 แสดงการทำงานในส่วนของ Message Scheduling 13
3.2 กราฟแสดง Degree of Membership ของ FTC 14
3.3 แสดงกราฟ Membership Function ของค่า Message Size 15
3.4 กราฟแสดง Degree of Membership ของ Buffer Section(BS) 15
3.5 Pseudo code สำหรับโปรโตคอล FMS 17
3.6 แสดงการทำงานของ Dropping Policy ของ FMSD 19
3.7 Pseudocode สำหรับโปรโตคอล FMSD 20
3.8 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) จำนวน 1000 ข้อความแรกของโปรโตคอล FMSD 22
3.9 กราฟแสดง Membership Function ของ Delta FTC 23
3.10 กราฟแสดง Membership Function ของ Delta Delay 24
3.11 กราฟแสดง Membership Function ของ Dropping Section 24
4.1 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) ของโปรโตคอล FMS ที่สภาพแวดล้อม 3MB 29
4.2 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) ของโปรโตคอล FuzzySpray ที่สภาพแวดล้อม 3MB 30
4.3 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) ของโปรโตคอล FMS ที่สภาพแวดล้อม 12MB 30
4.4 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) ของโปรโตคอล FuzzySpray ที่สภาพแวดล้อม 12MB 31
4.5 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) ของโปรโตคอล FMSD ที่สภาพแวดล้อม 3MB 33
4.6 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) ของโปรโตคอล FMSD ที่สภาพแวดล้อม 12MB 33
4.7 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) ของโปรโตคอล FMSD2 ที่สภาพแวดล้อม 3MB 35

สารบัญรูป (ต่อ)

4.8 Histogram เปรียบเทียบ FMSD2 กับ MaxProp ที่บัฟเฟอร์ขนาด 9MB 37
4.9 กราฟเปรียบเทียบ Delivery Probability ของโปรโตคอลต่างๆ 39
4.10 กราฟเปรียบเทียบ Overhead ของโปรโตคอลต่างๆ 39
4.11 กราฟเปรียบเทียบ Delivery Delay ของโปรโตคอลต่างๆ 39

บทที่ 1
บทนำ

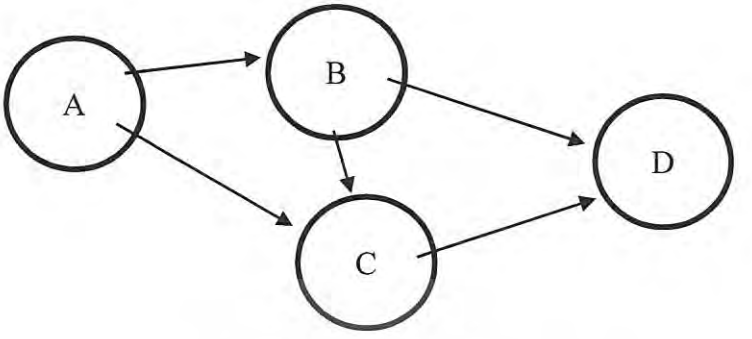
1.1 ความเป็นมาและความสำคัญ

ในโลกปัจจุบัน การเชื่อมต่อสื่อสารแลกเปลี่ยนข้อมูลกันนั้นเป็นสิ่งที่สำคัญต่อทุกคน เพื่อที่จะได้รับข่าวสารที่เราต้องการ สื่อกลางในการแลกเปลี่ยนข้อมูลกันนั้นก็คือเครือข่ายอินเตอร์เน็ต ดังนั้นจึงมีความพยายามที่จะทำให้ทุกๆตารางกิโลเมตรของบริเวณที่มีผู้คนอยู่อาศัยมีการเชื่อมต่อเข้าถึงอินเตอร์เน็ตตลอดเวลา แต่ในความเป็นจริงนั้นสามารถทำให้สำเร็จได้ยาก อาจจะเกิดจากข้อจำกัดในหลายๆด้าน เช่น การตั้งเสาสัญญาณในพื้นที่ทุรกันดารอาจทำได้ยาก หรือในกรณีที่เกิดภัยทางธรรมชาติแล้วโครงข่ายเสาสัญญาณเสียหายทำให้ไม่สามารถเข้าถึงอินเตอร์เน็ตได้ การศึกษาในเรื่องของเครือข่ายเคลื่อนที่เฉพาะกิจ (Mobile Ad-hoc Network : MANET) จึงเกิดขึ้น และในขณะเดียวกันก็มีการวิจัยเกี่ยวกับการส่งข้อมูลระหว่างกันในอวกาศ (Interplanetary Network : IPN) ที่ต้องมีความทนทานต่อความหน่วง (Delay) สูงเนื่องจากระยะทางและสภาพแวดล้อมที่ยากต่อการส่งข้อความระหว่างกัน เมื่อมีการนำความรู้เกี่ยวกับ IPN ไปปรับใช้กับเครือข่ายเคลื่อนที่เฉพาะกิจจึงเกิดเครือข่ายที่มีความทนทานต่อความหน่วงสูงเกิดขึ้น

เครือข่ายที่มีความทนทานต่อความหน่วงสูง(Delay-Tolerant Network) เป็นเครือข่ายที่มีความหนาแน่นของโหนดต่อขนาดของพื้นที่น้อย ไม่มีสามารถระบุเส้นทางการเชื่อมต่อระหว่างคู่โหนดได้ โดยแต่ละโหนดในเครือข่ายที่มีความทนทานต่อความหน่วงสูงมีทรัพยากรที่จำกัดทั้งในเรื่องของพลังงานที่โหนดมี ขนาดของบัฟเฟอร์ที่ใช้ในการเก็บข้อความเพื่อส่งออกไปยังโหนดอื่นๆ ในเครือข่าย ระยะเวลาที่คู่โหนดเจอกันนั้นมีเวลาที่จำกัด เป็นผลให้จำนวนของข้อความที่โหนดสามารถแลกเปลี่ยนกันในระยะเวลาที่เจอกันนั้นน้อยมาก โหนดในเครือข่ายที่มีความทนทานต่อความหน่วงสูงจึงใช้รูปแบบการส่งข้อความแบบเก็บแล้วส่งต่อ (Store and Forward) ที่โหนดรับข้อความเข้ามาเก็บในบัฟเฟอร์จนกว่าจะมีโอกาสที่จะส่งข้อความ (Relay) ต่อไปยังโหนดอื่นเพื่อให้ข้อความเดินทางไปถึงเป้าหมายของข้อความนั้นๆ

จาก**รูปที่ 1.1** โหนด A ซึ่งเป็นโหนดต้นทาง (Source Node) ของข้อความ M (Source : A, Destination D) เจอโหนด B และ C อยู่ในระยะที่สามารถแลกเปลี่ยนข้อความกันได้ (Transmission Range) โหนด A ก็จะทำการรีเลย์ข้อความไปยังทั้งโหนด B และ C เมื่อโหนด B หรือ C เคลื่อนที่เข้า

ไปในระยะที่สามารถแลกเปลี่ยนข้อความกันได้กับโหนด D ก็จะทำการรีเลย์ข้อความไปให้ยังโหนด D ซึ่งเป็นโหนดปลายทางของข้อความ M



รูปที่ 1.1 การทำ Store and Forward ในเครือข่าย DTN

1.2 วัตถุประสงค์

1.2.1 เพื่อคิดค้นกระบวนการและวิธีที่จะทำให้การจัดลำดับความสำคัญของข้อมูลรวมทั้งวิธีการจัดการบัฟเฟอร์ของโหนดในเครือข่ายที่มีความทนทานต่อความหน่วงสูงนั้น สามารถทำงานได้อย่างมีประสิทธิภาพมากขึ้น และส่งข้อความถึงปลายทางโดยใช้เวลาต่ำหรือใกล้เคียงโปรโตคอลอื่นๆ เกิด overhead น้อยกว่าเดิม และที่สำคัญต้องไม่มีการสูญหายของข้อมูล (lost) ภายในเครือข่าย

1.2.2 เพื่อศึกษาปัจจัยที่มีผลต่อความสำเร็จของการส่งข้อความและระยะเวลาในการส่งข้อความในเครือข่ายที่มีความทนทานต่อความหน่วงสูง

1.2.3 เพื่อออกแบบโปรโตคอลที่สามารถส่งข้อความได้เร็ว มีประสิทธิภาพทั้งในด้าน Delivery Ratio, Overhead Ratio, และ Delay

1.3 สมมติฐานของการศึกษา

เครือข่ายที่มีความทนทานต่อความหน่วงสูงเป็นเครือข่ายของโหนดเคลื่อนที่ที่มีความหนาแน่นของโหนดต่อขนาดของพื้นที่น้อย โหนดมักเป็นอุปกรณ์เคลื่อนที่จึงมีทรัพยากรด้านพลังงาน หน่วยความจำบัฟเฟอร์ในการเก็บข้อความที่จำกัด มีการเคลื่อนที่ในแบบสุ่ม ทำให้การคาดเดาว่าโหนดจะเคลื่อนที่ไปที่ไหนในเวลาถัดไปทำได้ยากไม่สามารถคาดเดาระยะเวลาที่โหนดสองตัวจะอยู่ในระยะที่สามารถแลกเปลี่ยนข้อความกันได้ (ไม่สามารถคาดเดา Contact time ของคู่โหนด) ทำให้การออกแบบโปรโตคอลในการจัดลำดับข้อความให้เหมาะสมเพื่อที่จะแลกเปลี่ยนในการเจอกันของโหนดครั้งถัดไปนั้นเป็นตัวแปรความสำคัญต่อประสิทธิภาพของ Routing Protocol

1.4 ขอบเขตของงานวิจัย

เป็นโครงงานที่ทำการวิจัยเกี่ยวกับการจัดลำดับความสำคัญในการส่งข้อความในเครือข่ายที่ความทนทานต่อความหน่วงสูง (Delay-Tolerant Network) โดยมีสภาพแวดล้อมการทำงานในการทำวิจัย คือ The One Network Emulator เพื่อจำลองการส่งข้อความระหว่างกันของ Network Node โดยทำการเปรียบเทียบผลการทดลองกับกระบวนการค้นหาเส้นทางต่างๆที่มีใช้อยู่ในปัจจุบัน เช่น Epidemic หรือ Spray and Wait

โดยค่าที่ใช้ในการวัดประสิทธิภาพของกระบวนการ (Routing Protocol) นั้นสามารถประเมินได้จากค่าอัตราความสำเร็จใจการส่งข้อความ (Delivery Probability) ค่าความหน่วงเวลาในการส่งข้อความ (Delivery Delay) ค่าอัตราการส่งข้อความสำเร็จต่อจำนวนครั้งการส่งทั้งหมด (Overhead)

แบบจำลองการเคลื่อนที่ของโหนดที่ใช้ทำการวิจัยจะเป็นแบบจำลองการเคลื่อนที่แบบสังเคราะห์ ที่มีชื่อว่า Random Waypoint ซึ่งโหนดจะทำการสุ่ม Path การเคลื่อนที่ภายในพื้นที่ของแผนที่ที่กำหนดไว้ เมื่อโหนดเคลื่อนที่ไปถึงจุดหมายที่กำหนดไว้แล้วก็จะหยุดรอเวลาเพื่อที่จะสุ่มและเคลื่อนที่ไปในจุดหมายถัดไป

1.5 ขั้นตอนการศึกษา

1.5.1 บทนำ : กล่าวถึงความสำคัญและความเป็นมาของปัญหาที่ทำให้เกิดการวิจัยขึ้น วัตถุประสงค์ของการวิจัย ขอบเขตของการวิจัย และขั้นตอนในการวิจัย

1.5.2 การทบทวนวรรณกรรมที่เกี่ยวข้อง : กล่าวถึงแนวคิดและกระบวนในการส่งข้อความในเครือข่ายที่มีความทนทานต่อความหน่วงสูงที่เคยได้มีการศึกษาวิจัยมาก่อนแล้ว และกล่าวถึงสภาพแวดล้อมจำลองที่มีในปัจจุบัน

1.5.3 วิธีการดำเนินการวิจัย : กล่าวถึงแนวคิดหลักของกระบวนการที่คิดค้นขึ้นใหม่ในการจัดลำดับความสำคัญของข้อความในเครือข่ายที่มีความทนทานต่อความหน่วงสูงและสภาพแวดล้อมที่จะใช้ในการทดลองเผื่อให้ได้มาซึ่งผลการวิจัย

1.5.4 ผลการทดลองและการเปรียบเทียบผลที่ได้กับ Routing Protocol ที่ถูกใช้ในเครือข่ายที่มีความทนทานต่อความหน่วงสูง

1.5.5 สรุปผลการวิจัย

บทที่ 2
การทบทวนวรรณกรรมที่เกี่ยวข้อง

2.1 Routing protocols ในเครือข่ายที่มีความทนทานต่อความหน่วงสูง

การส่งข้อความในเครือข่ายที่มีความทนทานต่อความหน่วงสูง (Delay-Tolerant Network) นั้นมีความแตกต่างจากเครือข่ายเคลื่อนที่เฉพาะกิจ (Mobile Ad-hoc Network) คือ ในเครือข่าย DTN จะตั้งสมมติฐานว่าโหนดไม่สามารถสร้างเส้นทางที่แน่นอนระหว่างกันในการส่งข้อความได้หรือถ้าหากสามารถสร้างได้ก็มีระยะเวลาที่เจอกัน (Contact time) ที่น้อยและไม่สามารถคาดเดาได้ โปรโตคอลเดิมที่ใช้งานในเครือข่ายเคลื่อนที่เฉพาะกิจจึงไม่สามารถใช้งานได้ในเครือข่ายที่มีความทนทานต่อความหน่วงสูงได้อย่างมีประสิทธิภาพ จึงเกิดโปรโตคอลสำหรับใช้งานในเครือข่ายที่มีความทนทานต่อความหน่วงสูงโดยเฉพาะ โดยยึดสมมติฐานที่ว่าโหนดในเครือข่าย DTN มีความหนาแน่นที่น้อย ไม่สามารถคาดเดา Contact time หรือทิศทางการเคลื่อนที่ได้แน่นอน

จากการศึกษาวรรณกรรมที่เกี่ยวข้อง จึงสามารถสรุปและแบ่งประเภทของโปรโตคอลที่มีอยู่ในปัจจุบันคร่าวๆจากหลักการจำลองข้อความ (Replication) นั่นคือ หลักการจำกัดจำนวนสำเนา (Copy) ของข้อความที่มีอยู่ในเครือข่าย การส่งต่อข้อความ (Forwarding) คือ หลักการเลือกข้อความที่อยู่ในบัฟเฟอร์เพื่อที่จะส่งหรือแลกเปลี่ยนในครั้งถัดไปของการเจอกันของโหนด โดยปกติจะเลือกในลักษณะเดียวกับการจัดการลำดับของข้อความในบัฟเฟอร์ แต่จะมีกรณีพิเศษบางกรณี เช่น โหนดผู้ส่งเจอโหนดปลายทางของข้อความนั้นพอดีทำให้เกิดการส่ง (Forwarding) แบบที่ไม่ได้เป็นไปตามลำดับของข้อความในบัฟเฟอร์ได้ และการจัดการลำดับของข้อความในบัฟเฟอร์ (Queue Management) คือ หลักการจัดลำดับของข้อความในบัฟเฟอร์ ได้ดังนี้

2.1.1 Epidemic-Based Routing Protocols

โปรโตคอลประเภทนี้จะพยายามกระจาย Copy ของข้อความออกไปในเครือข่ายให้ได้มากที่สุดโดยอาจจะ ใช้ข้อมูลที่โหนดเรียนรู้ในการช่วยตัดสินใจในการสร้าง Copy ของข้อความและการกระจายข้อความด้วยก็ได้

- Direct Delivery [1] เป็นโปรโตคอลที่โหนดต้นทาง (Source Node) จะเก็บข้อความไว้จนกว่าโหนดปลายทาง (Destination Node) และอยู่ในระยะที่สามารถส่งข้อความได้จึงจะส่งข้อความออกไป
- Epidemic [2] เป็นโปรโตคอลที่จะจำลอง (Replicate) ข้อความแล้วพยายามที่จะส่งข้อความไปยัง Relay Node อื่นๆที่อยู่ในระยะที่โหนดสามารถส่งข้อความได้ให้ได้มากที่สุด เพื่อที่จะได้ค่าอัตราความสำเร็จในการส่งข้อความที่สูงที่สุดเท่าที่เป็นไปได้ แต่ละโหนดใน

โปรโตคอลจะมีสิ่งที่เรียกว่า Message Summary Vector ช่วยในการทำ Queue Management ซึ่งเป็นค่า Hash ของ Message Id ที่โหนดมีอยู่ในบัฟเฟอร์ ทำหน้าที่สรุปว่าโหนดมีข้อความอะไรเก็บอยู่ในบัฟเฟอร์บ้าง ก่อนที่โหนดจะส่งข้อความหากันโหนดจะดูว่าโหนดปลายทางมีข้อความนี้อยู่ใน Summary Vector แล้วหรือไม่ ถ้าหากมีก็จะไม่ทำการส่งข้อความต่อไปยังโหนดปลายทางนั้น

- Encounter Based Fuzzy Logic Routing (EFLR) [5] โหนดในโปรโตคอลนี้จะมีตัวจับเวลาสองแบบ คือ Encounter Timer ซึ่งคือระยะเวลาที่โหนด i และ j อยู่ในระยะที่สามารถแลกเปลี่ยนข้อความกันได้ และ Non-Encounter Timer ซึ่งคือระยะเวลาที่โหนด i และ j ไม่อยู่ในระยะที่สามารถแลกเปลี่ยนข้อความให้กันได้ อยู่นอก Transmission Range) โหนดจะนำ Encounter Time และ Non-Encounter Time ที่ได้จากทั้ง Encounter Timer และ Non-Encounter Time ตามลำดับไปเป็น Input ของ Fuzzy Logic Controller เพื่อคำนวณออกมาเป็นค่าโอกาสในการส่ง (Transfer Opportunity) ของแต่ละคู่โหนด เมื่อโหนดผู้ส่งต้องการส่งข้อความ โหนดผู้ส่งก็จะถามโหนดระหว่างทางรอบๆที่อยู่ในระยะว่า Transfer Opportunity จากโหนดระหว่างทางนั้นไปยังโหนดปลายทางของข้อความมีค่ามากกว่า Transfer Opportunity ของโหนดผู้ส่งเองหรือไม่ ถ้าหาก Transfer Opportunity ของโหนดระหว่างทางนั้นมีค่ามากกว่า Transfer Opportunity ของโหนดผู้ส่ง โหนดผู้ส่งก็จะทำการรีเลย์ข้อความไปยังโหนดระหว่างทางนั้นเพื่อให้ข้อความมีโอกาสส่งไปถึงยังโหนดเป้าหมายของข้อความนั้น ในส่วนของการทำ Queue Management มี Dropping Policy คือ หากบัฟเฟอร์ของโหนดเต็มไม่สามารถรับข้อความเข้ามาอีกได้ โหนดก็จะเลือก Drop ข้อความที่มีค่า Transfer Opportunity ที่มาก เพราะมีโอกาสที่ข้อความนั้นมีจะส่งถึงเป้าหมายโดยโหนดอื่นๆแล้วมีมาก ทุกๆข้อความ m ที่ถูกสร้างขึ้นมาใหม่จะมีค่า Transfer Opportunity เริ่มต้นเป็น 0 ถ้าหากข้อความถูกส่งผ่านจากโหนด i ไปยังโหนด j แล้วค่า Transfer Opportunity ของข้อความ m จะมีค่าเท่ากับ $TROP_m + TROP_{i,j}$
- FuzzySpray [6] เป็นโปรโตคอลที่ใช้ Fuzzy Logic มาช่วยจัดลำดับข้อความ (Queue Management) ที่จะส่งในการพบกันครั้งต่อไปของโหนด โดยใช้ทั้ง Message Size และ ค่า Forward Transmission Count (FTC) ซึ่งเป็นค่าที่บอกจำนวนของ Message นั้นๆที่อยู่ในเครือข่ายโดยประมาณ เป็นค่า input ของ Fuzzy Logic Controller เมื่อประมวลผลแล้วจะออกมาเป็นค่า Priority เพื่อนำมาเรียงลำดับข้อความในบัฟเฟอร์ของโหนด โดยจะจัดเรียงจากข้อความที่มี Priority จากมากไปน้อย และมีการแปลกเปลี่ยน Summary Vector [2] เพื่อป้องกันการส่งข้อความซ้ำซ้อนกันของโหนด โปรโตคอลนี้จะสเปรย์ข้อความตามลำดับในบัฟเฟอร์ให้ทุกๆโหนดที่อยู่ในระยะที่สามารถแลกเปลี่ยนข้อความกันได้

- Adaptive Fuzzy Routing in Opportunistic Network (AFRON) [7] ใช้ Fuzzy Logic ในการจัดลำดับความสำคัญของข้อความเพื่อส่งในครั้งถัดไปที่โหนดไป contact กับโหนดอื่น เช่นเดียวกับ FuzzySpray [6] แต่ AFRON จะเพิ่ม input ของ Fuzzy Logic Controller ไปอีกหนึ่งตัวคือ ค่า Time-to-Live (TTL) ของข้อความ เพราะ ถ้าหากข้อความไม่ถูกรีเลย์ไปยังโหนดเป้าหมายในภายในระยะเวลาของ TTL ข้อความนั้นก็จะถูก Drop ออกจากบัฟเฟอร์ของโหนด ทำให้เกิด Lost เกิดขึ้นในเครือข่าย ดังนั้น input ของ Fuzzy Logic Controller จึงเป็น Message Size, Forward Transmission Count (FTC), และ Time-to-Live แล้วประมวลผลออกมาเป็นค่า Priority ของข้อความ การจัดเรียงลำดับของข้อความภายในบัฟเฟอร์จะมีลักษณะเป็นการเรียงจากข้อความที่มี Priority มากไปน้อย ทุกๆครั้งที่ทุกๆโหนดเจอกันจะมีการแลกเปลี่ยน Summary Vector ซึ่งเป็นค่า Hash ของ Message Id ที่เก็บอยู่ในบัฟเฟอร์ของโหนดนั้นๆ ข้อความที่มีอยู่ใน Summary Vector แล้วก็จะไม่ถูกรีเลย์ระหว่างโหนด ทำให้ไม่เกิดการรีเลย์ข้อความซ้ำซ้อนกัน ทำให้สามารถลดค่า Overhead ได้
- ลักษณะการจัดลำดับข้อความและการดรอปข้อความแบบ Global History-based Prediction (GHP) ใช้การค่าประมาณจำนวนสำเนาของข้อความ m ว่าข้อความมีทั้งหมดกี่ตัวในเครือข่าย มีโหนดไหนเห็นข้อความนั้นแล้วบ้าง และโหนดไหนเห็นข้อความ m แล้วมีพื้นที่ในบัฟเฟอร์เหลือเพื่อที่จะรับข้อความ m แล้วนำค่าต่างๆไปคำนวณเป็นค่า Priority หรือในเปเปอร์ของโปรโตคอลจะเรียกว่า Per-Message Utility ที่ใช้เป็นค่าตัดสินในการจัดเรียงลำดับของข้อความในบัฟเฟอร์และลำดับการถูกดรอปของข้อความในกรณีที่บัฟเฟอร์ของโหนดนั้นๆเต็ม GHP ออกแบบมาสำหรับโปรโตคอล Epidemic และโปรโตคอลจำพวก Two Hop Routing Protocol เช่น Spray and Wait, Spray and Focus เป็นต้น ใน GHP ข้อความที่มีค่า Priority สูงจะอยู่ด้านหน้าของ Sending queue หากข้อความมีค่า Priority ที่ต่ำก็จะถูกดรอปออกไปก่อนในกรณีที่บัฟเฟอร์ของโหนดเต็ม นอกจากรูปแบบการจัดเรียงและดรอปข้อความแล้ว GHP ยังได้มีลักษณะการตัดสินใจว่าจะรับข้อความที่โหนดอื่นกำลังจะส่งมาให้หรือไม่โดยใช้ค่า Priority ของข้อความที่โหนดอื่นกำลังจะส่งเข้ามาว่าค่า Priority ของข้อความนั้นมีค่าสูงกว่า Priority ของข้อความที่มี Priority ที่ต่ำสุดของโหนดผู้รับหรือไม่ ถ้าข้อความที่โหนดอื่นกำลังจะส่งมามีค่า Priority น้อยกว่าข้อความที่มี Priority ต่ำสุดของโหนดผู้รับ โหนดผู้รับก็จะตัดสินใจไม่รับข้อความที่กำลังจะเข้ามานั่นเอง
- MaxProp [10] เป็นโปรโตคอลที่ใช้ข้อมูลทางสถิติ เช่น ขนาดของข้อมูลเฉลี่ยที่แลกเปลี่ยนในแต่ละ Contact Time, ข้อมูลการเจอกันของโหนดเพื่อใช้ในการหาค่า Threshold ที่ใช้ในการกำหนดลำดับของข้อความที่จะถูกส่งใน Contact Time ครั้งต่อไป รวมทั้งเป็นข้อมูลที่ใช้สำหรับการตัดสินใจว่าโหนดจะดรอปข้อความใดหากบัฟเฟอร์ของโหนดเต็ม อีกทั้งยังมี

กระบวนการในการช่วยกระจายข้อความที่สร้างขั้นใหม่โดยการใช้ Priority สูงสุดต่อข้อความนั้นๆเพื่อที่จะถูกส่งก่อนใน Contact Time ครั้งต่อไป

2.1.2 Two-Hop Routing Protocols

โหนดแต่ละตัวใน Two-Hop Routing Protocol จะสามารถกระจาย Copy ของข้อความใดๆ ได้เป็นจำนวน L ข้อความ เมื่อข้อความใดมีค่า L เหลือเท่ากับ 1 โหนดที่เก็บข้อความนั้นจะสามารถส่งข้อความที่มี $L_m = 1$ ให้แก่โหนดเป้าหมายของข้อความนั้นเท่านั้น

- Spray and Wait [3] ในโปรโตคอลนี้สามารถแบ่งระยะ (Phase) ของข้อความได้ 2 ระยะ นั่นคือ Spray Phase และ Wait Phase ใน Spray Phase โหนดต้นทาง (Source Node) จะทำการรีเลย์ข้อความจำนวน L ข้อความไปยังโหนดที่ยังไม่มีข้อความนี้อยู่ในบัฟเฟอร์ ซึ่ง L คือ Replication token หรือจำนวน Copy ของข้อความที่มีอยู่ในเครือข่ายได้ เมื่อโหนดต้นทางทำการรีเลย์ข้อความที่ถูกสร้างขึ้นออกไปหนึ่งครั้ง ค่า L ของข้อความนั้นจะมีค่าเท่ากับ L/2 ถ้าหากค่า L ของข้อความใดมีค่าเท่ากับ 1 แล้ว ข้อความจะเข้าสู่ Wait Phase โหนดจะส่งข้อความนั้นในลักษณะของ Direct Delivery [1] นั่นคือจะเก็บข้อความนั้นไว้จนกว่าจะเจอโหนดเป้าหมาย (Destination Node) ของข้อความนั้น
- Spray and Focus [4] ในโปรโตคอลนี้สามารถแบ่งระยะ (Phase) ของข้อความได้ 2 ระยะ นั่นคือ Spray Phase และ Focus Phase ในระยะที่เป็น Spray เมื่อโหนดสร้างข้อความขึ้นมาก็จะทำการกำหนดค่า Forwarding Token ขึ้นมาด้วยค่าหนึ่ง ถ้าหากค่า Forwarding Token ยังมีค่ามากกว่า 1 โหนดก็จะสเปรย์ข้อความแบบ Binary Spray เช่นเดียวกับโปรโตคอล Spray and Wait [3] นั่นคือ เมื่อสเปรย์ข้อความไปยังโหนดที่ยังไม่มีข้อความนั้นอยู่ ค่า Forwarding Token ของข้อความที่โหนดต้นทางและโหนดที่ได้รับข้อความจะมีค่าเท่ากับ Forwarding Token/2 โดยโหนดผู้ส่งจะดูว่าโหนดผู้รับมีข้อความที่โหนดผู้ส่งกำลังจะส่งหรือไม่จาก Summary Vector ที่เป็นค่า Hash ของ Message Id ที่เก็บอยู่ในบัฟเฟอร์ของโหนดนั้นๆ [2] หากข้อความใดที่มีค่า Forwarding Token เท่ากับ 1 ข้อความนั้นจะเข้าสู่ระยะ Focus Phase ซึ่งอนุญาตให้โหนดสามารถรีเลย์ข้อความไปยังโหนดระหว่างทาง (Relay Node) ได้ถ้าหากโหนดนั้นมีโอกาสที่จะส่งข้อความนี้ไปถึงโหนดเป้าหมายได้สำเร็จมากกว่าโหนดเดิม โอกาสในการส่งสามารถประเมินได้จากค่าเวลาที่เจอกันครั้งล่าสุดและระยะทางระหว่างคู่โหนด ซึ่งการนำค่านี้มาช่วยตัดสินใจในการเลือกส่งข้อความถือว่าเป็นการทำ Utility Forwarding
- Adaptive Fuzzy Spray and Wait (AFSnW) [8] หลักการทำงานของ Adaptive Fuzzy Spray and Wait จะทำงานเหมือนกับ Spray and Wait คือ จะแบ่งระยะ (Phase) ของข้อความเป็น 2

ระยะ คือ เมื่อข้อความ *m* ถูกสร้างขึ้นจะอยู่ในระยะ Spray ซึ่งมีค่า Replication Token = L เมื่อโหนดต้นทางทำการรีเลย์ข้อความ *m* ออกไปยังโหนดผู้รับ ค่า L(*m*) = L(*m*)/2 ทั้งที่โหนดต้นทางและโหนดผู้รับ เมื่อข้อความ *m* มีค่า L(*m*) = 1 แล้ว ข้อความ *m* จะเข้าสู่ระยะ Wait ที่จะถูกรีเลย์ก็ต่อเมื่อโหนดเจอโหนดเป้าหมายของข้อความ *m* เท่านั้น AFSnW มีการใช้ Fuzzy Logic เพื่อช่วยในการเรียงลำดับข้อความที่จะส่งในบัฟเฟอร์ของโหนด เช่นเดียวกับ FuzzySpray [6] นั่นคือ มีการนำ Message Size และ Forward Transmission Count (FTC) เข้าไปเป็น input ของ Fuzzy Logic Controller เพื่อคำนวณออกมาเป็นค่า Priority ซึ่งลำดับของข้อความในบัฟเฟอร์จะเรียงจากข้อความที่มี Priority มากไปน้อย ในส่วนของ Dropping Policy ที่เลือกข้อความที่จะละทิ้ง (Drop)จากการสุ่มข้อความใดข้อความหนึ่งที่อยู่ในกลุ่ม Priority ที่น้อยที่สุด ซึ่งต่างจาก Dropping Policy แบบ Drop-Least-Priority ซึ่งจะละทิ้ง (Drop)ข้อความที่อยู่ท้ายบัฟเฟอร์(ตัวสุดท้ายในกลุ่มของ Priority ที่น้อยที่สุด) เสมอ ทำให้โปรโตคอลมีความยุติธรรมในเรื่องของการละทิ้ง (Drop)ข้อความเมื่อบัฟเฟอร์เต็มมากขึ้น

- Enhanced Fuzzy Logic-Based Spray and Wait [9] เป็นโปรโตคอลที่ใช้หลักมาจาก [3] แต่มีการนำ Fuzzy Logic Controller มาช่วยในการจัดลำดับข้อความเพื่อส่งในครั้งถัดไปที่โหนดอยู่ในระยะที่สามารถส่งข้อความได้ โดยมี Input ของ Fuzzy Logic Controller คือ ETR (Estimated Number of Replicas) หมายถึงตัวประมาณจำนวนของสำเนาข้อความที่อยู่ในเครือข่าย, Time-to-Live (TTL), และ Message Size ในแต่ละโหนดจะมีตารางเก็บข้อมูลของแต่ละข้อความที่อยู่ในบัฟเฟอร์ของโหนดนั้นๆ โดยตารางเก็บค่า message Id, my forwarding ซึ่งบอกว่าข้อความนี้เคยถูกรีเลย์จากโหนดไหนมาแล้วบ้าง, และค่า ETR

ก่อนที่ I จะเจอกับ J

ที่โหนด I	M1		1

ที่โหนด J	M2	G	2

หลังจากที่ I เจอกับ J : I ส่ง M1 ให้ J, J ส่ง M2 ให้ I

ที่โหนด I	M1	J	2
	M2	J,G	3

ที่โหนด J	M2	I	2
	M1	I	2

2.2 Mobility Model

การรูปแบบเคลื่อนที่ของโหนดในเครือข่ายที่มีความทนทานต่อความหน่วงสูงที่นิยมใช้ในผลงานวิจัยมักแบ่งออกเป็น 2 ประเภท ได้แก่ รูปแบบการเคลื่อนที่แบบสังเคราะห์ (Synthetic Mobility Model) ที่ใช้การสุ่มเส้นทาง (Path) ในการเคลื่อนที่ของโหนดเพื่อจำลองการเคลื่อนที่ให้มีลักษณะที่เหมือนของจริงให้มากที่สุด และรูปแบบการเคลื่อนที่แบบอาศัยข้อมูลจริง (Trace) เช่น การนำข้อมูลที่เก็บจากสถานที่จริงโดยการ Trace ข้อมูลของโหนดทดลองจากเครือข่ายจริง แล้วทำการแปลงข้อมูลที่เก็บได้ให้อยู่ในรูปแบบของ Path เพื่อนำไปใช้ในกับการจำลองสถานการณ์

2.2.1 Random Waypoint

ในการเคลื่อนที่แบบ Random Waypoint เป็น Synthetic Mobility Model โหนดจะเลือกจุดหมายปลายทางและความเร็วในการเคลื่อนที่แบบสุ่มใน Range ที่กำหนดไว้ เมื่อโหนดเคลื่อนที่ถึงจุดปลายทางที่ได้สุ่มไว้แล้วโหนดจะหยุดรอเวลาเพื่อทำการสุ่มจุดหมายปลายทางถัดไปแล้วจึงเคลื่อนที่ไปยังจุดหมายนั้น ในงานวิจัยนี้จะใช้ Random Waypoint ในการจำลองสถานการณ์ (รายละเอียดในบทที่ 3)

2.2.2 Random Walk

ในการเคลื่อนที่แบบ Random Walk เป็น Synthetic Mobility Model โหนดจะเลือกเส้นทาง ระยะทาง และความเร็วในการเคลื่อนที่โดยการสุ่มแล้วจึงเคลื่อนที่ไปยังจุดปลายทาง โดนระยะทางและความเร็วในการเคลื่อนที่จะถูกเลือกจากการสุ่มค่าที่กำหนดไว้ใน Range

บทที่ 3
กระบวนการจัดลำดับความสำคัญของข้อความในเครือข่ายที่มีความทนทานต่อความหน่วงสูง

การแลกเปลี่ยนข้อความในเครือข่าย Delay-Tolerant Network มีลักษณะเป็น Store and Forward นั่นคือ โหนดผู้รับจะรับข้อความมาจากโหนดผู้ส่งมาเก็บไว้ในบัฟเฟอร์ของโหนดผู้รับ (Store) เมื่อมีโอกาสที่จะส่งข้อความไปยังโหนดอื่นต่อจึงทำการส่ง (Forward) ไปยังโหนดอื่น วิธีการในการจัดลำดับของข้อความในการส่ง และวิธีในการเลือกโหนดเพื่อที่จะรีเลย์ข้อความต่อไป จึงเป็นสิ่งสำคัญมากในการที่จะทำให้โปรโตคอลมีประสิทธิภาพทั้งในด้านอัตราการส่งข้อความสำเร็จ ค่าความหน่วงของข้อความ รวมไปถึงในด้านของการประหยัดพลังงานของโหนดในเครือข่ายที่เป็นโหนดที่มีทรัพยากรด้านพลังงานและขนาดของบัฟเฟอร์ที่จำกัด เพราะโหนดใช้จำนวนครั้งในการส่งข้อความที่น้อยในการทำให้ข้อความเดินทางไปถึงยังโหนดเป้าหมาย

เพื่อที่จะสามารถออกแบบกระบวนการจัดลำดับความสำคัญและการ routing ในเครือข่าย Delay-Tolerant ได้อย่างมีประสิทธิภาพ เราจึงต้องแจกแจงปัจจัยต่างๆที่มีผลต่อประสิทธิภาพโดยรวมของโปรโตคอล จึงได้ผลจากการศึกษา ดังนี้

3.1 ปัจจัยที่มีผลต่อประสิทธิภาพของโปรโตคอลในด้านอัตราการส่งข้อความสำเร็จ (Delivery Rate) ค่าความหน่วง (Delivery Delay) และค่า Overhead

3.1.1 ความหนาแน่นของโหนดในเครือข่าย

ความหนาแน่นของโหนดในเครือข่าย เป็นค่าที่บ่งบอกถึงจำนวนโหนดต่อพื้นที่ในเครือข่าย หากในเครือข่าย DTN มีความหนาแน่นของโหนดสูง จะทำให้มีโอกาสที่โหนดใดๆจะเจอกัน (Contact) มากขึ้น ทำให้มีโอกาสที่ข้อความจะถูกรีเลย์จากโหนดหนึ่งไปยังอีกโหนดหนึ่งมากขึ้น ทำให้โอกาสที่ข้อความจะไปถึงโหนดเป้าหมายของข้อความมีมากขึ้น

3.1.2 จำนวนสำเนาของข้อความ (Copy of Message)

จำนวนสำเนาของข้อความเป็นปัจจัยที่สำคัญมากในโปรโตคอลหลายๆโปรโตคอลใน DTN [3][4] ยิ่งจำนวนของ Copy ของข้อความมีมาก โอกาสที่ข้อความจะไปถึงโหนดเป้าหมายก็จะมากขึ้นด้วย แต่ยิ่งมีการทำสำเนาของข้อความมากเท่าไร ก็ยิ่งทำให้เกิด ความซ้ำซ้อน (overhead) โดยไม่จำเป็นของข้อมูล ซึ่งจะทำให้สิ้นเปลืองเนื้อที่ในการจัดเก็บของข้อมูลตามมา

3.1.3 ขนาดของบัฟเฟอร์ของโหนด

บัฟเฟอร์คือหน่วยความจำของโหนดในการเก็บข้อความเพื่อรอการพบกันของโหนดครั้งต่อไปเพื่อรีเลย์ข้อความต่อไปยังโหนดอื่น ยิ่งขนาดบัฟเฟอร์ของโหนดมีขนาดใหญ่ก็สามารถเก็บ

ข้อความไว้ในบัฟเฟอร์ได้มาก ทำให้โหนดมีข้อความเพื่อรีเลย์ให้โหนดอื่นได้มาก เป็นผลทำให้อัตราการส่งข้อความสำเร็จของเครือข่ายโดยรวมมีมากขึ้น แต่ในทางกลับกัน หากโหนดมีขนาดของบัฟเฟอร์ที่น้อย ทำให้โหนดต้องดรอปข้อความใดข้อความหนึ่งในบัฟเฟอร์ทิ้งเพื่อให้มีที่ว่างสำหรับข้อความใหม่ที่กำลังเข้ามา ทำให้จำนวนครั้งการส่งจนกว่าจะถึงเป้าหมายของข้อความที่ถูกดรอปออกไปนั้นมีค่ามากกว่าปกติ เป็นผลทำให้ค่า Overhead โดยรวมของเครือข่ายสูงขึ้นได้

3.1.4 โปรโตคอลที่ใช้ในการส่งข้อความ

โปรโตคอลที่ใช้บนโหนดในเครือข่ายที่มีความทนทานต่อความหน่วงสูงนั้นมีผลในส่วนของลักษณะ วิธีในการเรียงลำดับและเลือกข้อความในการรีเลย์ไปยังโหนดอื่น รวมทั้งวิธีในการละทิ้ง (Drop) ข้อความในกรณีที่มีข้อความใหม่เข้ามาแต่ บัฟเฟอร์ของโหนดนั้นเต็ม

3.1.5 จำนวนและการกระจายตัวของข้อความในเครือข่าย

หากจำนวนข้อความในเครือข่ายมีจำนวนมาก จะทำให้เกิดการรีเลย์ข้อความระหว่างโหนดในจำนวนที่มาก ทำให้มีค่า Overhead โดยรวมในเครือข่ายสูงขึ้น

3.2 ผลของโมเดลการเคลื่อนที่ (Mobility Model) ต่อจำนวนการพบกันของโหนด

จำนวนการพบกันของโหนดซึ่งอาจบ่งบอกถึงความหนาแน่นของโหนดในเครือข่าย DTN มีผลต่ออัตราความสำเร็จในการส่งข้อความ (Delivery Rate) หากโหนดมีจำนวนครั้งของการพบกันมาก ก็มีโอกาสที่โหนดรีเลย์ข้อความให้กันระหว่างโหนดเยอะขึ้นตาม ทำให้ Delivery Rate โดยรวมสูงขึ้น

โมเดลการเคลื่อนที่ของโหนดในเครือข่ายของการวิจัยครั้งนี้จะใช้ Random Waypoint Model ซึ่งเป็นโมเดลแบบสังเคราะห์ ที่เป็นเคลื่อนที่แบบสุ่มทั้งจุดหมายปลายทางและความเร็วแบบสุ่มโดยค่าที่สุ่มจะอยู่ภายในช่วง (Range) ที่กำหนดไว้ เมื่อโหนดเคลื่อนที่ไปถึงจุดหมายปลายทางโหนดก็จะหยุดรอเพื่อสุ่มจุดหมายปลายทางและความเร็วถัดไป ซึ่งโมเดลการเคลื่อนที่แบบ Random Waypoint มีลักษณะตรงตามสมมติฐานในการวิจัยที่ตั้งไว้ใน**บทที่ 1**

3.3 สมมติฐานที่จะนำไปพัฒนาเป็นโปรโตคอลในเครือข่าย DTN

3.3.1 หากสามารถจัดลำดับของข้อความที่จะส่งในแต่ละ Contact Time ได้ดียิ่งขึ้น จะทำให้โปรโตคอลมีประสิทธิภาพที่ดีขึ้น

ในโปรโตคอลต่างๆในเครือข่าย DTN เช่น FuzzySpray [6], Enhanced Fuzzy-Based Spray and Wait [9] ใช้ค่า Priority ของข้อความในการจัดลำดับ Sending Queue ของโหนดนั้นๆ เพื่อให้โหนดสามารถส่งข้อความที่เหมาะสมต่อการส่งในขณะนั้นให้มากที่สุดเท่าที่จะเป็นไปได้ซึ่งเป็นการจัดลำดับข้อความของในแต่ละโหนดเท่านั้น เนื่องจากในการทดลองปัจจุบัน Connection ของโหนด

ทั้งหมดใน DTN จะยังเป็นแบบ Half-Duplex ซึ่งนั่นหมายความว่าฝั่งใดที่สร้าง Connection ก่อนก็จะเป็นฝ่ายส่งข้อความตามที่โหนดนั้นๆได้จัดลำดับไว้ ในหลายๆกรณีมีโอกาสที่โหนดอีกฝั่งหนึ่งซึ่งเป็นผู้รับ มีข้อความที่เหมาะสมกว่าข้อความที่โหนดอื่นฝั่งหนึ่งกำลังส่งมาใน Connection อยู่ (มีค่า Priority มากกว่าไม่ว่าจะเป็นผลมาจากมีค่า FTC ที่น้อยกว่า มีขนาดของข้อความที่น้อยกว่าหรือด้วยตัวแปรอื่นๆใดที่โปรโตคอลใช้ในการคิดค่า Priority) แต่โหนดผู้รับไม่สามารถส่งออกไปได้ ทำให้เสียโอกาสในการส่งข้อความที่เหมาะสมกว่าออกไป ซึ่งอาจเป็นผลให้โปรโตคอลทำงานได้ไม่เต็มประสิทธิภาพได้ หรือในบางโปรโตคอลสามารถตัดสินใจว่าจะรับข้อความที่กำลังเข้ามา (incoming message) หรือไม่ก็ได้โดยดูจากค่า Utility Metric บางอย่าง เช่นค่า TROP ในโปรโตคอล Encounter Fuzzy Logic Based Routing Protocol

ผู้วิจัยจึงนำเสนอวิธีการที่เรียกว่า Fuzzy-based Message Scheduling หรือเรียกว่า FMS มีหลักการทำงาน คือ เมื่อโหนดทั้งสองตัวเข้ามาอยู่ในระยะที่สามารถส่งข้อความกันได้และสร้าง Connection กันแล้วโหนดก็จะแลกเปลี่ยนข้อความที่มีโหนดอีกตัวเป็นโหนดเป้าหมายซึ่งจะเรียกข้อความแบบนี้ว่า Deliverable ในกรณีที่มี Deliverable จำนวนหลายข้อความ โหนดก็จะส่ง Deliverable ตามลำดับค่า Sending Priority ของ Deliverable นั้นจากมากไปน้อย หลังจากนั้นโหนดที่สร้าง Connection ก่อนจะจัดลำดับ Sending and Receiving Queue ที่ใช้ร่วมกันทั้งสองโหนดโดยนำข้อความของทั้งสองโหนดมาคิดค่า Sending Priority และจัดเรียงลำดับร่วมกันจะทำให้ได้ลำดับใน Sending and Receiving Queue ที่เหมาะสม ซึ่งข้อมูลที่จำเป็นในการทำ CSRQ จะได้มาจากการแลกเปลี่ยนข้อมูล (Summary) ที่จำเป็นในการจัดลำดับข้อความ ด้วยโปรโตคอลที่ผู้วิจัยใช้ในการทดลอง Implement ลักษณะการจัดลำดับข้อความคือโปรโตคอล FuzzySpray [6] ดังนั้นโหนดจึงต้องแลกเปลี่ยนข้อมูลค่า FTC และ Message Size ของข้อความทั้งหมด รายชื่อข้อความที่มีทั้งหมดของทั้งสองโหนดซึ่งข้อมูลทั้งหมดที่แลกเปลี่ยนกันนั้นเรียกว่า Summary Vector ซึ่งจำเป็นในการคิดค่า Sending Priority เมื่อได้แลกเปลี่ยนข้อมูล Summary Vector แล้วโหนดที่สร้าง Connection ก่อนจะเป็นฝ่ายคิด Sending Priority ของข้อความของทั้งสองโหนดเพื่อจัดลำดับการส่งข้อความด้วย Fuzzy Logic Controller แล้วจัดทำเป็น Common Sending and Receiving Queue (CSRQ) ซึ่งเป็น Queue ที่แสดงลำดับการรับและส่งข้อมูลที่ใช้ร่วมกันของทั้งสองโหนด แล้วข้อความจึงถูกแลกเปลี่ยนกันตามลำดับที่ถูกจัดไว้โดยโหนดที่เก็บข้อความนั้นๆ จะเป็นผู้ส่งข้อความ

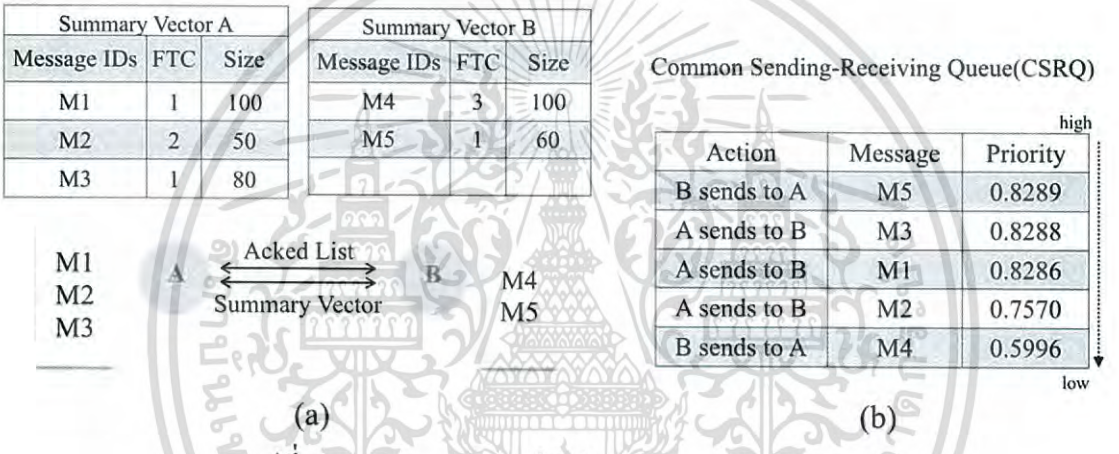
ในส่วนของ Dropping Policy ของโปรโตคอลนี้เป็นแบบ Drop Least Priority นั่นคือ ในกรณีที่มีข้อความที่ถูกสร้างขึ้นใหม่หรือมีข้อความกำลังเข้ามาแต่บัฟเฟอร์ของโหนดนั้นเต็ม โหนดก็จะทำการดรอปข้อความที่มี Sending Priority ต่ำที่สุดออกจากบัฟเฟอร์

Fuzzy-Based Message Scheduling (FMS)

Fuzzy-Based Message Scheduling (FMS) แบ่งการทำงานเป็นสองส่วนหลักๆ คือ Message Scheduling และ Fuzzy Logic Controller

Message Scheduling

ในส่วนของ Message Scheduling จะทำหน้าที่ในการสร้าง CSRQ และจัดลำดับข้อความในการรับส่งของ CSRQ จะใช้ข้อมูล Summary Vector ที่ประกอบไปด้วยรายชื่อข้อความทั้งหมดที่โหนดมี และข้อมูล Message Property ที่ประกอบไปด้วย ขนาดของข้อความและค่า FTC ของข้อความนั้น ที่แลกเปลี่ยนกันตอนที่โหนดแลกเปลี่ยนข้อความ Deliverable กันเสร็จเรียบร้อย มาใช้ในการทำเป็น CSRQ โดยการเรียงข้อความจะทำโดยการดูค่า Sending Priority จากมากไปน้อยซึ่งค่าSending Priority ของข้อความนั้นจะทำได้โดยการเรียกใช้ส่วนการทำงานของ Fuzzy Logic Controller ซึ่งจะอธิบายในส่วนถัดไป



รูปที่ 3.1 แสดงการทำงานในส่วนของ Message Scheduling

จาก**รูปที่ 3.1 (a)** เมื่อโหนด A และ B สร้าง Connection ระหว่างกันและแลกเปลี่ยนข้อความ Deliverable สำเร็จก็จะแลกเปลี่ยน Summary Vector กันเพื่อให้โหนดที่เป็นคนสร้าง Connection จัดทำ CSRQ ขึ้นจากข้อมูล Summary Vector ที่แลกเปลี่ยนกัน จะเห็นได้ว่าทั้งสองโหนดจะแสดงข้อมูลเกี่ยวกับข้อความที่จำเป็นในการคำนวณลำดับใน CSRQ นั่นคือ FTC และ Message Size ที่โหนด A มีข้อความ M1, M2, และ M3 อยู่ในบัฟเฟอร์ ในขณะที่โหนด B มีข้อความ M4, และ M5 อยู่ในบัฟเฟอร์ มีค่า FTC และ Message Size

จาก**รูปที่ 3.1 (b)** เมื่อทั้งสองโหนดได้แลกเปลี่ยน Summary Vector กันแล้ว โหนดที่เป็นผู้สร้าง Connection จะเป็นผู้สร้าง CSRQ ซึ่งเป็น Queue อ้างอิงลำดับการรับและส่งข้อความที่ทั้งสองโหนดจะใช้ร่วมกัน จะเห็นได้ว่า CSRQ จะแสดงข้อมูลว่าข้อความใดจะถูกส่งก่อนเพื่อให้ใช้ประสิทธิภาพของ Connection ได้สูงสุดโดยการเปรียบเทียบค่า Sending Priority ที่ถูกคำนวณด้วย Fuzzy Logic Controller โดยมีการเรียงข้อความที่มี Sending Priority จากมากไปน้อย (Buffer Section

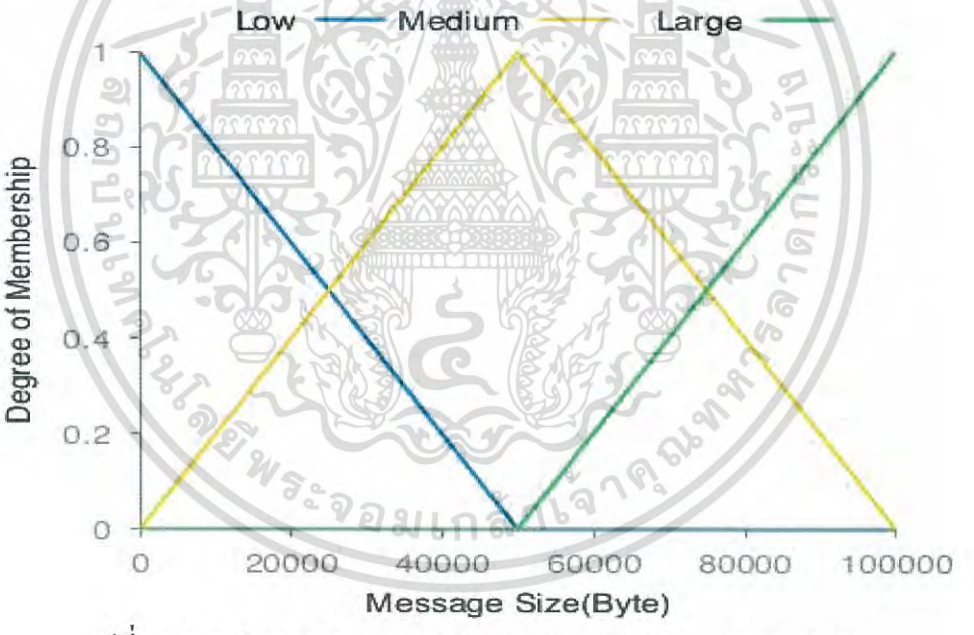
น้อยไปมาก) CSRQ ถูกสร้างขึ้น โหนดที่มีข้อความที่อยู่ในลำดับแรกของ CSRQ ก็จะเริ่มส่งข้อความไปยังอีกโหนดหนึ่ง

Fuzzy Logic Controller

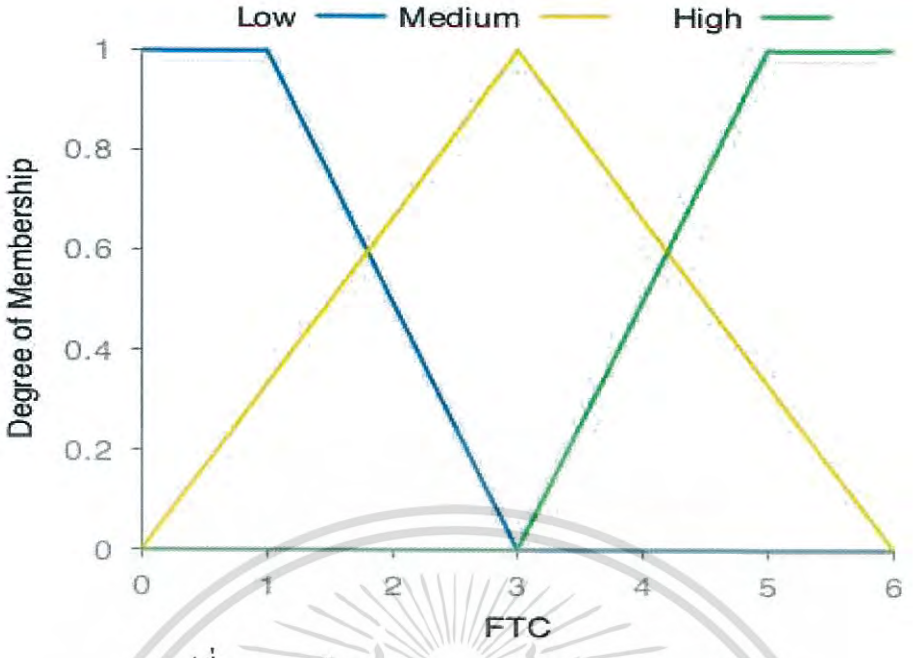
ในส่วนการทำงานนี้จะเป็นการใช้ Fuzzy Logic ในการคำนวณค่า Sending Priority ของข้อความเพื่อที่จะนำไปใช้ในการจัดลำดับการรับ-ส่ง ข้อความใน CSRQ

- Input ของ Fuzzy Logic Controller

ประกอบไปด้วย 2 ตัวแปรเช่นเดียวกับ FuzzySpray [6] นั่นคือ FTC ซึ่งเป็นค่าประมาณจำนวน Copy ของข้อความใดๆ เมื่อข้อความถูกสร้างขึ้นมาก็จะมีค่า FTC เท่ากับ 1 ซึ่งหมายความว่ามีข้อความนั้นแค่ข้อความเดียวในเครือข่าย ค่า FTC ของข้อความจะเพิ่มขึ้นทั้งในฝั่งผู้ส่งและผู้รับเมื่อมีการสเปรย์ข้อความเกิดขึ้นอย่างเสร็จสมบูรณ์ และ Message Size ซึ่งคือขนาดของข้อความมีหน่วยเป็น Byte ซึ่งสามารถแสดงเป็นกราฟได้ดังนี้



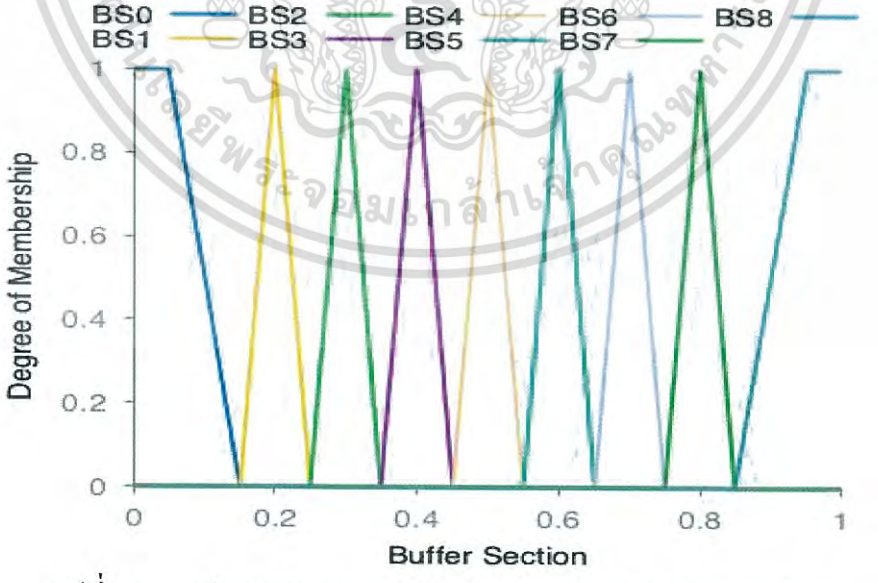
รูปที่ 3.2 กราฟแสดง Degree of Membership ของ Message Size(MS)



รูปที่ 3.3 กราฟแสดง Degree of Membership ของ FTC

- Output ของ Fuzzy Logic Controller

ผลลัพธ์ของ Fuzzy Logic Controller จะได้ออกมาเป็น Buffer Section (BS) ซึ่งแบ่งออกเป็น 9 Buffer Section ข้อความที่มี Sending Priority สูงที่สุดจะอยู่ใน Buffer Section 1 และข้อความที่มี Priority ที่น้อยจะอยู่ใน Buffer Section สุดท้ายนั่นคือ Buffer Section 8



รูปที่ 3.4 กราฟแสดง Degree of Membership ของ Buffer Section(BS)

ตารางที่ 3.1 แสดงค่า Buffer Section ของ FMSD2 ในการคิด Sending Priority ทั้ง 9 section

Sending Fuzzy Rule		
FTC	Message Size	Buffer Section
Low	Small	BS0
Low	Medium	BS1
Low	Large	BS2
Medium	Small	BS3
Medium	Medium	BS4
Medium	Large	BS5
High	Small	BS6
High	Medium	BS7
High	Large	BS8

Algorithm 1 Fuzzy-Based Message Scheduling

```
1: procedure FUZZYBASEDMESSAGESCHEDULING_A
2:     /*On node A, B is the current connected node */
3:     if ConnectionIsUp(B) then
4:         A.exchangeAckedList(B)
5:         A.exchangeDeliverables(B)
6:         exchangeSummayVector()
7:         deleteAckedMessages()
8:         CSRQ ← CSRQSorting(B)
9:         for each msg do in CSRQ
10:            if thisNode.contain(msg) then
11:                if B.makeRoomInBuffer(msg) then
12:                    sendMessage(msg, B)
13:
14: Function exchangeDeliverables(B)
15:     deliverables ← calPriority(deliverables)
16:     sendMessage(deliverables, B)
17: end functionexchangeDeliverables
18:
19: Function CSRQSorting(B)
20:     /* Message Property(MP) : Msg Id, Msg FTC,
21: Msg Size contains in Summary Vector*/
22:     allMsgs ← MP_thisNode + MP_B
23:     allMsgs ← calPriority(allMsgs)
24:     return sortByPriority(allMsgs)
25: end function CSRQSorting
26:
27: Function calPriority(msgs)
28:     for each (msg  in msgs) do
29:         msg.coaValue = fuzzy(msg)
30:         msg.priority = 1 - msg.coaValue
31:     return msgs
32: end function calPriority
33:
34: Function sendMessage(msg, toNode)
35:     otherNode.put(msg)
36:     if destination(msg) == otherNode then
37:         deleteMessage(msg)
38:         createACKMsg msg_(ack)
39: end function sendMsg
40:
41: Function fuzzy(msg)
42:     setFuzzyInput(FTC_msg, MS_msg)
43:     return getFuzzyOutput()
44: end function fuzzy
45:
46: Function makeRoomInBuffer(msg)
47:     if bufferFullByCreatingMessage then
48:         deleteMessage(Msg_Pmin)
49:     else if bufferFullByReceivingDeliverable
50:         deleteMessage(Msg_Pmin)
51: end function makeRoomInBuffer
```

รูปที่ 3.5 Pseudocode สำหรับโปรโตคอล FMS

3.3.2 หากโหนดในโปรโตคอล FMS มี Dropping Policy ที่เหมาะสม จะทำให้โปรโตคอลมีประสิทธิภาพโดยรวมที่ดี

โดยปกติโปรโตคอลในเครือข่าย DTN หากมีข้อความกำลังส่งเข้ามาที่โหนดผู้รับ โหนดผู้รับก็จะรับข้อความนั้นเก็บไว้ตามรูปแบบการทำงาน Store and forward เพื่อส่งต่อไปยังโหนดระหว่าง

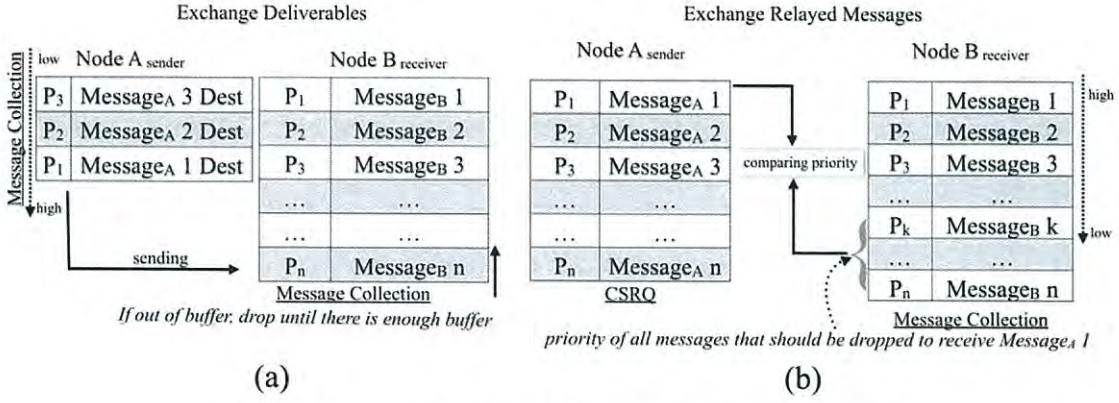
ทางอื่นๆ ซึ่งบางทีการที่โหนดรับข้อความโดยไม่มีหลักการใดๆเลยในการตัดสินใจว่าโหนดควรรับข้อความนั้นมาหรือไม่ ถ้าโหนดเลือกที่จะรับข้อความแต่บัฟเฟอร์ของโหนดไม่พอที่จะรองรับข้อความที่กำลังจะเข้ามาได้ โหนดควรจะต้องดรอปข้อความใดในบัฟเฟอร์จึงจะเหมาะสมที่สุด หากกระบวนการดังกล่าวออกแบบได้ไม่ดีก็อาจทำให้ประสิทธิภาพโดยรวมของโปรโตคอลแย่ลง เพราะในการที่โหนดจะรับข้อความใหม่เข้ามาอาจต้องทำการดรอปข้อความจำนวนหลายข้อความเพื่อให้มีพื้นที่ว่างพอสำหรับการรับข้อความใหม่ได้ ซึ่งอาจมีความเป็นไปได้ว่าข้อความที่โหนดรับเข้ามาอาจมีโอกาสโดยรวมที่จะส่งไปถึงเป้าหมายได้น้อยกว่าข้อความที่โหนดจะต้องดรอปไปเพื่อรับข้อความใหม่เข้ามาก็ได้

ดังนั้น ผู้วิจัยจึงเสนอ Dropping Policy สำหรับโปรโตคอล Fuzzy-Based Message Scheduling (FMS) ทำให้สามารถเรียกโปรโตคอลนี้ว่า Fuzzy-Based Message Scheduling and Dropping หรือ FMSD

ซึ่งมีส่วนการทำงานของ Message Scheduling และ Fuzzy Logic Controller ที่เหมือนกับ FMS ที่ได้เสนอไปในหัวข้อที่ 3.3.4 แต่เพิ่ม Dropping Policy ที่เหมาะสมเข้าไป

3.3.2.1 หลักการทำงานของ Dropping Policy

- ในกรณีที่บัฟเฟอร์ของโหนดเต็มเนื่องจากข้อความถูกสร้างขึ้นมาใหม่โดยโหนดนั้นๆ จะดรอปข้อความที่มีค่า Sending Priority ที่ต่ำที่สุด (ข้อความที่อยู่ลำดับสุดท้ายของ Sending Queue)
- ในกรณีที่บัฟเฟอร์ของโหนดผู้รับเต็มและโหนดอื่นกำลังจะส่งข้อความใหม่มาให้โหนดผู้รับ โหนดผู้รับจะเอา Sending Priority ของข้อความที่กำลังเข้ามาเทียบกับ Sending Priority ของข้อความทั้งหมดในบัฟเฟอร์ของโหนดผู้รับที่ละข้อความ หากข้อความที่กำลังจะเข้ามามีค่า Priority ที่สูงกว่าข้อความที่เทียบอยู่ ก็จะใส่ข้อความที่เทียบอยู่ในฝั่งผู้รับเข้าไปใน Dropping List ของข้อความที่จะถูกดรอป โดยโหนดผู้รับจะเทียบค่า Sending Priority แบบนี้ไปเรื่อยๆจนกว่าจะมีพื้นที่พอที่จะรับข้อความใหม่เข้ามาได้ แต่ถ้าหากข้อความที่กำลังจะเข้ามามีค่า Sending Priority น้อยกว่าข้อความใดข้อความหนึ่งในบัฟเฟอร์ของโหนดผู้รับ หรือเมื่อค่า Sending Priority รวมของข้อความใน Dropping List มีค่าสูงกว่า Sending Priority ของข้อความที่โหนดกำลังจะรับเข้ามา โหนดผู้รับก็จะปฏิเสธข้อความต่อโหนดผู้ส่งไป
- ในกรณีที่บัฟเฟอร์ของโหนดผู้รับเต็มและโหนดอื่นกำลังจะส่งข้อความใหม่มาให้โหนดผู้รับและข้อความนั้นเป็นข้อความที่มีโหนดผู้รับเป็น Final Recipient โหนดผู้รับจะต้องบังคับดรอปข้อความที่มี Sending Priority ที่ต่ำที่สุด



รูปที่ 3.6 แสดงการทำงานของ Dropping Policy ของ FMSD

จาก**รูปที่ 3.6 (a)** ในกรณีที่บัฟเฟอร์ของโหนดผู้รับเต็มและโหนดอื่นกำลังจะส่งข้อความใหม่มาให้โหนดผู้รับและข้อความนั้นเป็นข้อความที่มีโหนดผู้รับเป็น Final Recipient หรือเรียกข้อความประเภทนี้ว่า Deliverable โหนดผู้รับจะต้องบังคับดรอปข้อความที่มี Sending Priority ที่ต่ำที่สุดซึ่งก็คือ $Message_Bn$ ของโหนด B

จาก**รูปที่ 3.6 (b)** จะเห็นได้ว่าในกรณีที่เป็นการสเปรย์ข้อความระหว่างโหนดผู้รับและโหนดผู้ส่ง หากบัฟเฟอร์ของโหนดผู้รับเต็ม โหนดผู้รับก็จะเปรียบเทียบ Sending Priority ระหว่างข้อความที่กำลังเข้ามากับข้อความในบัฟเฟอร์ของโหนดผู้รับ แล้วจึงตัดสินใจว่าจะรับข้อความที่กำลังเข้ามาหรือไม่ หากตัดสินว่าจะรับข้อความที่กำลังเข้ามาต้องทำการดรอปข้อความใดในบัฟเฟอร์บ้าง ดังที่แสดงในตัวอย่าง โหนด B ต้องดรอปข้อความ $message_Bk$ ถึง $message_Bn$ เพื่อรับข้อความ $message_A1$ ที่มี Sending Priority P_1 สูงกว่า $\sum(P_{k_B}, \ldots, P_{n_B})$

Algorithm 1 Fuzzy-Based Message Scheduling

```
1: procedure FUZZYBASEDMESSAGESCHEDULING_A
2:     /*On node A, B is the current connected node */
3:     if ConnectionIsUp(B) then
4:         A.exchangeAckedList(B)
5:         A.exchangeDeliverables(B)
6:         exchangeSummayVector()
7:         deleteAckedMessages()
8:         CSRQ ← CSRQSorting(B)
9:         for each msg do in CSRQ
10:            if thisNode.contain(msg) then
11:                if B.canAcceptMessage(msg) then
12:                    sendMessage(msg, B)
13:
14: Function exchangeDeliverables(B)
15:     deliverables ← calPriority(deliverables)
16:     sendMessage(deliverables, B)
17: end functionexchangeDeliverables
18:
19: Function CSRQSorting(B)
20:     /* Message Property(MP) : Msg Id, Msg FTC,
21: Msg Size contains in Summary Vector*/
22:     allMsgs ← MP_thisNode + MP_B
23:     allMsgs ← calPriority(allMsgs)
24:     return sortByPriority(allMsgs)
25: end function CSRQSorting
26:
27: Function calPriority(msgs)
28:     for each (msg in msgs) do
29:         msg.coaValue = fuzzy(msg)
30:         msg.priority = 1 - msg.coaValue
31:     return msgs
32: end function calPriority
33:
34: Function sendMessage(msg, toNode)
35:     otherNode.put(msg)
36:     if destination(msg) == otherNode then
37:         deleteMessage(msg)
38:         createACKMsg msg_(ack)
39: end function sendMsg
40:
41: Function fuzzy(msg)
42:     setFuzzyInput(FTC_msg, MS_msg)
43:     return getFuzzyOutput()
44: end function fuzzy
45:
46: Function canAcceptMessage(msg)
47:     Priority_msg ← fuzzy(msg)
48:     if messageIsBiggerThanBuffer(msg) then
49:         return False
50:         /*incoming message is bigger than receiving node
51: buffer, reject it.
52:         while freeBuffer < Size_msg do:
53:             if hopCount_MPmin > 1 then
54:                 PriorityDropped += fuzzy(M_Pmin)
55:                 /* M_Pmin : message with least priority in buffer */
56:                 if Priority_msg <= PriorityDropped then
57:                     return False
58:                 freeBuffer += Size_m;
59:                 DroppingList.add(m)
60:     deleteMessageIn(DroppingList)
61:     return True
62: end function canAcceptMessage
63:
64: Function makeRoomInBuffer()
65:     if bufferFullByCreatingMessage then
66:         deleteMessage(Msg_Pmin)
67:     else if bufferFullByReceivingDeliverable
68:         deleteMessage(Msg_Pmin)
69: end function makeRoomInBuffer
```

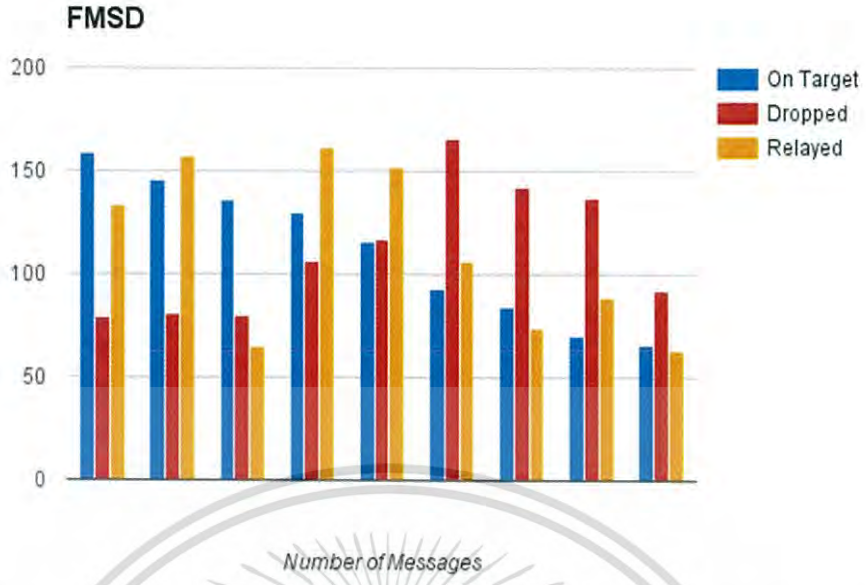
รูปที่ **3.7** Pseudocode สำหรับโปรโตคอล FMSD

3.3.3 หากโปรโตคอลมี Fuzzy Rule ที่ทำหน้าที่เฉพาะในการคิดค่า Dropping Policy ที่ดีขึ้น จะทำให้โปรโตคอล FMSD มีประสิทธิภาพที่ดีมากขึ้นทั้งในด้าน Delivery Probability, Overhead, และ Delivery Delay

โปรโตคอล Fuzzy-Based Message Scheduling and Dropping (FMSD) ในหัวข้อ 3.3.5 มี Dropping Policy โดยการเทียบค่า Sending Priority ของข้อความที่กำลังเข้ามากับข้อความที่กำลังจะถูกดรอป ค่า Sending Priority ที่ใช้เทียบนั้นคือค่า Sending Priority ที่คำนวณโดย Fuzzy Rule ตัวเดียวกับที่ใช้สำหรับการจัดลำดับการส่งข้อความซึ่งใช้ FTC และ Message Size เป็น Input ของ Fuzzy Logic Controller ซึ่งเมื่อดูผลการทดลองแล้ว ผู้วิจัยพบว่าพฤติกรรมการดรอปข้อความของ FMSD นั้นยังไม่มีความ Fairness เท่าที่ควร สังเกตได้จาก**รูปที่ 3.8** แสดงจำนวนการดรอปข้อความ FMSD มีการดรอปข้อความที่มีขนาดใหญ่มาก ขนาดกลางรองลงมาและดรอปข้อความที่มีขนาดเล็กน้อยที่สุดจนทำให้ Delivery Delay โดยรวมของโปรโตคอลไม่ดีเท่าที่ควร เพราะข้อความขนาดใหญ่ที่มีโอกาสถูกสเปรย์น้อยอยู่แล้วเนื่องจาก Sending Priority มีน้อย (เนื่องจากลักษณะของ Sending Fuzzy Rule ที่ใช้ FTC และ Message Size ในการคำนวณ Sending Priority) ทำให้ข้อความขนาดใหญ่มีโอกาสที่จะส่งไปถึงยังโหนดเป้าหมายของข้อความได้น้อยกว่าข้อความที่มีขนาดกลางหรือขนาดเล็กเป็นผลให้ Delivery Delay ของข้อความขนาดใหญ่ก็มีค่าที่สูงกว่าข้อความขนาดกลางและเล็กอยู่แล้ว เมื่อถูกดรอปออกจากบัฟเฟอร์ของโหนดจำนวนมากในเครือข่ายก็ยิ่งทำให้ Delivery Delay โดยรวมสูงขึ้นด้วย

ดังนั้นผู้วิจัยจึงคิด Fuzzy Rule ชุดใหม่สำหรับใช้ในการคิดค่า Dropping Priority สำหรับโปรโตคอล FMSD โดยเฉพาะ ผู้วิจัยจึงขอเรียกโปรโตคอลที่ใช้ Fuzzy Rule ชุดนี้ว่า Fuzzy-Based Message Scheduling and Dropping 2 (FMSD2) ซึ่งมีส่วนหลักการทำงานอยู่ทั้งหมด 3 ส่วน ได้แก่ 1). Message Scheduling ที่ทำงานเหมือนกับ FMS และ FMSD, 2). Fuzzy Logic Controller ที่ประกอบไปด้วย Fuzzy Rule 2 ตัว, และ 3). Dropping Policy

ในโปรโตคอล FMSD2 ส่วนที่ทำงานแตกต่างจาก FMS และ FMSD คือ เมื่อโหนดทั้งสองตัวสร้าง Connection กันสำเร็จเรียบร้อยแล้ว โหนดจะทำการแลกเปลี่ยนลิสต์ของข้อความที่ได้รับ Acknowledgement แล้วหรือเรียกว่า Ack List และแลกเปลี่ยนข้อความ Deliverable ในลักษณะเดียวกับโปรโตคอล FMS และ FMSD ต่อมาจึงทำการแลกเปลี่ยน Summary Vector ที่ใช้ในการคิดค่า Sending Priority และอัพเดทค่า Average FTC และ Average Delay ที่โหนดต้องใช้ในการคิดค่า Dropping Priority



รูปที่ 3.8 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) จำนวน 1000 ข้อความแรกของโปรโตคอล FMSD

วิธีการแลกเปลี่ยนและอัพเดทค่า Average FTC และ Average Delay ของโหนดนั้น คือ อาศัยข้อมูล FTC และ Delay ของข้อความที่โหนดนั้นๆส่งออกไปและได้รับ Ack กลับมาแล้วมาคิดเป็นค่าเฉลี่ย รวมทั้งเมื่อโหนดทั้งสองสร้าง Connection สำเร็จแล้วก็จะแลกเปลี่ยนข้อมูล FTC และ Delay ของข้อความที่โหนดนั้นๆส่งออกไปเพื่อเป็นข้อมูลให้ทั้งสองโหนดคิดเป็นค่า Average FTC และ Average Delay ซึ่งทำให้โหนดสามารถตีความได้ว่าค่าเฉลี่ยของจำนวน Copy และ Delay ของข้อความที่ส่งไปถึงโหนดเป้าหมายของข้อความมีค่าประมาณเท่าใด

3.3.3.1 Fuzzy Logic Controller

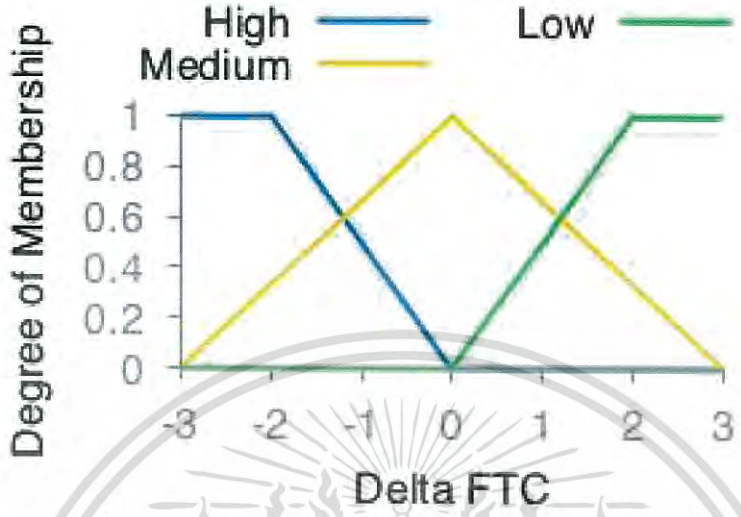
ในส่วนการทำงานของ Fuzzy Logic Controller จะประกอบไปด้วย Fuzzy Rule 2 ชุด ผู้วิจัยเรียกว่า Sending Fuzzy Rule(FTC, Message Size) ซึ่งมีชุดของ Rule ต่างๆเหมือนกับ Rule ของ FMS และ FMSD ทุกประการและ Dropping Fuzzy Rule ซึ่งสามารถอธิบายส่วนหลักของ Fuzzy Logic Controller ได้ดังนี้

- **Sending Fuzzy Rule** ที่มีรายละเอียดของชุดของ Rule เหมือนกับ FMS และ FMSD
- **Dropping Fuzzy Rule**

เป็น Fuzzy Rule ที่ใช้ในการคิดค่า Dropping Priority ที่ Dropping Policy จะใช้ในการจัดลำดับการดรอปข้อความ ใช้ Input สองค่าคือ Delta Delay และ Delta FTC และมีค่า Output เป็น Dropping Priority ซึ่งจะอธิบายในส่วนถัดไป

3.3.3.1.1 Input ของ Dropping Fuzzy Rule

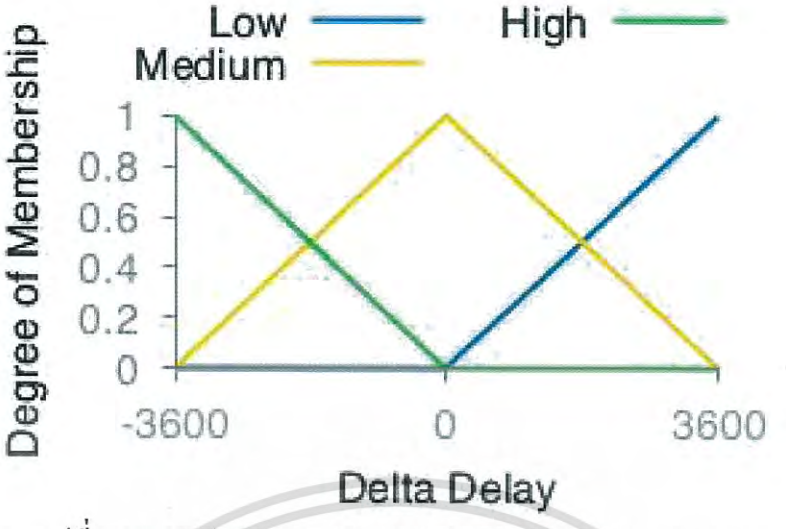
- **Delta FTC** คือ ค่าผลต่างของ Average FTC ที่โหนดนั้นๆส่งออกไปกับ FTC ของข้อความนั้น ๆ



รูปที่ 3.9 กราฟแสดง Membership Function ของ Delta FTC

หากค่า FTC ของข้อความนั้นๆ มีค่ามากกว่า Average FTC นั่นหมายความว่า ข้อความนั้นมีจำนวน copy มากกว่าจำนวน copy ของข้อความส่วนมากที่โหนดได้ส่งออกไปซึ่งหมายความว่าโหนดสามารถดรอปข้อความนี้ได้และมีโอกาสน้อยมากที่ข้อความจะเกิด lost ในทางกลับกันหากค่า FTC ของข้อความนั้นๆ มีค่าน้อยกว่า Average FTC แสดงว่าข้อความนั้นยังมีจำนวน copy ของข้อความที่น้อยกว่าค่าเฉลี่ย FTC ของข้อความที่โหนดได้ส่งออกไปจึงยังไม่ควรดรอปข้อความนี้เพื่อป้องกันการ Lost ของข้อความ

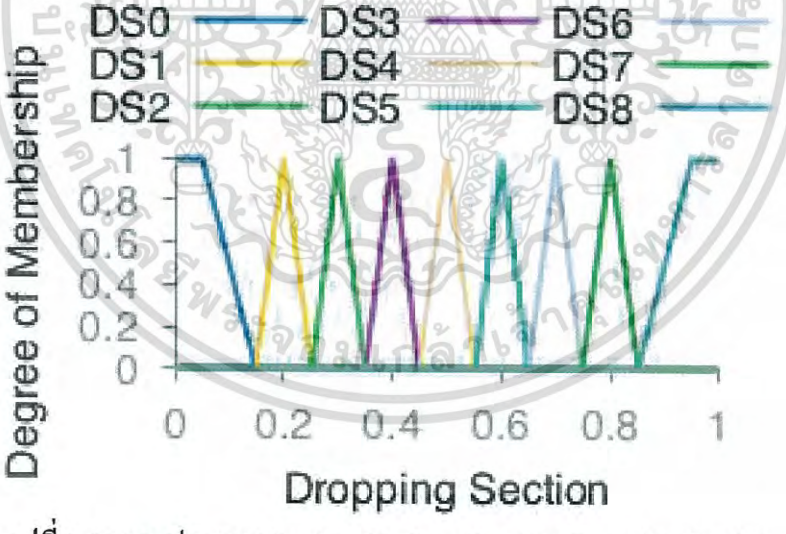
- **Delta Delay** คือ ค่าผลต่างของ Average Delay ของข้อความที่โหนดนั้นๆส่งออกไปกับ Elapsed Time ของข้อความ(เวลาตั้งแต่ข้อความถูกสร้างจนถึงปัจจุบัน) หาก Elapsed Time ของข้อความนั้นมีค่ามากกว่า Average Delay ของข้อความทั้งหมดที่โหนดนั้นๆส่งออกไป หมายความว่าโหนดยังไม่สามารถส่งข้อความนั้นไปยังเป้าหมายได้เมื่อบัฟเฟอร์ของโหนดเต็ม โหนดควรดรอปข้อความนั้นออกเพราะข้อความนั้นน่าจะยังไม่สามารถส่งไปยังเป้าหมายได้ ในทางกลับกัน หากข้อความนั้นมี Elapsed Time ที่น้อยกว่า Average Delay นั่นหมายความว่าข้อความนั้นถูกสร้างขึ้นมาเป็นเวลาไม่นานจึงน่าจะยังไม่มีโอกาสที่จะกระจาย (Relayed) ไปยังโหนดอื่นๆ โหนดจึงไม่ควรดรอปข้อความนั้นออก เพราะอาจทำให้เกิด Lost ได้



รูปที่ 3.10 กราฟแสดง Membership Function ของ Delta Delay

3.3.3.1.2 Output ของ Dropping Fuzzy Rule

ผลลัพท์ของ Dropping Fuzzy Rule จะได้เป็นค่า Dropping Priority ของข้อความนั้นๆ ซึ่งแบ่งออกเป็น 9 Dropping Section โดยข้อความที่อยู่ใน Dropping Section ที่ต่ำกว่าจะเป็นข้อความที่ถูกดรอปออกจากบัฟเฟอร์ก่อน สามารถแสดงเป็นกราฟได้ดังนี้



รูปที่ 3.11 กราฟแสดง Membership Function ของ Dropping Section

ตารางที่ 3.2 แสดงค่า Dropping Section ของ FMSD2 กระบวนการ Dropping policy ทั้ง 9 section

FTC	Delay	BS
low	high	BS_0
low	medium	BS_1
low	low	BS_2
medium	high	BS_3
medium	medium	BS_4
medium	low	BS_5
high	high	BS_6
high	medium	BS_7
high	low	BS_8

3.3.3.2 Dropping Policy

ส่วนการทำงานนี้จะถูกเรียกใช้เมื่อโหนดมีข้อความที่กำลังจะรับเข้ามาหรือมีข้อความถูกสร้างขึ้นใหม่ โหนดก็ต้องตัดสินใจว่าจะทำอย่างไร ในกรณีที่มีข้อความกำลังส่งเข้ามาที่โหนด โหนดควรจะรับข้อความที่กำลังเข้ามาหรือไม่ ถ้าตัดสินใจว่าจะรับข้อความก็ต้องตัดสินใจต่อว่าควรดรอปข้อความไหนออกเพื่อที่จะทำให้มีพื้นที่ว่างในบัฟเฟอร์ที่จะรับข้อความใหม่เข้ามา หรือในกรณีที่มีข้อความถูกสร้างขึ้นมาใหม่ที่โหนดแต่บัฟเฟอร์ของโหนดเต็ม โหนดก็ต้องตัดสินใจว่าจะดรอปข้อความใดทิ้งเพื่อให้มีที่ว่างสำหรับข้อความที่ถูกสร้างขึ้นมาใหม่ ในการเลือกดรอปข้อความนั้นโหนดจะเลือกดรอปข้อความที่มี Dropping Priority ที่มากที่สุด

3.3.3.2.1 หลักการทำงานของ Dropping Policy

- ในกรณีที่โหนดสร้างข้อความขึ้นมาใหม่ แต่บัฟเฟอร์ของโหนดเต็ม โหนดจะเรียกใช้ Dropping Policy ในการคิดค่า Dropping Policy ของข้อความทั้งหมดในบัฟเฟอร์แล้วจึงทำการดรอปข้อความที่มี Dropping Policy ค่าที่สูงที่สุด
- ในกรณีที่บัฟเฟอร์ของโหนดผู้รับเต็มและมีข้อความ m กำลังส่งมายังโหนดผู้รับ โหนดผู้รับจะคิดค่า Dropping Priority ของข้อความ m ที่กำลังส่งมายังโหนดผู้รับ แล้วเอาไปเทียบกับ Dropping Priority ของข้อความ n ที่สูงที่สุดในบัฟเฟอร์ของโหนดผู้รับ หาก Dropping Priority ของข้อความ n ในบัฟเฟอร์มีค่ามากกว่าของข้อความที่กำลังเข้ามา โหนดผู้รับก็จะเพิ่มข้อความ n ไปยัง Dropping List จนกว่าจะมีพื้นที่ในบัฟเฟอร์ของโหนดผู้รับเหลือพอที่จะรับข้อความ m ได้ แต่ถ้าหากค่า Dropping Priority รวมของ Dropping List มีค่ามากกว่า Dropping Priority ของ m โหนดผู้รับก็จะปฏิเสธการรับข้อความ m

- ในกรณีที่บัฟเฟอร์ของโหนดผู้รับเต็มและโหนดอื่นกำลังจะส่งข้อความใหม่มาให้โหนดผู้รับและข้อความนั้นเป็นข้อความที่มีโหนดผู้รับเป็น Final Recipient โหนดผู้รับจะต้องบังคับดรอปข้อความที่มี Dropping Priority ที่ต่ำที่สุด

บทที่ 4

ผลการทดลอง

ในการทดลองเพื่อให้ได้ผลการวิจัย ผู้วิจัยเลือกใช้ ONE Simulator ในการจำลองสภาพแวดล้อมในเครือข่าย Delay-Tolerant Network ซึ่งเป็น Simulator ที่ถูกพัฒนาด้วยภาษา Java

4.1 ตัวแปรที่ใช้วัดประสิทธิภาพของโปรโตคอลในเครือข่าย Delay-Tolerant Network

4.1.1 Delivery Probability

ค่าที่บอกว่าส่งข้อความสำเร็จไปแล้วกี่เปอร์เซ็นต์จากจำนวนข้อความทั้งหมดที่ถูกสร้างขึ้นในช่วงเวลาที่ทดลอง

4.1.2 Overhead Ratio

ค่าที่บอกว่าข้อความหนึ่งจะต้องใช้การรีเลย์ทั้งหมดกี่ครั้งถึงจะส่งได้ถึงโหนดเป้าหมาย โดยคิดจาก

$$\text{Overhead ratio} = \frac{\text{nrofRelayed} - \text{nrofDeliverd}}{\text{nrofDeliverd}} \times 1.00 \quad (4.1)$$

4.1.3 Delivery Delay

ค่าที่บอกว่าข้อความหนึ่งจะใช้เวลาทั้งหมดเท่าไหร่ตั้งแต่ข้อความถูกสร้างไปจนส่งไปถึงโหนดเป้าหมาย โดยจะ นำแค่ค่า Delivery Delay ของข้อความที่ส่งไปถึงโหนดเป้าหมายเท่านั้น

4.2 สภาพแวดล้อมที่ใช้ในการทดลอง

Time	**12 hours (43200 seconds)**
Message Size	**10kB – 100k**
Time Created	**0 second – 43200 second**
Message Creation Interval	**1 – 5 seconds**
Host	**60**
Buffer Size	**3MB, 6MB, 9MB, 12MB**
Moving Speed	**2.78m/s – 13.89m/s**
Time-to-Live	**600 minutes**
Transmission Speed	**250kbps**
Transmission Range	**30 m**
Map Size	**3220 m x 3220 m**

4.3 ผลการทดลอง

4.3.1 ผลการทดลองของสมมติฐานที่ 3.3.1

จาก**หัวข้อ 3.3.1** ที่เสนอสมมติฐานว่า **หากสามารถจัดลำดับของข้อความที่จะส่งในแต่ละ Contact Time ได้ดียิ่งขึ้น จะทำให้โปรโตคอลมีประสิทธิภาพที่ดีขึ้น** จะสังเกตได้ว่าโปรโตคอล FMS ที่การทดลองที่มีขนาดบัฟเฟอร์เป็น 3MB ที่มีเป้าหมายในการทดสอบประสิทธิภาพของโปรโตคอลในสภาพที่มีบัฟเฟอร์จำกัด ซึ่งการเลือกข้อความที่เหมาะสมในการแลกเปลี่ยนระหว่าง Contact Time แต่ละครั้งมีผลมากในการเพิ่มโอกาสที่ข้อความจะถูกส่งไปถึงโหนดเป้าหมายและลดโอกาสที่ข้อความจะถูกดรอป เห็นได้จากค่า Overhead ของ FMS ที่น้อยกว่า FuzzySpray เพราะในโปรโตคอล FMS จะแลกเปลี่ยนข้อความตาม Sending Priority ของทั้งสองโหนดในรูปแบบของ CSRQ ทำให้ใน Contact Time สั้นๆนั้นโหนดได้แลกเปลี่ยนข้อความที่มีโอกาสไปถึงเป้าหมายได้จริงๆ ทำให้โหนดที่รับข้อความไปมีโอกาสที่จะต้องดรอปข้อความต่างๆที่รับมาน้อยมาก ข้อความสามารถไปถึงเป้าหมายได้รวดเร็ว (สามารถสังเกตได้จากรูปที่ 4.8) ส่งผลดีต่อค่า Delivery Probability, Delivery Delay และค่า Overhead ที่ดีขึ้นอย่างเห็นได้ชัด แต่ FuzzySpray หรือ FS รวมทั้งโปรโตคอลอื่นๆจะมีแค่โหนดที่สร้าง Connection ก่อนเท่านั้นจะเป็นผู้ที่ส่งข้อความตาม Sending Priority ของโหนดนั้นๆเองภายใน Contact Time ที่จำกัดให้ได้มากที่สุด จะสังเกตจาก **รูปที่ 4.1** และ **รูปที่ 4.2** ได้ว่าโดยเฉลี่ยแล้ว FuzzySpray จะต้องใช้จำนวนการ Relay มากกว่า FMS ประมาณ 6%

ในกรณีที่มีบัฟเฟอร์ขนาดใหญ่นั่นหมายความว่าโหนดมีพื้นที่รองรับข้อความจำนวนมาก โอกาสที่จะเกิดการดรอปข้อความจึงน้อยกว่าในกรณีที่บัฟเฟอร์มีขนาดที่เล็ก ผลของ การจัดลำดับการรับ-ส่งข้อความภายใน Contact Time ที่มีต่อโอกาสเกิดการดรอปจึงน้อยลง เพราะพื้นที่บัฟเฟอร์ในโหนดผู้รับก็มีโอกาสเต็มได้น้อยอยู่แล้ว แต่มีผลโดยตรงต่อการเพิ่มหรือลดของโอกาสที่ข้อความจะส่งไปถึงโหนดเป้าหมายได้โดยใช้ Delivery Delay ที่น้อยกว่า ผลการทดลองปรากฏว่า FMS ได้ผลดีกว่า FuzzySpray เพียงเล็กน้อยในทุกๆ ด้าน เนื่องจาก FuzzySpray ซึ่งเป็นโปรโตคอลที่มีลักษณะเป็น Spray-Based Protocol นั่นคือ เป็นโปรโตคอลที่ถูกออกแบบให้ส่ง (สเปรย์) ข้อความให้มากที่สุดเท่าที่จะทำได้ภายใน Contact Time ที่จำกัด ด้วยสภาพแวดล้อมการทดลองที่บัฟเฟอร์ขนาด 12MB FuzzySpray จึงสามารถสเปรย์ข้อความให้โหนดผู้รับได้จำนวนมากซึ่งสังเกตได้จาก **รูปที่ 4.3** และ **รูปที่ 4.4** แสดงจำนวนการสเปรย์ข้อความ ในสภาพแวดล้อมการทดลองนี้ FMS สามารถจัดลำดับข้อความเพื่อให้ได้ Delivery Probability และ Delivery Delay ได้อย่างมีประสิทธิภาพซึ่งสามารถสังเกตได้จากรูปที่ 4.10 จะเห็นได้ว่า FMS ใช้จำนวนการ Relay ที่น้อยกว่า FuzzySpray ประมาณ 8.5% ทำให้มี Overhead ที่น้อยกว่า แต่ยังมี Delivery Probability ที่ดีกว่า FuzzySpray เล็กน้อย

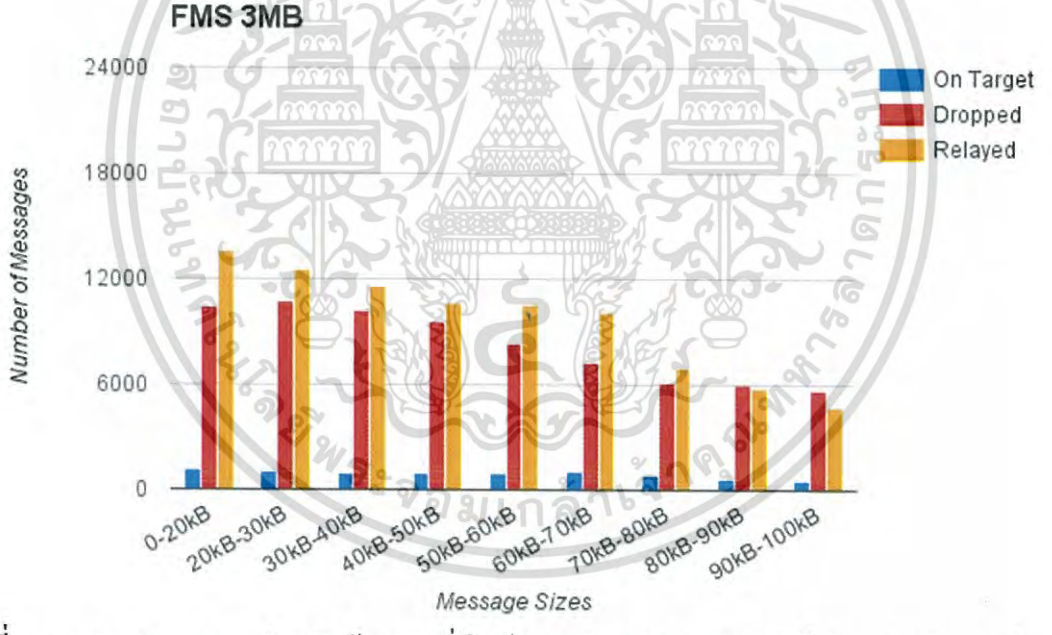
ดังนั้น จะเห็นได้ว่าโปรโตคอล FMS มีความ Fairness ในด้านของการส่งข้อความทุกๆขนาดไปถึงโหนดเป้าหมายในจำนวนที่เท่าๆกัน

ตารางที่ 4.1 แสดงผลการทดลองที่ขนาดบัฟเฟอร์ 3MB ของ FMS เปรียบเทียบกับ FuzzySpray

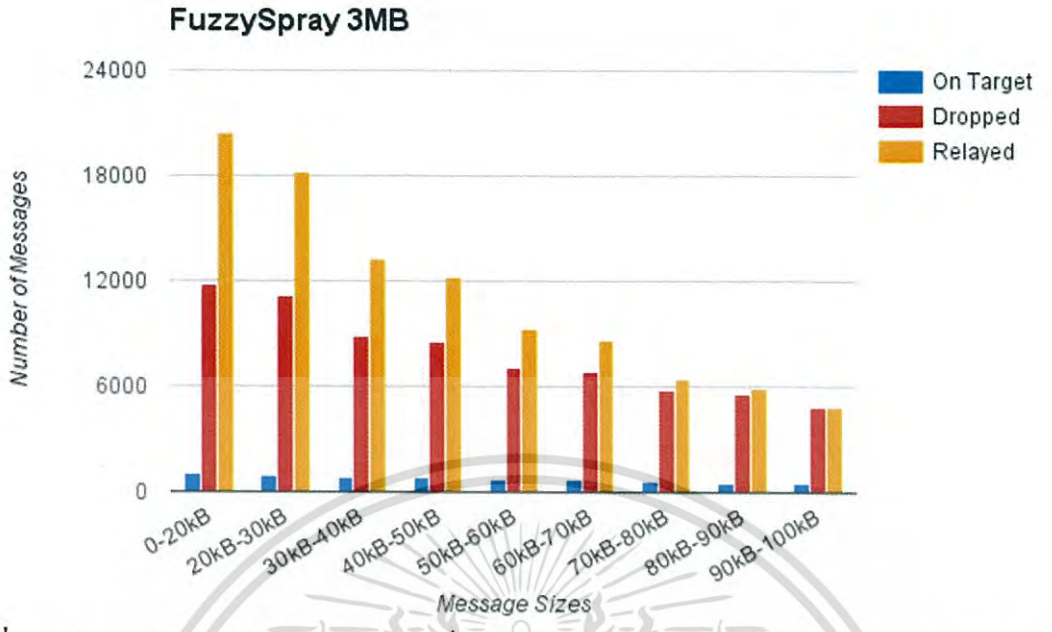
3MB	Delivery Probability	Overhead	Delivery Delay (s)
FMS	45.35%	11.15	2154.12
FuzzySpray	39.73%	15.08	2812.07

ตารางที่ 4.2 แสดงผลการทดลองที่ขนาดบัฟเฟอร์ 12MB ของ FMS เปรียบเทียบกับ FuzzySpray

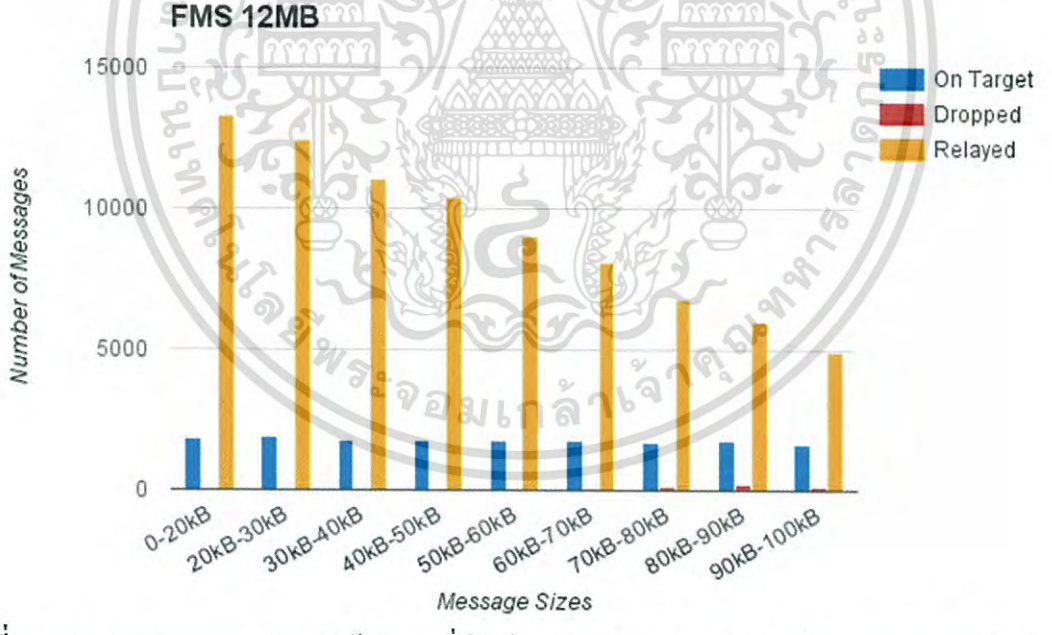
12MB	Delivery Probability	Overhead	Delivery Delay (s)
FMS	90.90%	5.06	3553.34
FuzzySpray	89.30%	5.53	4074.12



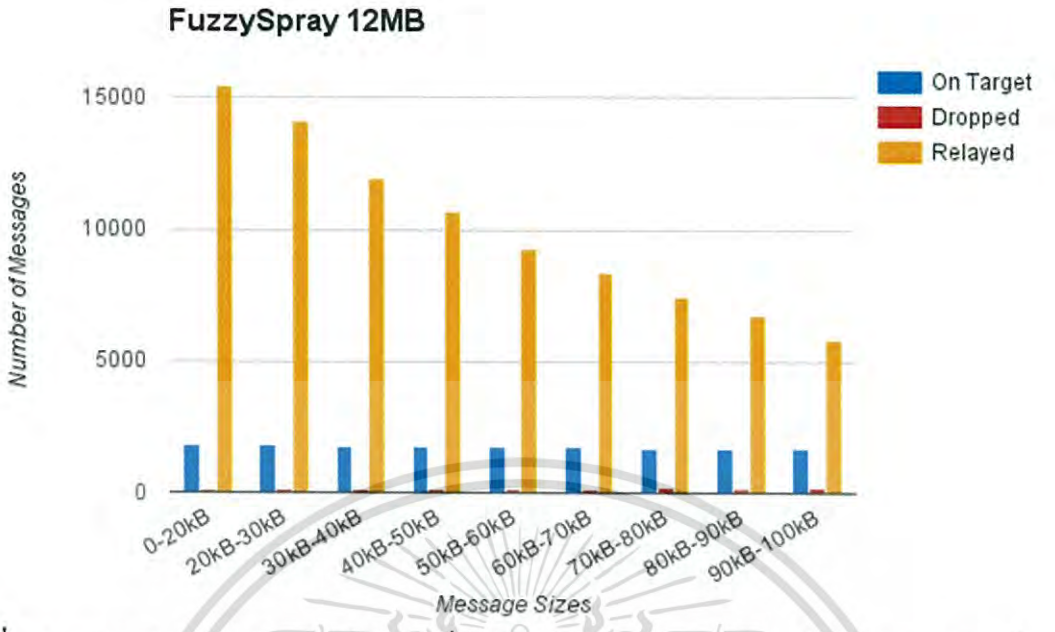
รูปที่ 4.1 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) ของโปรโตคอล FMS ที่สภาพแวดล้อม 3MB



รูปที่ 4.2 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) ของโปรโตคอล FuzzySpray ที่สภาพแวดล้อม 3MB



รูปที่ 4.3 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) ของโปรโตคอล FMS ที่สภาพแวดล้อม 12MB



รูปที่ 4.4 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) ของโปรโตคอล FuzzySpray ที่สภาพแวดล้อม 12MB

4.3.2 ผลการทดลองของสมมติฐานที่ 3.3.2

จากหัวข้อ 3.3.2 ที่เสนอสมมติฐานว่า **หากโหนดในโปรโตคอล FMS มี Dropping Policy ที่เหมาะสม จะทำให้โปรโตคอลมีประสิทธิภาพโดยรวมที่ดี**

จากตารางที่ 4.3 และรูปที่ 4.5 จะสามารถสังเกตได้ว่าในกรณีที่บัฟเฟอร์ขนาด 3MB ทำให้โอกาสที่บัฟเฟอร์เต็มมีมากขึ้นจึงทำให้สภาพแวดล้อมนี้เป็นการวัดประสิทธิภาพของ Dropping Policy ได้ดี ผลปรากฏว่าโปรโตคอล FMSD มีการ Relay ข้อความในจำนวนที่น้อยกว่า FMS โดยเฉลี่ยประมาณ 84% และมี Overhead ที่น้อยกว่า FMS ประมาณ 60% เป็นผลมาจากส่วนการทำงาน Dropping Policy ของ FMSD เนื่องจาก Dropping Policy ทำหน้าที่ในการตัดสินใจว่าโหนดควรดรอปข้อความหรือไม่ ถ้าหากควรดรอปข้อความจะต้องดรอปข้อความใดบ้างในกรณีที่ไม่มีพื้นที่บัฟเฟอร์เหลือพอในการรับข้อความที่กำลังเข้ามาโดยการตัดสินใจด้วยการเทียบค่า Sending Prioriy ของข้อความที่กำลังจะเข้ามากับข้อความที่อยู่ในส่วนท้าย Queue ของบัฟเฟอร์ของโหนดผู้รับว่าการที่จะรับข้อความที่กำลังเข้ามานั้น"คุ้มค่า" เมื่อเทียบกับที่โหนดต้องดรอปข้อความในบัฟเฟอร์ของโหนดออกไปหรือไม่ เนื่องจากข้อความบางข้อความถูกปฏิเสธโดยส่วนการทำงาน Dropping Policy ของโหนดผู้รับทำให้ Delivery Probability ลดลง 15.61% และ Delivery Delay เพิ่มขึ้น 60% เนื่องจากมีการแลกเปลี่ยนข้อความกันน้อยลง การกระจายตัวของข้อความในเครือข่ายน้อยลง ทำให้โอกาสที่ข้อความจะเจอโหนดเป้าหมายน้อยลงเช่นกัน

จาก**ตารางที่ 4.4** และ**รูปที่ 4.6** สังเกตได้ว่าจำนวนข้อความที่ถูกดรอปในขนาดต่างๆ มีจำนวนน้อยจนไม่มีนัยสำคัญ เนื่องจากบัฟเฟอร์ที่มีขนาด 12MB สามารถรองรับข้อความจำนวนมากได้ ส่วนการทำงานของ Dropping Policy จึงมีผลต่อประสิทธิภาพของโปรโตคอลอย่างไม่มีนัยสำคัญ (ไม่มีความจำเป็นที่จะต้องดรอปข้อความใดออกจากบัฟเฟอร์) ทำให้มีประสิทธิภาพในด้าน Delivery Probability, Overhead, และ Delivery Delay ที่ใกล้เคียงกับ FMS เพราะใช้ Fuzzy Rule ที่คำนวณค่า Sending Priority ที่เหมือนกันและมีประสิทธิภาพที่ดีกว่า FuzzySpray เล็กน้อย

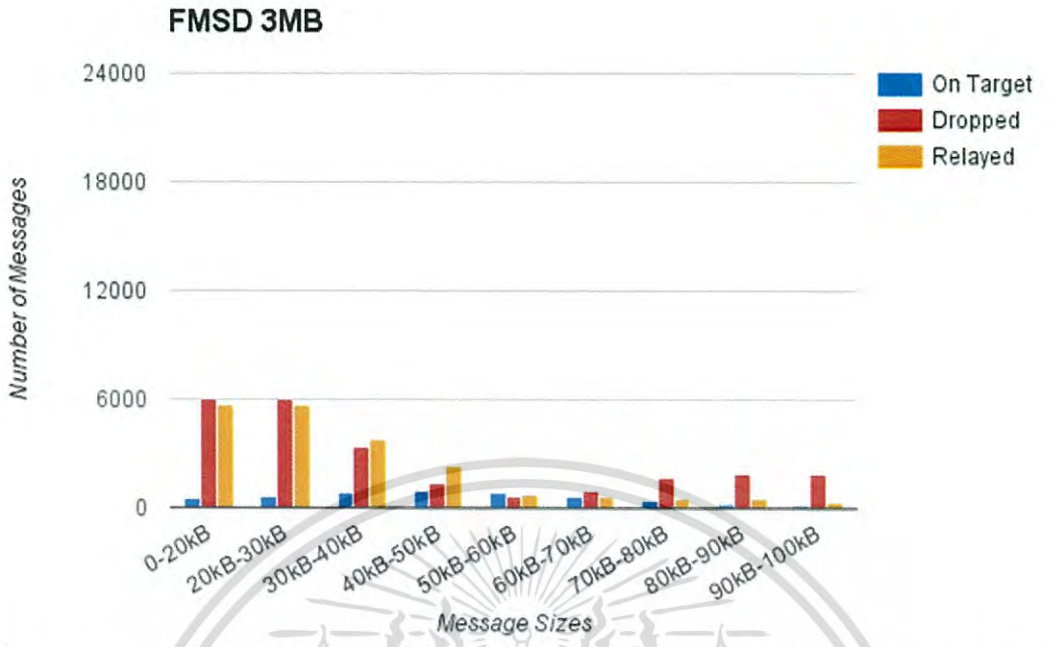
ดังนั้น สามารถสรุปได้ว่าโปรโตคอล FMSD มีประสิทธิภาพในด้านการลด Overhead ในสภาพแวดล้อมที่โหนดในเครือข่ายมีขนาดบัฟเฟอร์ที่น้อย

ตารางที่ 4.3 แสดงผลการทดลองที่บัฟเฟอร์ขนาด 3MB ของ FMSD เปรียบเทียบกับโปรโตคอลอื่นๆ

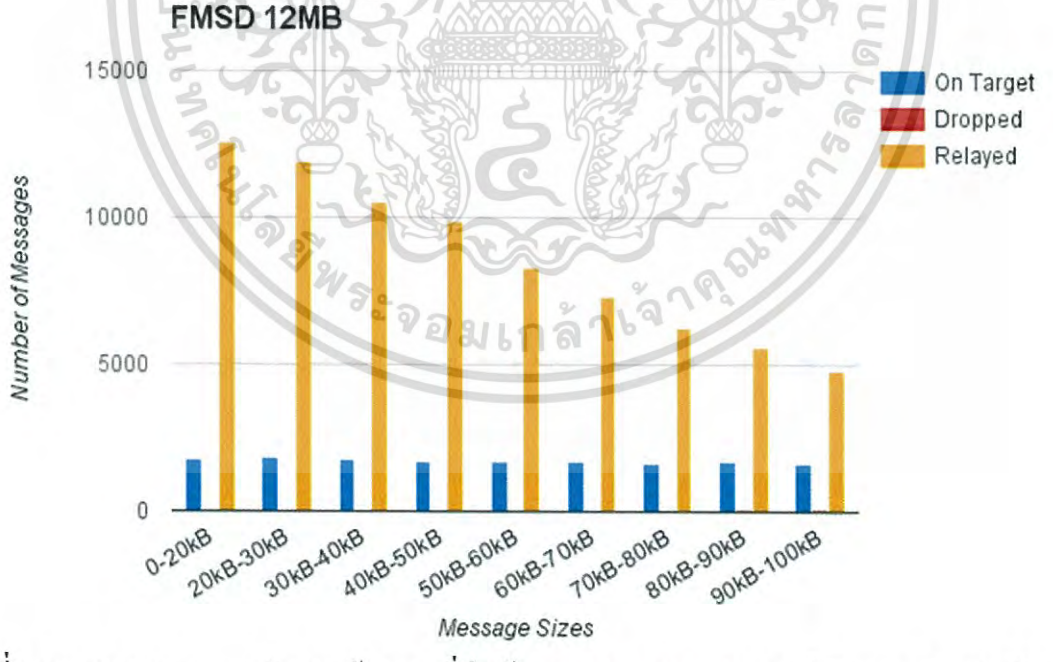
3MB	Delivery Probability	Overhead	Delivery Delay (s)
FMSD	29.74%	4.28	5404.06
FMS	45.35%	11.15	2154.12
FuzzySpray	39.73%	15.08	2812.07

ตารางที่ 4.4 แสดงผลการทดลองที่บัฟเฟอร์ขนาด 12MB ของ FMSD เปรียบเทียบกับโปรโตคอลอื่นๆ

12MB	Delivery Probability	Overhead	Delivery Delay (s)
FMSD	90.85%	4.99	3561.47
FMS	90.90%	5.06	3553.34
FuzzySpray	89.30%	5.53	4074.12



รูปที่ 4.5 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) ของโปรโตคอล FMSD ที่สภาพแวดล้อม 3MB



รูปที่ 4.6 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) ของโปรโตคอล FMSD ที่สภาพแวดล้อม 12MB

4.3.3 ผลการทดลองของสมมติฐานที่ 3.3.3

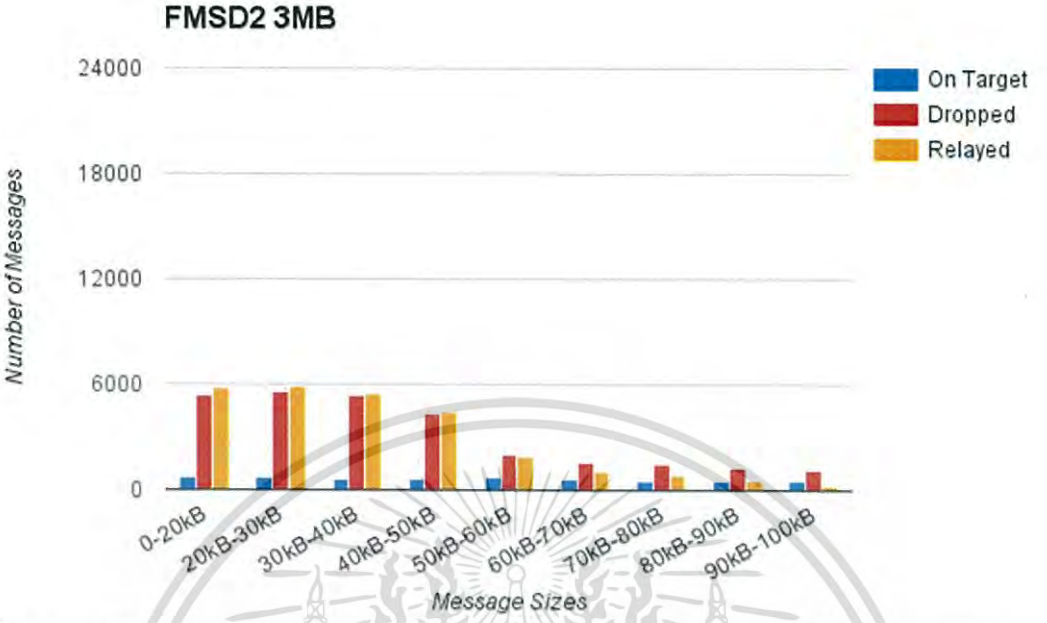
จาก**หัวข้อที่ 3.3.3** ที่เสนอสมมติฐานว่า **หากโปรโตคอลมี Fuzzy Rule ที่ทำหน้าที่เฉพาะในการคิดค่า Dropping Policy ที่ดีขึ้น จะทำให้โปรโตคอล FMSD มีประสิทธิภาพที่ดีมากขึ้นทั้งในด้าน Delivery Probability, Overhead, และ Delivery Delay** การพิสูจน์สมมติฐานจึงต้องดูจากสภาพแวดล้อมที่โหนดในเครือข่ายมีบัฟเฟอร์ขนาดเล็ก เช่น 3MB เพื่อดูประสิทธิภาพการทำงานของ Dropping Policy และ Dropping Fuzzy Rule ที่ผู้วิจัยทำขึ้นสำหรับโปรโตคอล FMSD

จาก**ตารางที่ 4.5** จะเห็นได้ว่าโปรโตคอล FMSD2 มีผลในด้าน Delivery Delay และ Overhead ที่ดีกว่าทุกๆโปรโตคอล เนื่องจากการทำงานของ Dropping Policy และ Dropping Fuzzy Rule ทำให้ภายในสภาพแวดล้อมที่บัฟเฟอร์ของโหนดมีขนาดแค่ 3MB ซึ่งมีโอกาสที่ Dropping Policy จะถูกเรียกใช้เป็นจำนวนมาก ซึ่ง Dropping Policy ของ FMSD2 ใช้ Dropping Fuzzy Rule ที่สามารถประเมิณได้ว่าข้อความในบัฟเฟอร์และข้อความที่เข้ามามีค่า Dropping Priority เท่าใดโดยประมาณการว่าข้อความนั้นมีจำนวน Copy ของข้อความเท่าใด (มากกว่าหรือน้อยกว่าค่าเฉลี่ยที่โหนดต่างๆส่งข้อความออกไปแล้วถึงโหนดเป้าหมาย) ข้อความนั้นมี Elapsed Time ที่มากเกินไปแล้วหรือไม่เมื่อเทียบกับค่าเฉลี่ย ทำให้โปรโตคอลมีค่า Overhead ที่ดีกว่าโปรโตคอลอื่นๆ ประมาณ 4% - 72% แต่มี Delivery Probability ที่น้อยกว่าโปรโตคอลอื่นๆเล็กน้อย จาก**รูปที่ 4.7** เป็นผลมาจากมีจำนวนการดรอปข้อความขนาดกลางค่อนไปทางใหญ่ (30kB - 40kB, 40kB – 50kB, 50kB – 60kB, 60kB – 70kB) ที่มาก เนื่องจากข้อความขนาดกลางๆ ก็มี Sending Priority ที่มากพอจะที่กระจายไปในเครือข่ายได้ ทำให้มีค่า FTC ที่สูงหรือใกล้เคียงกับค่า Average FTC ทำให้มี Dropping Priority ที่สูงพอสมควร

ดังนั้นสามารถสรุปได้ว่า FMSD2 มี Dropping Policy ที่ตัดสินใจรับหรือปฏิเสธข้อความและตัดสินใจดรอปข้อความที่มีประสิทธิภาพ ทำให้โปรโตคอลมีความ Fairness ในด้านของการส่งข้อความทุกๆขนาดไปถึงเป้าหมายได้ในจำนวนเท่าๆกันในสภาพแวดล้อมที่บัฟเฟอร์ของโหนดมีขนาดเล็ก

ตารางที่ 4.5 แสดงผลการทดลองที่บัฟเฟอร์ขนาด 3MB ของ FMSD2 เปรียบเทียบกับโปรโตคอลอื่นๆ

3MB	Delivery Probability	Overhead	Delivery Delay (s)
FMSD2	32.94%	4.15	5360.34
FMSD	29.74%	4.28	5404.06
FMS	45.35%	11.15	2154.12
FuzzySpray	39.73%	15.08	2812.07



รูปที่ 4.7 แสดง Histogram จำนวนข้อความที่ถึงเป้าหมาย (On Target), ดรอป (Dropped), และส่ง (Relayed) ของโปรโตคอล FMSD2 ที่สภาพแวดล้อม 3MB

4.3.4 ผลการทดลองเมื่อนำ FMSD2 มาเปรียบเทียบกับโปรโตคอลอื่นๆ

เพื่อตีความประสิทธิภาพของ FMSD2 ให้เข้าใจได้มากขึ้น ผู้วิจัยจึงนำ FMSD2 มาเปรียบเทียบกับโปรโตคอลอื่นๆ ที่ใช้ในปัจจุบัน จากตารางที่ 4.6 ที่บัฟเฟอร์ขนาด 3MB โปรโตคอล FMSD2 มีค่า Delivery Probability ที่น้อยกว่า FMS และ MaxProp เนื่องจาก FMSD2 มีการ Relay ข้อความในทุกๆขนาดที่น้อยกว่า FMS และ MaxProp เนื่องจากการทำงานของ Dropping Policy ของ FMSD2 ที่เห็นผลได้ชัดในสภาพแวดล้อมที่บัฟเฟอร์ขนาดเล็ก โดย FMSD2 ใช้การ Relay ข้อความในทุกๆ ขนาดน้อยกว่า FMS และ MaxProp โดยเฉลี่ย 74% และ 69% ตามลำดับ ทำให้ FMSD2 มี Overhead ที่น้อยกว่า 62% และ 60% ตามลำดับ ในส่วนของโปรโตคอล FMS ใช้ลักษณะการสเปรย์ข้อความที่เหมือนกับ FuzzySpray แต่ใช้ CSRQ ช่วยในการจัดลำดับการรับและส่งข้อความระหว่างคู่โหนดทำให้ได้ Delivery Probability ที่ดีแต่มี Overhead ที่เยอะเพราะไม่มีการควบคุมการรับข้อความจากโหนดผู้รับ

ในสภาพแวดล้อมการทดลองที่บัฟเฟอร์ขนาด 6MB จากตารางที่ 4.7 โปรโตคอล FMSD2 มีประสิทธิภาพในด้าน Delivery Probability และ Overhead ที่ดีที่สุด ส่วนหนึ่งเนื่องจากการทำงานของ CSRQ ที่เห็นผลทั้งใน FMSD และ FMS แต่เหตุที่ FMSD2 มีค่า Overhead ที่ดีกว่าทุกๆ โปรโตคอลเนื่องจากการทำงานของ Dropping Policy ของ FMSD2 ที่มี Fuzzy Rule สำหรับคิดค่า

Dropping Priority โดยเฉพาะเพื่อตัดสินใจในการรับและดรอปข้อความ ทำให้มีเมื่อเวลาผ่านไปในการทดลองเมื่อถึงจุดที่โปรโตคอล FMSD เริ่มดรอปข้อความขนาดกลางและใหญ่ Dropping Priority ของข้อความขนาดกลางและใหญ่ใน FMSD2 จะมีค่าน้อยกว่าข้อความขนาดเล็กเพราะค่า FTC ของข้อความขนาดเล็กจะมีเยอะขึ้นจนมากกว่าค่า Average FTC ทำให้มีการปฏิเสธการรับจากโหนดผู้รับหรือมีการดรอปข้อความขนาดเล็กออกจากโหนดผู้รับเพื่อเพิ่มโอกาสการรับข้อความขนาดกลางและใหญ่ทำให้มี Delivery Probability ที่ดีแต่ก็มี Delay ที่พอๆกับ FMSD เนื่องจากทั้งสองโปรโตคอลมีการปฏิเสธการรับข้อความบางข้อความที่มี Priority ที่คุ้มที่จะรับ ในขณะที่โปรโตคอล Epidemic มี Overhead ที่สูงเพราะว่าไม่มีการควบคุมจำนวนการ Relay ข้อความหรือเลือกส่งข้อความใดๆเลย เจอโหนดใดที่มีบัฟเฟอร์เหลือพอที่จะรับก็ส่ง ทำให้มีการกระจายข้อความที่สูงส่งผลให้ Delivery Delay ของโปรโตคอลมีน้อยแต่ Delivery Probability ก็มีน้อยด้วยเช่นกัน ส่วน EFSnW ที่มีหลักการเป็น Spray and Wait ที่มีการจัดลำดับข้อความด้วยค่า Priority ก็มีค่า Overhead ที่สูงเพราะมีจำนวนการ Relay ข้อความจำนวนมาก ซึ่งส่วนหนึ่งขึ้นอยู่กับค่า L (Replication Token) ของโปรโตคอลด้วย

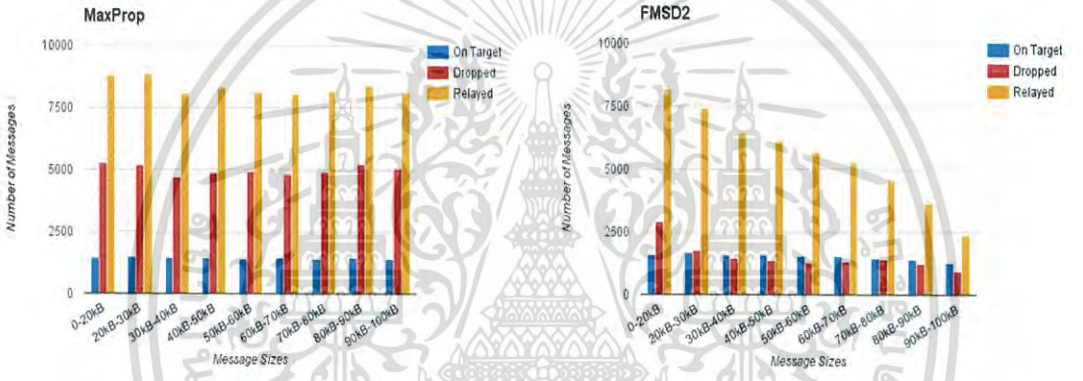
ตารางที่ 4.6 แสดงผลการทดลองของ FMSD2 เมื่อเทียบกับโปรโตคอลต่างๆ ที่ 3MB

3MB	Delivery Probability	Overhead	Delivery Delay (s)
FMSD2	32.94%	4.15	5360.34
FMSD	29.74%	4.28	5404.06
FMS	45.35%	11.15	2154.12
FuzzySpray [6]	39.73%	15.08	2812.07
EFSnW [9]	28.97%	13.9	2508.25
Epidemic [2]	26.34%	19.17	2319.23
MaxProp [10]	45.52%	10.39	7037.31

ในสภาพแวดล้อมการทดลองที่โหนดมีบัฟเฟอร์ขนาด 9MB ในกรณีนี้ถือว่าเป็นสภาพแวดล้อมที่บัฟเฟอร์เริ่มมีขนาดที่มากเมื่อเทียบกับขนาดของข้อความโดยเฉลี่ย ทำให้โหนดมีพื้นที่ว่างในการรับข้อความจำนวนมาก มีโอกาสที่จะเกิดการดรอปที่น้อย จึงทำให้ Message Scheduling ของโปรโตคอลเป็นส่วนสำคัญที่ส่งผลต่อประสิทธิภาพโดยรวมของโปรโตคอล จาก **ตารางที่ 4.8** จะเห็นได้ว่าโปรโตคอล FMSD2, FMSD, FMS มีประสิทธิภาพด้าน Delivery Probability และ Delivery Delay ที่ใกล้เคียงกันเป็นผลมาจากการใช้ CSRQ ที่ทำให้ได้ลำดับการรับและส่งข้อความภายใน Contact Time ที่จำกัดด้วยลำดับที่เหมาะสม

ตารางที่ 4.7 แสดงผลการทดลองของ FMSD2 เมื่อเทียบกับโปรโตคอลต่างๆ ที่ 6MB

6MB	Delivery Probability	Overhead	Delivery Delay (s)
FMSD2	80.18%	3.41	4313.58
FMSD	79.97%	3.55	4293.77
FMS	78.56%	6.05	3503.93
FuzzySpray [6]	76.34%	6.86	4068.02
EFSnW [9]	48.03%	8.23	3666.32
Epidemic [2]	38.59%	13.03	3272.37
MaxProp [10]	76.11%	5.94	5993.5



รูปที่ 4.8 Histogram เปรียบเทียบ FMSD2 กับ MaxProp ที่บัฟเฟอร์ขนาด 9MB

จาก**รูปที่ 4.8** จะเห็นได้ว่าหากเปรียบเทียบจำนวนการ Relay และ Drop จะเห็นได้ว่า FMSD2 ใช้จำนวนการ Relay ข้อความที่น้อยกว่าเพราะจัดลำดับข้อความที่เหมาะสมกับการแลกเปลี่ยนใน Contact Time นั้นจึงสามารถลดโอกาสที่ข้อความจะถูกส่งไปยังโหนดอื่นแต่ถูกดรอปในภายหลังได้ ทำให้มีประสิทธิภาพที่ดีทั้งในด้าน Delivery Probability, Delivery Delay, และ Overhead จาก Histogram จะเห็นได้ว่า MaxProp มีการ Relay ข้อความที่มากในทุกๆขนาดของข้อความเพราะ MaxProp ออกแบบให้ข้อความที่สร้างขึ้นมาใหม่ได้รับ Priority สูงสุดเพื่อเพิ่มโอกาสที่จะถูกส่งออกไปในเครือข่าย แต่ก็ยังสามารถรักษาระดับ Overhead ที่ดีได้ เพราะมีขั้นตอนการป้องกันข้อความส่งไปยังโหนดเดิมซ้ำๆ (คล้ายๆกับการป้องกันการ Flood)

ในสภาพแวดล้อมการทดลองที่โหนดมีบัฟเฟอร์ขนาด 12MB ที่ถือว่าเป็นขนาดบัฟเฟอร์ที่ใหญ่ จึงมีโอกาสที่บัฟเฟอร์ของโหนดจะเต็มได้น้อยด้วยขนาดของข้อความที่ใช้ในการทดลอง จาก**ตารางที่ 4.9** จะเห็นได้ว่าโปรโตคอล FMSD2, FMSD, และ FMS ที่ใช้ CSRQ ในการจัดลำดับการรับและส่งข้อมูลที่จะแลกเปลี่ยนใน Contact Time มีประสิทธิภาพด้าน Delivery Probability, Overhead,

และ Delivery Delay ที่ใกล้เคียงกัน เนื่อง Dropping Policy ของแต่ละโปรโตคอลมีผลน้อยมากในสภาพแวดล้อมนี้

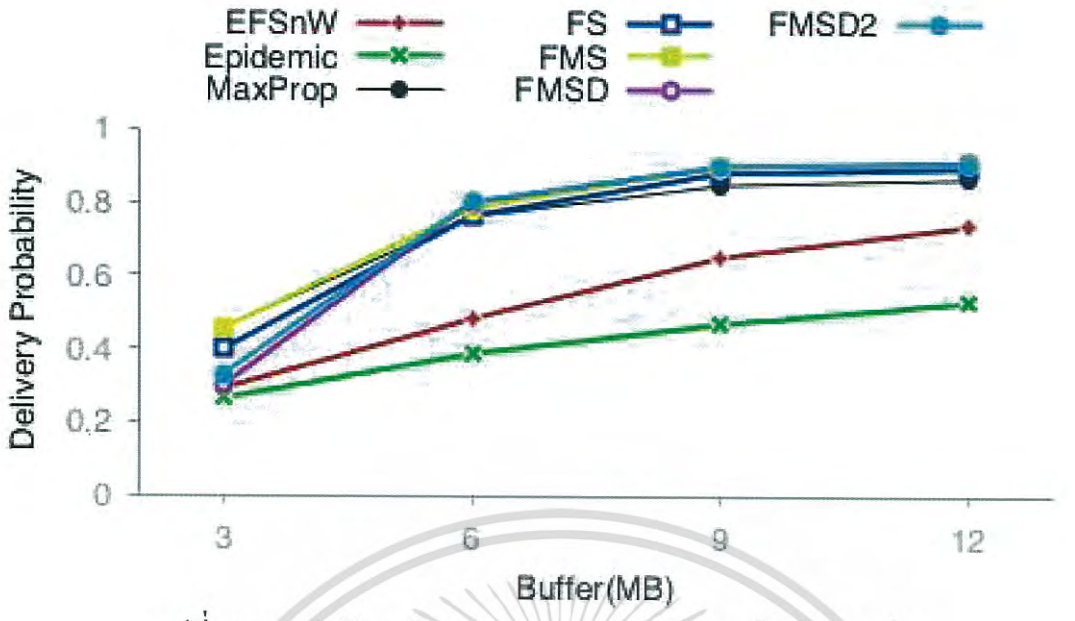
ในส่วนของ Epidemic ที่ Relay ข้อความให้ทุกๆโหนดที่เจอโดยไม่มีการจัดเรียงลำดับของข้อความ จึงมีการ Relay ข้อความจำนวนมากๆ ในทุกๆขนาดของข้อความ ทำให้เกิดการดรอปข้อความเกิดขึ้น ส่งผลโดยตรงต่อประสิทธิภาพของโปรโตคอล เพราะต้องใช้การ Relay เพิ่มขึ้นเพื่อส่งข้อความที่ถูกดรอปไป ทำให้ Delivery Delay และ Overhead เพิ่มขึ้นอีกทั้งการดรอปแบบสุ่มทำให้ไม่ได้ดรอปข้อความที่เหมาะสมเป็นผลทำให้ Delivery Probability ไม่ดีเท่าที่ควร

ตารางที่ 4.8 แสดงผลการทดลองของ FMSD2 เมื่อเทียบกับโปรโตคอลต่างๆ ที่ 9MB

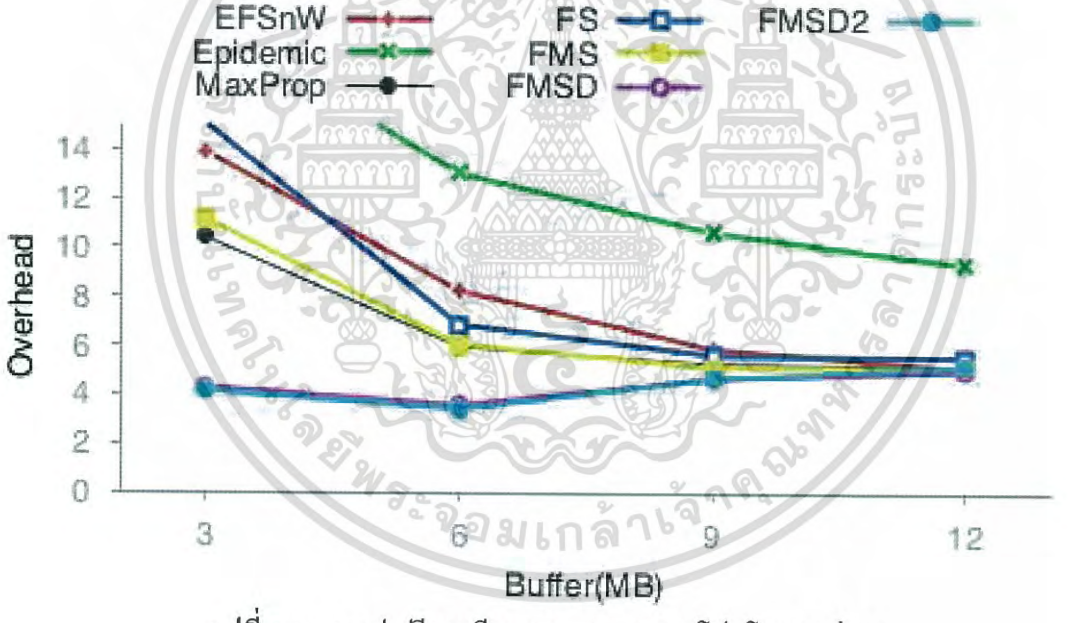
9MB	Delivery Probability	Overhead	Delivery Delay (s)
FMSD2	89.9%	4.67	3709.92
FMSD	89.84%	4.67	3730.68
FMS	89.81%	5.13	3663.66
FuzzySpray [6]	87.97%	5.63	4238.52
EFSnW [9]	64.91%	5.86	5387.12
Epidemic [2]	46.66%	10.62	3965.81
MaxProp [10]	84.66%	5.26	5002.16

ตารางที่ 4.9 แสดงผลการทดลองของ FMSD2 เมื่อเทียบกับโปรโตคอลต่างๆ ที่ 12MB

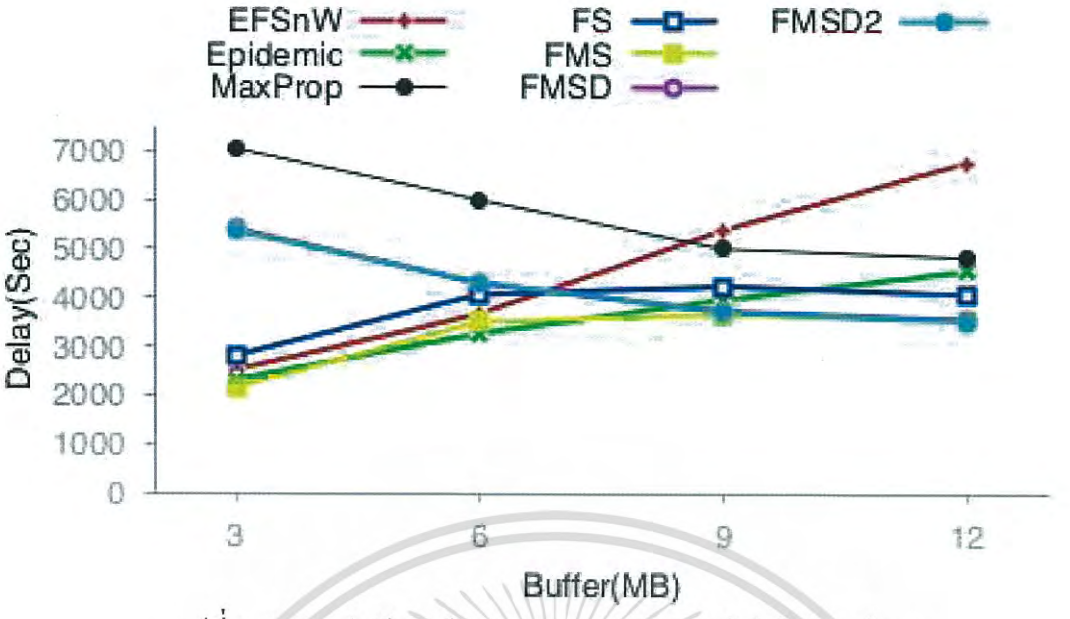
12MB	Delivery Probability	Overhead	Delivery Delay (s)
FMSD2	90.51%	5.17	3524.49
FMSD	90.85%	4.99	3561.47
FMS	90.90%	5.06	3553.34
FuzzySpray [6]	89.30%	5.53	4074.12
EFSnW [9]	73.46%	5.06	6758.54
Epidemic [2]	52.67%	9.3	4550.80
MaxProp [10]	86.25%	5.15	4816.38



รูปที่ 4.9 กราฟเปรียบเทียบ Delivery Probability ของโปรโตคอลต่างๆ



รูปที่ 4.10 กราฟเปรียบเทียบ Overhead ของโปรโตคอลต่างๆ



รูปที่ 4.11 กราฟเปรียบเทียบ Delivery Delay ของโปรโตคอลต่างๆ

บทที่ 5
สรุปผลการทดลอง

5.1 สรุปผล

หลังจากที่ผู้วิจัยได้ทำการศึกษาและสังเกตพฤติกรรมของโปรโตคอล FuzzySpray [6] ซึ่งเป็นผลงานเดิม พบว่าเป็นโปรโตคอลแบบ Epidemic-Based Routing Protocol ที่มีประสิทธิภาพดีทั้งในด้าน Delivery Probability และ Delivery Delay แต่ในสภาพแวดล้อมที่ขนาดของบัฟเฟอร์ของโหนดมีขนาดที่เล็ก เช่น 3MB ซึ่งในการใช้งาน DTN ในความเป็นจริงมักใช้กับอุปกรณ์เคลื่อนที่ที่มีพื้นที่บัฟเฟอร์และพลังงานที่จำกัด โปรโตคอล FuzzySpray มีค่า Overhead ที่สูงซึ่งหมายความว่าต้องใช้จำนวนครั้งที่เยอะในการส่งข้อความใดไปให้ถึงยังโหนดเป้าหมายของข้อความนั้นๆ ดังนั้นผู้วิจัยจึงเสนอโปรโตคอล FMS, FMSD, FMSD2 ที่มีเป้าหมายในการเพิ่มประสิทธิภาพในด้าน Overhead แต่ยังคงประสิทธิภาพในด้านอื่นๆที่ดีได้

ผลปรากฏว่า FMSD2 มีประสิทธิภาพที่ถือว่าดีกว่าทั้งในด้าน Delivery Probability, Overhead, และ Delivery Delay ที่ดีกว่า FMSD, FMS, และโปรโตคอลอื่นๆ ในทุกๆสภาพแวดล้อม ในสภาพแวดล้อมที่บัฟเฟอร์มีจำกัด FMSD2 สามารถรักษาระดับของ Delivery Probability ให้อยู่ในระดับที่ดีได้ โดยจะมีแค่ FMS และ MaxProp ที่ได้ค่า Delivery Probability ที่สูงกว่า FMSD2 แต่ FMSD2 สามารถรักษาระดับ Overhead ให้ได้ต่ำกว่าโปรโตคอลอื่นๆอย่างมีนัยสำคัญ ในสภาพแวดล้อมที่บัฟเฟอร์มีขนาดที่ใหญ่ เช่น 9MB, 12MB ผลปรากฏว่าโปรโตคอลที่ใช้ CSRQ ได้ผลที่ดีกว่าในด้านของ Delivery Probability และในด้าน Delivery Delay ที่น้อยกว่าโปรโตคอลที่ไม่ใช้ CSRQ อย่างมีนัยสำคัญ เนื่องจากโปรโตคอลที่ใช้ CSRQ สามารถส่งข้อความที่มีโอกาสที่ดีในการส่งไปถึงยังโหนดเป้าหมายได้ดีภายใน Contact Time ที่จำกัดได้ในจำนวนที่เยอะกว่าโปรโตคอลอื่นๆ สังเกตได้จากการใช้จำนวนการ Relay โดยรวมที่น้อยแต่ยังได้ Delivery Probability ที่ดีอยู่

บรรณานุกรม

[1] Matthias Grossglauser, David N. C. Tse, "Mobility Increases the Capacity of Ad Hoc Wireless Networks", **IEEE/ACM TRANSACTIONS ON NETWORKING**, vol. 10, no. 4, 2002.

[2] Vahdat D. Becker, "Epidemic routing for partially-connected ad hoc networks", **Duke University Technical Report Cs-2000-06**, Tech. Rep., 2000.

[3] Thrasyvoulos Spyropoulos, Konstantinos Psounis, Cauligi S. Raghavendra, **"Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks"**, [Online]. Available: http://chants.cs.ucsb.edu/2005/papers/paper-SpyPso.pdf, 2005.

[4] Thrasyvoulos Spyropoulos, Konstantinos Psounis, Cauligi S. Raghavendra, **"Spray and Focus: Efficient Mobility-Assisted Routing for Heterogeneous and Correlated Mobility"**, [Online].Available: http://www.bcf.usc.edu/~kpsounis/Papers/sprayfocus_icman.pdf, 2007.

[5] K. Sabeetha, A. Vincent Antony Kumar, R. S. D. Wahidabanu, W. A. M. Othman, "Encounter based fuzzy logic routing in delay tolerant networks", **Wireless Networks**, vol. 21, issue. 1, pp 173-185.

[6] Akadet Mathurapoj, Chotipat Pornavalai, Goutam Chakraborty, "Fuzzy-Spray: Efficient Routing in Delay Tolerant Ad-hoc Network Based on Fuzzy Decision Mechanism", ***IEEE FUZZ '09**, August 2009, pp 104-109, 2009.*

[7] Payam Nabhani, Amir Masoud Bidgoli, "Adaptive Fuzzy Routing in Opportunistic Network", **International Journal of Computer Applications**, vol. 52, no. 18, 2012.

[8] Jad Makhlouta, Hamza Harkous, Farah Hutayt, Hassan Artail, **"Adaptive Fuzzy Spray and Wait: Efficient Routing for Opportunistic Network"**, [Online].Available:http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6085837&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6085837, 2011.

[9] Sweta Jain, Meenu Chawla, Vasco N.G. J. Soares, and Joel J. Rodrigues, "Enhanced fuzzy logic-based spray and wait routing protocol for delay tolerant networks", **International Journal of Communication Systems**, vol. 28, issue. 18, 2014.

[10] J. Burgess, B. Gallagher, D. Jensen, and B.N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks", **Proc. IEEE Infocom**, 2006.

ประวัติผู้เขียน

นายกฤษกรช์ เอกวรรณัง เกิดเมื่อวันที่ 6 ธันวาคม พ.ศ. 2536 เข้าศึกษาในระดับปริญญาตรี หลักสูตรวิทยาศาสตรบัณฑิต สาขาเทคโนโลยีสารสนเทศ แขนงวิชาวิศวกรรมซอฟแวร์ คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีการศึกษา 2555

นายวิศรุต ตีรถะวรวรรณ เกิดเมื่อวันที่ 5 มิถุนายน พ.ศ. 2536 ที่จังหวัดกรุงเทพมหานคร เข้าศึกษาในระดับปริญญาตรี หลักสูตรวิทยาศาสตร์บัณฑิต สาขาเทคโนโลยีสารสนเทศ แขนงวิชา เทคโนโลยีเครือข่ายและระบบ คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีการศึกษา 2555

ALGORITHM FOR PRIORITIZING MESSAGES IN DELAY-TOLERANT NETWORK

Visarut Tirataworawan[1] and Kritsakorn Akwannang[2]

[1]Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok

[2]Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok

Emails: Visarut.tirataworawan@gmail.com, kritsakorn_krit@hotmail.co.th

ABSTRACT

Delay-Tolerant Network (DTN) is a sparse network of mobile wireless nodes where there is no end-to-end connectivity between nodes. DTN usually used to provide a network in extreme terrestrial environments, mobile environments, or even when there is a disaster happened. In DTN, contact time between nodes is usually low, so the amount of message exchanges between two nodes is limited causes the network to have high message transfer delay, hence the name Delay-Tolerant Network. To prioritizing a message to transfer is the next contact time is crucial to be able to ensure that the network can delivers a message efficiently, low overhead, with maximum delivery probability, low delay, and no loss in the network.

Index Terms - DTN; fuzzy; Routing

1. INTRODUCTION

Delay-Tolerant Network (DTN) is a sparse network of mobile wireless nodes where there is no end-to-end connectivity between nodes, and applications can tolerate high latency. DTN is usually used in extreme-terrestrial environments, mobile environments, or even when there is a disaster happened. Two or more nodes in DTN networks can exchange their messages only when they move within their transmission range. This event is called "contact" in DTN [13]. Once nodes move out of their transmission range, the communication breaks down. This "contact" duration or contact window depends on the mobility profile of the participating nodes [2]. It is usually short time and unpredictable [13] [11]. So the amount of message exchanges between two nodes is limited causes the network to have high message delivery delay, hence the name Delay-Tolerant Network.

Typical DTN uses store and forward mechanism [2] [12] for message to spread across the network. Store and forward mechanism is the mechanism that node sends message to the other intermediate nodes where message is kept in intermediate node's buffer and then send to destination node or other intermediate nodes later on their next contact time. Prioritizing messages to transfer in the next contact time is crucial to ensure that the network can deliver messages efficiently with low overhead, high delivery probability, low delay and no loss in the network. To properly drop appropriate messages in case of node's buffer is full, a dropping policy is also required.

There are two major classes of DTN protocol if we considered how it forwards (sprays) message out. They are Epidemic-based routing protocol and two-hop routing protocol [3]. Epidemic-based routing protocol [1] was designed to forward (spray) as many messages as possible to its contact node in the limited contact time. On the other hand, two-hop routing protocol allows node

to forward or spray just *L-1* copies of message to other intermediate nodes, and then intermediate nodes only allow to forward message only if the encountered node is the destination node of the message (*L* is usually predefined and fixed).

In this paper, we focus our study on message scheduling and dropping to prioritizing messages based on Epidemic-based routing protocol. We present the protocol and algorithm called "Fuzzy-based Message Scheduling and Dropping" or FMSD to decide the order of messages that should be sent by calculating priority of message using fuzzy logic based on the properties of the messages in the buffer. This information is called summary vectors and is exchanged during the starting of contact window. In case that node buffer is full, the fuzzy logic is also used to determine which messages should be dropped in order to receive incoming messages.

The proposed FMSD is simple and has low overhead. It uses only two parameters which are the properties of the message, namely Forward Transmission Count (FTC) and message size, to prioritize messages that are stored in its buffer. FTC information is updated whenever messages are transmitted during the contact time. The simulation results show that, in overall, FMSD has the best performance in terms of average success rate, average delay, overhead ratio, compared with many other existing DTN routing protocols.

2. RELATED WORKS

There are many Epidemic-based routing protocols use in Delay-Tolerant Network. Epidemic [1] is the simplest routing protocol in this type of routing. It assumed that all nodes in DTN have infinite buffer space and bandwidth. It uses flood-based approach where nodes replicate message and transmit that replicated message to every node it encounters that do not have a copy of this message. On the other hand, the two-hop routing protocols limit the number of copy of message *m* allowed to be replicated by nodes up to *L* copies [7] [10]. This can reduce overhead network consumption but the proper value of *L* is highly sensitive to the network environment [7].

Protocols in DTN may use the information that nodes in the network can learn during the contacts to improve the performance. This is called a utility metric. For example, MaxProp [8] uses historical statistics to schedule both message priority to send at the next contact time and message to be dropped in case of full buffer. It uses historical data to calculate a path that other node is likely to travel and calculates priority for each message.

Some protocols such as GHP [3] has a mechanism for node to decide whether node should accept message or in case that receiving node's buffer is full, and what message should the receiving node drop in order to receive the incoming message.

Recently, fuzzy logic has been successfully applied to DTN networks in many literatures. For example, Adaptive Fuzzy Spray and Wait [6] uses Fuzzy Controller to prioritize messages in node's buffer to be in a proper order for exchanging messages in the next contact time. Being based on Spray and Wait [7], *L* copies of message is allowed to be forwarded in the network. This protocol also implements dropping policy that receiving node randomly drop a message in case its buffer is full instead of just drop least-priority message or oldest message like most protocols do. Enhanced Fuzzy logic based Spray and Wait (EFSnW) [5] implemented an estimation on number of message copy called ETR that counts number of copy of message that this node has forwarded called "My Forwarding". ETR is updated every time the node encounters other nodes.

The protocol we proposed here (FMSD) is an extension of our previous work on FuzzySpray (FS) [2]. Similar to FS, the proposed FMSD protocol uses fuzzy logic to calculate message priority, and then messages to send during the next contact time are sorted according to their priorities. Message with higher priority will be sent before the lower ones. There are two inputs for Fuzzy Logic System namely Forward transmission count (FTC) and message size to calculate message priority. Forward transmission count is the estimation on the copy of particular message which is updated on both sending node and receiving node, incremented by one, every time message is sent. The FMSD extends FS to improve the performance in many ways.

3. FUZZY-BASED MESSAGE SCHEDULING AND DROPPING (FMSD)

Fuzzy-Based Message Scheduling (FMSD) is an extended work from FuzzySpray (FS) [2]. FMSD has three major modules namely (1) Fuzzy Logic System, (2) Common Sending and Receiving Queue Sorting, and (3) Dropping Policy.

When connection is established between a pair of node, each node starts exchanging its known *AckedList* information which is the list of known messages (and its *message-id*) that already delivered to its final destination (*AckedList*), and delete messages that already arrived at their destinations in node buffer. Then each node will send deliverable messages (meaning that the destination of messages is the connected node), in order of message priority if there are more than one deliverable messages, to each other. After that, node will exchange summary vectors that contains information about message properties of each message *m* that node has. These include *message-id*, Forward Transmission Count (FTC) of message (FTC_m), and message size (MS_m). After Summary Vector is exchanged, Summary Vector will be sent to Common Sending and Receiving Queue Sorting module.

Every message in node's buffer must be prioritized by Fuzzy Logic System before transmission is occurred. Message Priority (P_m) is calculated by Fuzzy Logic System using two inputs namely Forward Transmission Count (FTC) and Message Size (MS). Message with higher priority will be sent first. Messages from a pair of nodes will be merged into one Common Sending and Receiving Queue (CSRQ) which is sorted by Message Priority. When buffer of the receiving node is full, dropping policy will evaluate the incoming message priority and decide if node should accept this message or not by dropping some messages in its buffer in order to accommodate the incoming message. Algorithm Fig. 5 describes the process of FMSD protocol.

3.1. Fuzzy Logic System

Fuzzy Logic System simply takes only two properties of message *m* as two inputs to calculate priority. These two properties are Forward Transmission Count (FTC_m) and Message Size (MS_m)

Forward Transmission Count (FTC_m) : is used to estimate the current number of copies of message *m* in the network. Every message *m* generated by any node will have *FTC* of 1. FTC_m increases by one on both end of connection if message *m* is successfully forwarded (in other word, sprayed) to the connected node. The membership function of *FTC* has 3 categories which are low, medium, and high. The range of *FTC* is typically set to 10% of total number of nodes [2].

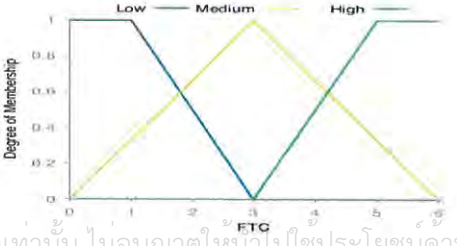


Figure 1. Membership function of FTC.

Message Size (MS_m) : size of a message m in byte. The membership function of message size has 3 categories which are small, medium, and large.

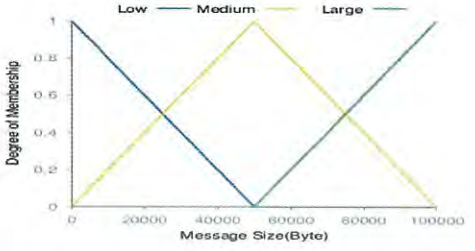


Figure 2. Membership function of Message Size.

In this protocol, we divided node buffer section into 9 sections which are BS0, BS1, ... , BS8. Buffer section is a section in node buffer. For example, BS0 is a buffer section where a message with higher priority is located in the node's buffer, in other word, the front of the sending queue. BS8 is at the tail of the sending queue.

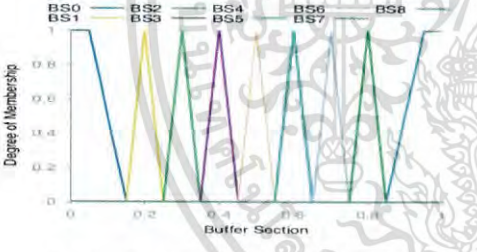


Figure 3. Membership function of Buffer Section.

Table 1. Fuzzy rule set for FMSD.

FTC	Message Size
Low	Small
Low	Medium
Low	High
Medium	Small
Medium	Medium
Medium	Large
High	Small
High	Medium
High	Large

3.2. Common Sending and Receiving Queue (CSRQ)

Typical message scheduling considers only sender node's buffer, and messages are sent according to the sending queue only. In order to sending/receiving messages in half duplex communication, each node in FMSD protocol prioritizes a collection of messages both from its own buffer and receiving side's buffer. We called this Common Sending and Receiving Queue or CSRQ. As shown in Fig. 4, initially when two nodes move within contact ranges, nodes exchange *AckedList*, any deliverable messages in order of message priority, and then Summary Vector which consists of message ID, FTC_m, MS_m. Then the node that make a connection first will calculate the priority for all message in the Common Sending Queue using all the information contained in Summary Vector. After all messages are prioritized, node that has a message that is at the head of Common Sending Queue will start sending to or receiving message on the other side on the connection.

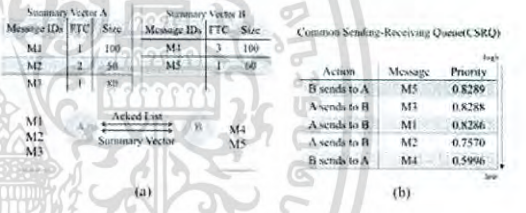


Figure 4. (a) Assuming that node A currently encounters and establishes a connection to node B, these two nodes then exchange *AckedList* and Summary Vector. Node A tells B that it has message M1, M2, and M3 which also contained FTC and Message Size of the afore mention messages in the summary vector. Node B does the same to A. Each node calculates message priority and sorts messages into CSRQ, (b) Each node then sends or receives messages according to order in CSRQ until the connection is terminated.

Algorithm 1 Fuzzy-Based Message Scheduling

```
1: procedure FUZZYBASEDMESSAGESCHEDULINGA
2:     /*On node A, B is the current connected node */
3:     if ConnectionIsUp(B) then
4:         A.exchangeAckedList(B)
5:         A.exchangeDeliverables(B)
6:         exchangeSummayVector()
7:         deleteAckedMessages()
8:         CSRQ ← CSRQSorting(B)
9:         for each msg do in CSRQ
10:            if thisNode.contain(msg) then
11:                if B.canAcceptMessage(msg) then
12:                    sendMessage(msg, B)
13:
14: Function exchangeDeliverables(B)
15:     deliverables ← calPriority(deliverables)
16:     sendMessage(deliverables, B)
17: end functionexchangeDeliverables
18:
19: Function CSRQSorting(B)
20:     /* Message Property(MP) : Msg Id, Msg FTC,
21: Msg Size contains in Summary Vector*/
22:     allMsgs ← MP_thisNode + MP_B
23:     allMsgs ← calPriority(allMsgs)
24:     return sortByPriority(allMsgs)
25: end function CSRQSorting
26:
27: Function calPriority(msgs)
28:     for each (msg in msgs) do
29:         msg.coaValue = fuzzy(msg)
30:         msg.priority = 1 - msg.coaValue
31:     return msgs
32: end function calPriority
33:
34: Function sendMessage(msg, toNode)
35:     otherNode.put(msg)
36:     if destination(msg) == otherNode then
37:         deleteMessage(msg)
38:         createACKMsg msg_(ack)
39: end function sendMsg
40:
41: Function fuzzy(msg)
42:     setFuzzyInput(FTC_msg, MS_msg)
43:     return getFuzzyOutput()
44: end function fuzzy
45:
46: Function canAcceptMessage(msg)
47:     Priority_msg ← fuzzy(msg)
48:     if messageIsBiggerThanBuffer(msg) then
49:         return False
50:         /*incoming message is bigger than receiving node
51: buffer, reject it.
52:         while freeBuffer < Size_msg do:
53:             if hopCount_MPmin > 1 then
54:                 PriorityDropped += fuzzy(M_Pmin)
55:                 /* M_Pmin : message with least priority in buffer */
56:                 if Priority_msg <= PriorityDropped then
57:                     return False
58:                 freeBuffer += Size_m;
59:                 DroppingList.add(m)
60:     deleteMessageIn(DroppingList)
61:     return True
62: end function canAcceptMessage
63:
64: Function makeRoomInBuffer()
65:     if bufferFullByCreatingMessage then
66:         deleteMessage(Msg_Pmin)
67:     else if bufferFullByReceivingDeliverable
68:         deleteMessage(Msg_Pmin)
69: end function makeRoomInBuffer
```

Figure 5. FMSD algorithm.

3.3. Dropping Policy

Having a limited buffer in node is one of the limitations in DTN. When node's buffer is full, node may choose to drop one or more message from its buffer to accommodate the incoming message, or reject the incoming message.

There are numbers of dropping policies in DTN have been proposed over the years. One of the simplest dropping policy is to drop the oldest message. Some protocols drop random messages that are in lower priority tier [6]. In our FMSD protocol, dropping policy is based on the estimating the likelihood that the incoming message to reach its destination compare to a message in receiving node's buffer to be dropped. Dropping policy is shown by Fig. 5 in function *canAcceptMessage* and *makeRoomInBuffer*. Dropping policy will be used in three main situations as follows.

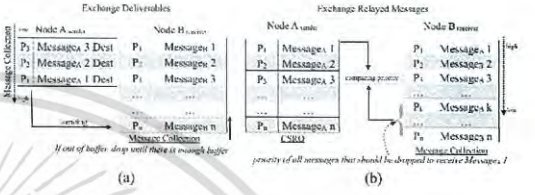


Figure 6. (a) actions when receiving node's buffer is full during exchange deliverables between nodes, (b) actions when sending node is sending a relayed message to a receiving node that its buffer is full.

4. EVALUATIONS

We used Opportunistic Network Environment (ONE) Simulator [9] as our simulation environment. ONE simulator is a tool for DTN protocol evaluation which implements numbers of popular DTN protocols.

4.1. Simulation Settings

Table 2. Simulation setting parameters.

Simulation Duration	12 Hours (43200 Seconds)
Number of Nodes	60
Message TTL	600 Minutes
Message Sizes	10kB – 100kB
World Size	3220m * 3220m
Transmit Range	30m
Transmit Speed	250kBps

4.2. Simulation Results

The performance metrics we use to evaluate protocols as follows

Delivery Probability: number of messages that have been delivered to its destination over number of all messages generated in simulation.

Overhead: shows the average number of times that a message is relayed until it reaches its destination node. It can be calculated by overhead $= \frac{nrofMessages_{Relayed} - nrofMessages_{Delivered}}{nrofMessages_{Delivered}}$

Delivery Delay: average delay a message need from it was generated till reach its destination node. If any message m is not successfully delivered to its destination node, $Delay_m$ is set to simulation time (in case of our work, 12 hours).

Each result in this section is the average result over 100 simulation runs. In these simulations, we tested how protocol handles DTN node limitations by varying node buffer. From Fig. 7, in 5MB scenario, our proposed FMSD has lower delivery probability than just MaxProp by 1% because MaxProp transmitted messages in greater number, in all message sizes, than FMSD in this scenario where in FMSD dropping policy refused some message to be transmitted, resulted in number of message arrived at destination for FMSD is around 11% - 24% of medium-sized messages less than that of MaxProp but FMSD achieved around 49% lower overhead than MaxProp in all scenarios.

For buffer of 10MB or more, it is clear that FMSD is the only protocol that achieved more than 90% on delivery probability because FMSD takes advantage of CSRQ Sorting. GHP is not effective in this environment due to our network is very sparse (In [3] uses 800m x 800m, with 130 node), resulted in lower amount of necessary information available for nodes to calculate message per-utility (message priority). Epidemic achieved second to lowest in term of delivery probability, and delay in all scenarios but due to the lack of space, we can not provide delay graph in this paper.

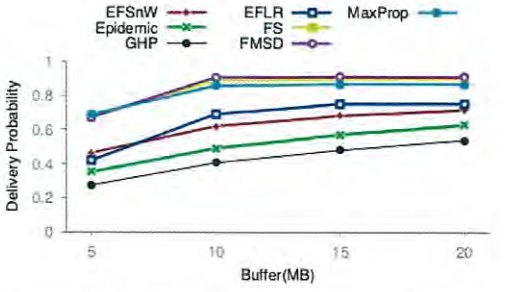


Figure 7. Delivery probability : Generating messages all time during simulation.

In Fig. 8, in term of overhead, FMSD achieved lowest overhead in 5MB and 10MB, the overhead is lower than FS by 57% and 12% respectively, and a 2% higher than EFLR in both 15MB and 20MB of buffer but FMSD achieved around 16% more than EFLR in term of delivery probability.

FMSD does not drop a message if that message's hop count is less than one to prevent loss occurred. So there is no redundant transmission for the particular message allowing FMSD to achieve low overhead and delay. EFLR achieved low overhead in most of the scenario by not relay any messages unless other intermediate node has better probability to encounter message destination node.

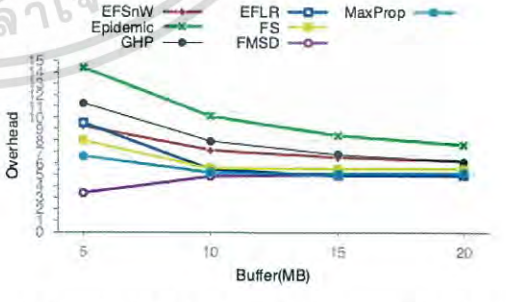


Figure 8. Overhead : Generating messages all time during simulation.

When buffer is full, MaxProp and FS have no mechanism for receiving node to decide if the receiving node should accept a message or not by priority, which means sender node will always send

message to the receiver, and receiver will need to drop message in order to accommodate the incoming message that leads to higher overhead and more message dropping will likely happen. In tight buffer space scenarios, MaxProp and FS have number of messages dropped, by average, 70% and 60% more than FMSD in all message size respectively.

5. CONCLUSION

This paper presents a message scheduling and dropping policy in DTN called Fuzzy-based Message Scheduling and Dropping or FMSD which composes of three main modules namely (1) Fuzzy Logic System, (2) Common Sending and Receiving Queue or CSRQ Sorting, and (3) Dropping Policy. To get the most out of limited contact time, FMSD prioritizes messages from both ends of connection by using two parameter which are FTC and Message Size [2]. By using CSRQ Sorting messages with higher priority are exchanged between two nodes. The dropping policy decides whether node should accept the incoming message or not. If node decides to accept the incoming message and receiving node's buffer is full, what message should the receiving node drop in order to accommodate the incoming messages. Our simulation results show that FMSD has better performance than most of the protocols we evaluated with in terms of delivery probability, overhead, and delivery delay.

REFERENCES

[1] A. Vahdat and D. Becker, Epidemic routing for partially connected ad hoc networks, Tech Report CS-200006, Duke University, April 2000.

[2] Akadet Mathurapoj, Chotipat Pornavalai and Goutam Chakraborty, FuzzySpray: Efficient Routing in Delay Tolerant Ad-hoc Network Based on Fuzzy Decision Mechanism, IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Jeju Island, 2009.

[3] Ahmed Elwhishi, Pin-Han Ho, K. Naik, and Basem Shihada, A Novel Message Scheduling Framework for Delay Tolerant Networks Routing, IEEE Transactions On Parallel And Distributed Systems, VOL. 24, NO. 5., May, 2013.

[4] K. Sabeetha, A. Vincent Antony Kumar,R. S. D. Wahidabanu and W. A. M. Othman, Encounter based fuzzy logic routing in delay tolerant networks, Wireless Networks, Volume 21, Issue 1, pp 173-185, January 2015.

[5] Sweta Jain, Meenu Chawla, Vasco N. G. J. Soares and Joel J. Rodrigues, Enhanced fuzzy logic-based spray and wait routing protocol for delay tolerant networks, International Journal of Communication Systems, 2014.

[6] Jad Makhlouta, Hamza Harkous, Farah Hutayt, Hassan Artail, Adaptive Fuzzy Spray and Wait: Efficient Routing for Opportunistic Networks, Mobile and Wireless Networking (iCOST), pp 64 - 69, 2011.

[7] T. Spyropoulos, K. Psounis, and C.S. Raghavendra, Spray and wait: an efficient routing scheme for intermittently connected mobile networks, ACM SIGCOMM workshop on Delay-tolerant networking, ACM, p. 259, 2005.

[8] J. Burgess, B. Gallagher, D. Jensen, and B.N. Levine, Maxprop: Routing for vehicle-based disruption-tolerant networks, Proc. IEEE Infocom, 2006.

[9] Ari Keranen, Jorg Ott, Teemu Karkkainen, The ONE Simulator for DTN Protocol Evaluation, SIMUTools 2009, Rome, Italy, 2009.

[10] T. Spyropoulos, K. Psounis, and C. Raghavendra, Spray and focus: Effi- cient mobility-assisted routing for heterogeneous and correlated mobility, Pervasive Computing and Communications Workshops '07, Fifth Annual IEEE International Conference, pages 79-85, 2007.

[11] Khalil Massri, Data Delivery in Delay Tolerant Networks, Sapienza University of Rome, Computer engineering, 2013.

[12] Yue Cao and Zhili Sun, Routing in Delay/Disruption Tolerant Networks: A Taxonomy, Survey and Challenges, IEEE Communications Surveys and Tutorials, Vol. 15, No. 2, Second Quater, 2013.

[13] DTN Research Group (DTNRG), http://www.dtnrg.org/

[14] Kevin Fall, A Delay-Tolerant Network Architecture for Challenged Internets, SIGCOMM ?03, August 25-29 '03, Karlsruhe, Germany, 2003.