QOS-BASED DYNAMIC SOURCE ROUTING PROTOCOL IN MOBILE AD-HOC NETWORKS

I WAYAN MUSTIKA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN COMPUTER ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2008

KMITL-2008-EN-M-070-077

หัวข้อวิทยานิพนธ์	โปรโตคอลค้นหาเส้นทางไดนามิกซอร์สที่สนับสนุนการรับประกันคุณภาพการให้บริการในเครือข่ายเคลื่อนที่แบบแอดฮอค
นักศึกษา	I Wayan Mustika
รหัสนักศึกษา	49060725
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
พ.ศ.	2551
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ผศ.ดร. ศักดิ์ชัย ทิพย์จักษุรัตน์

บทคัดย่อ

เครือข่ายเคลื่อนที่แบบแอดฮอคเป็นเครือข่ายไร้สายที่ประกอบด้วย โมไบล์โหนดไร้สายต่างๆ ที่เชื่อมต่อกันเป็นเครือข่ายที่สามารถติดต่อสื่อสารกันโดยปราศจากสายนำสัญญาณสถานีฐาน หรือศูนย์กลางการควบคุมการสื่อสาร และด้วยการเติบโตอย่างรวดเร็วของแอพพลิเคชันประเภทมัลติมีเดีย เช่น วีดีโอออนดีมานด์ และการประชุมทางไกล เป็นต้น ทำให้มีความจำเป็นที่จะต้องพิจารณาถึงการรับประกันคุณภาพของการให้บริการสำหรับแอพพลิเคชันประเภทนี้ อย่างไรก็ตามโปรโตคอลการค้นหาเส้นทางในปัจจุบัน ส่วนใหญ่จะเป็นโปรโตคอลที่ไม่มีการพิจารณาถึงคุณภาพของการให้บริการ กล่าวคือ ในการเลือกเส้นทางโปรโตคอลจะไม่มีการพิจารณาถึงการรับประกันความล่าช้าและแบนด์วิดธ์ที่ต้องการในการส่งข้อมูลจากโมไบล์โหนดต้นทางไปยังโมไบล์โหนดปลายทาง

วิทยานิพนธ์ฉบับนี้ เรานำเสนอโปรโตคอลการค้นหาเส้นทางที่พิจารณาการรับประกันคุณภาพของการให้บริการในเครือข่ายเคลื่อนที่แบบแอดฮอค เราเรียกโปรโตคอลที่นำเสนอว่า "โปรโตคอลค้นหาเส้นทางไดนามิกซอร์สที่สนับสนุนการรับประกันคุณภาพการให้บริการ (โปรโตคอลคิวดีเอสอาร์)" โดยการปรับปรุง "โปรโตคอลค้นหาเส้นทางไดนามิกซอร์ส (โปรโตคอลดีเอสอาร์)" เดิม โดยการเพิ่มกระบวนการในการพิจารณาถึงการรับประกันคุณภาพ ของการให้บริการในกระบวนการเลือกเส้นทางในโปรโตคอลดีเอสอาร์ โปรโตคอลคิวดีเอสอาร์สามารถแบ่งการทำงานออกเป็น 2 ขั้นตอน คือ ขั้นตอนแรกจะทำการจองแบนด์วิดธ์จากโมไบล์โหนดต้นทางถึงโมไบล์โหนดปลายทาง หลังจากนั้น ในขั้นตอนที่สองจะทำการค้นหาเส้นทางที่เหมาะสมที่สุดเพื่อให้ได้เส้นทางที่มั่นคงกว่าที่มีความล้าช้าของการส่งผ่านข้อมูลที่ต่ำกว่า ในขั้นตอนแรกโปรโตคอลคิวดีเอสอาร์จะใช้กระบวนการการจองเส้นทางโดยใช้อัลกอริทึมการจองแบนด์วิดธ์แบบทีดีเอ็มเอ เพื่อทำการจองสล๊อตที่ต้องการสำหรับการส่งข้อมูล สำหรับขั้นตอนที่สองจะคำนวณความล้าช้าและความเข้มของสัญญาณของเส้นทางในกระบวนการการค้นหาเส้นทาง และทำการรวมพารามิเตอร์ทั้งสองในฟังก์ชันการรับประกันคุณภาพ (ฟังก์ชันคิวโอเอส) เพื่อใช้ในการพิจารณาเส้นทางที่เหมาะสมที่สุด โดยวิธีการนี้โปรโตคอลคิวดีเอส

อาร์จะพยายามค้นหาเส้นทางที่มีเสถียรภาพมากที่สุด เพื่อให้สามารถใช้เส้นทางนั้นในการส่งข้อมูลได้นานกว่า ดังนั้นเราสามารถลดกระบวนการการบำรุงรักษาเส้นทางเพื่อเลือกเส้นทางใหม่ในกรณีที่เส้นทางที่เลือกไม่สามารถใช้งานได้

เราทำการประเมินประสิทธิภาพของโปรโตคอลคิวดีเอสอาร์ที่เรานำเสนอ โดยใช้วิธีการจำลองสถานการณ์ เราทำการเปรียบเทียบผลลัพธ์จากการจำลองสถานการณ์กับโปรโตคอลต้นแบบ เช่น โปรโตคอลดีเอสอาร์ และโปรโตคอลการค้นหาเส้นทางประเภทออนดีมานด์แบบอื่นที่พิจารณาคิวโอเอส เช่น โปรโตคอลคิวโอเอส-เอโอดีวี ผลจากการจำลองสถานการณ์แสดงให้เห็นว่าโปรโตคอลคิวดีเอสอาร์สามารถปรับปรุงประสิทธิภาพให้กับระบบในเทอมของค่าทรูพุทเฉลี่ย อัตราการส่งแพ็กเก็ตที่สูงกว่า และสามารถลดอัตราการสูญเสียของแพ็กเก็ต และมีโอเวอร์เฮดของการค้นหาเส้นทาง ได้ดีกว่าผลจากการจำลองสถานการณ์ของโปรโตคอลดีเอสอาร์และโปรโตคอลคิวโอเอส-เอโอดีวี

Thesis Title	QoS-Based Dynamic Source Routing Protocol in Mobile Ad-Hoc Networks
Student	I Wayan Mustika
Student ID.	49060725
Degree	Master of Engineering
Program	Computer Engineering
Year	2008
Thesis Advisor	Asst. Prof. Dr. Sakchai Thipchaksurat

ABSTRACT

Mobile Ad-hoc Networks (MANETs) consist of a collection of wireless mobile nodes that form a network and are able to communicate each other without cable, backbone, based station or centralized administration. With the rapidly growing of multimedia applications in MANET such as on-demand video streaming and teleconferencing, the Quality of Service (QoS) requirements for such applications have raised new challenge. However, the current routing protocols in MANETs are mainly based on non-QoS requirement, for example the source node attempts to transmit data to the destination node without any delay or bandwidth guarantee.

We present a QoS routing scheme in MANET called QoS-Based Dynamic Source Routing (QDSR) protocol which constitutes an extension of Dynamic Source Routing protocol, in which QoS features are embedded in the routes selection procedures. QDSR protocol can be classified into two steps: first is to provide the bandwidth reservation from source to destination and the second is attempt to find the optimal route through stronger path with lower delay. For the first step, QDSR employ the route reservation procedure using a TDMA-based bandwidth reservation algorithm to reserve slots required for transmitting data. The second step is to calculate the link delay and link signal strength during route discovery process and combines these metric into a QoS function for selecting the optimal route. By using this approximation, QDSR attempts to find the most stable links which lead to longer-lived routes and reduce route maintenance process.

We evaluate the performance of QDSR protocol by means of simulation. The simulation results prove that QDSR provides the significant improvement in terms of average throughput, packet delivery ratio, lower packet loss rate and lower routing overhead comparing with both the original DSR and another on-demand QoS routing protocol, such as QoS-AODV.

ACKNOWLEDGEMENTS

Many thanks are given to my great advisor, Asst. Prof. Dr. Sakchai Thipchaksurat, for his continuous support and help on all aspects from technical to material. His guidance and advice helped me overcome many technical obstacles which otherwise would take much more efforts. I also would like to thank Assoc. Prof. Dr. Ruttikorn Varakulsiripunth, a leader of Communication Networks Laboratory in Research Center for Communications and Information Technology (ReCCIT), KMITL, for his extensive contributions, comments and encouragement to improve my research as well as my thesis.

Thanks also go to my co-advisor at Tokai University, Prof. Dr. Hiroshi Ishii. I am very grateful for his time to visit KMITL and give valuable comment which brought up some interesting idea to conduct the collaboration research. Financial support from AUN/SEED-Net JICA project for this research and scholarship is greatly acknowledged.

Many of my colleagues had contribution for this thesis. I would like to thank my special friends, Mr. Saiyan Saiyod, Mr. Tawatchai Chontong and Mr. Boonchai Jindathawornkij, who have accompanied me throughout my master journey. They have always been there when I needed them. Without their support and helpful discussion, I could not have succeeded in my research as well as my life in Thailand. I will always remember the time we had together and wish them all success in their careers and happiness in their lives. I also want to thank all members of Communication Network Laboratory for their help and support to my research in the Laboratory.

Special thanks are given to Ketut Sriasih for her love, constant support and patience, which make this long journey full of joy.

This thesis is dedicated to my father and mother. Their love, encouragement, belief and understanding made everything I have possible.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT)

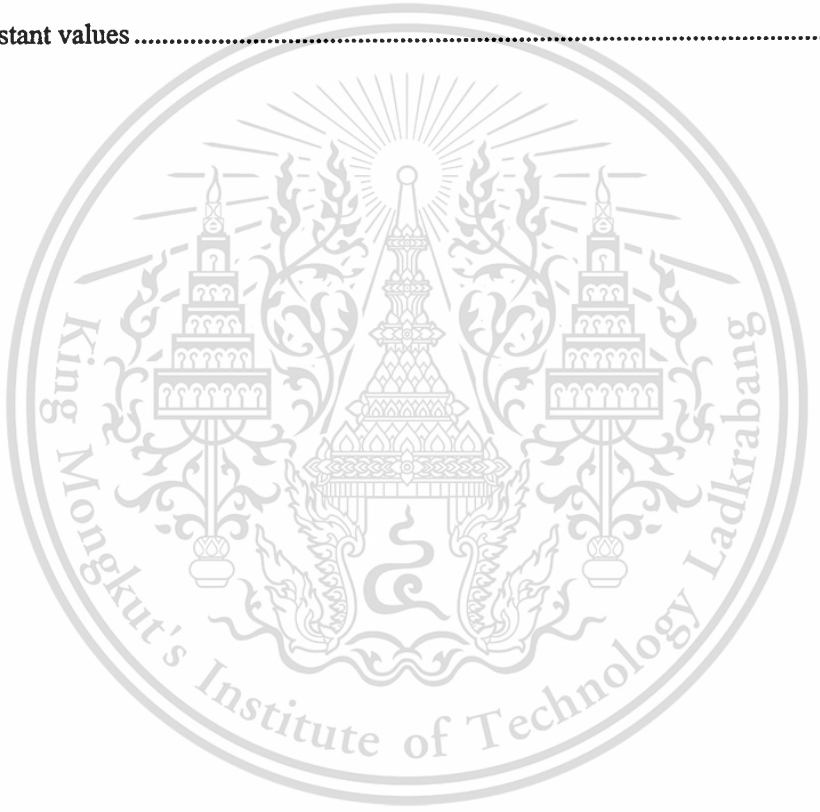
TABLE OF CONTENTS (CONT)

TABLE OF CONTENTS (CONT)

LIST OF FIGURES

LIST OF FIGURES (CONT)

LIST OF TABLES



Chapter 1

Introduction

1.1 Statement of the Problems

The ability to communicate with anyone from anywhere has been mankind's dream for a long time. Wireless is the only medium that can enable such kind of communication. However, with the recent technological advances, together with the demand for flexibility and mobility of wireless systems, the development of an emerging "anyone, anywhere, anytime" paradigm of mobile ad hoc networking becomes a reality. Mobile Ad Hoc Networks (MANETs) have the potential way to serve as the basic building blocks of the future "ubiquitous communication and computing" systems, capable of interconnecting thousands of heterogeneous devices.

As the emergence of multimedia applications such as live voice or video conference, web TV and radio generated much interest in MANETs, QoS routing features are desired to support such kind of applications. Multimedia applications are typically delay-sensitive and have specific bandwidth requirement. In order to guarantee the real-time and multimedia traffic in MANETs where nodes can move unpredictable and randomly, routing protocol must be able to find the route that satisfy the QoS requirement. There are many researches on routing protocols in MANETs [1, 4, 5, 10], however, the most existing work is based on non-QoS requirement for example the source node attempts to transmit data to the destination node without any delay or bandwidth guarantee. In addition, the existing QoS routing protocols [2, 3, 14, 15] merely focus only on one application's QoS requirement based on a single QoS constraint which is not usually suitable to the highly unpredictable and dynamic ad hoc environment.

In this research, we consider with the combination among multiple QoS constraint to find the efficient route that satisfy the QoS requirement. We consider the bandwidth metric as the fist priority because bandwidth guarantee is the most critical requirements for real time applications. While combination between delay and signal strength metrics are considered as the secondary importance in order to provide mobile nodes a better chance for surviving over a period of time from node movement.

1.2 Goal and Objective

Over the last few years, many routing protocols for MANETs have been proposed and enhanced to efficiently route data packets between two nodes in a network. However, most of those routing protocols are based on non-QoS requirement and provide the best effort for data traffic. Real time and multimedia traffic have different characteristics than the non-multimedia traffic in which multimedia traffic is sensitive with delay and bandwidth. Therefore, the routing protocol must be able to find the efficiently route from source to destination that satisfy the QoS requirement.

The objectives of this thesis are to 1) propose an on-demand routing protocol for QoS support in MANETs, 2) to simulate and analyze proposed routing protocol under the same network parameters and mobility scenarios, and 3) to evaluate that protocol based on its performance. This evaluation will be done through simulation. It was also desirable to compare the results with the results in both on demand routing protocol and another on demand QoS routing protocol.

The goal of this thesis was to:

- Get a general understanding of mobile ad-hoc networks (MANETs), routing protocols, and QoS issues in MANETs.
- Develop the on-demand QoS routing protocol to support the multimedia traffic in MANETs.
- Analyze the protocol through simulation.
- Compare the performance of propose protocol with the existing routing protocol in MANET

1.3 Theory

A Mobile Ad-Hoc Network (MANET) is an autonomous collection of wireless mobile nodes with no predetermined topology or central control. The nodes intercommunicate through single-hop or multi-hop paths in a peer-to-peer fashion and operate themselves as hosts as well as routers and thus, the routing takes place without existence of fixed infrastructure as shown in Figure 1.1

MANETs technology can broadly applied in two main areas. The first area extends the current wired and wireless networks by adding new mobile nodes that use MANET technology at the edge of the network, for example MANET that used in home office, educational

buildings, organization conferences and many other similar situations. The second area where MANETs technology provide a practical way to rapidly build a decentralized communication network in areas where either there is no any wireless communication infrastructure available or the pre-existing infrastructure cannot be used. MANETs can be used in any situation that involves emergency, such as search-and-rescue operations, battlefields, disaster environments [1] and many others.

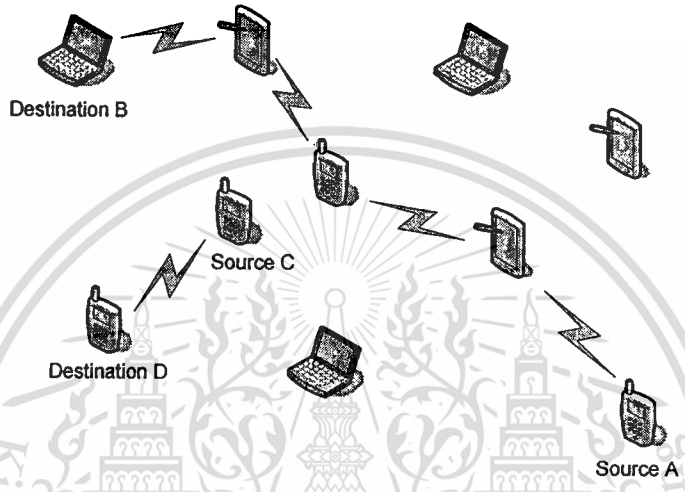


Figure 1.1 A mobile ad-hoc network

A central challenge in the design of ad hoc networks is the development of dynamic routing protocols that can efficiently find routes between two communicating nodes. It is not clear, however, how different protocols behave in different environments. A protocol may be the best, one to use in one network topology and mobility scenario, but the worst to use in another. In addition, the routing protocol must be able to keep up with the high degree of node mobility that often changes the network topology drastically and unpredictably. Generally, routing protocols in ad hoc networks can be classified into three different groups [1, 4, 5, 10]: *Proactive* (*Table Driven*), *Reactive* (*On-Demand*) and *Hybrid*. In proactive routing protocols, each node has a number of tables to maintain routing information to every other node in the network and these tables are updated periodically, while reactive routing protocols finds the routes only when desired. Hybrid routing protocols combine proactive and reactive routing protocols where proactive is used in a cluster called zone whereas reactive is used to find nodes outside the cluster. Proactive and reactive routing protocols are used in flat architecture, while hybrid routing protocols are used in hierarchical architecture instead.

1.4 Process of the Study

The research was begun with the comprehensive review of the concept mobile ad-hoc network (MANET), review of the existing routing protocols in MANET including its properties and the comparisons of each protocol. Then, we observe the QoS issues and challenge in MANETs which concern with the problems in the dynamic environment. Finally, we proposed the on-demand QoS routing scheme in which QoS feature are embedded in the route selection and route maintenance process. Our proposed algorithm is done by modifying the algorithm inside DSR routing protocol module. We write the simulation script in OTCL to define the network (number of nodes), the traffic in the network (sources, destination, type of traffic) and which protocol we will use in simulation. The result of simulations is an output trace file. Finally, we analyze the results by using the *awk* command and Perl scripts.

1.5 Scope and Organization of this Thesis

In the second chapter, we will study the proposed and most "popular" routing protocols for mobile ad-hoc wireless networks that may be suitable for use in a dynamic environment.

In the third chapter, we will describe the detail of Dynamic Source Routing protocol (DSR) including mechanism of the route discovery and route maintenance process. The key advantage and optimizations of DSR is also described in this chapter.

In the fourth chapter, we will explain in detail of our proposed scheme, called Quality of Service Based Dynamic Source Routing which constitutes an extension of DSR routing protocol, in which QoS features are embedded in the routes selection and the route maintenance procedures.

In the fifth chapter, we evaluate and compare the performance of QDSR with the original DSR and another QoS routing protocol. The performance analysis of our proposed scheme will be done using the simulation software. As for the simulation software, we will use the Network Simulator *ns*-2 under Linux platform (Ubuntu), as this software has been used in several citations related to this thesis's simulation experiments, and thus, we can have comparable results.

Finally, in the sixth chapter we summarize the results obtained in chapter five into conclusions.

Chapter 2

Routing Protocols in Mobile Ad Hoc Networks

2.1 Protocols Overview

The Internet Engineering Task Force MANET working group suggests two different types of metrics for evaluating the performance of routing protocols of MANETs [7]. In accordance with [7], routing protocols should be evaluated in terms of both qualitative metrics and quantitative metrics in which for measure its suitability and performance. Qualitative metrics include:

Distributed operation: This is an essential property which the protocol should be distributed and should not be dependent on centralized controlling node. This is the case even for stationary networks. The difference is that nodes in ad hoc network can enter or leave the network easily and because of mobility of the network can be partitioned.

Loop Freedom: This refers mainly, but not only, to all protocols that calculate routing information based on the Bellman-Ford algorithm. In a wireless environment with limited bandwidth, interference from neighboring nodes' transmissions and a high probability of packet collisions, it is essential to prevent a packet from "looping" in the network and thus consuming both processing time and bandwidth. Ad hoc solutions such as TTL values can bound the problem, but a more structured and well-formed approach is generally desirable as it usually leads to better overall performance.

On-Demand Routing Behavior: Due to bandwidth limitations in the wireless network, on-demand or reactive-based routing minimizes the dissemination of control packets in the network, increases the available bandwidth for user data, and conserves the energy resources of the mobile nodes. Reactive routing protocols introduce a medium to high latency.

Proactive Behavior: Proactive behavior is preferable when low latency is the main concern and where bandwidth and energy resources permit such behavior.

Unidirectional Link Support: Nodes in the wireless environment may be able to communicate only through unidirectional links. It is preferable that routing protocols be able to support both unidirectional and bidirectional links.

Sleep mode: In general, nodes in a MANET use batteries for their energy source and the life of battery is limited. The protocol should be able to operate, even though some nodes

are in "sleep mode" for short periods, without any adverse consequences in the protocol's performance.

Security: The wireless environments, along with the nature of the routing protocols in MANETs, which require each node to participate actively in the routing process, introduce many security vulnerabilities. Therefore, routing protocols should efficiently support security mechanisms to address these vulnerabilities.

In conclusion, a routing protocol for MANETs should keep a balance between latency and routing overhead, energy consumption, and node participation in the routing process, and should employ security mechanisms. For tactical and emergency communications, low latency and high packet delivery ratio are more important than low routing overhead. Energy consumption is not always an issue in emergency communications, as nodes may well be suited in vehicular platforms with unlimited energy resources. However, for group communications using portable man-pack radio devices, energy consumption is an important issue. For multimedia communications and applications, low latency, high throughput and low routing overhead are more important.

Quantitative metrics broadly include:

End-to-end data throughput and delay: Many metrics can be used to measure the effectiveness of the routing protocol. Design flaws that increase delay and minimize data throughput can be revealed by these metrics.

Route Acquisition Time: How much time does a protocol need for a route discovery? This is a main concern in reactive routing protocols, as the longer the time is the higher the latency is in the network.

Percentage Out-of-Order delivery: The percentage of packets that are delivered out-of-order will affect the performance of higher layer protocols such as TCP, which prefers in-order data delivery of packets.

Efficiency: Additional metrics can be used to measure the efficiency of the protocol. One can use them to measure the portion of the available bandwidth that is used by the protocol for route discovery and maintenance. Another measurement calculates the data packet delivery ratio over the total number of packets transmitted and the energy consumption of the protocol for performing its task.

All the above quantitative metrics should be based on the same network attributes, such as mobility, network density, data density, bandwidth, energy resources, transmission and receiving power, antenna types and any other "component" that a simulation tool could provide.

The performance evaluation of our proposed scheme will be based on some of the above quantitative metrics, which will be defined more precisely in chapter 4.

2.2 Classification of Routing Protocols

Routing protocols for MANETs can be classified into three main categories. (1) Proactive routing protocols, which each node determines the routes to all destinations at start up time and all routing information are maintained and updated periodically. (2) Reactive routing protocols, where routes are established when they are required by a source node using route discovery process. (3) Hybrid routing protocols combine proactive and reactive routing protocol together where proactive routing protocol is used in a cluster and reactive routing protocol is used to discover route between clusters. Normally Hybrid routing protocol is used in hierarchical network structure and the first two routings protocols are used in flat network structure.

2.2.1 Proactive routing protocols

In proactive routing protocols, each node has a number of tables to maintain routing information to every other node in the network and these tables are updated periodically. Examples of proactive routing protocols are Optimized Link State Routing (OLSR), Destination Sequenced Distance Vector (DSDV), Wireless Routing Protocol (WRP), Cluster-Head Gateway Switch Routing (CGSR) and so on. These protocols have some differences in term of the way for updating the routing information. Proactive protocols present low latency, but medium to high routing overhead, as the nodes periodically exchange control messages and routing-table information in order to keep up-to-date routes to any active node in the network. Therefore, these protocols may allow some significantly overheads and consume bandwidth in the network

2.2.1.1 Destination Sequenced Distance Vector (DSDV)

Destination Sequenced Distance Vector is a loop free routing protocol based on the classical Bellman-Ford algorithm [4]. Data packets are transmitted between the nodes using routing tables stored at each node. Each routing table contains all the possible destinations from a node to any other node in the network and also the number of hops to each destination.

The protocol has three main attributes: to avoid loops, to resolve the "count to infinity" problem, and to reduce high routing overhead. Each node issues a sequence number that is

attached to every new routing-table update message and uses two different types of routing-table updates, named "full" and "incremental dumps", respectively. The full dump packet carries all the available routing information and the incremental packet carries only the information changed since the last full dump. The full dump update messages are perform infrequently because the update size scales with the network size. The incremental update messages are sent more frequently to minimize the number of control messages disseminated in the network. Each node also keeps statistical data concerning the average setting time of a message that the node receives from any neighboring node. The data is used to reduce the number of rebroadcasts of possible routing entries that may arrive at a node from different paths but with the same sequence number.

The construction of routing-table in DSDV starts with the condition that every node in the network periodically exchange control messages with its neighbors to set up multihop paths to any other node in the network. Each individual route to every destination is tagged with a destination sequence number, which is issued by the destination node. Any route to a destination with a higher destination sequence number replaces the same route with a smaller destination sequence number in the node's routing table, regardless of the number of hops to this destination. Higher destination sequence number indicate the most recent or fresh route on the routing table. Every node then immediately advertises any significant change in its routing table, such as a link failure to its neighboring node(s), but waits for a certain amount of time to advertise other changes.

The average time to get all the updated advertisement called the "settling time". It is calculated by maintaining, for every destination, weighted average of the most recent updates of the routes. By implementing this advertising scheme, DSDV tries to minimize the number of route updates transmitted by a node. Thus, when a node receives a route update for a destination from one of its neighboring nodes, and a few seconds later, it receives a second update from a different neighboring node for the same destination with the same destination sequence number, but a lower number of hops, the node does not immediately broadcast the change in its routing table. This is highly possible in a MANET, in which the network topology changes very dynamically. For this purpose, each node maintains a table with the destination address, the last settling time and the average settling time of this address. The node uses the information in this table to check the stability of the route to a destination. Table 2.1 shows the structure of a node's routing table.

Table 2.1 DSDV routing table

Destination	Next hop	Number of Hops	Sequence Number	Install Time	Stable Data
A	A	0	A_221	Time1	Ptr_A
B	C	1	B_109	Time2	Ptr_B
C	B	4	C_222	Time2	Ptr_C
D	B	2	D_325	Time3	Ptr_D

As the position of a node in a network constantly changes over time, each node needs to advertise to its neighboring nodes any change in its routing table. However, this approach can lead to a high overhead of control messages in the network, leaving no available bandwidth for user data. With the concept of full and partial routing-table advertisement, each node that encounters a change in the path to any destination may choose to advertise only this particular change instead of its complete routing table. This partial advertisement of a node's routing table is called an "incremental update". If many changes occur in a node's routing table, the node may choose to advertise full routing-table entries, called a "full update", instead of many incremental updates.

DSDV has certain advantages that cannot be overlooked. First, the simplicity of the protocol is very similar to the classic Distance Vector, with only small modifications to avoid loops, with the use of destination sequence numbers. DSDV also presents low latency, as every node always has a route to any destination in the network. However, DSDV does not scale well in networks with high mobility, as the broken links create a "storm" of route updates. This situation may severely degrade network performance, in which the available bandwidth is limited. Another disadvantage of DSDV is that every node in the network must periodically broadcast changes or full updates of its routing table. Those frequent and periodic route updates in the network will also result in high routing overhead and consume bandwidth in the network.

2.2.1.2 Optimized Link State Routing (OLSR)

Optimized Link State Routing [6] is based on the link state algorithm and it is proactive (table-driven) in nature. This protocol has been modified and optimized to efficiently operate MANET environment. The main concept of the protocol is to adapt the changes of the network without creating control messages overhead due to the protocol flooding nature. OSLR have only a subset of the nodes, named Multipoint Relays (MPRs), in the network that responsible

for broadcasting control messages and generating link state information. The idea of MPR is to minimize the flooding of broadcast packet in the network by reducing duplicate retransmission in the same region. A second optimization is that every MPR may choose to broadcast link state information only between itself and the nodes that have selected it as an MPR.

In OSLR, only multipoint relays (MPR) are designated for link state updates and packet forwarding. To select the MPRs, each node periodically broadcasts a list of its one hop neighbors using hello messages. From the list of nodes in the hello messages, each node selects a subset of one hop neighbors, which covers all of its two hop neighbors.

In a typical flooding-based approach, a node broadcasts a message either if it is the originator or if it has not received this message before. Thus, the number of messages transmitted in the network is almost as large as the number of the nodes in the network. Figure 2.1a shows a typical flooding scenario. Figure 2.1b shows the flooding in the entire network when using MPRs.

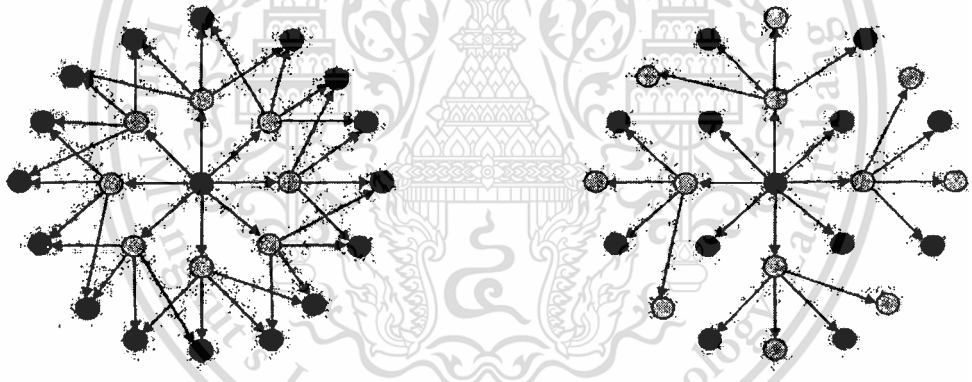


Figure 2.1 Pure flooding and MPR flooding

As we can see in Figure 2.1b, only the MPRs, the grey-colored nodes, broadcast messages in the network. It is clear that the number of broadcasted messages can be greatly reduced by the MPRs' implementation.

Although the optimal number of MPRs for a node reflects a smaller number of broadcasting messages in the network, MPR selection is based only on a heuristic that is proposed in [6]. The set that consists of the nodes that are multipoint Relays is called *MPR set.* Each node *N* in the network selects an MPR set that processes and forwards every link state packet that node *N* originates. The neighboring nodes of *N* that are not in the MPR set process this packet but do not further broadcast it. A node *N* also maintains a subset of neighbors, named *MPR selectors,* which is the set of the neighbors that have selected *N* as one of their

MPRs. Each node may have one or more MPRs. A condition for the selection of an MPR node is the assurance of bidirectional links between it and its selectors.

Each node in a network maintains a routing table that enables a source node to send data packets to a destination node. Table 2.2 shows the structure of a node's routing table in OLSR. Four different types of information are used for the construction, calculation and maintenance of routing information. Every node in the network obtains all the information necessary for the construction of its routing table with a periodic transmission of messages. The node, upon receiving this information, updates and recalculates its routing table. When a link breaks or if the network topology changes, no messages other than those defined above are required for the update of the routing table.

Table 2.2 OLSR routing table

Destination address	Next hop address	Number of hops to destination	Network interface
Destination address	Next hop address	Number of hops to destination	Network interface
Destination address	Next hop address	Number of hops to destination	Network interface

The main advantages of OLSR are low latency and high data delivery ratio because each node in the network maintains an up-to-date routing table with all the destinations in the network. Thus, no additional connection set-up time is required for a node to send data packets to another node in the network. However, the main disadvantage of this protocol comes from its proactive nature and the flooding mechanism, despite the use of the MPRs. OLSR may introduce high routing overhead, consuming a large portion of the available bandwidth.

2.2.2 Reactive routing protocols

In reactive routing protocols, every node in the network obtains a route to a destination on a demand fashion. When the source node wants to send data packet, the protocol initiates a route discovery process, if such a route does not already exist, to find a path to the destination. Route discovery usually occurs by flooding a route request packets through the network. When a node with a route to the destination (or the destination itself) is reached, a route reply is sent

back to the source node. Several reactive routing protocols have been proposed such as Ad hoc On-demand Distance Vector (AODV), Dynamic Source Routing Protocol (DSR), Temporally Ordered Routing Algorithm (TORA) and so on. These protocols can be classified into two groups: source routing and hop-by-hop routing [1]. Reactive protocols do not maintain up-to-date routes to any destination in the network and do not generally exchange any periodic control messages. Thus, they present low routing overhead, but high latency as compared to proactive protocols.

2.2.2.1 Ad Hoc On-Demand Distance Vector (AODV)

Ad Hoc On-Demand Distance Vector [8] is a reactive routing protocol that is based on the Bellman-Form algorithm and uses originator and destination sequence numbers to avoid both "loops" and the "count to infinity" problems that may occur during the routing calculation process.

AODV, as a reactive routing protocol, does not explicitly maintain a route for any possible destination in the network. However, its routing table maintains routing information for any route that has been recently used within a time interval; so a node is able to send data packets to any destination that exists in its routing table without flooding the network with new Route Request (RREQ) messages. In this way, AODV try to minimize the routing overhead in the network caused by the frequent generation of routing control messages.

The route discovery process is performed as follow. When a source node or originator has data to send to the destination node, the source node looks in its routing table to find a route to desired destination node. If there is no such route, or the route is marked as invalid by an appropriate flag, the source node broadcasts a RREQ message to its neighboring nodes. Before sending the RREQ message, the RREQ ID and the originator sequence number in the message header are incremented by one. In this way, each RREQ message is uniquely identified by combining the above numbers with the originator IP address. Any intermediate node that receives an RREQ message, takes one of the following three actions: First, the intermediate node discards the RREQ message if it has previously received the same RREQ message. If the intermediate node has a valid route to the destination node, it reverses a RREP message back to the originator. If the intermediate node does not have a valid route to the destination, it further propagates the message to its neighboring nodes. The destination node, which finally receives the RREQ message, increments the destination sequence number and reverses an RREP message back to the source node. When the source node receives the RREP message, it updates

its routing table with the "fresh" route. Figure 2.2 shows the route discovery process from source node 1 to destination node 10.

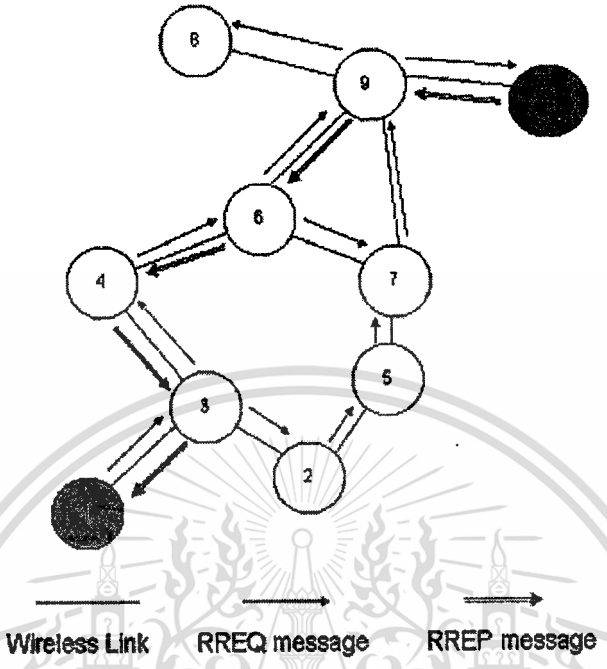


Figure 2.2 AODV route discovery process

AODV uses mainly two mechanisms to avoid high routing overhead caused by its flooding nature. The first mechanism involves a binary exponential back off to minimize congestion in the network. The second one involves an expanding ring-search technique in which the originator node starts broadcasting a RREQ message and the TTL value is set to a minimum default value, normally set by one. If the originator node does not receive a RREP message within a certain time interval, it exponentially increments the time interval and increases the diameter of the searching ring. The maximum value for the ring diameter is set by default to 35, which is the maximum value of the network diameter for AODV. The route maintenance process in AODV is very simple. When the link in the path between node 1 and node 10 breaks (Figure 2.3) the upstream node that is affected by the break, in this case node 4 generates and broadcasts a RERR message. The RERR message eventually ends up in source node 1. Upon receiving the REER message, source node 1 will generate a new RREQ message.

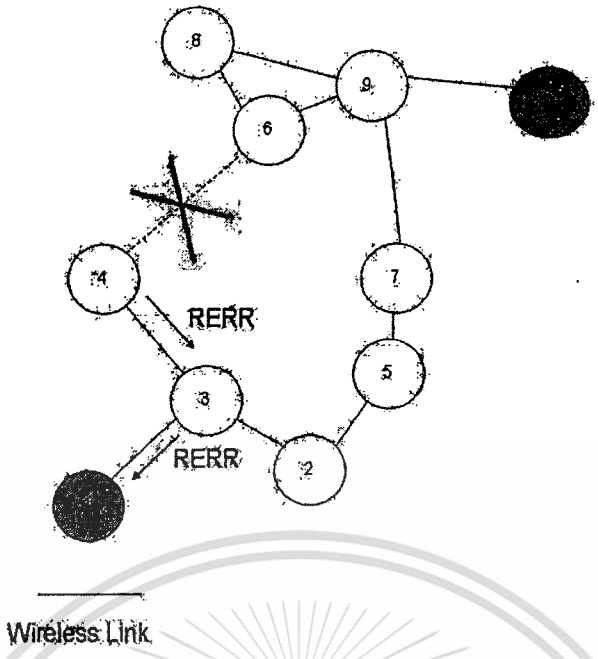


Figure 2.3 AODV RERR message generation

Finally, if node 2 already has a route to node 10, it will generate a RREP message, as indicated in Figure 2.4. Otherwise, it will re-broadcast the RREQ, as in Figure 2.2.

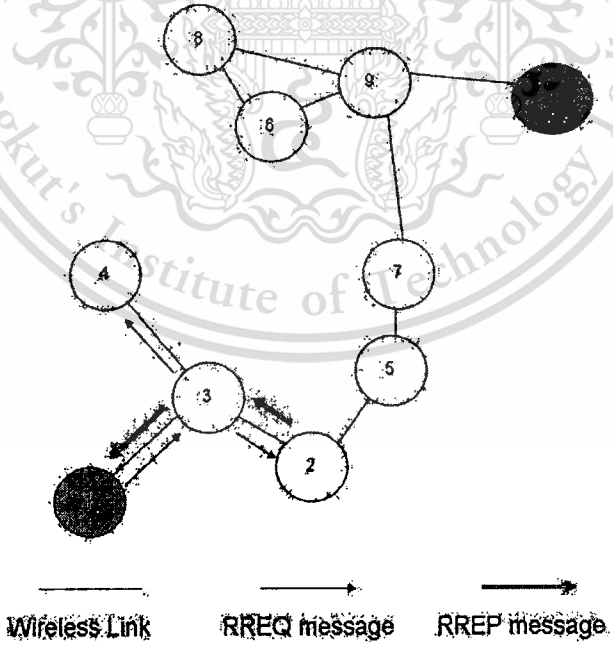


Figure 2.4 AODV route maintenance process

The advantages of AODV are its reactive nature, which reduces the routing overhead in the network and the use of destination sequence numbers that address routing loops and the "count to infinity" problem. However, control message overhead can be introduced when every

intermediate node originates a RREP message, to satisfy a route discovery request if it has a valid route to the destination, causing a RREP messages "storm". Another advantage of AODV is adaptable to highly dynamic networks. However, node may experience large delays during route construction, and link failure may initiate another route discovery, which introduces extra delays and consumes more bandwidth as the size of the network increases. Another disadvantage of AODV is that the propagation of periodic HELLO messages from a node, to maintain connectivity with its neighboring nodes, will lead to bandwidth consumption.

2.2.2.2 Dynamic Source Routing Protocol (DSR)

Dynamic Source Routing [5] is a simple reactive protocol that is based on two main mechanisms: route discovery and route maintenance. Both mechanisms are implemented in an on-demand fashion and in the absence of any kind of periodic control messages. As the name implies, DSR uses the "source routing" concept, in which means the sender knows the complete hop-by-hop route to the destination. Each node "caches" the routes to any destination it has recently used, or discovered by overhearing its neighbors' transmission. When there is not such route, a route discovery process is initiated. The detail process of the route discovery and route maintenance of DSR is presented in chapter 3.

The main advantage of DSR is the absence of any periodic control messages that would take over a portion of the available bandwidth. The route discovery and maintenance optimization techniques further eliminate the propagation and dissemination of control messages. However, DSR does not employ any local repair of a broken link and as any intermediate node can respond with a RREP message to a RERR message, based on its route cache, there is a possibility for unstable routes in the network. DSR was designed for a network with a limited number of nodes. High mobility in the network will cause frequent link breaks that will result in high routing overhead.

2.2.2.3 Temporally Ordered Routing Protocol (TORA)

The Temporally Ordered Routing Protocol (TORA) is a highly adaptive loop free distributed routing protocol that is based on the link reversal algorithm [9]. TORA was proposed to operate in highly dynamic mobile networking environment. The main concept of this protocol is that the network for any source node can be "visualized" as a Directed Acyclic Graph (DAG) rooted at the destination node. When a link between the source and the destination fails, the nodes reverse the direction of the links and update the previous nodes in

the path. Additionally, each node maintains multiple paths to a given destination and is capable of detecting any partitions in the network.

To accomplish such behavior, a value, "height," is associated with each node at all times. These values can be ordered in comparison to the "height" of each neighboring node. Data flow occurs from a node with a higher value to a node with a lower value. When a node cannot detect the height value of one of its neighbors, it does not forward data packets to that node.

TORA disseminates control messages in a small local area, not in the entire network, thus preserving bandwidth and minimizing processing time in the nodes. When a link failure occurs, there is no need for a large-scaled dissemination of control packets, as they can be limited to the small region where the link failure occurs.

TORA requires bidirectional links between the nodes in the network and synchronization from an internal or external mechanism, e.g., Global Positioning System (GPS). The protocol is loop-free, as the route formation is based on the DAG that is a loop-free data structure, and supports only unicasting routing. Finally, TORA is not a self-operating protocol, but requires the existence of the Internet MANET Encapsulation Protocol (IMEP) as the underlying network layer protocol.

TORA has four basic functions: route discovery, route maintenance, route erasing, and route optimization. The route discovery process is performed as follow. When data packets have to be sent from a source node to a destination node and there is no such route, the source node broadcasts a query (QRY) packet to its neighboring nodes. This QRY packet is further forwarded from the intermediate nodes and finally reaches the destination node. When the destination node receives the QRY packet, it replies with an update (UPD) packet and sets its distance, "height," to the destination to the lowest value, as it is the destination node. Any intermediate node that receives the UPD packet sets its distance to the destination to a value higher than the sender's value of the UPD packet. If any intermediate node has a route to the destination, it replies with an UPD packet and sets accordingly its distance to the destination. Eventually, all nodes in the path will have adjusted their heights, based on the heights of their neighboring nodes, and a DAG from the source to the destination node will be created. Figure 2.5 is a visual presentation of the DAG formation from source node 1 to destination node 9.

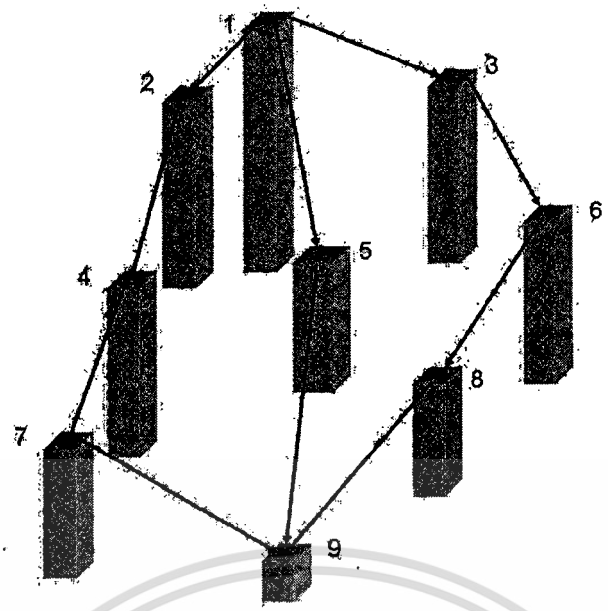


Figure 2.5 DAG formation in TORA

Height values [9] are assigned to each node in the paths, and data "flows" from source node 1, which has the higher value, to destination node 9, with zero height value.

TORA also employs a route optimization mechanism by which the destination node initiates the route discovery process by sending an optimization (OPT) packet. The destination node sets its height in the OPT packet and broadcasts the packet to its neighbors. Any intermediate node that receives the packet reselects its height and further broadcasts an OPT packet to the network. The scope behind the optimization function is to enable a proactive behavior in networks with a low rate of topology changes, so that nodes can have in advance one or more routes to any reachable destination in the network.

Route maintenance in TORA is required when a link failure occurs in the path between the source and the destination node. However, the protocol bounds the link-repair procedure to a small area close to the link failure. If the link between nodes 7 and 9 breaks due to network topology changes, node 7 will reverse the link between itself and node 4, it will change its distance to node 9 to a higher value than its neighbors' value, and it will generate an UPD message. Node 4, which will receive the message, it will reverse the link between itself and node 2 and will forward an UPD message to node 2, which eventually will reach source node 1. Figure 2.6 shows the changes in the DAG due to the link failure.

In this example, node 1 has multiple alternate routes to destination node 9 that it will use to continue sending data packets. If node 1 did not have any alternate routes, it would generate a QRY message, as was described in the previous section. TORA also has the ability

to detect any partition in the network. A partition may occur when one or more nodes are unreachable from the source node. In Figure 2.7, we can see how the protocol reacts when the link between nodes 4 and 2 fails. Node 4 reverses the link between itself and node 2, and sends an UPD message to node 7. However, the link reversal conflicts with the previous one in which the link between nodes 7 and 9 failed. This is an indication of a network partition. When a node detects a partition, it updates its link status table and broadcasts a clear (CLR) message to inform other nodes of the network partition.

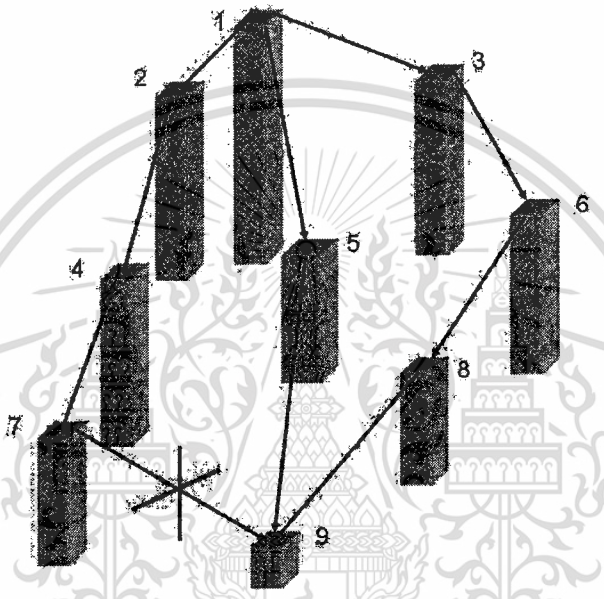


Figure 2.6 Link break in TORA

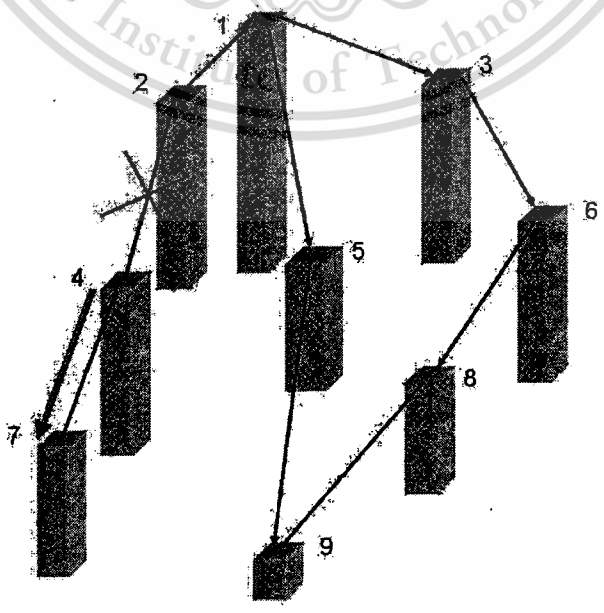


Figure 2.7 TORA network partition detection

The reactive nature of the protocol, along with its capacity to limit the dissemination of control messages to the region close to a link failure, minimizes the amount of routing control messages in the network. In TORA, there is a potential for oscillations to occur, especially when multiple sets of coordinating nodes are concurrently detecting partitions, erasing routes, and building new routes based on each other. Because TORA uses internodal coordination, its instability problem is similar to the count-to-infinity problem in distance vector routing protocols except that such oscillations are temporary and route convergence will ultimately occur. This is regarded as a disadvantage of the protocol. However, TORA could be a good choice for large mobile ad-hoc networks, in which link-state and distance-vector based algorithms will produce a significantly large number of control messages due to frequent changes of the network topology.

2.2.3 Hybrid routing protocols

Hybrid routing protocols are a new generation of protocol, which are combine both proactive and reactive routing protocols. These protocols are designed to increase scalability by allowing nodes with close proximity to work together to form some sort of a backbone to reduce the route discovery overheads. In these protocols, every node acts reactively in the region close to its proximity and proactively outside of that region, or zone. Several hybrids routing protocols have been proposed such as Zone Routing Protocol (ZRP), Greedy Perimeter Stateless Routing (GPSR) Zone-based Hierarchical Link State (ZHLS) and so on, but the most popular protocol is ZRP. Hybrid protocols take advantage of both reactive and proactive protocols, but may require additional hardware, such as GPS, separated or integrated into the communication device.

2.2.3.1 Zone Routing Protocol (ZRP)

Zone Routing Protocol [10] is a distributed routing protocol that integrates both a proactive and a reactive scheme for route discovery and maintenance. The basic idea of the protocol is the creation of areas, or zones, where every node proactively maintains one route or multiple routes to any destination inside the zone and reactively obtains routing information for any node outside of the zone. Therefore, for nodes within the routing zone, routes are immediately available. For nodes that lie outside the routing zone, routes are determined on-demand (i.e. reactively), and it can use any on-demand routing protocol to determine a route to the required destination.

The zone diameter may be defined in advance, before nodes form the network, or it may be optimized by every node, based on ZRP traffic measurements. The radius of a node's zone plays a significant role in the proper behavior of the protocol. If the network consists of a large number of nodes with medium to low mobility or the demand for routes is high, a large value for the radius is preferable to avoid the frequent dissemination of routing requests and reply messages. On the other hand, if the network consists of a small number of nodes with high mobility or the demand for routes is small, the radius value should also be small to avoid overhead of periodic routing update messages.

ZRP consists of two main protocols. The *Intrazone Routing Protocol* (IARP) [10] is responsible for finding and maintaining valid routes in the internal zones between any source/destination pair at all times. The *Interzone Routing Protocol* (IERP), [10] is responsible for finding any available route outside of the node's internal zone. The scope behind this implementation is to reduce routing overhead and delay and to respond better in the topological changes of the network.

Figure 2.8 shows the routing zones of node 10 with two different values for the radius. In the first case, where *radius* = 2, node 10 maintains a small number of available routes in its routing table. In the second case, where *radius* = 3, node 10 maintains more routes available in its routing table, in exchange for a larger number of routing update messages. When *radius* = 1, ZRP behaves like a pure proactive protocol.

ZRP is a loop-free protocol and provides support for unidirectional links, hierarchical routing, and interconnection with other non-ZRP routing domains when every node's network interface is assigned a unique IP address.

The route discovery process in ZRP depends on the location of the destination node. If the destination node is located inside the source node's intra zone, the protocol acts like any other proactive protocol, thus ensuring that there is always a route to any destination in the intra zone. When the destination node is located outside of the source's intra zone, the source node initiates a route discovery process and the IERP is assigned to accomplish this task. To avoid large-scaled dissemination of routing request messages ZRP employs a third protocol, the *Bordercast Resolution Protocol* (BRP) [10], which is a sub-layer of the IERP protocol. The BRP identifies the nodes that are located in the source node's zone perimeter and forwards the route request messages only to those peripheral nodes. Figure 2.9 shows the route discovery process from node 10 to node 8 for an intra zone, *radius* = 2.

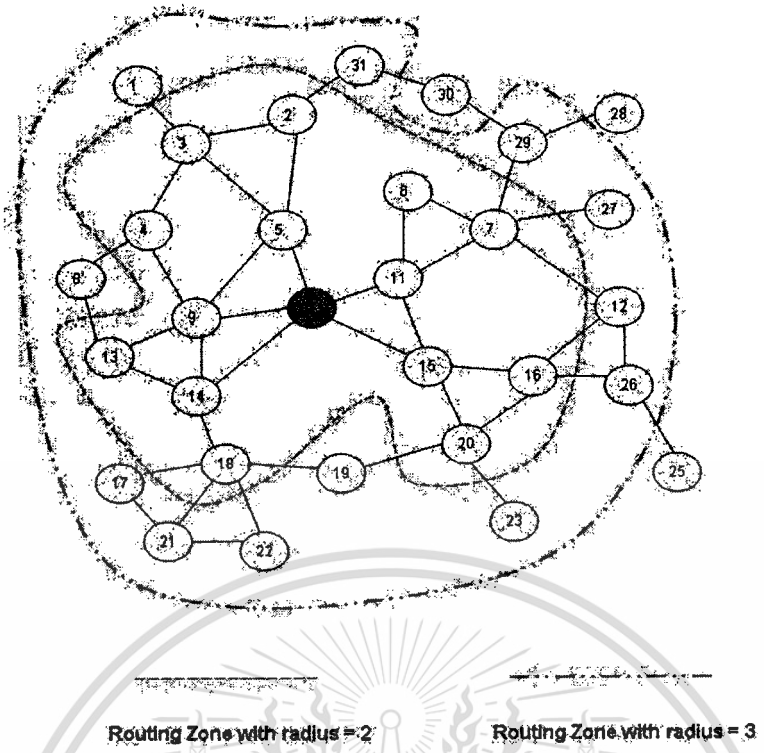


Figure 2.8 Routing zones in ZRP

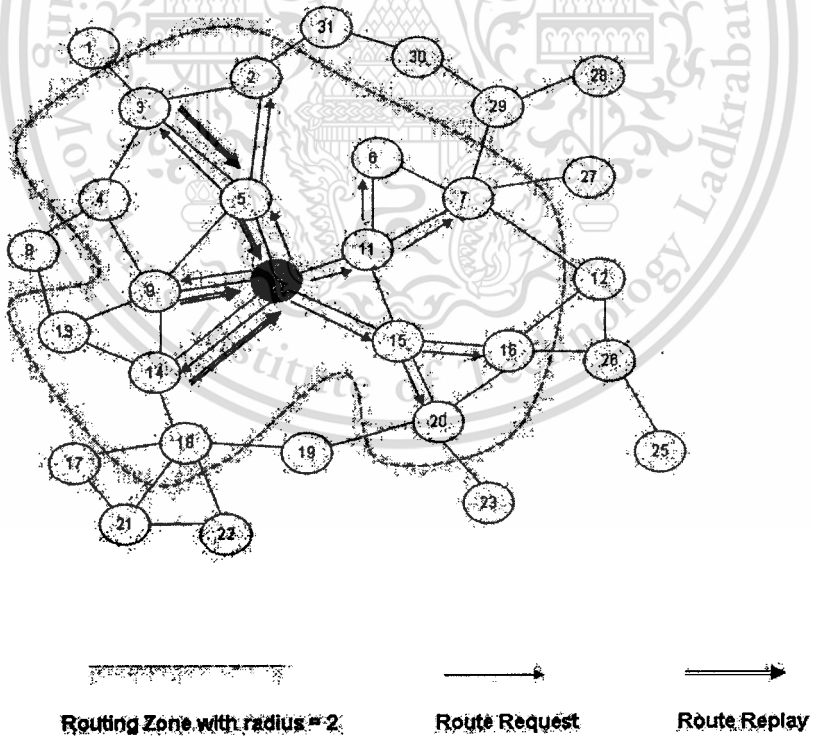


Figure 2.9 Route discovery in ZRP

As node 8 is located outside the node 10 intra-zone, the BRP underlying protocol border-casts the RREQ messages to the peripheral nodes 2, 3, 4, 6, 7, 13, 16, 18, and 20.

However, as nodes 3, 4, 9, and 14 already have a route to node 8, they initiate a RREP message without further forwarding the RREQ message. When node 10 receives multiple routes to the same destination, it stores the routes in its routing table and selects the shortest one, based on the number of hops. It can be observed in the Figure 2.9 that there is a possibility of collisions when multiple nodes transmit the RREP messages back to the source. However, the border-casting scheme minimizes the propagation of RREQ messages within a small region, except when the source/destination pair is located at opposite edges of the network. When a peripheral node does not have a route to the destination node, it originates a RREQ message and border-casts the message to its peripheral nodes. That procedure continues until a route to the destination is found.

Route maintenance takes place when a node in an active route detects a link failure in the route: the node employs a local reconfiguration of the path by searching for an alternate route to the destination. If such a route exists, the node originates an update message to inform all other nodes in the path and the source node of a change in the path. The source node may continue sending data packets in the new non-optimized route. If the source node wants to obtain a new optimal route, it regenerates a RREQ message, as previously discussed.

ZRP seems to employ the best characteristics of both reactive and proactive protocols. It avoids flooding the network with large-scaled Route Request messages, as reactive protocols do, and the periodic exchange of HELLO messages in the proactive scheme. Thus, ZRP reduces routing overhead in an inexpensive way. The only visible drawback of the protocol is, perhaps, that its performance depends heavily on the zone radius. By optimizing the zone radius, we can succeed in the high performance of the protocol, or at least a better performance than any other "pure" proactive or reactive protocol.

2.2.3.2 Greedy Perimeter Stateless Routing (GPSR)

Greedy Perimeter Stateless Routing is a hybrid protocol whose functionality depends on knowledge of the geographic location of the nodes in network [11]. That knowledge can be obtained by integrating a GPS device into the communication device or by other available means. Every node in the network must know its own location and the location of its neighboring nodes. Thus, every node periodically broadcasts its address and its location in x and y coordinates to all of its neighboring nodes. Data-packet forwarding decisions are based on the locations of both the source and the destination node. An address-to-location look-up algorithm is implemented to map a node address to its location.

A periodic exchange of beacons, which encapsulate the node address and location, is similar to the behavior of proactive protocols. The absence of any periodic route table information is closer to the philosophy of reactive protocols.

GPSR employs two algorithms to forward data packets from a source to a destination node: the greedy forwarding algorithm and the perimeter forwarding. The objective of the protocol's design is to minimize routing overhead and increase the packet delivery ratio in a network, by effectively responding to network topology changes without the dissemination of large scaled control messages.

GPSR presents certain advantages over other protocols we have studied. First, it does not use any type of control messages, such as route requests and error messages. Second, it does not flood the network with any other type of control messages, except those between a node and its neighbors, for location-finding purposes. Perhaps the only visible drawback of GPSR is its dependence on "external" devices, such as GPS, that increase the implementation cost. The presence of a GPS device introduces another drawback if it does not provide accurate node-location information. Any malfunction of the GPS device will degrade the protocol's performance and may lead to network crash.

Chapter 3

Dynamic Source Routing Protocol

The Dynamic Source Routing protocol (DSR) is a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes [5]. This protocol was proposed by David B. Johnson, David A. Maltz, and Josh Broch. DSR is designed to reduce overhead and to provide highly reactive services to ensure successful delivery of data packet from source to destination in spite of node movement or other changes in network conditions. By using DSR, the network is completely self-organizing and self-configuring, without the need for any existing network infrastructure or administration. Network nodes such as computers cooperate to forward packets for each other to allow communication over multiple "hops" between nodes that not directly within wireless transmission range of one another. DSR routing protocol is also automatically determined and maintained all routing as nodes in the network move about or join or leave the network, and as wireless transmission conditions such as sources of interference change.

DSR protocol is composed of the two main mechanisms of "Route Discovery" and "Route Maintenance", which work together to allow nodes to discover and maintain routes to arbitrary destinations in the ad hoc network. Route Discovery is the mechanism by which a node **S** wishing to send a packet to a destination **D** obtains a source route to **D**. To reduce the cost of Route Discovery, each node maintains a Route Cache of source routes it has learned or overheard. Route Maintenance is the mechanism by which a packet's originator **S** detects if the network topology has changed such that it can no longer use its route to the destination **D** because some of the nodes listed on the route have moved out of range of each other. The basic operation of the DSR protocol is shown in Figure 3.1.

All aspects of the protocol operate entirely "on-demand" fashion. In particular, unlike other protocols, DSR requires no periodic control packets of any kind at any layer within the network. For example, DSR does not use any periodic routing advertisement, link status sensing, or neighbor detection packets, and does not rely on these functions from any underlying protocols in the network. Thus, the pure on-demand properties allow the routing packet overhead of DSR to scale automatically to only that needed to react to changes in the routes currently in use.

The protocol allows multiple routes to any destination and allows each sender to select and control the routes used in routing its packets, for example for use in load balancing or for increased robustness. Other advantages of the DSR protocol include easily guaranteed loop-free routing, support for use in networks containing unidirectional links, use of only "soft state" in routing, and very rapid recovery when routes in the network change.

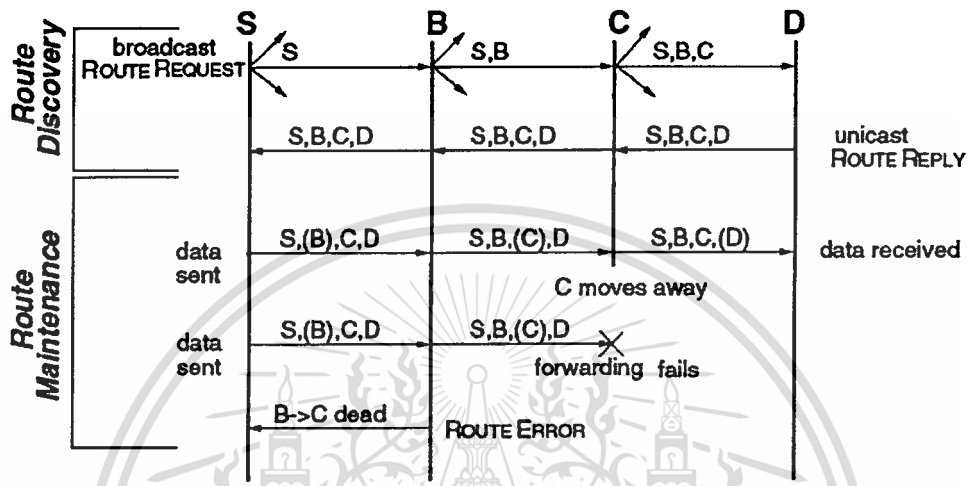


Figure 3.1 Basic operation of the DSR protocol

3.1 Source Routing

The main concept of DSR protocol is "source routing", in which the sender learns the complete, ordered sequence of network hops necessary to reach the destination, and, at a conceptual level, each packet to be routed carries this list of hops in its header. The key advantage of a source routing design is that intermediate nodes do not need to maintain up-to-date routing information in order to route the packets that they forward, since the packets themselves already contain all the routing decisions.

Basing the routing protocol on source routes also has two additional benefits. First, the protocol can be trivially proved to be loop-free, since the source route used to control the routing of a packet is, by definition, of finite length, and it can be trivially checked for loops. Second, each source route is a statement that a particular path is believed to exist through the network. As source routes travel through the network riding on control packets, such as Route Requests or Route Replys, or the data packets whose forwarding they control, any node overhearing a source route can incorporate the information it contains into its Route Cache. At the cost of no overhead above that used to carry out the normal operation of the protocol, the protocol itself spreads topology information among the nodes in the network. The information

carried by a source route on a data packet also has the useful property that the more frequently heard routes and the most recently heard routes are the most likely to contain accurate information, since those routes are currently being tested by the packets flowing along them.

Although DSR uses source routes, and each packet is routed based on a discovered source route, recent improvements to DSR have made it so that most packets do not need to incur the overhead of carrying an explicit source route header [5]. This allows DSR to achieve the benefits of information aggregation, loop-freedom, and source routing with out the significant overhead cost in terms of bytes.

3.2 Route Discovery

Route discovery works by flooding a request through the network to find a route to some target destination. In its simplest form, when a source node wants to send a data packet to the destination node in the network, it first looks in its "Route Cache" to find a route to the destination. If such a route exists, the source node attaches to the packet header the complete route to the destination and forwards the packet to the next node. The next node checks the packet header and forwards the packet to the next node. The process terminates when the packet reaches the destination. If the source node cannot find a route to the destination in its "Route Cache", it initiates a route discovery process. Before originating the Route Request, source node chooses a *request_id* to place into the Request such that the pair <*originating_address*, *request_id*> is globally unique. The route request (RREQ) is then broadcasted to its neighboring nodes. Each of the neighboring nodes checks in its Route Cache, and if it finds such a route, it sends a Route Reply (RREP) message back to the originator with the complete path to the destination. Otherwise, the destination node is obliged to do this task. DSR, by default, determines that a node may retransmit a RREQ message up to sixteen times. However, to avoid a big number of control messages being disseminated in the network, and sometimes for an inactive node, DSR uses expanding ring and exponential back off mechanisms in the route discovery process. Figure 3.2 shows the Route Discovery process in a simple network scenario contains five nodes. The source node (A), intermediate node (B, C, and D), and destination node (E) have different task in the Route Discovery process.

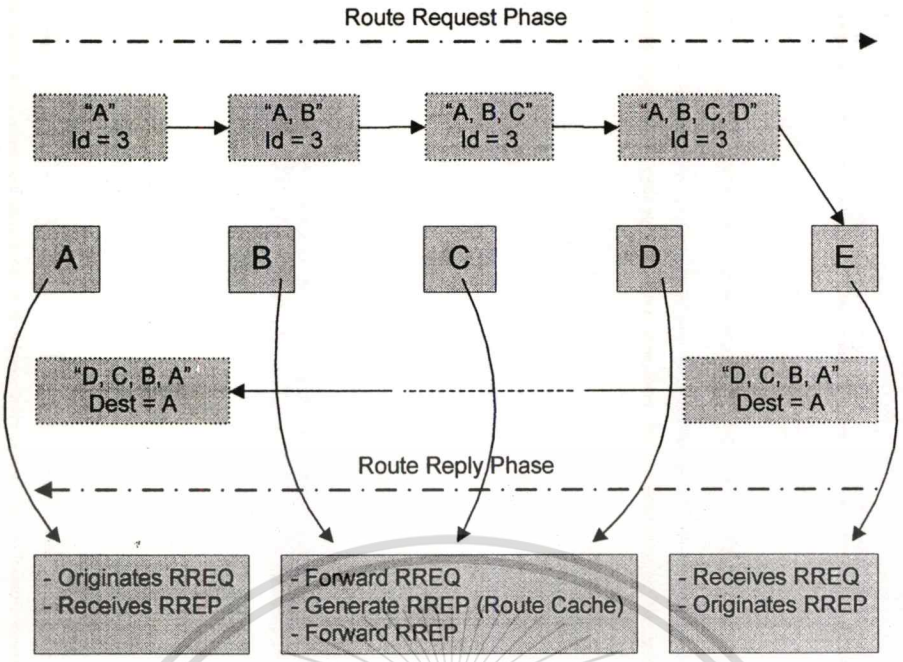


Figure 3.2 DSR route discovery process

In Figure 3.3, we assume that there is no path from source node 1 to destination node 10, so node 1 initiates the route discovery process (route request phase) by generating the RREQ packet. Node 9 discards the RREQ packet forwarded by node 7, as it has already received the same packet from node 6.

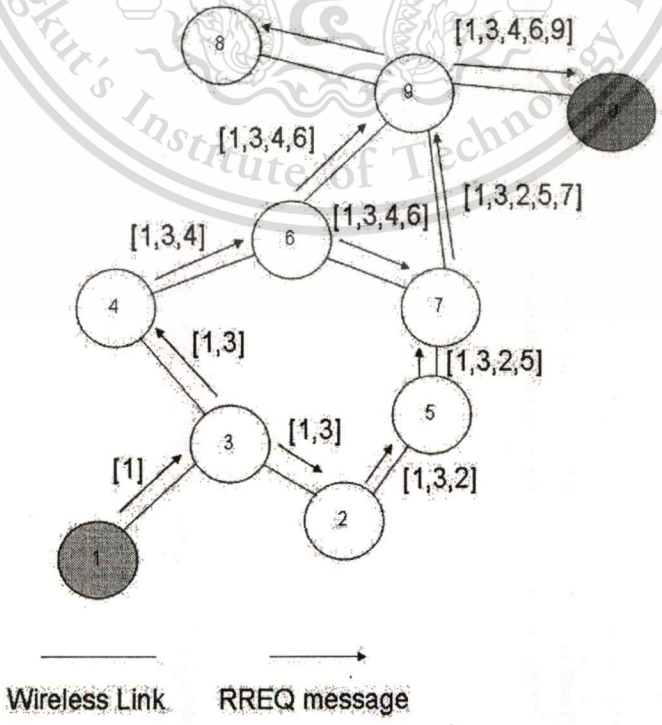


Figure 3.3 DSR RREQ message broadcasting

When node 10 receives the RREQ packet, it initiates a RREP packet and attaches in the packet header the reverse path to node 1. Figure 3.4 shows the RREP packet process.

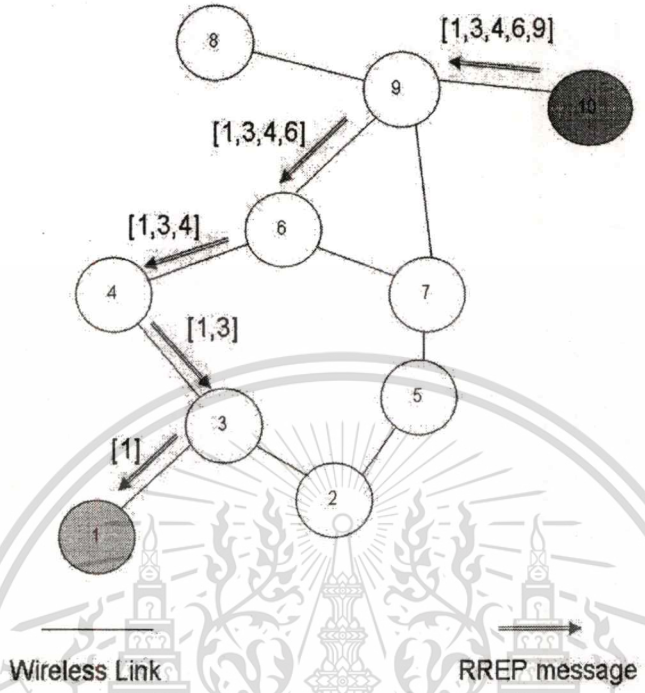


Figure 3.4 DSR RREP message processing

The route discovery process utilizes two types of messages, a Route Request and a Route Reply to actively search a route to the desired destination. Figure 3.5 and 3.6 show the format of both messages. Nodes related to Route Discovery process are classified into three groups, that is, source node, intermediate node and destination node as described in the section 3.2.1, 3.2.2 and 3.2.3 respectively.

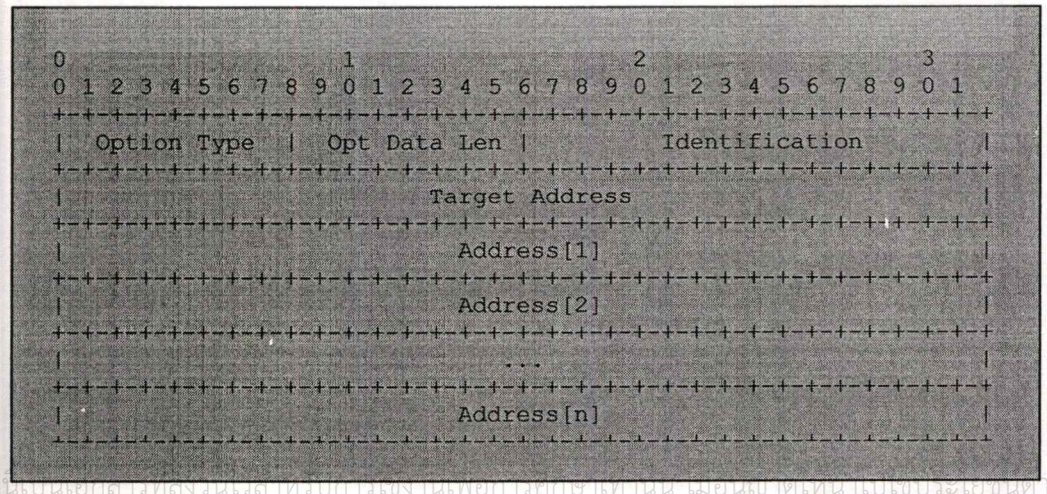


Figure 3.5 DSR route request message format

IP fields:

Source Address: The address of the node originating this packet. Intermediate nodes that retransmit the packet to propagate the Route Request MUST NOT change this field.

Destination Address: The IP limited broadcast address (255.255.255.255).

Hop Limit (TTL): Varies from 1 to 255, for example to implement non-propagating Route Requests and Route Request expanding-ring searches.

Route Request fields:

Option Type: Value: 2. A node that does not understand this option MUST discard the packet and the Option Data may change en-route.

Opt Data Len: 8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

Identification: A unique value generated by the initiator (original sender) of the Route Request. This value allows a recipient to determine whether or not it has recently seen a copy of this Route Request; if it has, the packet is simply discarded. When propagating a Route Request, this field must be copied from the received copy of the Request being forwarded.

Target Address: The home address of the node that is the target of the Route Request.

Address[1..n]: Address[i] is the address of the *i*-th node recorded in the Route Request option. The address given in the Source Address field in the IP header is the address of the initiator of the Route Discovery and must not be listed in the Address[i] fields. Thus the address given in Address[1] is thus the address of the first node on the path after the initiator. The number of addresses present in this field is indicated by the Opt Data Len field in the option (n = (Opt Data Len - 6) / 4).

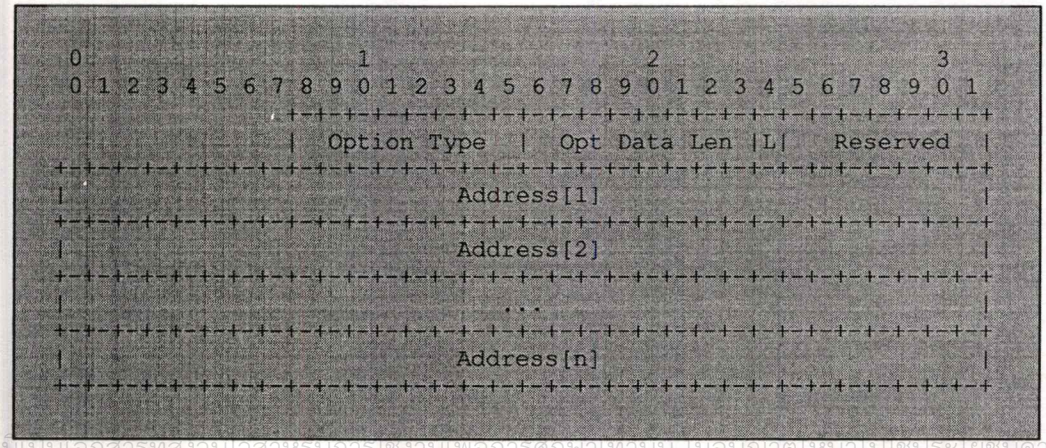


Figure 3.6 DSR route reply message format

IP fields:

Source Address: The address of the node sending the Route Reply.

Destination Address: The address of the source node of the route being returned. Copied from the Source Address field of the Route Request generating the Route Reply, or in the case of a gratuitous Route Reply, copied from the Source Address field of the data packet triggering the gratuitous Reply.

Route Reply fields:

Option Type: Nodes not understanding this option will ignore this option.

Opt Data Len: 8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

Last Hop External (L): Indicate that the last hop given by the Route Reply (the link from Address[n-1] to Address[n]) is actually an arbitrary path in a network external to the DSR network; the exact route outside the DSR network is not represented in the Route Reply.

Reserved: Must be sent as 0 and ignored on reception.

Address[1..n]: The source route being returned by the Route Reply. The route indicates a sequence of hops, originating at the source node specified in the Destination Address field of the IP header of the packet carrying the Route Reply, through each of the Address[i] nodes in the order listed in the Route Reply, ending with the destination node indicated by Address[n]. The number of addresses present in the Address[1..n] field is indicated by the Opt Data Len field in the option (n = (Opt Data Len - 1) / 4).

3.2.1 Source Node

As depicted in Figure 3.2, a source node involving with Route Discovery has two main tasks, originating route query (RREQ) and receiving route reply (RRPL).

3.2.1.1 Originating a Route Request (RREQ)

Before broadcasting a route request packet for discovering destination node, the source node initializes a Route Request option (Figure 3.5) in a DSR options header. The Route Request option must be included in a DSR options header in the packet. To initialize the Route Request option, the source node performs the following sequence of steps:

Step 1) The Option Type in the option must be set to the value 2.

Step 2) The Opt Data Len field in the option must be set to the value 6. The total size of the Route Request option when initiated is 8 octets; the Opt Data Len field excludes the size of the Option Type and Opt Data Len fields themselves.

Step 3) The Identification field must be set to a new value, different from that used for other Route Requests recently initiated by this node for this same target address. For example, each node may maintain a single counter value for generating a new Identification value for each Route Request it initiates.

Step 4) Set the Target Address field to be the IP address that is the target of this Route Discovery. The Source Address in the IP header of this packet must be the node's own IP address. The Destination Address in the IP header of this packet must be the IP "limited broadcast" address (255.255.255.255).

Step 5) Broadcasts the route request packet

Besides the fives steps above, the source node must maintain the information about Route Request in its Route Request Table. When initiating a new Route Request, the node must use the information recorded in the Route Request Table entry for the target of that Route Request and it must update that information in the table entry for use in the next Route Request initiated for this target.

The Route Request Table entry for the target records two important information, (1) Time To Live (TTL) field in the Route Query for the last Discovery, which allows the source node to implement various algorithms for controlling the spread of its route request on each Route Discovery initiated for a target and (2) the number of consecutive Route Requests initiated for this target since receiving a valid Route Reply giving a route to that target node, and the remaining amount of time before which this node may next attempt at a Route Discovery for that target node. A node must use these values to implement a back-off algorithm to limit the rate at which this node initiates new Route Discoveries for the same target address.

3.2.1.2 Receiving Route Reply (RREP)

When receiving route reply packet, the source node must do the following steps:

Step 1) Add the new route to its Route Cache including the information added from Route Reply option.

Step 2) Check each packet in its Send Buffer, a queue of packets before sending, to determine whether a route to that packet's IP Destination Address now exists in the node's Route Cache including the information just added to the Cache.

Step 3) If so, the packet should be sent using the route found in the Route Cache and remove from the node's Send Buffer.

3.2.2 Intermediate Node

Intermediate nodes involving with Route Discovery have three main tasks, (1) receiving and forwarding route request, (2) receiving and forwarding route reply (RREP) as illustrated in Figure 3.2 and (3) generating route reply by using the Route Cache and (3). These three tasks are going to be described in the following sections.

3.2.2.1 Receiving and Forwarding Route Request (RREQ)

When a node receiving route request packet, it performs the following steps:

Step 1) If the Target address field in the Route Request matches this node's IP address, then the node should originate a Route Reply (Section 3.2.3) and return a Route Reply packet to the initiator of this Route Request (the Source Address in the IP header of the packet).

Step 2) Else, the node must examine the route recorded in the Route Request option to determine whether this node's IP address already appears in the list of addresses. If so, the node must discard the Route Request packet.

Step 3) Else, the node decreases TTL value in the Route Request option to control the flooding of Route Request in the network. If TTL countdown to 0, the node must drop this route request packet and stop re-broadcast further.

Step 4) Else, the node must search its Route Request table for the entry of source node of this Route Request (the IP Source Address field). If such an entry is found in the table, then the node must search the node's cache for Identification values and the target node address of the Route Request to determine whether they just recently received. If such an (identification, target address) entry is found in this cache in the Route Request Table, then the node must discard the Route Request packet.

Step 5) Else, the node must process the Route Request according to the following steps:

- Copy identification and target address of this Route Request into its cache.
- Append this node's IP address to the list of Address[i] values in the Route Request option and increase the value of the Opt Data Len field in the Route Request by 4 (the size of an IP address).
- Search its own Route Cache for a route to the target of this Route Request packet. If such a route is found in its Route Cache, this node must generate a Route Reply from Route Cache (next section) to the source node of this Route Request.
- If no route found from its cache, then this node should broadcast this route request packet to its neighbors.

3.2.2.2 Receiving and Forwarding Route Reply (RREP)

Once a node receives Route Request packet and in order to avoid propagating the route request packet toward the target of the Request, this node checks a route from this node to the target node in its Route Cache. If there is such a route, then Route Reply is generated by the node from its own cached route to the target of a Route Request. This task is called a "cached Route Reply", and this mechanism can greatly reduce the overall overhead of Route Discovery on the network by reducing the flood of Route Request.

If there is a route from the Route Cache for the Route Request to the target node, the node should construct and return a cached Route Reply as follows:

Step 1) The source route for this Route Reply is the sequence of hop addresses:

Initiator, Address[1], Address[2], ..., Address[n], *c*-route

where Initiator is the address of the initiator of this Route Request, each Address[i] is an address from the Route Request, and *c*-route is the sequence of hop addresses in the source route to this target node, obtained from the node's Route Cache.

Step 2) Send a Route Reply to the source node of the Route Request. The initiator of the Route Request is indicated in the Source Address field in the packet's IP header.

3.2.2.3 Generating Route Reply using the Route Cache

Once a node receives any route reply packet, it then adds some new information gathered from that packet to its own cache. Every time that a node adds new information to its

route cache, the node checks each packet in its own Send Buffer to determine whether a route to that packet's IP Destination Address now exists in the node's Route Cache. If so, the packet should be sent by using that route and removed from the Send Buffer.

3.2.3 Destination Node

As depicted in Figure 3.2, the destination node involving with Route Discovery has two tasks, receiving Route Request (RREQ) and Originating Route Reply Packet (RREP). Route Request packet can reach at any node in wireless range of the node that broadcasts or forwards RREQ, but every node receives this RREQ, it will check whether this RREQ is searching for itself as mentioned in section 3.2.2. If the node is the target of RREQ, then the node must originate Route Reply in order to reply to the source and control information is included in the Route Reply packet's field as described in Figure 3.6. The process of originating route reply can be described as below:

Step 1) The Option Type in the option must be set to the value 3.

Step 2) The Opt Data Len field in the option must be set to the value (n * 4) + 3, where *n* is the number of addresses in the source route being returned (excluding the Route Discovery initiator node's address).

Step 3) The Last Hop External (L) bit in the option must be initialized to 0.

Step 4) The Reserved field in the option must be initialized to 0.

Step 5) The Route Request Identifier must be initialized to the Identifier field of the Route Request to which this Route Reply is sent in response.

Step 6) The sequence of hop addresses in the source route are copied into the Address[i] fields of the option. Address[1] must be set to the first-hop address of the route after the initiator of the Route Discovery, Address[n] must be set to the last-hop address of the source route (the address of the target node), and each other Address[i] must be set to the next address in sequence in the source route being returned.

3.3 Route Maintenance

Route Maintenance is the mechanism by which a source node is able to detect if the network topology has changed such that it can no longer use its route to the destination because a link along the route no longer works. When Route Maintenance indicates that a source route

is broken, the source node can attempt to use any other route it happens to know to the destination node, or can invoke Route Discovery again to find a new route for subsequent packets to the destination node. Route Maintenance for this route is used only when the source node is actually sending packets to the destination node.

When forwarding a packet, a node must attempt to confirm the reach ability of the next-hop node, unless such confirmation had been received in the last MaintHoldoffTime period. If no confirmation is received after the retransmission of MaxMaintRexmt acknowledgement requests, after the initial transmission of the packet, and conceptually including all retransmissions provided by the MAC layer, the node determines that the link for this next-hop node of the source route is "broken". This confirmation from the next-hop node for Route Maintenance can be implemented using:

- Link-layer acknowledgement (Section 3.3.1)
- Passive acknowledgement (Section 3.3.2)
- Network-layer acknowledgement (Section 3.3.3)

If no acknowledgement is received after MaxMaintRexmt retransmissions, the node should originate a Route Error (Section 3.3.4) to the original sender of the packet (the source node).

3.3.1 Link Layer Acknowledgment

If the MAC protocol in use provides feedback as to the successful delivery of a data packet, then the use of the DSR Acknowledgement Request (Figure 3.7) and Acknowledgement Option (Figure 3.8) are not necessary. If such link-layer feedback is available, it should be used instead of any other acknowledgement mechanism for Route Maintenance, and the node should not use either passive acknowledgements or network-layer acknowledgements for Route Maintenance.

When using link-layer acknowledgements for Route Maintenance, the retransmission timing and the timing at which retransmission attempts are scheduled and generally controlled by the particular link layer implementation used in the network. For example, in IEEE 802.11, the link-layer acknowledgement is returned after the data packet as a part of the basic access method of the IEEE 802.11 Distributed Coordination Function (DCF) MAC protocol; the time

at which the acknowledgement is expected to arrive and the time at which the next retransmission attempt will occur are controlled by the MAC protocol implementation.

When a node receives a link-layer acknowledgement for any packet in its Maintenance Buffer, that node should remove that packet as well as any other packets in its Maintenance Buffer with the same next-hop destination.

3.3.2 Passive Acknowledgement

Passive acknowledgement is used for Route Maintenance when link-layer acknowledgements are not available or when originating or forwarding packets along any hop other than the last hop (the hop leading to the IP Destination Address node of the packet). In particular, passive acknowledgement is used for Route Maintenance in such cases if the node can place its network interface into "promiscuous" receive mode, and if network links used for data packets generally operate bi-directionally.

In order to use passive acknowledgements for a packet that it originates or forwards, a node considers the later receipt of a new packet (e.g., with promiscuous receive mode enabled on its network interface) to be an acknowledgement of this first packet if both of the following two tests succeed:

- The Source Address, Destination Address, Protocol, Identification, and Fragment Offset fields in the IP header of the two packets must match, and
- If either packet contains a DSR Source Route header, both packets must contain one, and the value in the Segments Left field in the DSR Source Route header of the new packet must be less than that in the first packet.

When a node hears such a passive acknowledgement for any packet in its Maintenance Buffer, that node should remove that packet, as well as any other packets in its Maintenance Buffer with the same next-hop destination.

3.2.3 Network-layer Acknowledgement

If both acknowledgement mechanisms above are not available for a node to determine the reach-ability of the next hop node in the source route for Route Maintenance, the node should request a network-layer acknowledgement from that next-hop node. In order to do this acknowledgement, the node inserts an Acknowledgement Request option in DSR Options header

in the packet. The Identification the packet The Identification field in that Acknowledgement Request option must be set to a value unique over all packets recently transmitted by this node to the same next-hop node.

When a node receives a packet containing an Acknowledgement Request option, then that node performs the following tests on the packet:

- If the indicated next-hop node address for this packet does not match any of this node's own IP addresses, then this node must not process the Acknowledgement Request option. The indicated next-hop node address is the next Address[i] field in the DSR Source Route option in the DSR Options header in the packet, or is the IP Destination Address in the packet if the packet does not contain a DSR Source Route option or the Segments Left there is zero.
- If the packet contains an Acknowledgement option, then this node must not process the Acknowledgement Request option.

If neither of the tests above fails, then this node must process the Acknowledgement Request option by sending an Acknowledgement option to the previous-hop node. In order to do this acknowledgement, the node performs the following sequence of steps:

- Create a packet and set the IP Protocol field to the protocol number assigned for DSR.
- Set the IP Source Address field in this packet to the IP address of this node, copied from the source route in the DSR Source Route option in that packet (or from the IP Destination Address field of the packet, if the packet does not contain a DSR Source Route option).
- Set the IP Destination Address field in this packet to the IP address of the previous-hop node, copied from the source route in the DSR Source Route option in that packet (or from the IP Source Address field of the packet, if the packet does not contain a DSR Source Route option).
- Add a DSR Options header to the packet, and set the DSR Options header's Next Header field to the "No Next Header" value.
- Add an Acknowledgement option to the DSR Options header in the packet; set the Acknowledgement option's Option Type field to 6 and the Opt Data Len field to 10.

- Copy the Identification field from the received Acknowledgement Request option into the Identification field in the Acknowledgement option.
- Set the ACK Source Address field in the Acknowledgement option to be the IP Source Address of this new packet (set above to be the IP address of this node).
- Set the ACK Destination Address field in the Acknowledgement option to be the IP Destination Address of this new packet (set above to be the IP address of the previous-hop node).
- Send the packet.

When using network-layer acknowledgements for Route Maintenance, a node should use an adaptive algorithm in determining the retransmission timeout for each transmission attempt of an acknowledgement request. For example, a node should maintain a separate round-trip time (RTT) estimate for each to which it has recently attempted to transmit packets, and it should use this RTT estimate in setting the timeout for each retransmission attempt for Route Maintenance.

Figure 3.7 Acknowledgement request message format

Acknowledgement Request fields:

Option Type: Nodes not understanding this option will remove the option and return a Route Error.

Opt Data Len: 8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

Identification: The Identification field is set to a unique value and is copied into the Identification field of the Acknowledgement option when returned by the node receiving the packet over this hop.

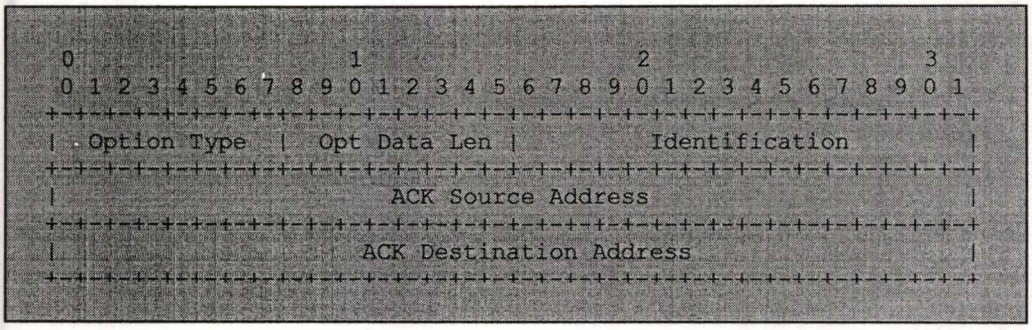


Figure 3.8 Acknowledgement option format

Acknowledgement Option fields:

Option Type: Nodes not understanding this option will remove the option.

Opt Data Len: 8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

Identification: Copied from the Identification field of the Acknowledgement Request option of the packet being acknowledged.

ACK Source Address: The address of the node originating the acknowledgement.

ACK Destination Address: The address of the node to which the acknowledgement is to be delivered.

3.3.4 Originating a Route Error

When a node is unable to verify reach-ability of a next-hop node after reaching a maximum number of retransmission attempts, a node should send a Route Error as described in Figure 3.8 to the IP Source Address of the packet.

A node transmitting a Route Error must perform the following steps:

- Create a new packet which its Source Address field is set by the value of the address of this node.
- If the Salvage field in the DSR Source Route option in the packet triggering the Route Error is zero, then copy the Source Address field of the packet triggering the Route Error into the Destination Address field in the new packet's IP header; otherwise, copy the Address[1] field from the DSR Source Route option of the packet triggering the Route Error into the Destination Address field in the new packet's IP header
- Insert a DSR Options header into the new packet.

- Add a Route Error Option to the new packet, setting the Error Type to NODE_UNREACHABLE, the Salvage value to the Salvage value from the DSR Source Route option of the packet triggering the Route Error, and the Unreachable Node Address field to the address of the next-hop node from the original source route. Set the Error Source Address field to this node's IP address, and the Error Destination field to the new packet's IP Destination Address.
- If the packet triggering the Route Error contains any Route Error or Acknowledgement options, the node may not append to its Route Error each of these options.
- Send the packet back to the source node.

3.3.5 Processing a Received Route Error Option

Besides originating route error of a node that detects broken route, another node that receives this route error packet that node must process the Route Error option as follow:

- Remove some dead links in this node's Route Cache which are identified by the Error Source Address field to the node identified by the Unreachable Node Address field (if this link is present in its Route Cache).
- If the option following the Route Error is an Acknowledgement or Route Error option sent by this node, copy the DSR options following the current Route Error into a new packet with IP Source Address equal to this node's own IP address and IP Destination Address equal to the Acknowledgement or Error Destination Address. Transmit this packet with the salvage count in the DSR Source Route option set to the Salvage value of the Route Error.

After processing the Route Error as described above, the node may initiate a new Route Discovery for any destination node for which it then has no route in its Route Cache as a result of processing this Route Error, if the node has indication that a route to that destination is needed.

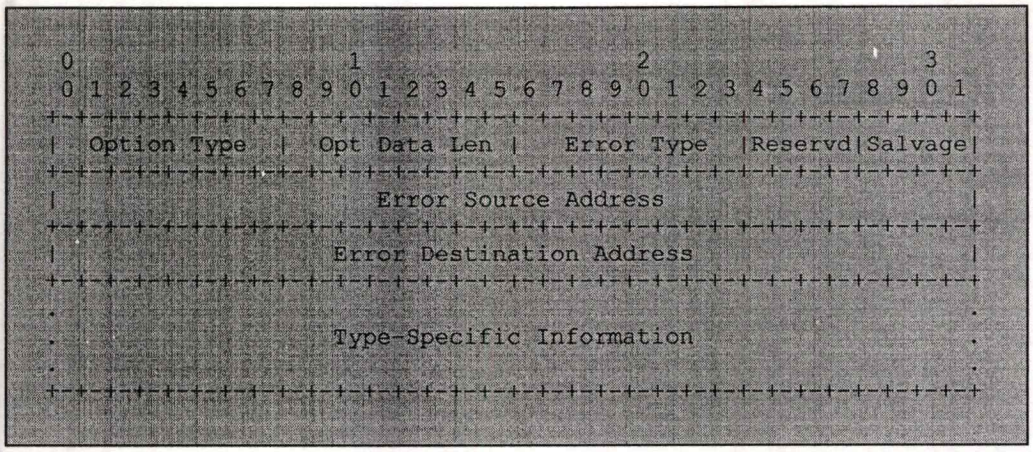


Figure 3.9 Route error option format

Route Error Option fields:

Option Type: Value is 3. Nodes not understanding this option will ignore this option.

Opt Data Len: 8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.

Error Type: The type of error encountered. Currently, the following type values are defined:

1 = NODE_UNREACHABLE

2 = FLOW_STATE_NOT_SUPPORTED

3 = OPTION_NOT_SUPPORTED

Other values of the Error Type field are reserved for future use.

Reservd: Reserved. Must be sent as 0 and ignored on reception.

Salvage: A 4-bit unsigned integer. Copied from the Salvage field in the DSR Source Route option of the packet triggering the Route Error.

Error Source Address: The address of the node originating the Route Error (e.g., the node that attempted to forward a packet and discovered the link failure).

Error Destination Address: The address of the node to which the Route Error must be delivered. For example, when the Error Type field is set to NODE_UNREACHABLE, this field will be set to the address of the node that generated the routing information claiming that the hop from the Error Source Address to Unreachable Node Address (specified in the Type-Specific Information) was a valid hop.

Type-Specific Information: Information specific to the Error Type of this Route Error message.

3.4 DSR Optimizations

DSR protocol incorporates optimizations based on aggressive caching and analysis of the available topology information. Aggressive caching can reduce the frequency of route requests broadcasts. More information about the topology can be obtained by operating the network interfaces on the nodes in the promiscuous mode and this enables eavesdropping on routes being used by nearby nodes. Additional topology information can be deduced by combining information from several routes. DSR optimizations are performed in the route discovery process, route maintenance process and caching strategy.

3.4.1 Optimization to Route Discovery

3.4.1.1 Non-propagating Route Requests

Each Route Request message contains a "hop limit" that may be used to limit the number of intermediate nodes allowed to forward the copy of Route Request. This hop limit is implemented using the Time-to-Live (TTL) field in the IP header of the packet carrying the Route Request. As the Request is forwarded, this limit is decremented, and the Request packet is discarded if the limit reaches zero before finding the target. DSR use this mechanism to send a "non-propagating" Route Request to determine if the target is currently a neighbor of the source node of if the neighbor node has a route to the target cached (effectively using the neighbor's caches as an extension of the initiator's own cache).

When performing Route Discovery, nodes first send a Route Request with the maximum propagation limit (hop limit) set to zero, such that any neighboring node receiving the initial transmission of the Route Request will not forward the Request to other nodes by re-broadcasting it. At the cost of a single broadcast packet, this mechanism allows a node to query the route caches of all its neighbors for a route and optimizes the case in which the destination node is adjacent to the source. If the non-propagating Route Request fails to elicit a reply within a short time limit, a "propagating" Route Request with a hop limit set to the maximum value is sent.

The hop limit mechanism is also considered to implement an "expanding ring" search for the target. For example, a node could send an initial non-propagating Route Request as above; if no Route Reply is received for it, a node could initiate another Route Request with a hop limit of 2. For each Route Request initiated, if no Route Reply is received for it, the node

doubles the hop limit used on the previous attempt, to progressively explore for the target node without allowing the Route Request to propagate over the entire network. However, this expanding ring search approach could have the effect of increasing the average latency of Route Discovery, since multiple Discovery attempts and timeouts may be needed before discovering a route to the target node.

3.4.1.2 Replying from Cache

If a node receives a Route Request for a destination to which it has a route, the node may generate a Route Reply based on its cached information instead of re-broadcasting the Route Request. In the Route Reply, this node sets the route record to list the sequence of hops over which this copy of the Route Request was forwarded to it, concatenated with the source route to this target obtained from its own Route Cache. This optimization is intended to both reduce the latency of Route Replys and prevent Route Requests from flooding through the entire network.

However, before transmitting a Route Reply packet that was generated using information from its Route Cache in this way, a node must verify that the resulting route being returned in the Route Reply, after this concatenation, contains no duplicate nodes listed in the route record.

3.4.1.3 Gratuitous Route Replies

When a node operating in promiscuous receive mode overhears a packet for which it is not the next hop, it scans through the list of addresses in the unprocessed portion of the source route looking for its own address. If its address is listed in this part of the source route, the node knows that packet could bypass the unprocessed hops preceding it in the source route. The node then sends a gratuitous Route Reply message to the packet's source, giving it the shorter route without these hops. Upon receiving the Route Reply, the originator will insert the shorter route into its route cache. The Route Cache can store more than one route to a destination, so the shorter route will not necessarily overwrite the longer route already in the cache. If the shorter route is found not to work, the originator can immediately revert to the longer route.

When deciding whether to return a gratuitous Route Reply, a node may factor in additional information beyond the fact that it was able to overhear the packet. For example, the node may decide to return the gratuitous Route Reply only when the overheard packet is received with a signal strength or signal-to-noise ratio above some specific threshold. In

addition, each node should maintain a Gratuitous Route Reply Table to limit the rate at which it originates gratuitous Route Replies for the same returned route.

3.4.1.4 Preventing Route Reply Storms

The ability for nodes to reply to a Route Request not targeted at them by using their Route Caches can result in a Route Reply "storm". In particular, if a node broadcasts a Route Request for a node that its neighbors have in their Route Caches, each neighbor may attempt to send a Route Reply, thereby wasting bandwidth and increasing the rate of collisions in the area. When the target the Route Discovery is a node with which many nodes communicate, such as a server, these simultaneous replies from the mobile nodes receiving the broadcast may create packet collisions among some or all of these replies and may cause local congestion in the wireless network. In addition, it will often be the case that the different replies will indicate routes of different lengths, since some nodes will be closer to the server than others.

If nodes are able to promiscuously listen to the channel, they can reduce the number of Route Replys sent to the originator of the Request by deferring their Reply for a time period based on the length of the source route in their Reply. If a node with a deferred Reply hears the originator of the Route Discovery use a source route to the target shorter than the one it is deferring, the node knows that the originator already has a shorter route to the destination, and it can cancel its deferred Route Reply.

3.4.2 Optimization to Route Maintenance

3.4.2.1 Salvaging

When an intermediate node forwarding a packet discovers that the next hop in the source route for the packet is unreachable, it examines its route cache for another route to the same destination. If a route exists, the node replaces the broken source route on the packet's header with the route from its cache and retransmits the packet. If a route does not exist in its cache, the node drops the packet and does not send a Route Request. In either case, the node attempting to perform salvaging returns a Route Error to the source of the data packet.

The intermediate node itself does not initiate a Route Discovery to attempt to heal the broken link because it is likely to be wasteful of network resources. The originating node is likely to have an alternate route to the destination, so any work done to heal the broken route is unneeded extra overhead. If the design of DSR were to change so that intermediate nodes did

attempt Route Discovery to heal the break, DSR would need a new type of Route Request packet that can request a route to any of several destinations. Since the intermediate node does not know which the node between itself and the final destination is the best node at which to rejoin the old route, it should send a Route Request targeting all of them. Simply requesting a route to what had been the next node on the broken route may result in a very suboptimal repair.

3.4.2.2 Gratuitous Route Errors

When a source node receives a Route Error for a packet that it originated, source node propagates this Route Error to its neighbors by piggybacking it on its next Route Request. In this way, stale information in the caches of nodes around source node will not generate Route Replys that contain the same invalid link for which source node received a Route Error.

3.4.3 Optimization to Caching Strategies

3.4.3.1 Snooping

When a node forwards a data packet, it "snoops" on the unprocessed portion of the source route and adds to its cache the route from itself to the final destination listed in the source route.

3.4.3.2 Tapping

Nodes operate their network interfaces in promiscuous mode, disabling the interface's address filtering and causing the network protocol to receive all packets that the interface overhears. These packets are scanned for useful source routes or Route Error messages and then discarded. This optimization allows a node to prime its route cache with potentially useful information, while causing no additional use of the limited network bandwidth.

Chapter 4

QoS-Based Dynamic Source Routing Protocol

With the rising popularity of multimedia applications in MANETs such as digital video, live voice and video conference, QoS features have raised new challenge. However, the current routing protocols in MANETs are mainly based on non-QoS requirement, for example the source node attempts to transmit data to the destination node without any delay or bandwidth guarantee. In order to support QoS guarantee in MANETs, we propose a routing scheme called QoS-Based Dynamic Source Routing (QDSR) protocol [12, 13], which extent some mechanism in Dynamic Source Routing protocol as described previously in Chapter 3.

4.1 Protocol Design

In QDSR protocol, data transmission is separated into two groups. The first group is the non-real time data transmission which no need bandwidth guarantee and allows the delay during transmission. Examples of this service are file transfer, web documents and other traditional datagram applications. The routing protocol will use the function that similar with original DSR protocol to provide the best effort for such data traffic. The second group is real time data transmission or multimedia transmission which needs QoS guarantee such as on-demand multimedia stream, audio and video conference, etc. QDSR constitutes an extension of DSR routing protocol, in which QoS features are embedded in the routes selection procedures [13].

When the real-time or multimedia traffic is applied, the protocol searches the route that satisfies the bandwidth requirement as the first priority. This is because bandwidth guarantee is one of the most critical requirements for real time applications. Once the bandwidth is available on the intermediate node, the protocol will attempt to find the optimal route such as more stable route and lower delay along the path to destination node. The optimal route selection is considered as the secondary importance. In order to find the optimal route, the protocol calculates the heuristic QoS function based on delay and signal strength metric obtained on the route discovery process. The higher value of QoS function implies the higher probability of lower delay and more stable link on the route.

4.2 Route Discovery

QDSR adds a QoS header to an ordinary route request (RREQ) packet and has the following fields: *<packet_type, source_address, destination_addr, sequence_number#, route_list, qos_enabled, link_info (delay, signal_strength), slot_array_list, bandwith_required, qos_function, TTL, data>*. Table 4.1 and 4.2 describe the fields used in both Route Request and Route Reply packets in QDSR.

Table 4.1 Route request packet fields in QDSR

Fields	Description
Source Address	The address of the node originating this packet.
Dest. Address	The IP limited broadcast address (255.255.255.255).
Hop Limit (TTL)	Varies from 1 to 255, for example to implement non-propagating Route Requests and Route Request expanding-ring searches.
Option Type	Value: 2. A node that does not understand this option MUST discard the packet and the Option Data may change en-route.
Opt Data Len	8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.
Sequence_number	A unique value generated by the initiator (original sender) of the Route Request.
Target Address	The home address of the node that is the target of the Route Request.
Route_list[1..n]	Route_list[i] is the address of the *i*-th node recorded in the Route Request option.
QoS enabled	Value = 1/0. If value is 1 indicate requesting the route for multimedia traffic. Otherwise, route request for common data traffic.
Link_info (delay, signal strength) [0..n]	Record the information of link (delay and power signal received) which a route request packet has traversed.
Slot_array_list [0.. n]	Record the state of slots of every node which a route request packet has traversed.
Bandwidth required (slots)	Number of slots required by the originator to send the multimedia traffic.
QoS function	Function value obtained from combination between delay metric and signal strength metric.

Table 4.2 Route reply packet fields in QDSR

Fields	Description
Source Address	The address of the node sending the Route Reply.
Dest. Address	The address of the source node of the route being returned.
Option Type	Nodes not understanding this option will ignore this option.
Opt Data Len	8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Opt Data Len fields.
Last Hop External (L)	Indicate that the last hop given by the Route Reply (the link from Address[n-1] to Address[n]) is actually an arbitrary path in a network external to the DSR network.
Reserved	Must be sent as 0 and ignored on reception.
Address[1..n]	The source route being returned by the Route Reply. The route indicates a sequence of hops, originating at the source node specified in the Destination.
Slot_array_list [0.. n]	State of slots of every node along the path and used for reservation process on when Route Reply traverse back to the source node.

Figure 4.1 shows an example of how the route discovery of QDSR is performed, starting from the source node S to the destination node D. When a source node S receives a call from the Application layer to create a new VC with specific bandwidth requirements, it broadcasts RREQ packet containing field of QoS enabled in RREQ packet is set to 1 and number of slots required for the reservation (Figure 4.1a).

When the RREQ message is received by A and E, it knows that both A and E are its neighbor, so the available path bandwidth from S to A or S to E is equal to receive link bandwidth from S to A or S to E. Path bandwidth S-A or S-E is calculated as the portion of the receive link bandwidth S-A or S-E. Then both A and E forwards the received RREQ packet with link bandwidth of S-A and S-E, add route address, compute the link information including link delay and link signal strength and combine them into a QoS function and finally rebroadcast RREQ message (Figure 4.1b).

When B and C get a RREQ packet from A and E respectively, B figures out that S is A's neighbor, so the available path bandwidth S-A-B is calculated using S-A and A-B receive link bandwidth. QoS function of path S-A-B and S-E-C is also calculated. Then B checks whether path bandwidths S-A and A-B are sufficient, and if they are B adds address of [S-A-B]

together with QoS function of path S-A-B to RREQ packet and rebroadcast it (Figure 4.1c). E also does the same procedure as done by B and finally rebroadcast RREQ packet which contains address [S-E-C} and QoS function of path S-E-C.

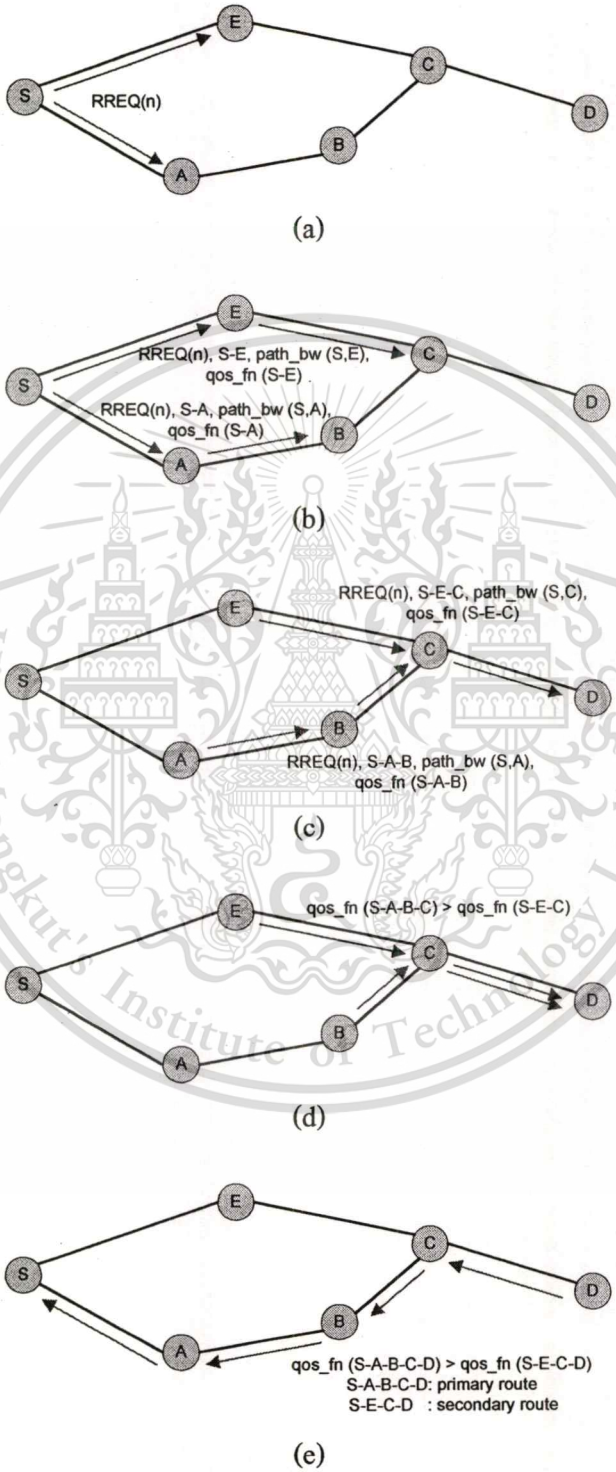


Figure 4.1 Example of QDSR route discovery initiated at source node S for destination node D. A solid line between two nodes indicates that node can hear each other. The arrow out coming from the node means that a node broadcasts messages to its neighbors.

When C receives RREQ from B, C know that it has recently received RREQ packet from E. In order to get opportunity of better path, C compares the QoS function of newly arrived RREQ from B with recent RREQ packet from E. If C figures out that RREQ packet from B contains higher QoS function than recent RREQ packet from E, then C checks whether path bandwidth S-A-B-C are sufficient, and if they are C adds address of [S-A-B-C], path bandwidth S-A-B-C and QoS function of path S-A-B-C to RREQ packet and rebroadcast it (Figure 4.1d). Otherwise, if C founds that QoS function of newly arrived RREQ packet from B is lower than recent RREQ packet from E, C just simply discard this RREQ packet.

At last, after D gets both RREQ from C, which contains two possible routes S-E-C-D and S-A-B-C-D, it performs calculations similar to the ones done on C in order to get available path bandwidth. If path bandwidth S-E-C-D and S-A-B-C-D are sufficient, D compare QoS function of both RREQ packets (Figure 4.1e). RREQ packet that contains the highest QoS function is selected as the primary route and D generates a corresponding RREP packet and sends it back to S. During RREP traverses back to the S, the route reservation process is performed. If the route reservation has failed, the backup route is used to reply the RREP packet and perform reservation process.

Contrarily to DSR protocol, we defined new three algorithms related to Route Discovery process, that is, source node, intermediate node and destination node algorithm as described in the section 4.1.1, 4.1.2 and 4.1.3 respectively.

4.2.1 Source Node Algorithm

Once the source node receives the QoS call request from its application layer, for example the source node wishes to send the multimedia packets, it needs to establish the virtual connection (VC) from the source to the destination node. To do so, it first set the QoS enabled field in RREQ packet to 1 to indicate that source node searches the QoS route to destination node. Then, the source node checks if it has enough link bandwidth (slots) available to any of its neighbors. If there is no enough bandwidth (slots) available, the request for QoS connection is denied and the upper layer is informed. However, if it has sufficient bandwidth, then the source node will check the route in the route cache table whether there is a fresh route to the destination that satisfies the traffic requirement. If a fresh route is found in route cache, the source node uses it. Otherwise, if no valid route is stored in the route cache table, then the source node creates a routing table entry for the requested QoS call and the destination address. Finally, the source node broadcasts the RREQ packet to all of its neighbors. To support QoS,

the RREQ packet in QDSR has some additional information which contains the QoS enabled, number of slots required for reservation, slot state information, link delay and link signal strength information and QoS function. Figure 4.2 shows the process of a source node that wishes to send data packet.

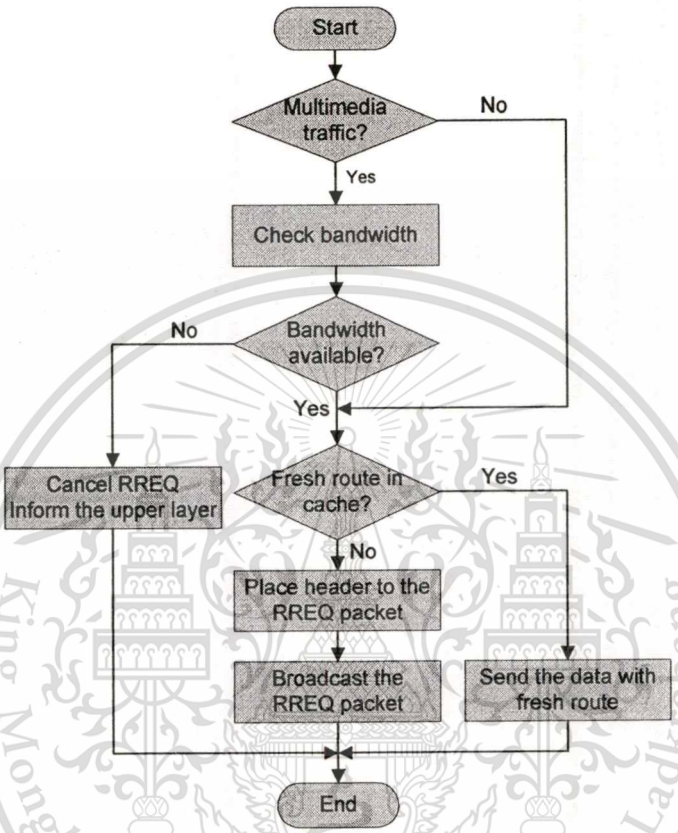


Figure 4.2 Source node algorithm

4.2.2 Intermediate Node Algorithm

Figure 4.3 shows the process on the intermediate node. When an intermediate node receives a RREQ packet it checks whether the pair *<destination_address, sequence_number#>* for the RREQ is seen recently. If so, it checks whether Time to Live (TTL) is zero or not, if TTL counts down to zero, the intermediate node drops the RREQ and do not process it further. TTL can limit the length of delivery path and control the flooding traffic. However, this path will be difficult to be maintained in dynamic network environment.

Based on the state of slot information recorded in RREQ packet, the intermediate node computes the link bandwidth and path bandwidth from the sources to this intermediate node. The link bandwidth and path bandwidth calculation algorithm will be described in section 4.1.2.3. If the calculated path bandwidth to the source is more or equal to the bandwidth

required, the intermediate node records the state of slot information to the *slot_array_list*. Otherwise, if the bandwidth is insufficient, the RREQ will be dropped and the upper layer is informed. Once the bandwidth is sufficient, the intermediate node then attempts to find the optimal route as the secondary importance by calculating the link delay and link signal strength and record the information in RREQ packet. Then it calculates the QoS function based on information of link delay and link signal strength recorded in RREQ packet. The detail of QoS function will be described in section 4.1.2.5.

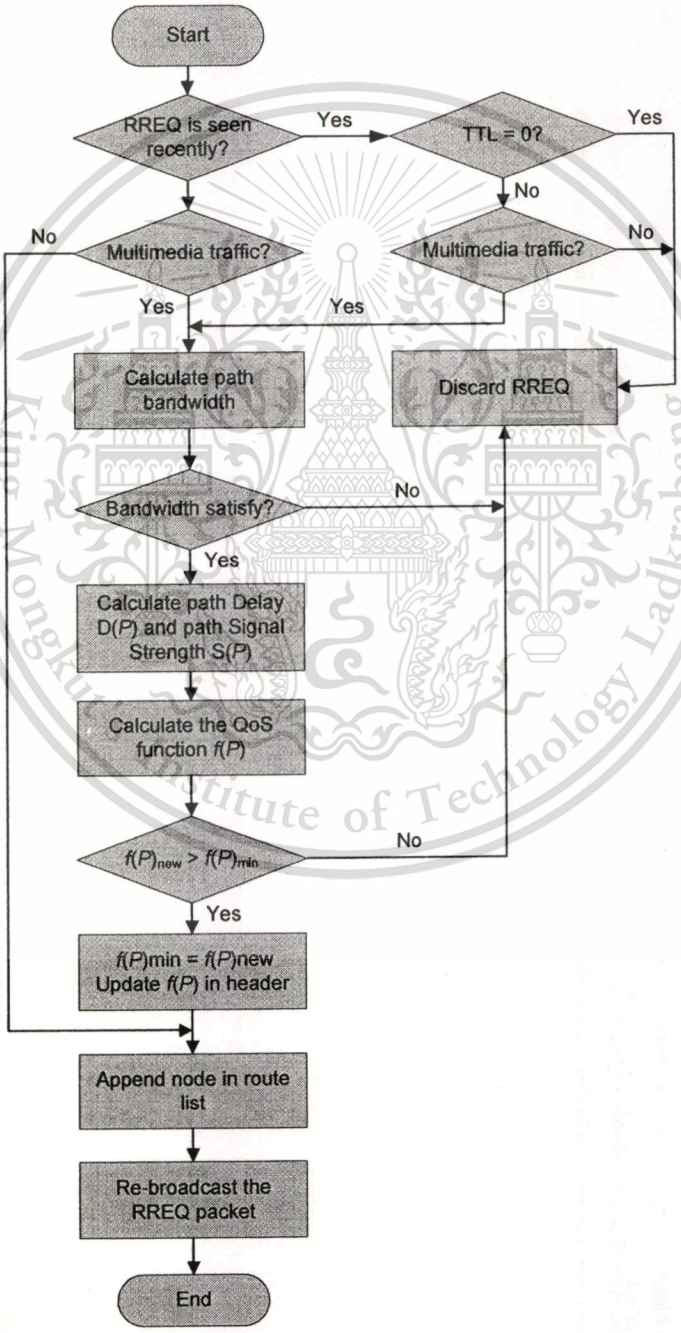


Figure 4.3 Intermediate node algorithm

The intermediate node decrements TTL by one and append the address of this node to the *route_list* in order to track the route which the packet has traversed.

Finally, it performs the selective re-broadcast mechanism in order to reduce the broadcast storm and routing overhead. If additional RREQ packet arrive with same destination and sequence number, the value of newly arrived packet is compare to Max QoS function. If new packet has a higher QoS function value, Max QoS function is changed to this new value and the new RREQ packet is forwarded. Otherwise, the new RREQ packet is dropped.

4.2.2.1 Bandwidth Reservation Mechanism

Multimedia applications such as digital audio and video have much more stringent QoS requirements than traditional datagram applications. A major challenge in multihop, multimedia networks is the ability to account for resources so that bandwidth reservations can be placed on them. In cellular (single hop) networks, all stations learn of each other's requirements either directly or through a control station (e.g., the base station in cellular systems). However, this solution cannot be extended to the MANET environment. To support QoS for real-time applications, we need to know the available bandwidth to the destination and a Virtual Connection (VC) should be accepted only if there is enough available bandwidth. Otherwise, it would disrupt the existing VC's. "Bandwidth" in time slotted network systems is measured in terms of the amount of "free" slots. The main goal of our QoS routing algorithm is to find a feasible path from source to destination such that the available bandwidth on the path is above the minimal requirement through the stable route. To compute the "bandwidth" path, we not only have to know the available bandwidth on each link along the path, but we also have to determine the scheduling of free slots.

4.2.2.2 Bandwidth Calculation

The transmission time scale is organized in frames, each containing a fixed number of time slots. The entire network is synchronized on a frame and slot basis. The frame/slot synchronization mechanism is not described here, but can be implemented with techniques similar to those employed in the wired networks (e.g., "follow the slowest clock" [19]) and properly modified to operate in a wireless mobile environment. Propagation delays will cause imprecision in slot synchronization. However, slot guard times (fractions of a microsecond) will amply absorb propagation delay effects (in the order of microseconds).

Each frame is divided into two phases, namely, the control phase and the data phase, as shown in Figure 4.4. The size of each slot in the control phase is much smaller than the one in the data phase. The control phase is used to perform all the control functions, such as slot and frame synchronization, code assignment, power measurement, slots request, VC setup, etc. The amount of data slots/frame assigned to a VC is determined according to a QoS requirement.

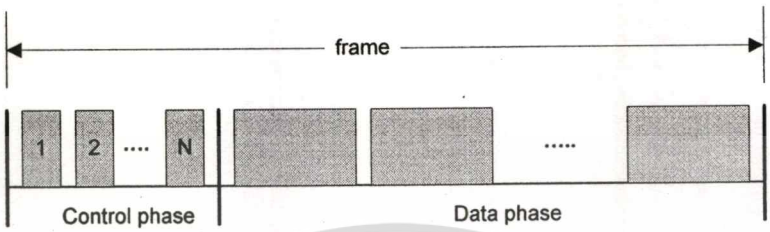


Figure 4.4 Frame structure

As depicted in Figure 4.4, the control phase uses pure TDMA with full power transmission in a common code. That is, each node takes turns to broadcast its information to all of its neighbors in a predefined slot, such that the network control functions can be performed distributively. We assume the information can be heard by all of its adjacent nodes. Ideally, at the end of the control phase, each node has learned the channel reservation status of the data phase. This information will help one to schedule free slots, verify the failure of reserved slots, and drop expired real-time packets.

Because only adjacent nodes can hear the reservation information and the network is multihop, the free slots recorded at every node may be different. We define the set of the common free slots between two adjacent nodes to be the *link bandwidth*. Consider the example shown in Figure 4.5 in which node *C* intends to compute the bandwidth to node *A*. We assume the next hop is node *B*. If node *B* can compute the available bandwidth to node *A*, then node *C* can use this information and the "link bandwidth" to node *B* to compute the bandwidth to node *A*.

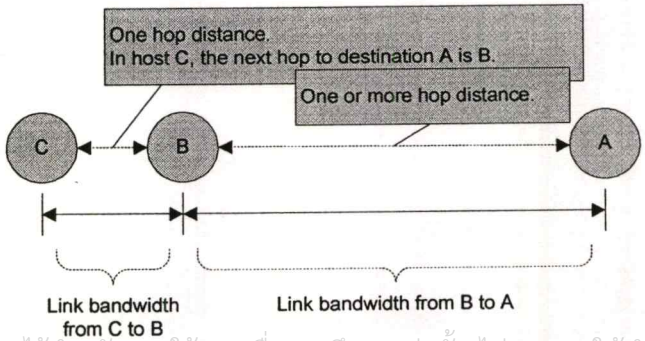


Figure 4.5 End-to-end bandwidth calculation overview

We define the *path bandwidth* (or called *end-to-end bandwidth*) between two nodes, which are not necessary to be adjacent, to be the set of available slots between them. Consider the example in Figure 4.4 and assume that one hop distance is between *A* and *B*. If *C* has free slots {1, 3, 4}, and *B* has free slots {1, 2, 3}, then the *link bandwidth* between *C* and *B* is {1, 3}. This means that we can only exploit slot 1 and slot 3 for packet transmission from *C* to *B*. Thus, if a VC session needs more than two slots in a time frame, then it will be rejected to pass through (C, B). We can observe that *link_BW*(A, B) = *free_slot*(A) $\cap$ *free_slot*(B). The definition of *free_slot*(*X*) is the slots which are not used by any adjacent host of *X* to receive or to send packets from the point of view at node *X*. Next, we can further employ link bandwidth to compute end-to-end bandwidth. End-to-end bandwidth can provide us an indication of whether there exits a QoS route between a given source-destination pair. We will use the following four cases as example to show how to calculate the path bandwidth.

Case 1: Assume the link bandwidth of both (*A*, *B*) and (*B*, *C*) are the same, say {1, 2, 3, 4}, as in Figure 4.6. If node C uses slots 1, 2 to send packets to node *B*, then node *B* can only use slots 3, 4 to forward packets to node *A*. This is because node *B* cannot be in transmitting mode and listening mode simultaneously. So the path bandwidth from node *C* to node *A*, denoted as path_BW(C, A), can be {1, 2}, and its size is two. In this case, four free slots can only contribute two slots for path bandwidth. Namely, [4/2] = 2. Similarly, if there are only three free slots on both links, then the size of path bandwidth is [3/2] = 1.

Case 2: Assume link_BW(A, B) and link_BW(B, C), as in Figure 4.7. Namely, link_BW(A, B) $\subset$ link_BW(B, C). If node *C* uses slot 2, then node *B* cannot use slot 2 any more. So in this case, node *C* should first use slots in link_BW(B, C) - link_BW(A, B) to maximize system utilization. Therefore, if node *C* uses slots 1 and 4, then node *B* can use slots 2, 3. So path_BW(C, A) = {1, 4 }, and its size is two. Similarly, we can use the same way to process the case of link_BW(A, B) $\subset$ link_BW(B, C). In this case, node *B* must use slots in "link_BW(A, B) - link_BW(B, C)" first.

Case 3: If link_BW(A, B) $\cap$ link_BW(B, C) = $\varnothing$, no conflict will occur. Figure 4.8 shows this example. Node C can choose either slot 3 or 4, and node B chooses slot 2. So path_BW(C, A) = {3}, and its size is one.

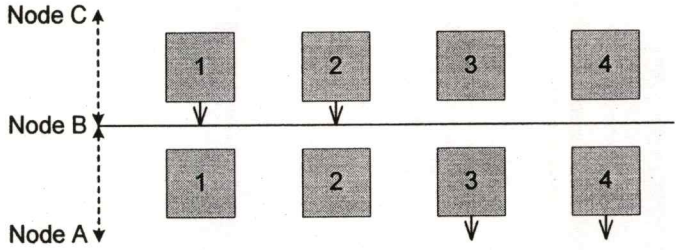


Figure 4.6 Equal case of bandwidth calculation

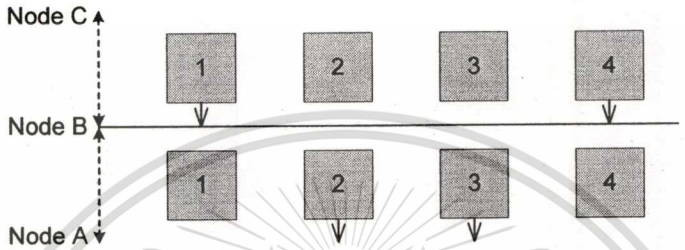


Figure 4.7 Containing case of bandwidth calculation

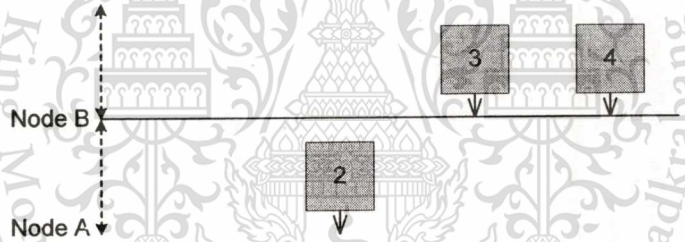


Figure 4.8 Exclusive case of bandwidth calculation

Case 4: This is a general case, as shown in Figure 4.9. We will find any general case can be regarded as a combination of the previous three cases. Follow the slot assignment policy in Case 2. We assign slot 9 to node *C* and slot 1 to node *B* first. Figure 4.10 shows the slots left. Next, we assign slot 10 to node *C* and slot 4 to node *B*. Figure 4.11 shows only slots {5, 6, 7, 8} left (slot 4 cannot be used by node *C* any more since node *B* is in transmitting mode). At present, this is the same situation as in Case 1. So node *C* can be assigned slots 5, 6, and is assigned slots 7, 8, as shown in Figure 4.12. Here we let the path_BW(C, A) = {5, 6, 9, 10}. That is, node *C* can use slots {5, 6, 9, 10} to send packets to node *B*, and then uses {1, 4, 7, 8} to forward packets to node *A*. The size of path bandwidth from node *C* to node *A* is four.

The last case is a general case. Observe that it is, in fact, a combination of Cases 1–3. Our slot assignment policy is to consider the slots not in link_BW(B,C) $\cap$ link_BW(A,B) first, until one of the special cases (i.e., Cases 1–3) occurs. The detail of the bandwidth calculation

algorithm is shown in Figure 4.13. To further generalize Case 4, if the distance between node *A* and node *B* is no longer one hop, as shown in Figure 4.5, the same algorithm can still work. Node *C* can calculate path_BW(C, A) by using link_BW(C, B) and path_BW(B, A) by using the algorithm in Figure 4.13.

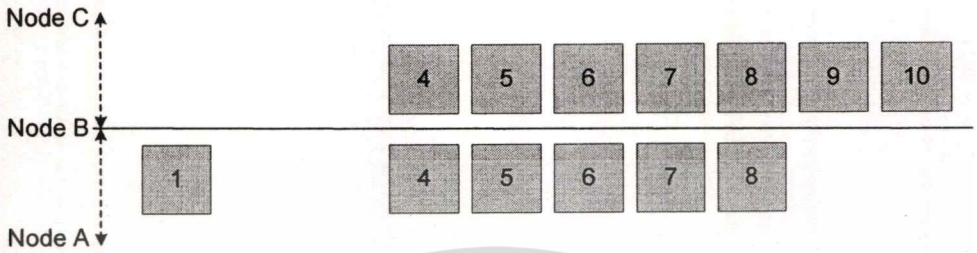


Figure 4.9 General case of bandwidth calculation

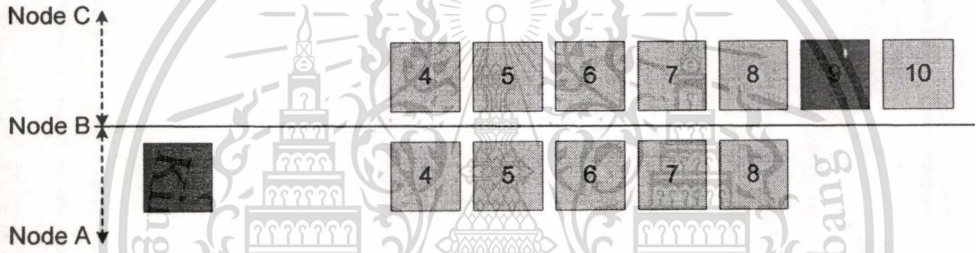


Figure 4.10 Step 1 bandwidth calculation in node *C*

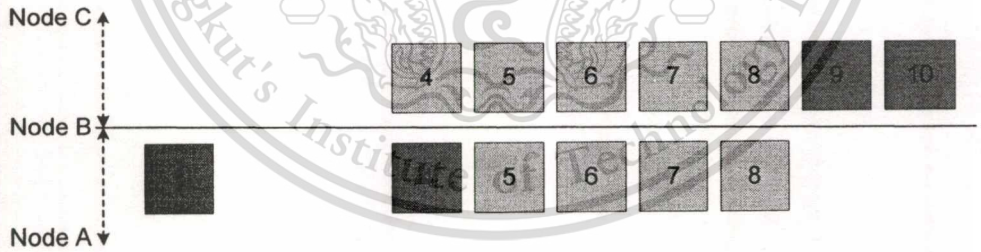


Figure 4.11 Step 2 bandwidth calculation in node *C*

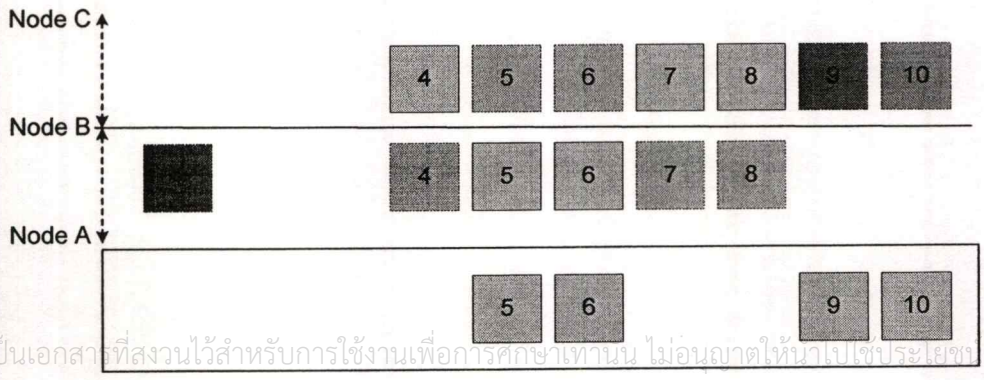


Figure 4.12 Final result of bandwidth in node *C*

```
Path Bandwidth Calculation Algorithm:

int bandwidth_calculation( int i){
if (the numbers record in slot_list or i == 0)
      printf("Error \n")
      return -2;
else if (the numbers record in slot_list or i == 1 ) {
      link_BW = slot_list[1] & slot_list[2] ;
      //slot_list[1] & state of slots of the node receiving route query
      request(RREQ)
      return link_BW;
}
else if (the numbers record in slot_list or i == 2){
      i = i - 1;
      link_bw_i = slot_list()[i] & slot_list()[i+1]
      //( state of slots of node receiving RREQ)
      bandwidth = bandwidth_calculation(i);
      common_BW = bandwidth & link_bw_i;
      common_BW_size = sizeb(common_BW);
      diff2 = bandwidth ^ common_BW;
      difference_BW1 = sizeb(link_bw_i ^common_BW); // the symbol ^ means
      XOR difference_BW2 = sizeb(bandwidth ^ common_BW);
      if ( difference_BW1 <= difference_BW2) {
            bandwidth_size = difference_BW1;
            remain_BW_size = difference_BW2-difference_BW1;
      }
      else {
           bandwidth_size = difference_BW2;
           remain_BW_size = difference_BW1 - difference_BW2;
            }
      if ( ( remain_BW_size > 0 ) || ( common_BW_size > 0)) {
           if ( common_BW_size <= remain_BW_size)
                bandwidth_size = bandwidth_size + common_BW_size;
      else {
           bandwidth_size = bandwidth_size + remain_BW_size;
           common_BW_size = ( common_BW_size - remain_BW_size)/2;
           if ( common_BW_size > 0)
                bandwidth_size= bandwidth_size + common_BW_size; }
           }
           if (bandwidth_size >= requested bandwidth)
                return link_bw_i;
           else
                return -2; // no bandwidth available
      }
}
```

Figure 4.13 Path bandwidth calculation algorithm

4.2.2.3 Slot Information and Assignment

We have already described how to calculate path bandwidth from source node to destination node during route request mechanism of QDSR. In this section, we will discuss how to do the slot assignment. Once a call is accepted, the system needs to allocate slots to the VC hop-by-hop along the path to the destination. Each node must run a slot assignment algorithm. The algorithm in Figure 4.13 only calculates the size of link bandwidth and path bandwidth by using the bandwidth slot information in RREQ packet. However, it does not tell us how to

assign the available slots efficiently during the call setup in each host. In this section, we will explain how to do the slot assignment.

Once a RREQ packet arrives at destination, the slot assignment procedure must be done in order to assign slots to every node that the route request packet has traversed previously. As result, the information of slot assignment is then copied in the header of route reply packet that is going to be sent back to the source. Once any intermediate node receives route reply packet, it must set its slots to be occupied according to the slot information in route reply packet header. We use Figure 4.9 as an example to describe how the slot assignment algorithm works. For a given path, the source node, immediate nodes, and the destination node will do different task. We will start to describe the process of slot assignment at the destination node because it is the only node that does the process of slot assignment algorithm which information of slot assignment is placed in the packet header of RREP packet. From this packet, any intermediate or the source node can set their slots' status according to the slot assignment information in the RREQ packet. Assume the new VC session from node *C* (source node) to node *A* (destination node) needs four data slots in each time frame (i.e., the QoS requirement).

Destination Node: When the destination receives the RREQ packet, then it calculates the end-to-end bandwidth from source to this node using the state of slot information in RREQ packet. If the bandwidth from source to destination does not satisfy the QoS requirement, the destination node drops the RREQ packet. Otherwise, it processes the slot assignment procedure to every node that RREQ has traversed. The information of this process will be used in RREP packet as information for every intermediate node to set their status to be occupied by this connection. According to link_BW(C, B) and link_BW(B,A), we can compute path bandwidth as illustrated in Figure 4.12. Slots {5, 6, 9, 10} and Slots {1, 4, 7, 8} are recorded in an *slot_array_list* of route reply packet. Finally the destination node does some other tasks such reserving slots {1, 4, 7, 8} for this connection and sending out route reply packet (RREP) back to the source by using reversing route in RREQ. This algorithm is given in Figure 4.14.

Intermediate Nodes: When node B receives RREP packet containing bandwidth information from node A, then it uses this information to check its slots' status. First it checks whether the incoming slots {5, 6, 9, 10} are free. If so, it can receive data packets from node *C*. In addition, node *B* must check if there are free slots such as {1, 4, 7, 8}, outgoing slots, which can be used for forwarding packets to next hop. If any one slot in {5, 6, 9, 10} is busy or if there are no enough free slots {1, 4, 7, 8} to forward the packets, this bandwidth reservation will fail. So node B must reject the reservation request as result of lacking of available either

incoming free slots or outgoing free slots. Because node *C* has already reserved slots {5, 6, 9, 10}, node *B* needs to send a route error (RERR) packet with type of unsuccessful reserved message (*URSV*) to node *C* to ask for freeing the slots {1, 4, 7, 8}. If all slots are available, then this intermediate node reserves all these slots for this connection and the status of these slots are set to be occupied. This algorithm is given in Figure 4.15 and these checking operations have to be done because the topological change may affect the free slots.

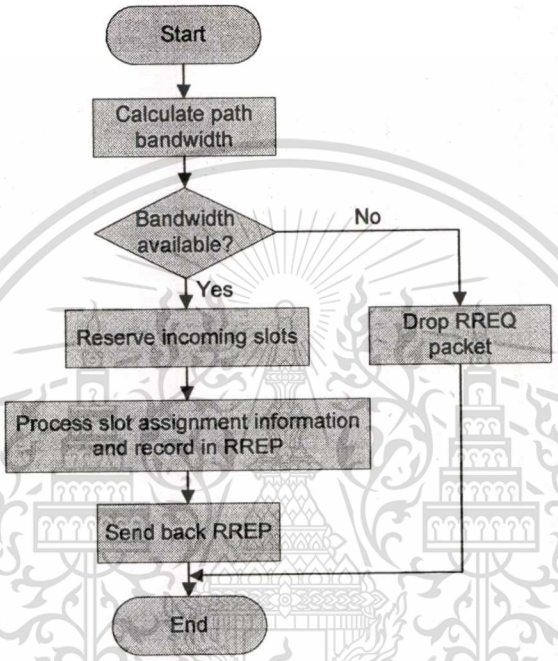


Figure 4.14 Slot assignment procedure at destination node

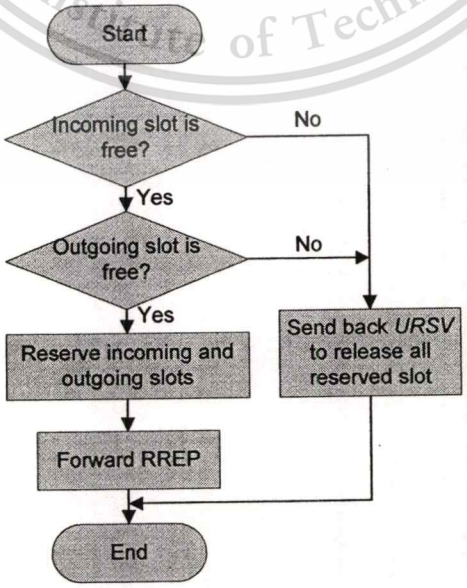


Figure 4.15 Slot assignment procedure at intermediate node

Source Node: Once route reply packet arrives at the source node, it needs to reserve slots to be used for transmitting. It checks only outgoing slots, in this example, the source node checks only {5, 6, 9, 10}. If these slots are available, the source node reserves these slots for outgoing transmission. After reserving these slots, it records this route in its Route Cache for further use and starts transmitting data packets. If either one of {5, 6, 9, 10} is not free then the unreserved packet (*URSV*), a type of RERR message, needs to be sent back to the node that sent RREP packet in order to release its slots reservation for this node. The algorithm of this process is given in Figure 4.16.

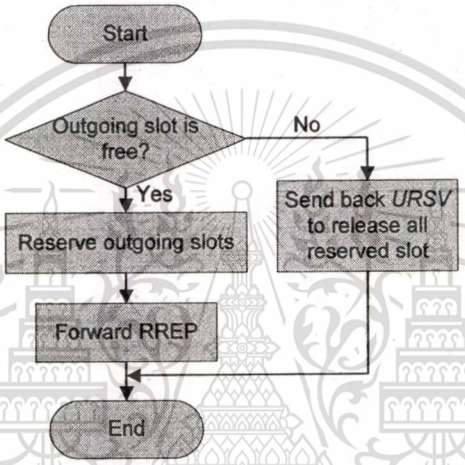


Figure 4.16 Slot assignment procedure at source node

In the previous example, we discuss how to reserve the bandwidth. When every node receives a reservation packet, it must check if the slots that the sender will use for transmitting packets are free and find if there are free slots that can be reserved for forwarding packets. If both are satisfied, the reservation can be passed to the next hop. Finally, the destination sends a *REPLY* backward to the source to acknowledge having set up the connection. The reservation algorithm is shown in Figure 4.14 – 4.16. In the algorithm, when the call setup fails, it is necessary to free all reserved slots along the path backward to the source. This operation is very important since the bandwidth in the wireless network is quite limited. A topological change may make it impossible to send either a *REPLY* or a *URSV* back along the path that has been established so far. Thus, if an intermediate node does not receive a *REPLY* by a predefined time, the reserved slots will be freed automatically.

4.2.2.4 QoS Function for Remaining Metrics

Consider the example of simple network topology as illustrated in Figure 4.17. The point A, B, C, ..., H represent the mobile nodes in MANETs. The weight of each edge is expressed with a two-tuples, which denote the delay and the signal strength of the relevant link. Suppose we want to find a route from Source Node A to a Destination Node F. For the pure routing algorithm such as Dijkstra, the route <A-G-F> will be the result. However, QoS route problem is entirely different. The shortest path route <A-G-F> may not be adequate for satisfying the requirement of QoS routing. The problem of QoS routing in MANETs can be described as following formally.

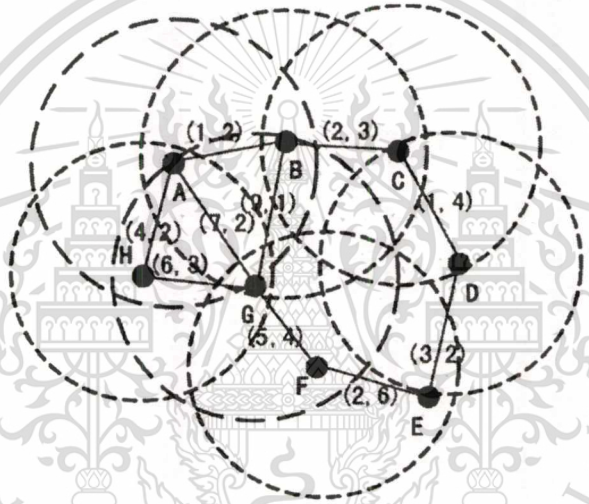


Figure 4.17 Example of simple network topology

A MANETs can be modeled as a undirected graph with weight G, $G = \{V, E\}$, V is the set of the mobile nodes, and the E the set of the bidirectional wireless link, for a link $e \in E$, have e = (v_i, v_j), and node $v_i \in V$, $v_j \in V$, $i \neq j$, then v_i and v_j are the neighbor nodes. G and V are both dynamic set. For $s \in V$ is the source, $p \in \{V - \{s\}\}$ is the destination, if $e \in E$, define the metric function on each link:

Delay function: $D(e) : E \longrightarrow \mathcal{R}^+$

Signal strength function: $S(e) : E \longrightarrow \mathcal{R}^+$

Then, the QoS parameters of path p can be represented as:

$$\begin{cases} D(p) = \sum_{i=1}^{k-1} D(v_i, v_{i+1}) \\ S(p) = \min(S(v_i, v_{i+1}), (v_{i+1}, v_{i+2}), \ldots, (v_{i+1}, v_{i+2}) \end{cases}, \quad (4.1)$$

For optimal route selection which is considered as the secondary importance, the QoS function for remaining metrics on path P is given by a heuristic formula (4.2):

$$f(P) = \left(\frac{d - D(P)}{d}\right) + \left(\frac{S(P)}{R_t}\right) \qquad (4.2)$$

where $D(P)$ is the delay at given path, d is the maximum end-to-end delay that can be tolerated, $S(P)$ is the minimum signal strength along the path, and R_t is the received signal threshold. The higher value of QoS function implies the higher probability of lower delay and more stable link route.

4.2.2.4.1 Delay Metric

The delay on each link $d(e)$ is a measure of total delay that a packet would experience when traversing the link e. The end-to-end delay of a path is the summation of the *node-delay* at each node and the *link-delay* at each link on the path. *node-delay* (i, j) includes the protocol-processing time and the queuing delay of a normal data packet at node i for link (i, j). *link-delay* (i, j) is the propagation delay on link (i, j). The delay metric is defined as

$$delay(i,j) = node\_delay(i,j) + link\_delay(i,j) \qquad (4.3)$$

Let p be a probe and suppose the path that p has traversed is $P = i \rightarrow j \rightarrow \ldots \rightarrow r \rightarrow s$. Therefore, the end-to-end delay from source to destination along the path P can be calculated as the following formula:

$$Delay\,(p) = delay(i,j) + \ldots + delay(r,s) \qquad (4.4)$$

In our scheme, each intermediate node and destination node calculate the delay from their previous one hop. This information is then added in RREQ header and used to calculate the QoS function further. The objective of our proposed scheme is to select the path from source to destination that has the minimum end-to-end delay.

4.2.2.4.2 Signal Strength Metric

A radio channel between a transmitter unit u and a receiver unit v is established if and only if the power of the radio signal received by node v is above a certain threshold, called the *sensitivity threshold*. Formally, there exists a direct wireless link between u and v if $P_r \geq \beta$, where P_r is the power of signal received by v, and β denotes the sensitivity threshold. The

exact value of β depends on the features of the wireless transceiver and on the communication data rate: for a given radio, the higher the data rate, the higher the value of β, implying a stronger requirement on the received power. In order to simplify notation, in the following we assume that β has the conventional value of 1.

The received power P_r depends on the power P_t used by u to transmit the radio signal, and on the *path loss*, which models the radio signal degradation with distance. Denoting with $PL(u, v)$ the path loss between units u and v, we can write

$$P_r = \frac{P_t}{PL(u,v)}. \qquad (4.5)$$

Thus, the occurrence of a radio channel between any two network nodes can be predicted if the path loss model is known.

In our network, we use the two ray ground as the propagation model. It is because it is seldom the case that the single direct path between the transmitter and the receiver is the only physical means of propagation of the radio signal. Therefore, the free space propagation model, which model that used to predict radio signal propagation when the path between the transmitter and the receiver is clear and unobstructed (line-of-sight, or LOS, path), is often inaccurate. In addition, the two ray ground model can improve accuracy, in which such model considers two propagation paths: the directed path and a ground reflected propagation between the transmitter and the receiver as shown in Figure 4.18.

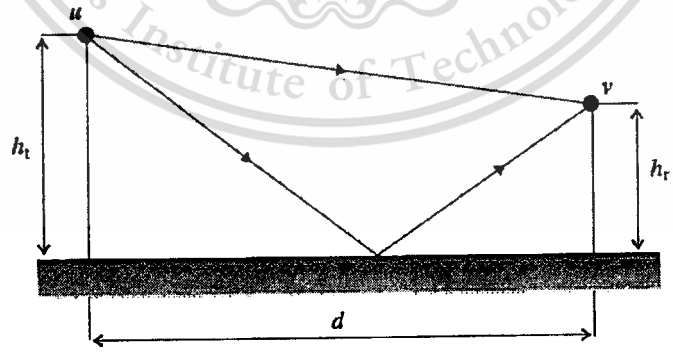


Figure 4.18 The two-ray propagation model: the radio signal sent by node u reaches node v through the direct path, and through a ground reflected path.

In the two-ray ground propagation model, the received power at distance d is given by the following formula:

$$P_r(d) = P_t G_t G_r \frac{h_t^2 h_r^2}{d^4}, \quad (4.6)$$

where h_t is the transmitter antenna height and h_r is the receiver antenna height. If the distance between the sender and the receiver is relatively large ($d >> \sqrt{h_t h_r}$), and abstracting the features of the radio transceivers, we can write the following simplified formula:

$$P_r(d) = C_t \cdot \frac{P_t}{d^4}, \quad (4.7)$$

where C_t (t stands for two-ray ground) is a constant that depends on the characteristics of the radio transceivers.

In our proposed scheme, every intermediate node and destination node that receives the RREQ packet measure the power signal received from its previous one hop. The power signal received often called as signal strength. If the distance between receiver and transmitter is close enough, the receiver will receive the higher signal strength. Otherwise, if the receiver is far away from the transmitter, the lower signal strength will be received by receiver. This information is then recorded in the RREQ header. The signal strength on each link on a path is then combined to calculate the path signal strength with the following formula:

$$S(p) = \min[s(i,j), s(j,k), ..., (s(l,m)] \quad (4.8)$$

Where $s(i, j)$ is a metric for link (i, j) in a path $p(i, j, k, ..., l, m)$. We use this approximation since the weakest link signal strength on a path will have the higher probability of broken route. Therefore, our proposed scheme attempts to find the path that has stronger link.

4.2.3 Destination Node Algorithm

When the first RREQ packet with QoS enabled reaches the destination node, it does the path bandwidth calculation using the same algorithm as the intermediate node. Afterward, if there is path bandwidth available, the node waits for a threshold interval (T_r). During that time, the destination node examines the value of QoS function of every arrived RREQ packet and the route is listed into a Route Information Table. When the timer interval (T_r) expired, the destination node selects the RREQ packet that has the highest rank of QoS function in the Route Information Table, generate a corresponding route reply (RREP) packet carrying the route information and unicast it back to the source node along the coming route and does the

reservation process. The route records in Route Information Table that has the lower rank of QoS function is used for backup route in case of reservation process to source node using the primary route is fail. When another RREQ packet with the same destination and sequence number arrives after the time interval, it will not be considered and do not process it further. Figure 4.19 shows the process of destination node when receive RREQ packets.

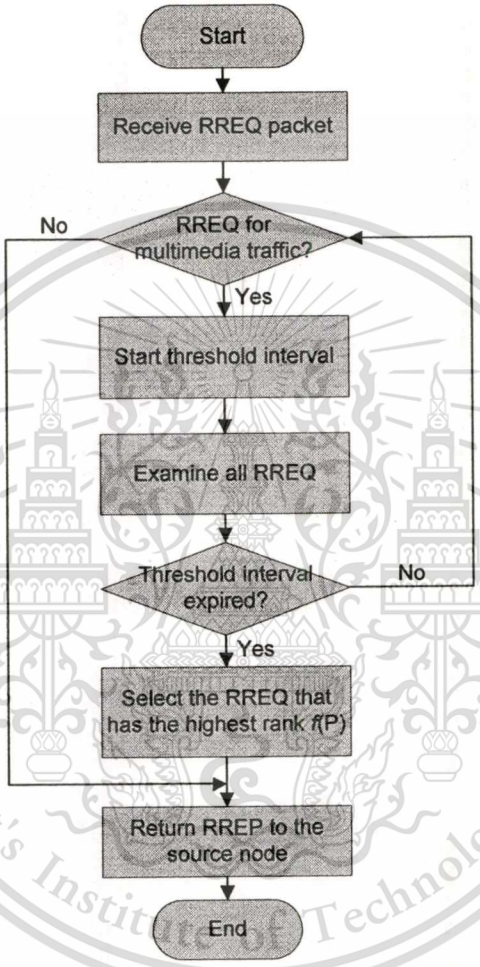


Figure 4.19 Destination node algorithm

4.3 Route Reservation

On receiving the RREQ packet, the destination node reserves resources using a TDMA-based bandwidth reservation model by algorithms explained in section 4.1.2.3. As a RREQ travels from the source to the destination, it automatically sets up the reverse path from the destination back to the source. From the RREQ packet, we can obtain the state of data slots in according to the information recorded within this packet. The destination node can set up a QoS route and reserves resources (slots) hop-by-hop backward to the source node. If the reservation

operation fails or the resources are not available, a route error (RRER) message is sent to the destination along the coming route of RREP. When intermediate node receives RERR message, this node will release slots reserved for the traffic. Upon receives RERR message, destination node will check the Route Information Table to select the secondary route recorded and resend RREP. When RREP reaches the source, a QoS guarantee route is established and the source can start to send data.

4.3.1 Virtual Connection (VC) Establishment, Maintenance and Release

Once the destination node receives RREQ message and figures out that there is bandwidth available to the source node, it generates a corresponding RREP and starts a route reservation procedure. To set up the reverse path, a destination node records the address of the intermediate node from which it received the copy of the RREQ. From the RREQ packet, we can obtain the state of data slots in *slot_array_list* field. Using the source routing algorithm, the fields *<route_list, slot_array_list>* are copied from RREQ to RREP which *route_list* field contains the routes which RREQ packet has traversed from source to destination, while *slot_array_list* field contains the state of data slots which a RREQ packet has traversed. The destination node then reserved its incoming slots and send RREP back to the source node as shown in Figure 4.15.

When an intermediate node receives the RREP message, it starts a reservation procedure by reserves incoming and outgoing slots as shown in Figure 4.14. If the slot reservation was made successfully, intermediate node sets the status of these slots to be occupied. If the slot reservation has failed, which can happen because other neighboring nodes may reserve the same slots simultaneously, then resource release is made, and an unreserved message, *URSV,* is sent along the path back to the destination. When source node receives the RREP message, it does reservation for outgoing slots as shown in Figure 4.13. After reserving these slots, the end-to-end bandwidth reservation is successful and the application layer is notified that the VC is established. Then, the application layer in turn starts sending data packets.

After the application terminates the connection, the VC is released by generating a *URSV* message that is sent along the path to the destination. When node receives the *URSV* message it releases resources as described above.

Established VCs can be broken when nodes go beyond each others transmission range or when new nodes enter node transmission range with the slots already reserved for the

transmission or receiving. After QDSR discovers that VC is broken, it releases resources and sends *URSV* messages to the source and the destination nodes. When the source node receives *URSV*, it checks whether it has more messages to transfer. If it does, then it goes through the path discovery session.

4.4 Route Maintenance

If the destination node does not receive the data packets in reserved time slot, it considers that the link is broken and invokes the route maintenance procedure. The destination node then uses its dedicated control time slot to broadcast a route error (RERR) message with finite TTL. Any node in the QoS route will forward the RERR message to the source node. On receiving the RERR message, the source node can re-establish a new route for subsequent packets. We define three type of RRER message for the route maintenance purpose as shown in Table 4.3.

Table 4.3 RERR message types and their function

RERR Message Type	Function
UNSUCESSFUL_RESERVATION (*URSV*)	Nack for unsuccessful reservation
ROUTE_BROKEN	Nack for route broken
NO_ROUTE	Nack for finding no route

During the active period of connection, a topological change may destroy the VC. It could be resulted from nodes go beyond each others transmission range or when new nodes enter node transmission range with the slots already reserved for the transmission or receiving. The connection control must re-route or re-establish the VC over a new path. When a route is broken, the breakpoints send a special RERR packet with type *ROUTE_BROKEN* to the source and the destination. That is, once the next hop becomes unreachable, the breakpoint which is near the source sends an unsolicited RERR to the source, and the other breakpoint does to the destination. Each node along the path forwards this *ROUTE_BROKEN* to its active neighbors and so on. Furthermore, they release all reserved slots for this connection by sending another type of RERR such as *URSV* and then drop all data packets of this connection which are still waiting for sending in the queue. Upon receiving the *ROUTE_BROKEN,* the source re-starts another discovery process to re-establish a VC over a new path. This procedure is repeated until

either the completion of data delivery or timeout. If timeout occurs, the source will reject the call request.

If a link on the VC is broken before the completion of a session, the last data packet may be still on the way to the destination. This packet, thus, can not reach the destination, and is suspended within an intermediate node. In this situation, some resources are still occupied by this connection and can not be used by the others. In order to solve this problem, we use the timeout scheme for each reserved slot. If a reserved slot is not used to deliver data packets for a couple of data frames, then timeout occurs and this slot is freed automatically. Such free slots will be fully utilized by the other new sessions.

USRV and *NO_ROUTE* messages are generally used in route reservation process. When the RREP travels back to the source, the reservation operation may not be successful. This can be resulted from the fact that the slots which we want to reserve are occupied by another VC or the path breaks. If this is the case, we must give up the route. The interrupted node sends a *USRV* message back to the destination, and the destination re-starts the reservation process again along the next feasible path recorded on the Route Information Table. If there is no VC can be setup along all feasible QoS routes, the destination broadcasts another RERR message (i.e., *NO_ROUTE*) to notify the source. Upon receiving *NO_ROUTE*, the source node can re- start the discovery process if it still requires a route to the destination. If there is no any response back to the source node before the timeout occurs, the source node will also re-perform the route discover operation.

Once a VC is established, the source can begin sending datagrams in the data phase. At the end of the session, all reserved slots must be released. These free slots will be contended by all new connections. However, if the last packet is lost, we will not know when the reserved slots should be released. This problem is solved by implementing the timeout scheme for each reserved slot as explained before.

Chapter 5

Performance Evaluation

In this chapter, we describe the methodology for evaluating our proposed scheme in an ad hoc environment, concentrating on the simulation system and the metrics used for analyzing and comparing our proposed scheme with the other protocols later in this chapter.

5.1 Simulation Software

The Network Simulator ns-2 [16] is discrete event simulation software for network simulations. Ns-2 began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. The latest version, ns-allinone-2.30, supports for simulation of routing protocols for ad-hoc wireless networks such as DSDV, AODV, TORA and DSR. Ns-2 is written in C++ programming language and Object Tool Common Language (OTCL). Although ns-2 can be built on various platforms, we chose a Linux platform [17] for this thesis, as Linux offers a number of programming development tools that can be used along with the simulation process.

For this thesis, we developed our proposed algorithm by modifying the algorithm inside DSR routing protocol module. Then, we write the simulation script in OTCL to define the network (number of nodes), the traffic in the network (source, destination, type of traffic) and which protocol we will use. The result of simulations is an output trace file. Finally, we analyze the results by using the *awk* command and Perl scripts. Ns-2 also offers a visual representation of the simulated network by tracing nodes' movements and events and writing them in a network animator (NAM) file.

However, ns-2 is open source software that has been built by a number of different developers, suffers from a number of known and unknown bugs.

5.2 Simulation Environment

Our simulation modeled an ad hoc network where 8 mobile nodes placed randomly within 670 meter × 670 meter area. Radio transmission range for each node was 250 meter. That is, two nodes can hear each other if their distance is within the transmission range. For the mobility model, we used random waypoint model. With such mobility model, each node

randomly selects a position inside the simulation site, and moves toward that a position with a speed distributed uniformly between some minimum speed and some maximum speed. Once the node reaches that position, it becomes stationary for a short predefined pause time. After that pause time, the node selects another position with random speed and proceed there are described previously, repeating this behavior for the duration of simulation. We varied the lower bound and upper bound speed to simulate the different average mobility speed. The average speed was from 0 to 10m/s.

A traffic generator was developed to represent the multimedia traffic which simulates constant bit rate sources and during 500 seconds of simulation, 3 source nodes send the constant bit rate (CBR) packet at rate of 25 packets/second. The link bandwidth is assumed as 2Mbps, and a resource scheduling and reservation protocol is assumed to be used to schedule and reserve the required resource at each mobile node along the route.

In the experiments, we will pay more attention to the effect of mobility to the system performance. The detail of simulation parameters and constant values for *f(P)* in Equation 4.2 are listed in Table 5.1 and 5.2, respectively.

Table 5.1 Simulation parameters

Parameter	Value
Simulation area	670 x 670m
Number of nodes	8 nodes
Mobility	0 ~ 10 m/s
Pause time	2 sec
Movement model	Random waypoint
Transmission range	250 m
Source / Packet size	CBR / 512 bytes
Sending rate	25 packets/sec
Number of CBR sources	3 sources

Table 5.2 Constant values

Constant	Value
Max delay (*d*)	1 sec
R_t	5.844×10^{-9} w
Threshold time interval (*Tr*)	2* 1st_rreq_time

5.3 Performance Metrics

In evaluating QDSR protocol, we used several sets of metrics to characterize the basic performance of the protocols. The following metrics are used to evaluating QDSR protocol and the other protocols studied in this thesis.

5.3.1 Packet Delivery Ratio

The packet delivery ratio is defined as the fraction of all the received data packets at the destinations over the number of data packets sent by the sources and can be formulated as shown in Equation 5.1. This is an important metric in networks as it described the loss rate that will be seen by transport protocols, which in turn the effect the maximum throughput that the network can support.

$$Packet\_Delivery\_Ratio = \frac{\sum Num\_data\_pkts\_received}{\sum Num\_data\_pkts\_sent} \quad (5.1)$$

5.3.2 Packet Loss Rate

The packet loss rate is defined as the ratio of packet lost or dropped during transmission. If the application uses TCP as the layer 2 protocol, high packet loss at the intermediate nodes will result in retransmissions by the sources that will result in network congestion. To calculate the packet loss rate, we can use the Equation 5.2 and 5.3, respectively.

$$\sum Num\_data\_pkts\_lost = \sum Num\_data\_pkts\_sent - \sum Num\_data\_pkts\_recvd \quad (5.2)$$

$$Packet\_Loss\_Rate = \frac{\sum Num\_data\_pkts\_lost}{\sum Num\_data\_pkts\_sent} \quad (5.3)$$

5.3.3 Throughput

Throughput is defined as the amount of data per time unit that is delivered through the network. The throughput is usually measured in bits per second (bits/s or bps). This metric can be measured by calculate the total bits of data received by destination over the simulation time as shown in Equation 5.4. This is also an important metric in networks as it described how fast the data can be sent over the network in a period of time (second).

$$Throughput\ (bps) = \frac{\sum Data\_pkts\_recvd\ (bit)}{\sum Time\_of\_simulation\,(\text{sec})} \quad (5.4)$$

5.3.4 Average End-to-End Delay

End-to-end delay includes all possible delays in the network caused by route discovery latency, retransmission by the intermediate nodes, processing delay, queuing delay, and propagation delay. To measure the average end-to-end delay, we sum every delay for each successful data packet delivery and divide that sum by the number of successfully received data packets as described in Equation 5.5. This metric is important in delay sensitive applications such as video and voice transmission.

$$Average\_End\_to\_End\_Delay = \frac{\sum (Time\_received - Time\_sent)}{\sum Num\_data\_pkts\_received} \quad (5.5)$$

5.3.5 Routing Overhead

There are two kinds of routing overhead that can be measured during simulation. First is routing overhead measured in term of number of routing packets sent per second. This metric can be measured by using Equation 5.6.

$$Routing\_Overhead\ (packets/\sec) = \frac{\sum Num\_routing\_pkts\_sent(packets)}{\sum Time\_of\_simulation(\sec)} \quad (5.6)$$

The second option to measured routing overhead is in term of routing bytes sent per second. This metric can be measured by calculate the total byte of routing packet sent by all nodes over the simulation time as shown in Equation 5.7.

$$Routing\_Overhead\ (bytes/\sec) = \frac{\sum Routing\_pkts\_sent(byte)}{\sum Time\_of\_simulation(\sec)} \quad (5.7)$$

Protocols that generate large amount routing overhead can increase the probability of packet collision and may delay data packets in network interface transmission queues.

5.3.6 Normalized Routing Load

The normalized routing load is defined as the fraction of all routing control packets sent by all nodes over the total number of data packets received at the destination nodes. This metric can be measured by using Equation 5.8.

$$Normalized\_Routing\_Load = \frac{\sum Num\_routing\_pkts\_sent}{\sum Num\_data\_pkts\_received} \quad (5.8)$$

Routing load is an important metrics in the network, as it measures the efficiency of routing protocol and the degree which it will function in congested environments. The bigger this fraction, the less efficient the protocol is.

5.3.7 Power Factor

There is no any protocol that superior in all kind of metric performance. One protocol may be the best in one metric performance but poor in another metric. However, we still can compare the performance by using combination of several metrics. In this simulation, we will compare the power factor of each protocol using normalized throughput and normalized RREQ delay metric.

The normalized throughput can be considered in term of medium utilization. To measure the normalized throughput, we have to know the theoretical throughput of our system. As the assumption of fixed size for all data packets, the theoretical throughput for one source can be calculated by using Equation 5.9.

$$X(bits) = Packet\_rate(pkts/\sec) \times Size\_of\_data\_packet(byte) \times 8(bits/byte) \quad (5.9)$$

For the system that has many sources, the total theoretical throughput can be measured as the sum of throughput of each source as shown in Equation 5.10.

$$X_{in}(bits) = X_1 + X_2 + ... + X_n \quad (5.10)$$

Then, the normalized throughput (T) can be formulated as shown in Equation 5.11.

$$Normalized\_Throughput(T) = \frac{Theoritical\_throughput(X_{in})}{Measured\_throughput(X_{out})}, \quad (5.11)$$

where $0 \leq T \leq 1$ and the measured throughput (X_{out}) is the total throughput can be achieved during simulation.

In the same way, we can calculate the normalized RREQ delay based on the method explained above. Assume that the system has more than one source, for example d_1, d_2, ... d_n, which each source have different delay in searching the route. The average delay of the system can be measured by using Equation 5.12.

$$d' = \frac{d_1 + d_2 + ... + d_n}{n} \tag{5.12}$$

Then, the normalized RREQ delay (D) can be formulated as shown in Equation 5.13.

$$Normalized\_RREQ\_Delay(D) = \frac{d'}{Max(d_1, d_2, ...d_n)}, \tag{5.13}$$

where $0 \leq D \leq 1$ and the value of D will be maximum if $d_1 = d_2 = d_n$.

Finally, the power factor can be measured by using Equation 5.14.

$$Compound\_Factor(T/D) = \frac{Normalized\_throughput(T)}{Normalized\_RREQ\_delay(D)} \tag{5.14}$$

Power factor is also a performance metric that should be considered in evaluating the performance of the system, as it measures the how good the performance of our system over the combination of some metrics.

5.4 Simulation Results

To evaluate the performance of our proposed protocol, we conducted the experiment by means of simulation using *ns*-2 network simulator. The results of simulation are then compared with the other protocol on an identical scenario network model. For the comparison, we consider with the original DSR since QDSR is an extension of DSR protocol, in which QoS features are embedded in the routes selection and maintenance procedures. We also compare our proposed protocol with another on-demand QoS routing protocol, such as QoS-AODV, to be fairly to compare with the protocol that involves the QoS features. QoS-AODV is the extension of the Ad hoc On-demand Distance Vector (AODV) routing mechanism which tries

to find a suitable path and combined with bandwidth reservation mechanism for providing end-to-end QoS in an ad hoc environment [18].

5.4.1 Packet Delivery Ratio

Figure 5.1 and 5.2 highlight the performance of QDSR in term of delivery data packet comparing with DSR and QoS-AODV. All of the protocols deliver a greater percentage of the originated application data packets when there is little node mobility (i.e., at lower speed), converging to 100% delivery when there is no node motion as shown in Figure 5.1. However, QDSR performs particularly well at the highest rates of mobility, where it delivers about 80.07% of its packets at average mobility speed 10m/s. While DSR and QoS-AODV deliver about 76.94% and 71.78%, respectively.

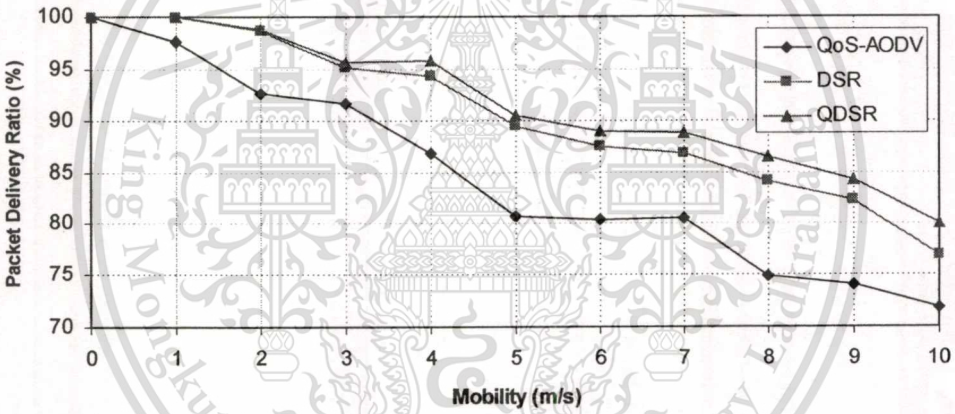


Figure 5.1 Packet delivery ratio versus mobility

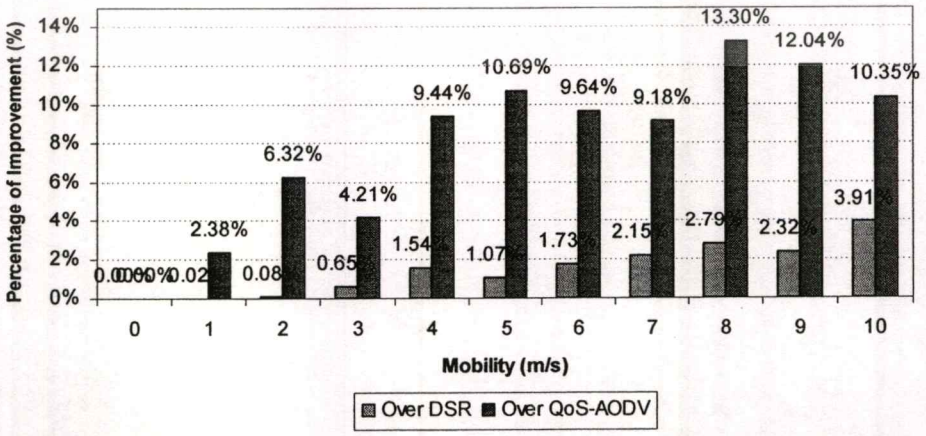


Figure 5.2 Improvement of packet delivery ratio in QDSR

Moreover, figure 5.2 shows the percentage of improvement of data packets delivery achieved by QDSR over DSR and QoS-AODV. At lower rates of mobility (1 m/s), QDSR improves over DSR and QoS-AODV by 0.02% and 2.38%, respectively. However, at highest rates of mobility (highest speed), QDSR improves by more than 3% and 10%. This proves that the routes chosen by QDSR are more stable and the traffic load is more balanced at the higher rates of mobility.

5.4.2 Packet Loss Rate

The packet loss occurs when the network becomes congested or the route is broken frequently due the mobility. From figure 5.3, we observe that QoS-AODV has the worst performance among the other protocols, which about 21.09 packets/sec are lost during transmission at the highest rate of mobility. The reason is that the network is congested at some points by the periodic transmission of HELLO messages and AODV routing packets. Therefore, the existence of valid routes between a source/destination pair in a node's routing table, does not necessarily guarantee a better packet delivery ratio with lower packet loss rate.

QDSR has almost the same performance with DSR to present a lower packet loss rate at the lower rates of mobility (lower speed). However, QDSR performs better at the highest rates of mobility, which its loss rate about 14.95 packets/sec, while DSR suffers of loss rate about 17.28 packets/sec. Since QDSR considers with signal strength metric on selecting the optimal route, the chosen hop along the path is generally shorter and therefore, it can reduced the probability of broken route, packet loss and route maintenance process compared with DSR and QoS-AODV.

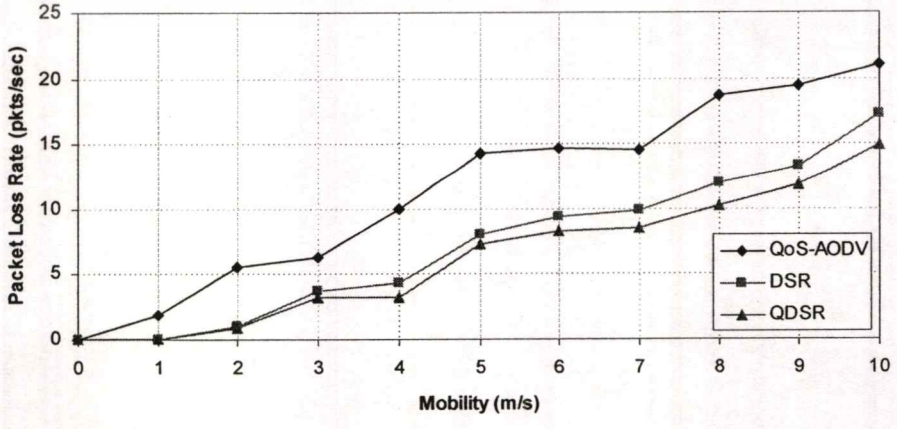


Figure 5.3 Packet loss rate versus mobility

5.4.3 Throughput

For the next experiment, we consider the effect of mobility with the throughput. Since the packet rate in our system is 25 packets/sec (102.4 Kbps), the demand for efficient routing and wireless medium utilization for multimedia traffic is higher in that scenario. We will observe how the three protocols can scale in that demanding network. From Figure 5.4, we observe that at higher data rates with increasing mobility, the performance of the network is significantly degraded. At the highest rate of mobility, the total throughput achieved by three source-destination pairs in QDSR is degraded to 246.78 Kbps or in other way, it is decreased by 19.66% compared with those when there is no node mobility. DSR also performs well at the lower rates of mobility, but it is decreased by 22.71% at the highest rate of mobility. While QoS-AODV decreased its throughput by 27.97% at the highest rate of mobility.

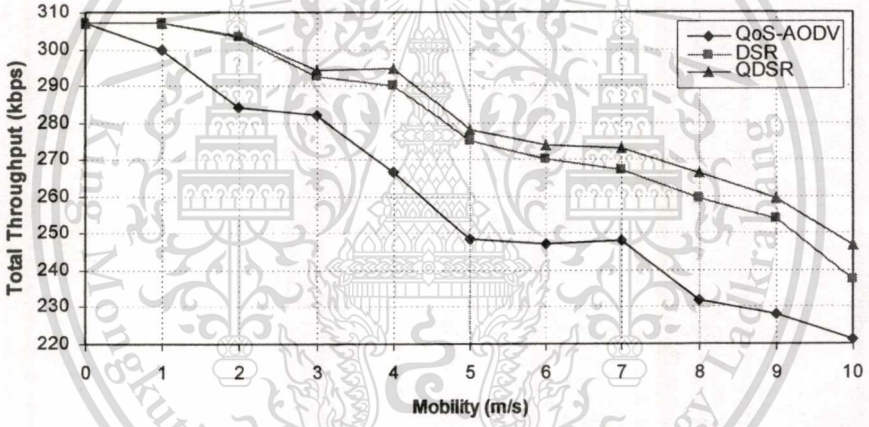


Figure 5.4 Total throughput versus mobility

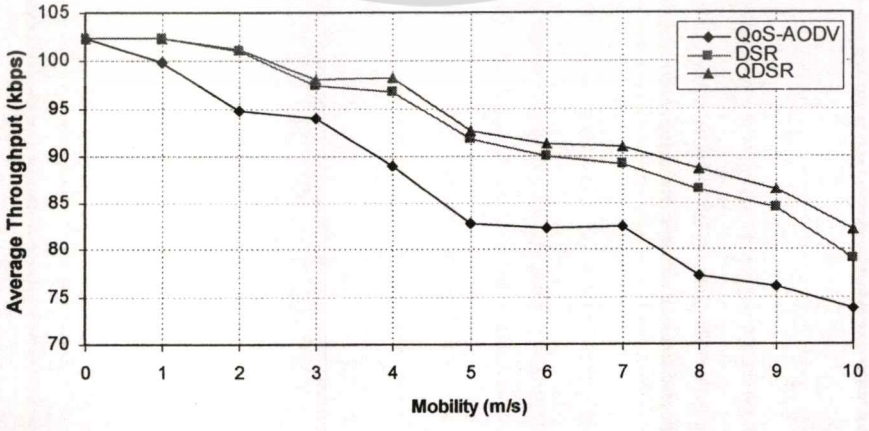


Figure 5.5 Average throughput versus mobility

Figure 5.5 shows the average throughput achieved by each source-destination pair. We observe when there is no node mobility, each protocol achieve the higher throughput which is about 102.4 Kbps or can be mathematically calculated from multiplication of packet size (bit) and the packet's sending rate (packets/sec). This means all multimedia packets receive the destination without any congestion occurred in the network. However, when the mobility is increased, each protocol achieved different throughput. It is resulted from the congestion and dropped packet in the network. Since adding more QoS criteria in the selection of intermediate nodes along the path, QDSR provides more reliable data transmission and achieves the higher throughput over DSR and QoS-AODV.

Moreover, as shown in Figure 5.6, QDSR improves the throughput compared with DSR and QoS-AODV by about 3.78% and 10.34%, respectively. This proves that QDSR provides the efficient routing and higher wireless medium utilization for multimedia traffic.

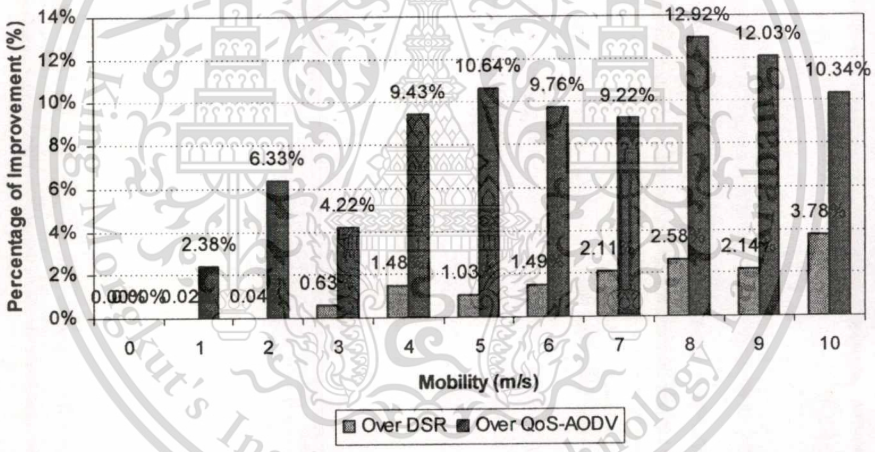


Figure 5.6 Improvement of throughput in QDSR

5.4.4 Average End-to-End Delay

Figure 5.7 shows the average end-to-end delay of a successful connection between the source node and the destination node. DSR has the lowest and-to-end delay, which increases almost linearly with the rate of mobility. QDSR has slightly more end-to-end delay compared with DSR. It is because some QoS routes are not the shortest and sometimes are harder to find than a best effort route at high mobility. However, the routes created by QDSR guarantee the bandwidth requirement, while DSR only finds the shortest path and is not concerned with available of bandwidth. Additionally, in congested conditions, routes become invalid more

easily in DSR and the buffers is full quickly, resulting in a significant increase of packet dropping, routing overhead and time to find new routes or repairing routes.

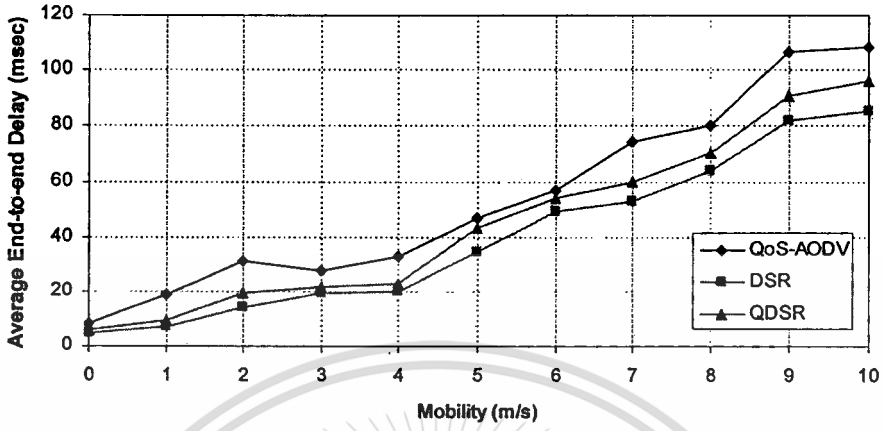


Figure 5.7 Average end-to-end delay versus mobility

QoS-AODV has the highest end-to-end delay among the other protocols. When the mobility is increased, the delay is also increased which is higher than those of DSR and QDSR. It may be caused from the route broken frequently due of mobility and harder to find the QoS path when the mobility is increased. Since QoS-AODV also has proactive behavior to propagate HELLO message periodically, the probability of congestion in the network is higher and that leads to higher delay in network interface transmission queues.

5.4.5 Routing Overhead

Figure 5.8 and 5.9 shows the average number of routing protocol packets sent per second by each protocol in obtaining the delivery ratios as shown in Figure 5.1. DSR and QDSR are plotted on a same scale, but QoS-AODV is plotted on different scales to best show the effect of mobility and offered load on overhead.

DSR and QDSR, which are implemented in an on-demand fashion and in the absence of any kind of periodic control messages, have almost identically shaped curves. QDSR slightly perform well, which sends less routing packet compared with DSR. This is because QDSR maintain the stable path that can reduce the probability of broken route and route maintenance process. However, the routing packet sent by QoS-AODV is very different. which requires about 9.63 times of routing packet the overhead of DSR when there is highest node motion (speed 10 m/s). This dramatic increase of routing packets in QoS-AODV occurs because of its

proactive behavior which propagates HELLO messages periodically from every node in order to maintain connectivity with its neighboring nodes and this leads to the higher routing packets disseminated in the network.

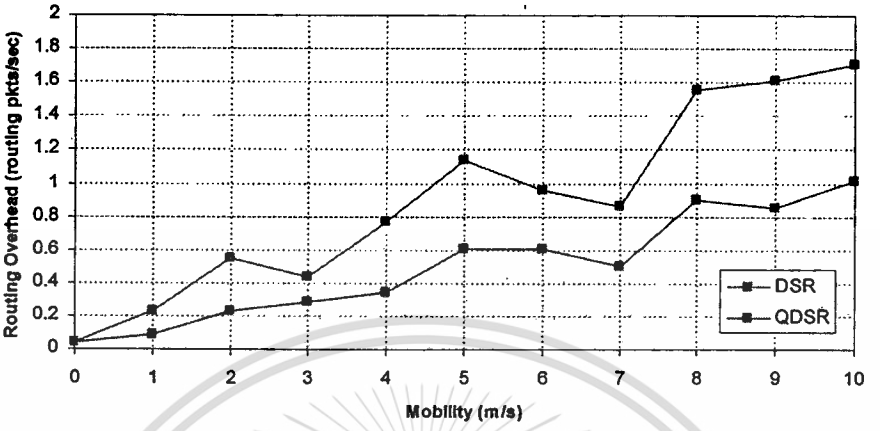


Figure 5.8 Routing overhead (packets/sec) of DSR and QDSR

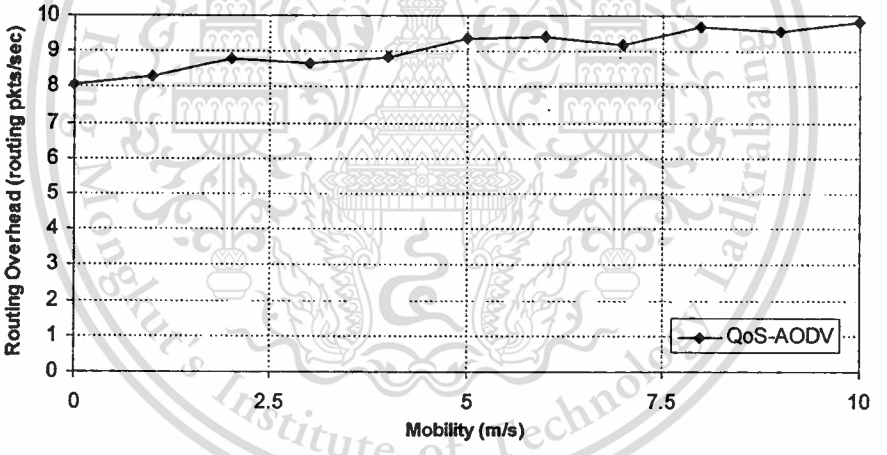


Figure 5.9 Routing overhead (packets/sec) of QoS-AODV

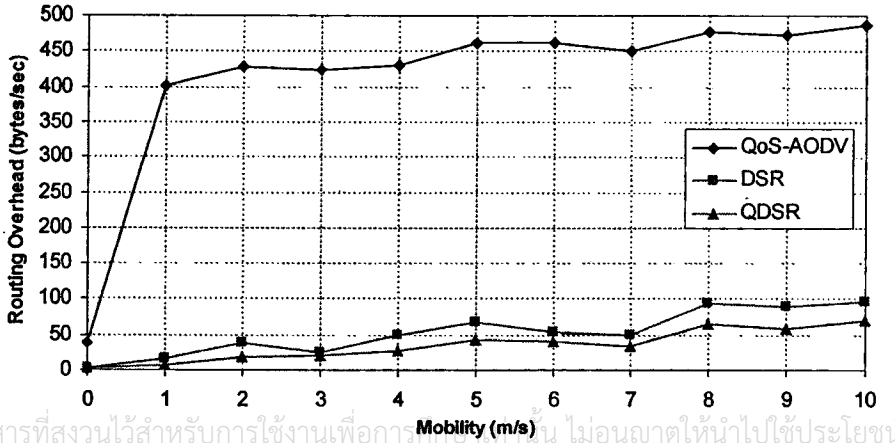


Figure 5.10 Routing overhead (bytes/sec) versus mobility

Moreover, Figure 5.10 shows the routing overhead in term of bytes/sec. From Figure 5.10, we can explain that although the RREQ and RREP packets on QDSR have more fields to record the QoS information on the route, QDSR still achieves less overall routing overhead. It is resulted from the previous analysis that QDSR achieves lower route error and combined with selective re-broadcast mechanism based on QoS function in the intermediate nodes. Since the routing packets in QDSR are sent less often, the lower overall routing overhead in term of bytes/sec can be achieved.

5.4.6 Normalized Routing Load

Normalized routing load informs the fraction of all routing control packets sent by all nodes over the number of received data packets at the destination nodes. This metric discloses how efficient the routing protocol is. The lower this fraction, the more efficient the protocol is. Since the routing control packets in QDSR is sent less often compared with DSR and QoS-AODV as shown in Figure 5.11, the lower overall routing load can be achieved.

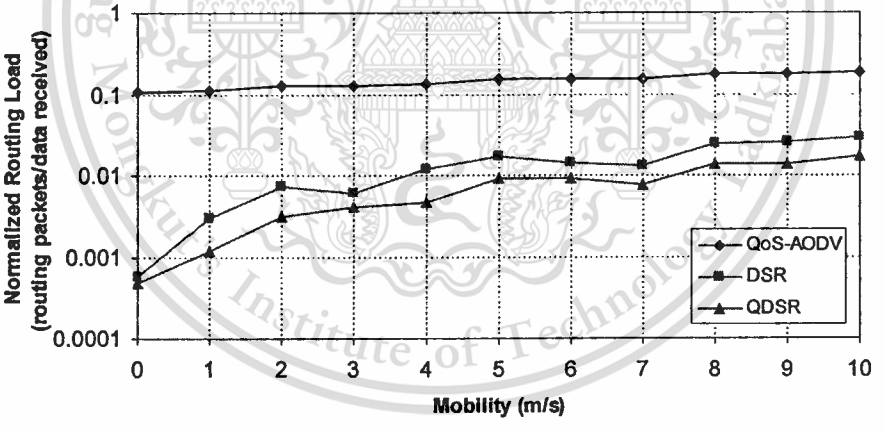


Figure 5.11 Normalized routing load versus mobility

5.4.7 Power Factor

Figure 5.12 shows the comparison of power factor of each protocol. The combination metrics that used for power factor are normalized throughput over the normalized RREQ delay. At the zero mobility, in which all nodes are stationary, QoS-AODV achieves the good performance over DSR and QDSR. It is because QoS-AODV has the lowest delay in searching the route while the normalized throughput of all protocols is converging to 1. However, at the medium and highest rate of mobility, the performance of QoS-AODV is poorly compared with

DSR and QDSR since its throughput is fall below the other protocols. DSR achieves the good performance at the medium and highest rate of mobility since the delay in searching the route is keep low which is based on the shortest path. Although QDSR achieves the higher throughput compared with the other protocol, the delay in searching the route is not as low as the delay in DSR. It is because QDSR find the route that satisfies the QoS requirement and sometimes the routes are not the shortest one. As the result, the performance of power factor of QDSR is below the DSR's performance but still higher than performance that achieved by QoS-AODV.

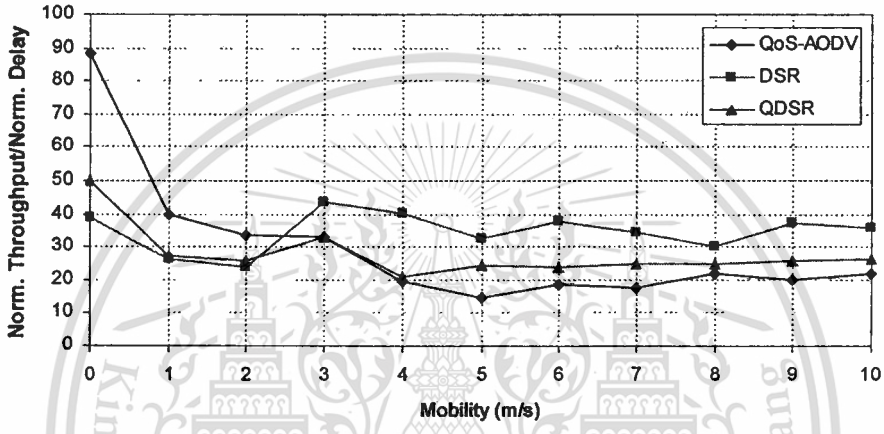


Figure 5.12 Power factor versus mobility

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Mobile ad hoc networks (MANETs) are rapidly evolving and the concept of mobile ad hoc networking has become one of the most challenging research areas of wireless communications. This type of network offers unique benefits and versatility for a variety of applications and situations, such as organizational conferences, emergency search-and-rescue operations and disaster environments. This comes at the expense of overcoming extremely difficult challenges posed namely due to the absence of fixed infrastructure, bandwidth and energy-constrained operation and random mobility patterns of nodes, which are unique to such networks. With the rising popularity of multimedia applications among end users in MANETs has stimulated a spur of research in providing QoS support in such networks. However, the mobile nature, limited resources, and dynamic topology of MANETs cause it to be complicated to provide QoS guarantee in such network.

We proposed a new approach for QoS-aware routing to support multimedia applications in mobile ad hoc networks called Qos-Based Dynamic Source Routing (QDSR) protocol. The proposed scheme requires modifications to an on-demand routing protocol which QoS features are embedded in the routes selection and maintenance procedures. QDSR considers with multiple QoS constraints such as delay, bandwidth, and signal strength to find the most feasible route from the source node to the destination node.

The proposed QoS routing protocol provides not only a bandwidth guarantee mechanism, but also a routing optimally that is considered as the second priority. With the routing optimally mechanism, QDSR selects the most stable links which leads to longer-lived routes and reduces route maintenance.

Simulation results showed that QDSR successfully improves the packet delivery ratio and average throughput over both original DSR and another on-demand QoS routing protocol, QoS-AODV. Results imply that the packet loss rate decreases significantly when the rate of mobility increases since the routes chosen are more stable and that decrease the probability of broken route. In addition, the number of routing packets disseminated in a network is also lower with the absence of any periodic control packet. However, since a QoS route is harder to

find and not usually imply the shortest route, as a consequence, QDSR has slightly higher the end-to-end delay than the best effort routing protocols, such as DSR.

6.2 Future Work

Previous chapter has made an attempt to evaluate each key building block of our proposed QoS routing scheme through simulations. Potential directions for extensions to the research work were presented at the end of this chapter. Although every model that this thesis came up with works well with CBR traffic, it is necessary to analyze their robustness with different traffic mixes, namely with the aid of variable-bit-rate (VBR). We will let this task for our future work.

Once the above tasks have been completed successfully, it would be interesting to implement the complete model in an experimental test-bed to see its practical viability. With the availability of advanced and cheap hardware systems, this task appears feasible. In addition to the other possible directions of future work identified at this chapter, I am also eager to extend my research carrier in practical implementations (prototyping). This would finally enable me to find a suitable model for supporting multimedia applications in mobile ad hoc networks.

REFERENCES

[1] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, **"A Review of Routing Protocols for Mobile Ad Hoc Networks,"** Ad Hoc Networks, Volume 2, Issue 1, pp. 1 – 22, January 2004.

[2] P. Mohapatra, J. Li, and C. Gui, **"QoS in Mobile Ad Hoc Networks,"** IEEE Wireless Communications, pp. 44 - 52, June 2003.

[3] B. Zhang and H.T. Mouftah, **"QoS Routing for Wireless Ad Hoc Networks: Problems, Algorithms, and Protocols,"** IEEE Communications Magazine, Volume 43, Issue 10, pp. 110 – 117, Oct. 2005.

[4] C.E. Perkins and P. Bhagwat, **"Highly Dynamic Destination-Sequenced Distance-Vector routing (DSDV) for Mobile Computers,"** Proceedings of SIGCOMM '94, pp. 234 - 244, August 1994.

[5] D.B. Johnson, D.A. Maltz, and Y-C Hu, **"The Dynamic Source Routing for Mobile Ad-hoc Networks,"** IETF Internet draft, 19 July 2004.

[6] T. Clausen, Ed. and P. Jacquet, Ed., **"Optimized Link State Routing Protocol"**, Project Hipercom, INRIA, October 2003.

[7] S. Corson and J. Macker, **"Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations"**, January 1999.

[8] C. Perkins, E. Belding-Royer, and S. Das., **"Ad hoc On-Demand Distance Vector (AODV) Routing"**, RFC 3561, July 2003.

[9] V.D. Park and M.S. Corson, **"Temporally-Ordered Routing Algorithm (TORA) version 1 functional specification"**, IETF Draft: draft-ietf-manet-tora-spec-04.txt, 20 July 2001.

[10] Z.J. Haas, M.R. Pearlman, and P. Samar, **"The Zone Routing Protocol (ZRP) for Ad Hoc Networks"**, IETF Draft: draft-ietf-manet-zone-zrp-04.txt, July 2002.

[11] B. Carp and H.T. Kung, **"GPSR: Greedy Perimeter Stateless Routing for Wireless Networks"** Proceedings of the sixth annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000).

[12] I.W. Mustika, S. Thipchaksurat, R. Varakulsiripunth, and H. Ishii, **"A QoS-Based Dynamic Source Routing for Multimedia Service in Mobile Ad-Hoc Network"**, The Electrical/Electronics, Computer, Telecommunications and Information

Technology International Conference (ECTI-CON 2007), Volume 2, pp. 805 - 808, 8 - 11 May 2007.

[13] I.W. Mustika, S. Thipchaksurat, R. Varakulsiripunth, and H. Ishii, "**Performance Improvement of QoS-Based Dynamic Source Routing Protocol for Multimedia Services in Mobile Ad-Hoc Networks**", The International Conference on Engineering, Applied Sciences, and Technology (ICEAST 2007), pp. 535 – 538, 21 - 23 November 2007.

[14] C.R. Lin and J-S Liu, "**QoS Routing in Ad Hoc Wireless Networks**", IEEE Journal on Selected Areas in Communications, Vol. 17, No. 8, pp. 1426 - 1438, August 1999.

[15] R. de Renesse, M. Ghassemian, V. Friderikos, and A.H. Aghvami, "**QoS Enabled Routing in Mobile Ad Hoc Networks**", Fifth IEEE International Conference on 3G Mobile Communication Technologies, pp. 678 – 682, 2004.

[16] The Network Simulator – NS2, http://www.isi.edu/nsnam/ns/.

[17] Ubuntu Linux, http://www.ubuntu.com/.

[18] Y. Zhang and T.A. Gulliver, **Quality of Service for Ad Hoc On-Demand Distance Vector Routing,** IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'2005), Volume 3, pp. 192 – 196, , 22-24 Aug. 2005.

[19] Y. Ofek, "**Generating a Fault Tolerant Global Clock using High-Speed Control Signals for the MetaNet Architecture,**" IEEE Transaction on Communication, vol. 42, no. 5, pp. 2179 – 2188, 1994.

Appendix

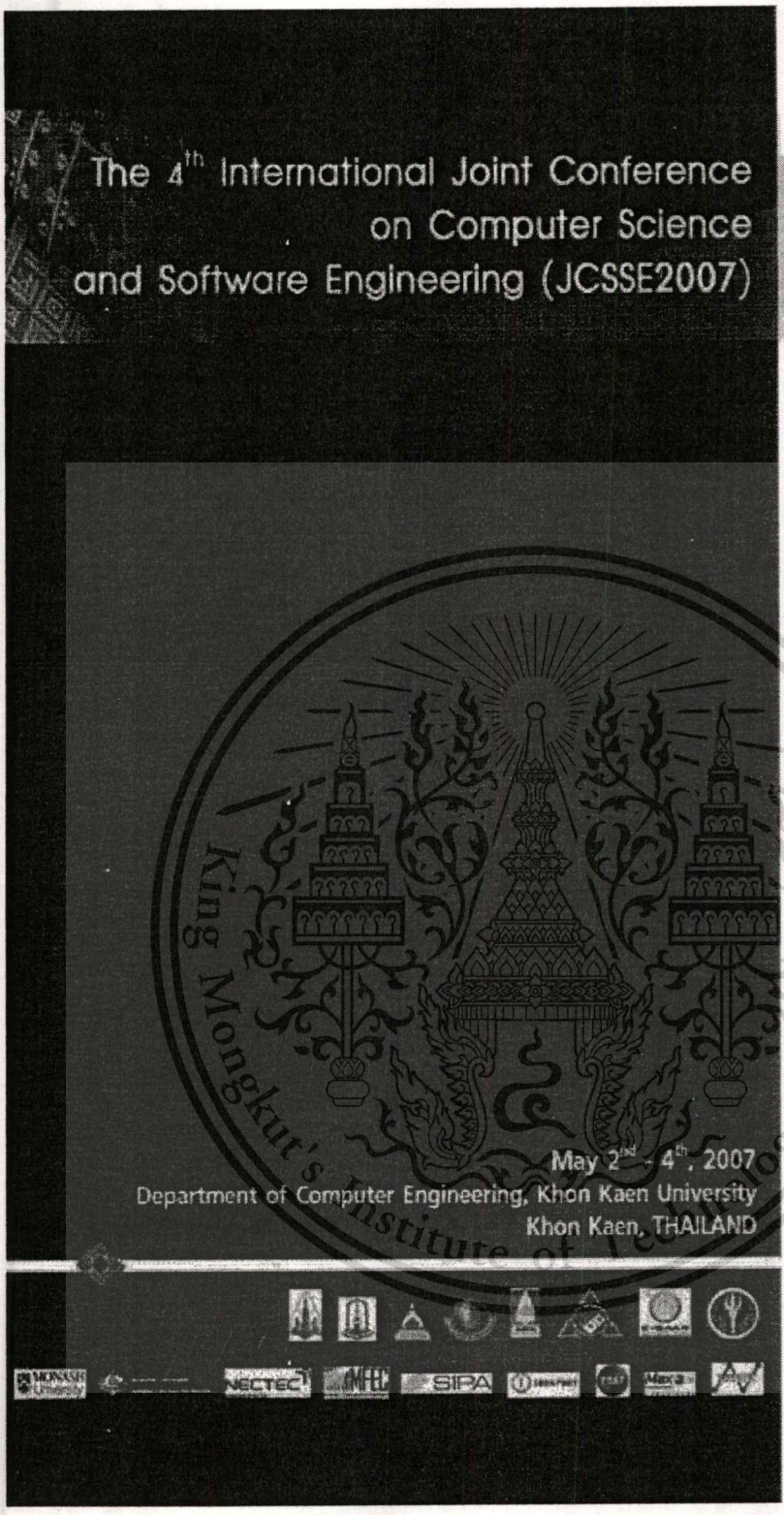
Appendix A

List of Publications

[1] I.W. Mustika, S. Thipchaksurat, R. Varakulsiripunth, and H. Ishii, **"An On-Demand QoS Routing with Multiple Metrics for Multimedia Service in Mobile Ad-Hoc Network"**, International Joint Conference on Computer Science and Software Engineering (JCSSE 2007), 3 - 4 May 2007, pp. 180 – 184.

[2] I.W. Mustika, S. Thipchaksurat, R. Varakulsiripunth, and H. Ishii, **"A QoS-Based Dynamic Source Routing for Multimedia Service in Mobile Ad-Hoc Network"**, The Electrical/Electronics, Computer, Telecommunications and Information Technology International Conference (ECTI-CON 2007), 8 - 11 May 2007, Volume 2, pp. 805-808.

[3] I.W. Mustika, S. Thipchaksurat, R. Varakulsiripunth, and H. Ishii, **"Performance Improvement of QoS-Based Dynamic Source Routing Protocol for Multimedia Services in Mobile Ad-Hoc Networks"**, The International Conference on Engineering, Applied Sciences, and Technology (ICEAST 2007), 21 - 23 November 2007, pp. 535 – 538.



International Joint Conference on Computer Science & Software Engineering (JCSSE2007)
May 2-4, 2007

An On-Demand QoS Routing with Multiple Metrics for Multimedia Service in Mobile Ad Hoc Networks

I Wayan MUSTIKA*, Sakchai THIPCHAKSURAT*, Ruttikorn VARAKULSIRIPUNTH*, Hiroshi ISHII**
*Faculty of Engineering and Research Center for Communication and Information Technology (ReCCIT)
King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok, Thailand 10520
**Department of Communication Engineering, School of Information Technology and Electronics, Tokai University, Japan
Email: mus_thicks@yahoo.com, {ktsakcha, kvruttik}@kmitl.ac.th, ishii@dt.u-tokai.ac.jp

Abstract **- The emergence of real-time and multimedia applications in mobile ad hoc networks (MANETs) has recently generated much interest. To support such kind of applications, Quality of Service (QoS) features are desired. However, the mobile nature, limited resources, and dynamic topology of MANETs make it complicates to provide QoS guarantee in such network. In this paper, we propose an on-demand QoS routing scheme for multimedia service based on multiple QoS metrics called QoS-Based Dynamic Source Routing (QDSR). QDSR gathers information about end-to-end delay, available bandwidth and signal strength during route discovery and uses the information in route decision process. We evaluate the performance of QDSR by means of simulation. The result shows that QDSR provides the higher throughput, better packet delivery ratio and lower routing overhead than those of DSR.**

Keywords: **QoS routing, multimedia, mobile ad hoc networks, DSR.**

I. INTRODUCTION

A Mobile Ad Hoc Network (MANET) is an autonomous collection of wireless mobile nodes with no predetermined topology or central control. The nodes intercommunicate through single-hop or multi-hop paths in a peer-to-peer fashion and operate both as hosts as well as routers as shown in Fig. 1. Such network can be used in the situation where either there is no any wireless communication infrastructure present or such infrastructure cannot be used such as in battlefields, emergency search-and-rescue operations, and disaster environments [1]. Collaborative computing and communication in smaller areas (home office, educational buildings, organization conferences, etc.) also can be set up using MANETs.

Since MANET is characterized by its fast changing topology, extensive research efforts have been devoted to the design of routing protocols for MANETs [1, 4, 5]. However, the most existing work is based on non-QoS requirement for example the source node attempts to transmit data to the destination node without any delay or bandwidth guarantee. As the real-time applications such as multimedia stream, live voice and video conference grow rapidly and sensitive with delay and bandwidth, QoS features become more important. However, the mobile nature, limited resource, and dynamic topology of MANETs make it complicates to provide QoS guarantee in such network.

There are many researches on QoS support in MANETs include QoS models [6, 7], Layered QoS [2], QoS Medium Access Control (MAC) [8], and QoS routing [3, 9, 10] which only consider with a single QoS constraint and sometimes it is not suitable to the highly unpredictable ad hoc environment. In order to guarantee the real-time and multimedia applications with multiple QoS requirement, we propose a QoS-Based Dynamic Source Routing (QDSR) protocol. The protocol finds the route that meets the multiple QoS constraints and has a better chance for surviving over a period of time from node movement. The multiple QoS constraints considered here are delay, bandwidth and signal strength. The rest of this paper is organized as follows. In Section II, we briefly introduce MANETs routing protocols and QoS routing. The proposed scheme QDSR is presented in Section III. Section IV presents our simulation and result. Finally, conclusion and future work are given in Section V.

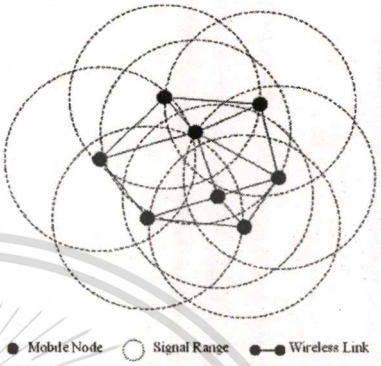


Figure 1. A mobile ad hoc network

II. RELATED WORK

A. MANETs Routing Protocols

Routing protocols in ad hoc networks can be classified into three groups [1]: *Proactive* (*Table Driven*), *Reactive* (*On-Demand*) and *Hybrid* routing protocols as shown in Fig. 2.

A.1 Proactive (Table Driven) routing protocols

Proactive routing protocols come as a natural extension of protocols for the wired network. In these routing protocols,

each node has a number of tables to maintain routing information to every other node in the network and these tables are updated periodically. Examples of proactive routing protocols are Destination Sequence Distance Vector (DSDV), Wireless Routing Protocol (WRP), Cluster-head Gateway Switch Routing (CGSR) [1, 4] and so on. Since using periodic update, these protocols allow some significantly overheads and consume bandwidth in the network.

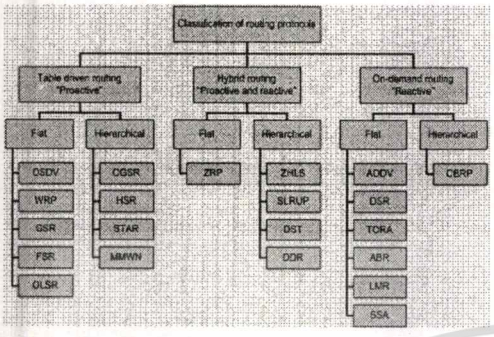


Figure 2. Classification of routing protocol

A.2 Reactive (On-Demand) routing protocols

Unlike proactive routing protocols, reactive routing protocols neither maintain nor periodically update their route tables. Several reactive routing protocols have been proposed such as Dynamic Source Routing (DSR), Ad Hoc On-Demand Distance Vector (AODV), Temporally Ordered Routing Algorithm (TORA) [1, 5] and so on. These routing protocols were designed to overcome the overheads occurred in proactive protocols.

DSR is one of the more commonly accepted routing protocols [5]. In DSR, when a node wishes to send data packets and does not have any routing information, it initiates a route discovery by flooding all of its neighbors with route request (RREQ) packets. Each neighbor broadcasts the RREQ, adds its own address in the header of packet. When the RREQ is received by the destination or by a node that has a route to destination, a route reply (RREP) is generated and sent back to the sender along with the addresses accumulated in the RREQ header. Route maintenance is invoked when the source node detects the broken link and then it can choose any other route to the destination node or invokes the route discovery again to find the new route to destination.

A.3 Hybrid routing protocols

Hybrid routing protocols combine proactive and reactive routing protocols. These protocols are designed to increase scalability when the wireless networks become larger. All nodes in the network are separated into groups. Each group is called a cluster. All clusters form a hierarchical infrastructure. In these routing protocols, the proactive maintains routes in a cluster and reactive maintains routes between clusters. Several Hybrid routing protocols have been proposed such as Zone Routing Protocol (ZRP), Zone-based Hierarchical Link State (ZHLS), Distributed Dynamic Routing (DDR) [1], and so on.

B. QoS Routing

QoS routing is the critical issue in MANETs due to mobility, mobile nature, and resource limitation. Some technical challenges are described in terms of designing optimal routes, providing reliable transmission and robustness, efficiency in resource utilization, adaptability, and unlimited mobility.

Normally, QoS routing protocols can be classified into two main groups. The first group concerns with some traditional protocols extended with QoS features. Their goal is to eliminate among the selected routes that do not meet the given criteria. The second group takes into account QoS aspect within routing algorithm. Two types of constraints exist such as application specific [9, 12] (e.g. bandwidth or delay) and network specific features (e.g. congestion, energy constraint, link stability) [10, 11, 13]. We can conclude that most of QoS routing protocols merely focus only on one application's QoS requirement based on a single QoS constraint which is not usually suitable to the highly unpredictable and dynamic ad hoc environment.

III. QoS-Based Dynamic Source Routing Protocol

In MANETs, the data service can be classified into two groups. The first group is the non real-time data service which bandwidth is not so sensitive and allows the delay during transmission. The routing protocol will do the best effort for such data service. Examples of this service are file transfer, web documents and other traditional datagram applications. The second group is the real-time data service which bandwidth and delay are very sensitive such as on-demand multimedia stream, video and audio conference, etc. This kind of traffic needs QoS guarantee. The proposed scheme constitutes an extension of DSR routing protocol, in which QoS features are embedded in the selection and maintenance of routes.

A. QDSR procedures

QDSR relies on three procedures: route discovery, route reservation and route maintenance.

1. Route discovery

QDSR adds a QoS header to an ordinary route request (RREQ) packet and has the following fields: <*packet_type*, *source_addr*, *dest_addr*, *sequence#*, *route_list*, *link_info* (*delay*, *bandwidth*, *signal*), *slot_array_list*, *qos_enabled*, *bw_slot_req*, *qos_function*, *TTL*, *data*>. The following QoS metrics are used in our proposed scheme to provide the network layer with QoS requirements that are used during route discovery:

- **Delay**: The end-to-end delay from a source to a destination that the application can tolerate.
- **Bandwidth**: The lowest bandwidth that application can tolerate on the route.
- **Signal Strength**: More stable route through dynamic network topology that has probability of surviving longer.

In this paper, we consider the combination of delay, bandwidth, and signal strength as the QoS function for selecting the appropriate route.

a) Source node algorithm

As illustrated in Fig. 3, when a source node wishes to send the real-time data packets, it first checks its resource availability such as free slots for bandwidth. If there is no resource available, the route discovery is canceled and the upper layer is informed. If the source node has sufficient resource, it will check the route in the route cache whether there is a route to the destination that satisfies the traffic requirement. If no valid route is stored in the cache, the source node places the information in RREQ header, initializes the QoS function and broadcasts the RREQ packet to its neighbors.

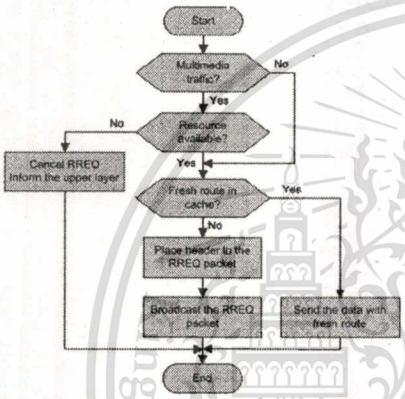


Figure 3. Source node algorithm

b) Intermediate node algorithm

This algorithm is executed by intermediate nodes on receiving a RREQ packet. From Fig. 4, we can explain the process of intermediate nodes as the following steps:

1. If the pair *<dest_addr, sequence#>* for the RREQ is seen recently, check whether Time to Live (TTL) is zero or not, if TTL counts down to zero, we drop the RREQ and do not process it further. TTL can limit the length of delivery path and control the flooding traffic.
2. Check whether the route request is searching for the path of multimedia traffic or common data traffic. If the traffic is multimedia traffic, the intermediate node calculates the link delay and available bandwidth from the sources to this node. If the result satisfies the QoS requirement, the intermediate node records the state of the data slots to the *slot_array_list*. Otherwise, we drop the RREQ.
3. Compute the QoS function including the link delay, available bandwidth and received signal strength. The QoS function of path P is given by a heuristic formula (1):

$$f(P) = (d - D(P))^{\alpha} + BW(P)^{\beta} + (S(P)/Rt)^{\gamma} \qquad (1)$$

where $D(P)$ is the delay at given path, d is the maximum end-to-end delay that application can tolerate, $BW(P)$ is the available path bandwidth, $S(P)$ is the minimum signal strength along the path, and Rt is the received signal threshold. α, β and γ are fixed parameters used to show relative importance about delay, bandwidth and signal strength in QoS routing based on application requirement.
4. Decrement TTL by one and append the address of this node to the *route_list* to track the route which the packet has traversed.
5. Selectively re-broadcast RREQ packet based on the QoS function. In order to reduce the broadcast storm and routing overhead, an intermediate node can re-broadcast a second RREQ packet only if it has higher QoS function than the first RREQ packet, otherwise the RREQ will be discarded.

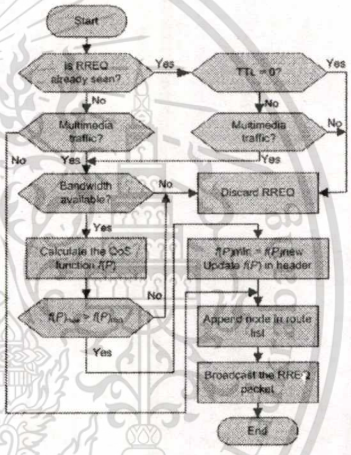


Figure 4. Intermediate node algorithm

c) Destination node algorithm

When the first RREQ packet with QoS enabled reaches the destination, the node starts the timer interval and during that time the node examines the QoS function of every arrived RREQ packet. When the timer interval expired, the destination node selects the RREQ packet that has the highest

QoS function and sends the route reply (RREP) packet carrying the route information to the source node. When another RREQ packet with the same sequence number arrives after the threshold time interval, it will not be considered and do not process it further. Figure 5 shows the process of destination node when receive RREQ packets.

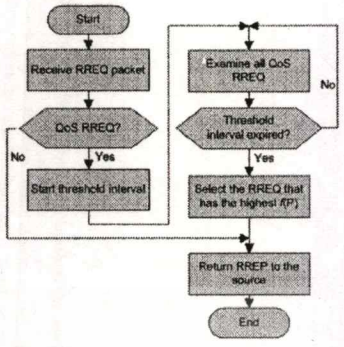


Figure 5. Route decision algorithm on the destination node

2. Resource reservation procedure

On receiving the RREQ packet, the destination node reserves resources using a TDMA-based bandwidth reservation model. As a RREQ packet travels from the source to the destination, it automatically sets up the reverse path from the destination back to the source. From the RREQ packet, we can obtain the state of data slots and according to the information recorded within the RREQ packet. The destination can set up a QoS route and reserves resources (slots) hop-by-hop backward to the source. If the reservation operation fails or the resources are not available, a route error (RRER) packet is sent.

3. Route maintenance

If the destination node does not receive the data packet in reserved time slot, it considers that the link is broken and uses its dedicated control time slot to broadcast a route error (RERR) packet with finite TTL. Any host in the QoS route will forward the RERR packet to the source. On receiving the RERR packet, the source can re-establish a new RREQ with QoS enabled.

IV. Simulation and Result

Network Simulator (NS-2) is used to analyze the performance of QDSR. Multiple QoS constraints are applied to the QoS function such as end-to-end delay, bandwidth and signal strength. The performance of our proposed scheme is then compared to the original DSR. Only the route discovery procedure has been implemented and the maintenance procedure is the one implemented in DSR.

A. Simulation model and parameters

The simulations were run using an ad hoc network model composed of 8 nodes that moved randomly at uniform speed over a rectangular area 800 x 800m flat space. The simulation parameters and constant values for $f(P)$ in Eq.(1) are shown in Table I and II, respectively.

TABLE I. Simulation Model

Parameter	Value
Simulation area	800 x 800m
Number of nodes	8 nodes
Max. speed	0 - 10 m/s
Node movement	Random
Transmission range	250 m
Source / Packet size	CBR / 512 bytes
Sending rate	20 packets/sec

TABLE II. Constant Values

Constant	Value
α	0.3
β	0.3
γ	0.4
Max delay (d)	1 sec
Rt	3.65×10^{-19}

B. Simulation results

The result of simulation is shown in Fig. 6. We notice that QDSR improves the average throughput in comparison to DSR. When the mobility is increased, the performance of DSR will be significantly degraded. Adding more QoS criteria in the selection of intermediate nodes along the path leads to a more reliable data transmission which improves the throughput.

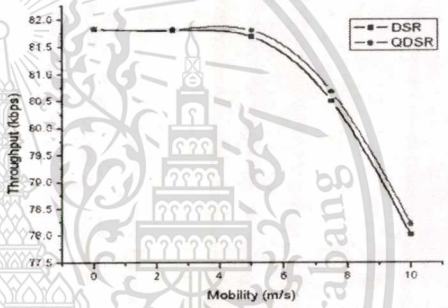


Figure 6. Average throughput versus mobility

From Table III, we can see that on average, QDSR sends about 43.6% fewer routing packets than those by DSR. Since one of the multiple constraint that considered by QDSR on selecting the route is signal strength, the chosen hop along the path is generally shorter and therefore, it can reduced the probability of route error. Moreover, normalized routing overhead informs the number of routing packets transmitted per data packet and it is proportional to node's mobility. From Fig. 7, we also shows that although the RREQ and RREP packets on QDSR have more fields that record the QoS information on the route, QDSR still achieves less overall

overhead. It is resulted from the previous analysis that QDSR achieves lower route error and combined with selective re-broadcast mechanism based on QoS function in the intermediate nodes. Since RREQ packet on QDSR is sent less often as shown in Table III, lower overall routing overhead for multimedia traffic can be achieved.

TABLE III. NUMBER OF ROUTING PACKETS

Mobility (m/s)	0	2.5	5	7.5	10	Sum
DSR (non-multimedia)	21	47	97	124	209	498
QDSR (multimedia)	19	25	41	55	77	217

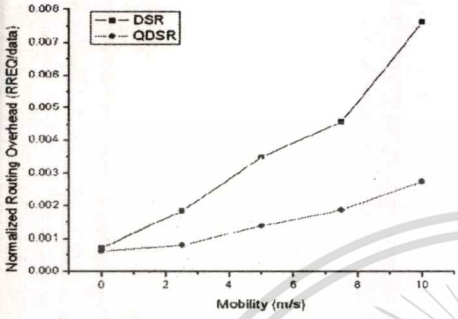


Figure 7. Normalized routing overhead versus mobility

As shown in Fig. 8, the packet delivery ratio of multimedia traffic on QDSR is higher than one of common data traffic on DSR. When the network becomes congested or the route is broken frequently due the mobility, many packets will be discarded or lost. The lower packet loss rate implies the more stable route. Since QDSR selects the most feasible and stable route according to combination of delay, bandwidth and signal strength, the probability of broken route can be decreased and that leads to improvement on the packet delivery ratio which is higher than those on DSR.

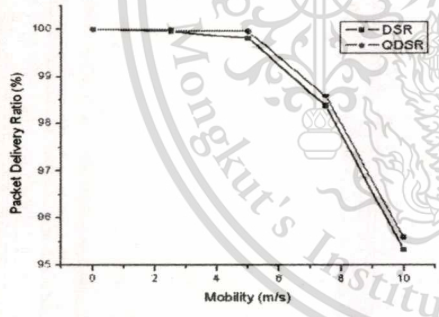


Figure 8. Packet delivery ratio versus mobility

V. CONCLUSION AND FUTURE WORK

We propose a new approach for QoS-aware routing to support the real-time and multimedia applications in mobile ad hoc networks. Our proposed scheme, QDSR, considers with multiple QoS constraints such as delay, bandwidth, and signal strength to find the most feasible route from the source node to the destination node. It also selects the most stable links which leads to longer-lived route and reduces route maintenance. We can conclude that our proposed scheme provides the improvement in terms of throughput, packet delivery ratio and lower routing overhead that is suitable for real-time and multimedia services in small networks.

Our future works will consequently consist in adding more functionalities such as improving the maintenance procedure, scalability issues and analyzing the effect of QoS function parameters on the performance of our proposed scheme.

ACKNOWLEDGMENT

This work was supported by ASEAN University Network/Southeast Asia Engineering Education Development Network (AUN/SEED-Net) JICA Project.

REFERENCES

[1] Mehran Abolhasan, Tadeusz Wysocki, and Eryk Dutkiewicz,"A Review of Routing Protocols for Mobile Ad Hoc Networks," *Ad Hoc Networks*, Volume 2, Issue 1, Pages 1 – 22. January 2004.

[2] P. Mohapatra, J. Li, and C. Gui, "QoS in Mobile Ad Hoc Networks," IEEE Wireless Communications, pp. 44 - 52, June 2003.

[3] B. Zhang and H.T. Mouftah, "QoS Routing for Wireless Ad Hoc Networks: Problems, Algorithms, and Protocols," *IEEE Communications Magazine*. Volume 43. Issue 10, Page(s):110 – 117, Oct. 2005.

[4] C.E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector routing (DSDV) for Mobile Computers," *Proceedings of SIGCOMM '94*, pp. 234 - 244, August 1994.

[5] D.B. Johnson, D.A. Maltz, and Y-C. Hu, "The Dynamic Source Routing for Mobile Ad-hoc Networks," *IETF Internet draft*. 19 July 2004.

[6] H. Xiao,W.K.G. Seah, A. Lo, K.C. Chua, "A Flexible Quality of Service Model for Mobile Ad-Hoc Networks", *Proc. of IEEE VTC2000*, Tokyo, May 2000.

[7] G.S. Ahn, A.T. Campbell, A. Veres, L.H. Sun, "Supporting Service Differentation for Real-Time and Best-Effort Traffic in Stateless Wireless Ad Hoc Networks (SWAN)", *IEEE Transactions on Mobile Computing*, Vol.1, No.3, pp.192-207, 2002.

[8] IEEE 802.11e/D6.0, "Draft Amendment to Standard for Information Technology – Telecommunications and Information Exchange between Systems – LAN/MAN Specific Requirements – Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Quality of Service (QoS) Enhancements," Nov. 2003.

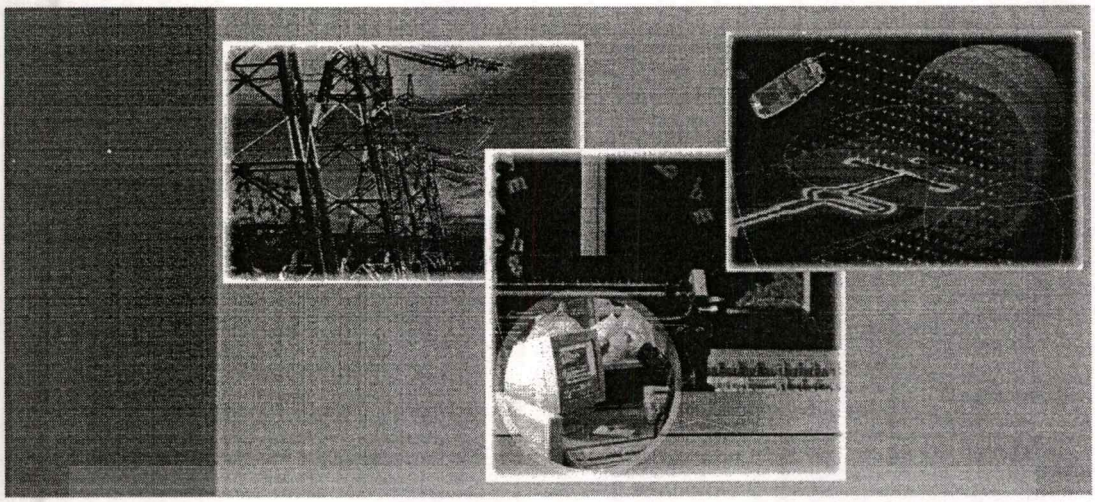
[9] C.R. Lin and J.-S. Liu, "QoS Routing in Ad Hoc Wireless Networks", *IEEE Journal on Selected Areas in Communications*. Vol. 17, No. 8, pp. 1426-1438. August 1999.

[10] C.-K. Toh, "A Novel Distributed Routing Protocol to Support Ad-Hoc Mobile Computing," *Wireless Personal Communication*, Jan. 1997.

[11] J. Li, D. Cordes and J. Zhang, "Power-Aware Routing Protocols in Ad Hoc Wireless Networks," *IEEE Wireless Communications*, December 2005.

[12] R. Sivakumar, P. Sinha, V. Bharghavan, "CEDAR : A Core Extraction Distributed Ad Hoc Routing Algorithm ", *IEEE JSAC*, vol.17, no. 8, pp 1454-65, Aug. 1999.

[13] R. Dube, C. D. Rais, K-Y Wang, and S. K. Tripathi, "Signal Stability-Based Adaptive Routing (SSA) for Ad Hoc Mobile Networks," *IEEE Personal Communications*, Volume 4, Issue 1, pp. 36 - 45, Feb. 1997.

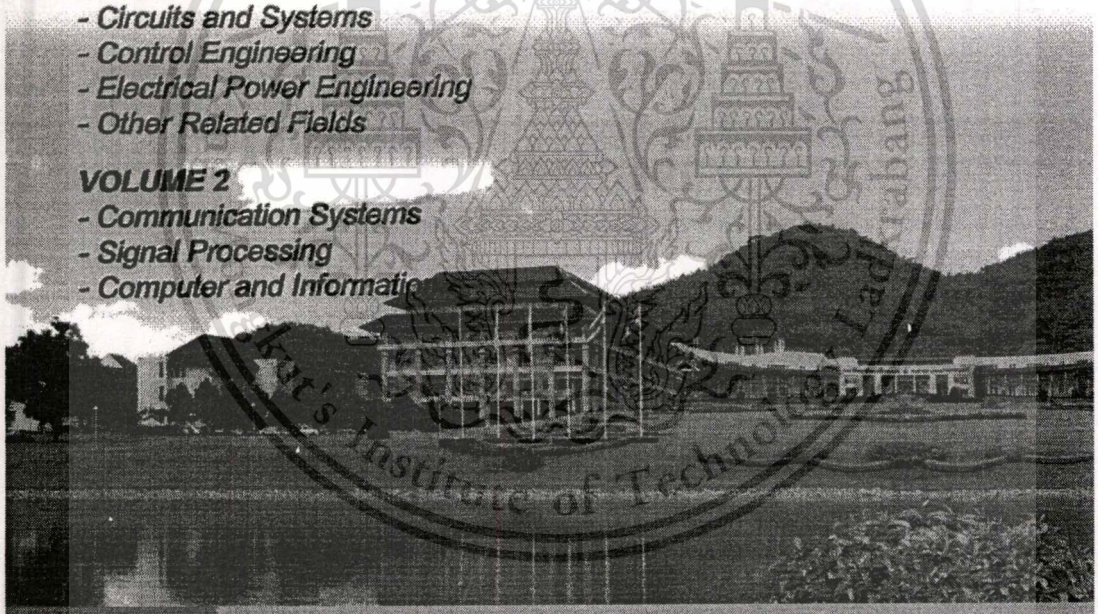


ECTI-CON 2007

Mae Fah Luang University, Chiang Rai, Thailand
May 9-12, 2007

VOLUME 1
- Circuits and Systems
- Control Engineering
- Electrical Power Engineering
- Other Related Fields

VOLUME 2
- Communication Systems
- Signal Processing
- Computer and Informatic



ECTI Association | IEEE THAILAND SECTION | NECTEC a member of NSTDA | Western Digital

A QoS-Based Dynamic Source Routing for Multimedia Service in Mobile Ad Hoc Networks

Wayan MUSTIKA*, Sakchai THIPCHAKSURAT*, Ruttikorn VARAKULSIRIPUNTH*, Hiroshi ISHII**
*Faculty of Engineering and Research Center for Communications and Information Technology (ReCCIT)
King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok, Thailand 10520
**Department of Communication Engineering, School of Information Technology and Electronics, Tokai University, Japan
Email: mus_thicks@yahoo.com, {ktsakcha, kvruttik}@kmitl.ac.th, ishii@dt.u-tokai.ac.jp

***Abstract* - The growth of real-time multimedia applications in mobile ad hoc networks (MANETs) will be increasingly popular in the near future. To support such kind of applications, Quality of Service (QoS) features are desired. However, the mobile nature, limited resources, and dynamic topology of MANETs make it complicates to provide QoS guarantee in such network. In this paper, we propose a routing scheme for multimedia service based on multiple QoS constraint called QoS-Based Dynamic Source Routing (QDSR). It applies the source routing mechanism defined by the Dynamic Source Routing (DSR) to reduce routing overheads and improve throughput. QDSR gathers information about end-to-end delay, available bandwidth and signal strength during route discovery and uses the information in route decision process. We evaluate the performance of our proposed scheme by means of simulation. The result shows that QDSR provides the higher throughput and lower packet loss rate than those of DSR.**

***Keywords*: QoS routing, mobile ad hoc networks, Dynamic Source Routing.**

I. INTRODUCTION

A Mobile Ad Hoc Network (MANET) consists of wireless mobile nodes with no predetermined topology or central control. The nodes intercommunicate through single-hop or multi-hop paths in a peer-to-peer fashion and operate both as hosts as well as routers. Such network can be used in the situation where either there is no any wireless communication infrastructure present or such infrastructure cannot be used such as in battlefields, emergency search-and-rescue operations, and disaster environments [1]. Collaborative computing and communication in smaller areas (educational buildings, conferences, etc.) also can be set up using MANETs.

Since MANET is characterized by its fast changing topology, extensive research efforts have been devoted to the design of routing protocols for MANETs [1, 4, 5]. However, the most existing work is based on non-QoS requirement for example the source node attempts to transmit data to the destination node without any delay or bandwidth guarantee. As the real-time applications such as multimedia stream, live voice and video conference grow rapidly and sensitive with delay and bandwidth, QoS features become more important. However, the mobile nature, limited resource, and dynamic topology of MANETs make it complicates to provide QoS guarantee in such network.

There are many researches on QoS support in MANETs include QoS models [6], Layered QoS [2], QoS Medium Access Control (MAC) [7], and QoS routing [3, 8] which only consider with a single QoS constraint and sometimes it is not suitable to the highly unpredictable ad hoc environment. In order to guarantee the real-time and multimedia applications with QoS requirement, we propose a QoS-Based Dynamic Source Routing (QDSR) protocol. The protocol finds the route that meets the multiple QoS constraints and has a better chance for surviving over a period of time from node movement. The multiple QoS constraints considered here are delay, bandwidth and signal strength. The rest of the paper is organized as follows. In Section II, we introduce all related works. The proposed scheme QDSR is presented in Section III. Section IV presents our simulation and results. Finally, conclusion and future work are given in Section V.

II. RELATED WORK

A. MANETs Routing Protocols

Routing protocols in ad hoc networks can be classified into three groups [1]: *Proactive* (*Table Driven*), *Reactive* (*On-Demand*) and *Hybrid* as shown in Fig. 1.

A.1 Proactive (Table Driven) routing protocols

Proactive routing protocols come as a natural extension of protocols for the wired network. In these routing protocols, each node has a number of tables to maintain routing information to every other node in the network and these tables are updated periodically. Examples of proactive routing protocols are DSDV, WRP, CGSR [1] and so on. Since using periodic update, these protocols allow some significantly overheads and consume bandwidth in the network.

A.2 Reactive (On-Demand) routing protocols

Unlike proactive routing protocols, reactive routing protocols neither maintain nor periodically update their route tables. Several reactive routing protocols have been proposed such as DSR, AODV, TORA [1, 5] and so on. These routing protocols were designed to overcome the overheads occurred in proactive protocols.

DSR is one of the more commonly accepted routing protocols [5]. In DSR, when a node wishes to send data packets and does not have any routing information, it initiates a route discovery by flooding all of its neighbors with route request (RREQ) packets. Each neighbor broadcasts this

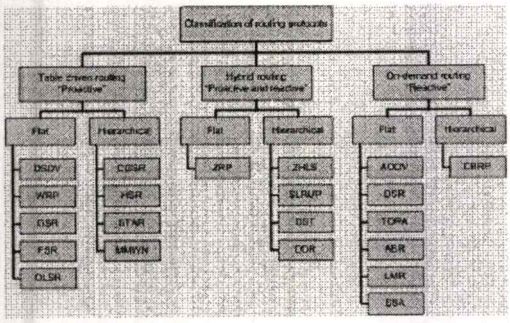


Figure 1. Classification of routing protocols

RREQ, adding its own address in the header of packet. When the RREQ is received by the destination or by a node that have a route to destination, a route reply (RREP) is generated and sent back to the sender along with the addresses accumulated in the RREQ header. Route maintenance is invoked when the source node detects the broken link and then it can choose any other route to the destination node or invokes the route discovery again to find the new route to destination.

A.3 Hybrid routing protocols

Hybrid routing protocols combine proactive and reactive routing protocols. These protocols are designed to increase scalability when the wireless networks become larger. All nodes in the network are separated into groups. Each group is called a cluster. All clusters form a hierarchical infrastructure. In these routing protocols, the proactive maintains routes in a cluster and reactive maintains routes between clusters. Several hybrid routing protocols have been proposed such as ZRP, ZHLS, DDR [1] and so on.

B. QoS Routing

QoS routing is the critical issue in MANETs due to mobility, mobile nature, and resource limitation. Some technical challenges are described in terms of designing optimal routes, providing reliable transmission, robustness, efficiency in resource utilization, adaptability, and unlimited mobility.

Normally, QoS routing protocols can be classified into two main groups. The first group concerns with some traditional protocols extended with QoS features. Their goal is to eliminate among the selected routes that do not meet the given criteria. The second group takes into account QoS aspect within the routing algorithm. Two types of constraints exist such as application specific [8, 10] (e.g. bandwidth or delay) and network specific features (e.g. congestion, energy constraint and link stability) [9, 11]. We can conclude that most of QoS routing protocols merely focus only on one application's QoS requirement based on a single QoS constraint which is not usually suitable to the highly unpredictable and dynamic ad hoc environment.

III. QoS-Based Dynamic Source Routing Protocol

In MANETs, the data service can be classified into two groups. The first group is the non real-time data service which bandwidth is not so sensitive and allows the delay during transmission. Examples of this service are file transfer, web documents and other traditional datagram applications. The second group is the real-time data service which bandwidth and delay are very sensitive such as on-demand multimedia stream, video and audio conference, etc. This kind of traffic needs QoS guarantee on the route. The proposed scheme constitutes an extension of DSR routing protocol, in which QoS features are embedded in the selection and maintenance of the routes.

A. QDSR procedures

QDSR relies on three procedures: route discovery, route reservation and route maintenance.

1. Route discovery

QDSR adds a QoS header to an ordinary route request (RREQ) packet and has the following fields: <*packet_type, source_addr, dest_addr, seq_num#, route_list, link_info* (*delay, bandwidth, signal*), *slot_array_list, traffic_type, bw_slot_req, qos_function, TTL, data*>. The following QoS metrics are used in our proposed scheme to provide the network layer with QoS requirement that are used during route discovery:

- **Delay**: The end-to-end delay from a source to a destination that the application can tolerate.
- **Bandwidth**: The lowest bandwidth that the application can tolerate on the route.
- **Signal Strength**: More stable route through dynamic network topology that has probability of surviving longer.

In this paper, we consider the combination of delay, bandwidth, and signal strength as the QoS function for selecting the appropriate route from source to destination.

a) Source node algorithm

When a source node wishes to send the real-time data packets, it first checks its resource availability such as free slots for bandwidth. If there is no resource available, the route discovery is canceled and the upper layer is informed. If the source node has sufficient resource, it will check the route in the route cache whether there is a route to the destination that satisfies the traffic requirement. If no valid route is stored in the route cache, the source node will place the information in RREQ header, initializes the QoS function and broadcasts the RREQ packet to its neighbors.

b) Intermediate node algorithm

This algorithm is executed by intermediate nodes on receiving a RREQ packet. From Fig. 2, we can explain the process of intermediate nodes as the following steps:

1. If the pair <*dest_addr, seq_num#*> for the RREQ is seen recently, check whether Time to Live (TTL) is zero or not, if TTL counts down to zero, we drop the RREQ and do not

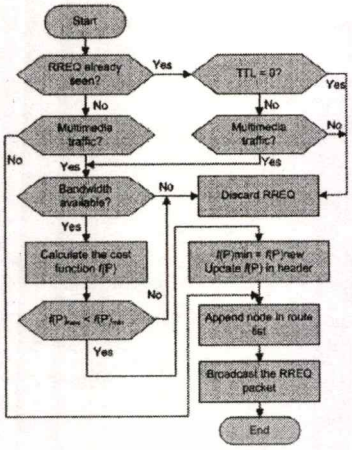


Figure 2. Intermediate node algorithm

process it further. TTL can limit the length of delivery path and control the flooding traffic.

2. Check whether the route request is searching for the path of multimedia traffic or common data traffic. If the traffic is multimedia traffic, the intermediate node calculates the available bandwidth from the source to this node using path bandwidth calculation and records the state of the data slots to the *slot_array_list*. If the result does not satisfy the QoS requirement, we drop the RREQ.
3. Compute the QoS function including the link delay, available bandwidth and received signal strength. The QoS function of path P is given by a heuristic formula (1):

$$f(P) = 1/\left((d - D(P))^{\alpha} + BW(P)^{\beta} + (S(P)/Rt)^{\gamma}\right) \quad (1)$$

where $D(P)$ is the delay at given path, d is the maximum delay that application can tolerate, $BW(P)$ is the available path bandwidth, $S(P)$ is the minimum signal strength along the path, and Rt is the received signal threshold. α, β and γ are fixed parameters used to show relative importance about delay, bandwidth and signal strength in QoS routing.

4. Decrement TTL by one and append the address of this node to the *route_list* to track the route which the packet has traversed.
5. Selectively re-broadcast RREQ packet based on the QoS function. In order to reduce the broadcast storm and routing overhead, an intermediate node can re-broadcast a second RREQ packet only if it has lower QoS function than the first RREQ packet, otherwise the RREQ will be discarded.

c) Destination node algorithm

When the first RREQ packet with QoS enabled reaches the destination, the node starts the timer interval and during that time the node examines the QoS function of every arrived RREQ packet. When the timer interval expired, the destination node selects the RREQ packet that has the lowest QoS function and sends the route reply (RREP) packet carrying this route to the source node. When another RREQ packet with the same sequence number arrives after the threshold time interval, it will not be considered and do not process it further. Fig. 3 shows the process of destination node when receive RREQ packets.

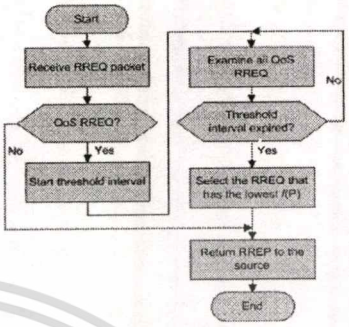


Figure 3. Destination node and route decision algorithm

2. Resource reservation procedure

On receiving the RREQ packet, the destination node reserves resource using a TDMA-based bandwidth reservation model. As a RREQ packet travels from the source to the destination, it automatically sets up the reverse path from the destination back to the source. From the RREQ packet, we can obtain the state of data slots and according to the information recorded within the RREQ packet. The destination can set up a QoS route and reserves resources (slots) hop-by-hop backward to the source. If the reservation operation fails or the resource are not available, a route error (RRER) packet is sent.

3. Route maintenance

If the destination node does not receive the data packet in reserved time slot, it considers that the link is broken and uses its dedicated control time slot to broadcast a route error (RERR) packet with finite TTL. Any node in the QoS route will forward the RERR packet to the source. On receiving the RERR packet, the source can re-establish a new RREQ with QoS enabled.

IV. Simulation and Result

Network Simulator (NS-2) is used to analyze the performance of QDSR. The performance of our proposed scheme is then compared to DSR. Only the route discovery procedure has been implemented and the maintenance procedure is the one implemented in DSR.

A. *Simulation model and parameters*

The simulations were run using an ad hoc network model composed of 8 nodes that moved randomly at uniform speed over a rectangular area 800 x 800m flat space. The simulation parameters and constant values for $f(P)$ in Eq.(1) are shown in Table I and II, respectively.

TABLE I. SIMULATION MODEL

Parameter	Value
Simulation area	800 x 800m
Number of nodes	8 nodes
Max speed	0 - 10 m/s
Node movement	Random
Transmission range	250 m
Source / Packet size	CBR / 512 bytes
Sending rate	20 packets/sec

TABLE II. CONSTANT VALUES

Constant	Value
α	0.3
β	0.3
γ	0.4
Max delay (d)	1 sec
Rt	3.65×10^{-10}

B. *Simulation results*

The result of simulation is shown in Fig. 4. We notice that QDSR improves the average throughput in comparison to DSR. When the mobility is increased, the performance of DSR will be significantly degraded. Adding more QoS criteria in the selection of intermediate nodes along the path leads to a more reliable data transmission which improves the throughput.

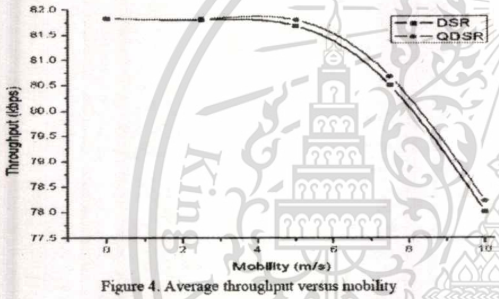


Figure 4. Average throughput versus mobility

In Fig. 5, we show the packet loss rate versus mobility. It also shows that the packet loss rate is proportional to the node's mobility. When the network becomes congested or the route is broken frequently due the mobility, many packets will be discarded or lost. The lower packet loss rate implies the more stable route. Since our proposed scheme involves the multiple QoS constraints and selects the most feasible and stable route, the probability of broken route can be decreased and that leads the lower packet loss rate than those on DSR.

V. CONCLUSION AND FUTURE WORK

We propose a new approach for QoS-aware routing to support multimedia applications in mobile ad hoc networks.

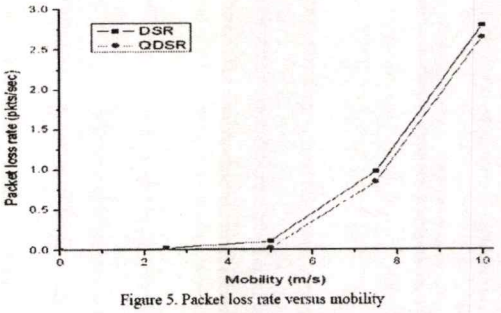


Figure 5. Packet loss rate versus mobility

Our proposed scheme, QDSR, considers with multiple QoS constraints such as delay, bandwidth, and signal strength to find the most feasible route from the source node to the destination node. It also selects the most stable links which leads to longer-lived routes and reduces route maintenance. We can conclude that our proposed scheme provides the improvement in terms of throughput and lower packets lost for real-time traffic and multimedia services in small networks.

Our future works will consequently consist in adding more functionalities such as improving the maintenance procedure and analyzing the effect of QoS function parameters on the performance of our proposed scheme.

REFERENCES

[1] Mehran Abolhasan, Tadeusz Wysocki, and Eryk Dutkiewicz."A Review of Routing Protocols for Mobile Ad Hoc Networks." *Ad Hoc Networks*, Volume 2, Issue 1, Pages 1 – 22, January 2004.

[2] P. Mohapatra, J. Li, and C. Gui, "QoS in Mobile Ad Hoc Networks," IEEE Wireless Communications, pp. 44 - 52, June 2003.

[3] B. Zhang and H.T. Mouftah, "QoS Routing for Wireless Ad Hoc Networks: Problems, Algorithms, and Protocols," *IEEE Communications Magazine*, Volume 43, Issue 10, Page(s):110 – 117, Oct. 2005.

[4] C.E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector routing (DSDV) for Mobile Computers," *Proceedings of SIGCOMM '94*, pp. 234 - 244, August 1994.

[5] D.B. Johnson, D.A. Maltz, and Y-C. Hu, "The Dynamic Source Routing for Mobile Ad-hoc Networks," *IETF Internet draft*, 19 July 2004.

[6] G.S. Ahn, A.T. Campbell, A. Veres, L.H. Sun, "Supporting Service Differentation for Real-Time and Best-Effort Traffic in Stateless Wireless Ad Hoc Networks (SWAN)", *IEEE Transactions on Mobile Computing*, Vol.1, No.3, pp.192-207, 2002.

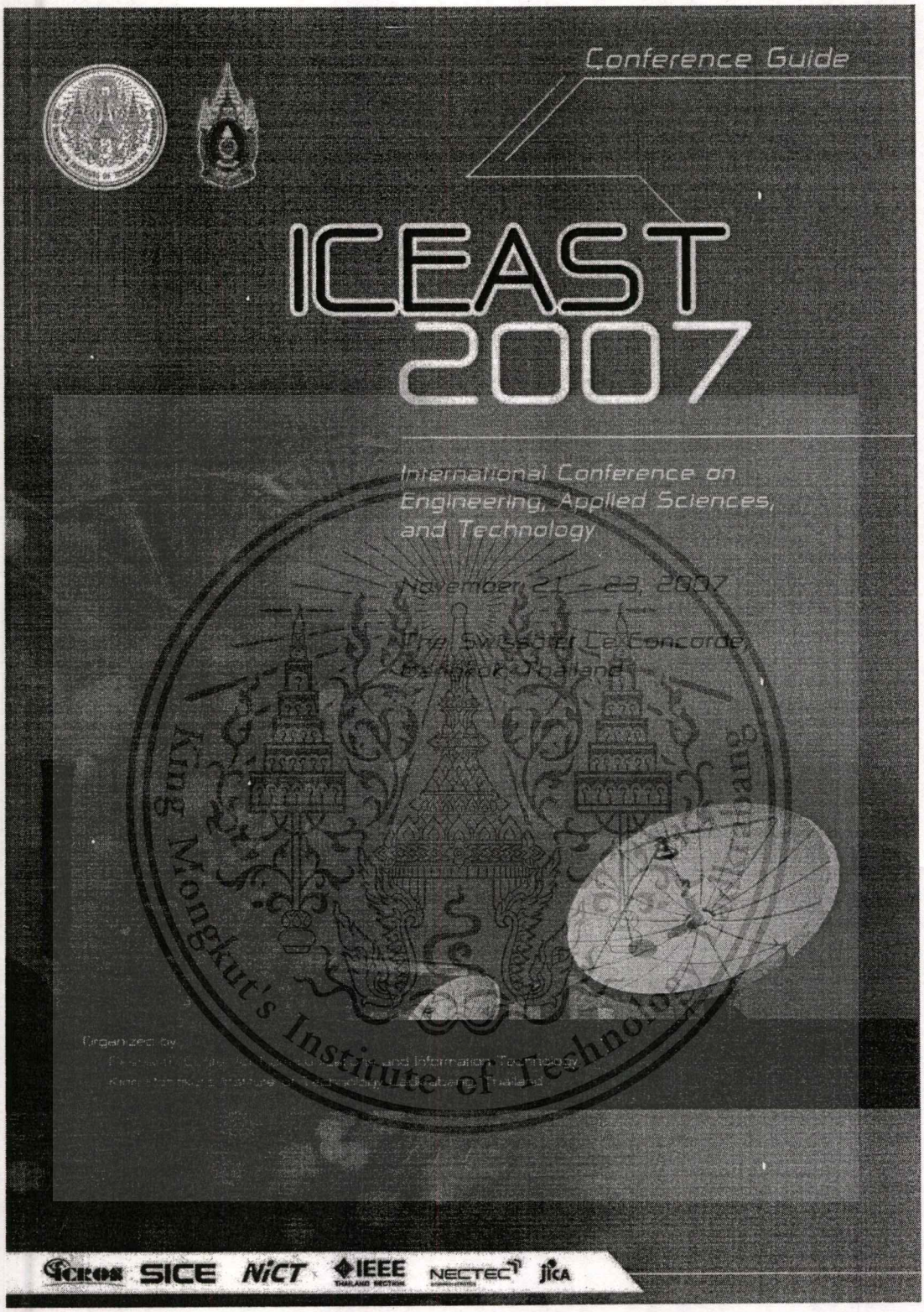
[7] IEEE 802.11e/D6.0, "Draft Amendment to Standard for Information Technology – Telecommunications and Information Exchange between Systems – LAN/MAN Specific Requirements – Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Quality of Service (QoS) Enhancements," Nov. 2003.

[8] C.R. Lin and J.-S. Liu, "QoS Routing in Ad Hoc Wireless Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 8, pp. 1426-1438, August 1999.

[9] J. Li, D. Cordes and J. Zhang, "Power-Aware Routing Protocols in Ad Hoc Wireless Networks," *IEEE Wireless Communications*, December 2005.

[10] R. Sivakumar, P. Sinha, V. Bharghavan, "CEDAR : A Core Extraction Distributed Ad Hoc Routing Algorithm ", IEEE JSAC, vol.17,no. 8, pp 1454-65, Aug. 1999.

[11] R. Dube, C. D. Rais, K-Y Wang, and S. K. Tripathi, "Signal Stability-Based Adaptive Routing (SSA) for Ad Hoc Mobile Networks," *IEEE Personal Communications*, Volume 4, Issue 1, pp. 36 - 45, Feb. 1997.



Performance Improvement of QoS-Based Dynamic Source Routing Protocol for Multimedia Services in Mobile Ad Hoc Networks

I Wayan MUSTIKA[1], Sakchai THIPCHAKSURAT[1], Ruttikorn VARAKULSIRIPUNTH[1], Hiroshi ISHII[2]
[1]Faculty of Engineering and Research Center for Communications and Information Technology (ReCCIT)
King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok, Thailand 10520
[2]Department of Communication Engineering, School of Information Technology and Electronics, Tokai University, Japan
Email: mus_thicks@yahoo.com, {ktsakcha, kvruttik}@kmitl.ac.th, ishii@dt.u-tokai.ac.jp

Abstract—**With the rising popularity of multimedia application in Mobile Ad Hoc Networks (MANETs), Quality of Service (QoS) features become more important. However, most of QoS routing protocols merely focus on a single QoS constraint which is not usually suitable to the highly unpredictable and dynamic ad hoc environment. In this paper, we propose a routing scheme for multimedia service based on multiple QoS constraints in MANETs called QoS-Based Dynamic Source Routing (QDSR). QDSR gathers information about available bandwidth, link delay and signal strength during route discovery and uses the information in route decision process. We evaluate the performance of our proposed scheme via simulations. The simulation results show that QDSR provides the higher throughput and lower routing load than those of DSR.**

Keywords—**Mobile Ad Hoc Networks, QoS routing, Dynamic Source Routing.**

I. INTRODUCTION

A Mobile Ad Hoc Network (MANET) is an autonomous collection of wireless mobile nodes with no predetermined topology or central control. The nodes intercommunicate through single-hop or multi-hop paths in a peer-to-peer fashion and operate themselves as hosts or routers. MANETs provide a practical way to rapidly build a decentralized communication network in areas where either there is no any wireless communication infrastructure provided or such infrastructure cannot be used such as in battlefields, emergency search-and-rescue operations, and disaster environments [1].

Since MANET is characterized by its fast changing topology, some extensive researches had been devoted to the design of routing protocols for MANETs [1, 4, 5]. However, the most existing work is based on non-QoS requirement, for example the source node attempts to transmit data to the destination node without any delay or bandwidth guarantee. With the rising popularity of multimedia application in MANETs such as multimedia stream, live voice and video conference, QoS features become more important. However, the mobile nature, limited resources, and dynamic topology of MANETs cause it to be complicated to provide QoS guarantee in such network.

There are many researches on QoS support in MANETs that include QoS models [6], Layered QoS [2], and QoS routing [3, 7] which only consider with a single QoS constraint and sometimes they are not suitable to the highly unpredictable ad hoc environment. In order to guarantee the real-time and multimedia applications with QoS requirement, we propose a QoS-Based Dynamic Source Routing (QDSR) protocol. This protocol finds the route that satisfies the multiple QoS constraints and has a better chance for surviving over a period of time after node movement. The multiple QoS constraints considered here are delay, bandwidth and signal strength. The rest of the paper is organized as follows. In Section II, we introduce all related works. The detailed description of QDSR's algorithm and its operations are given in Section III. Section IV presents our simulation and results. Finally, section V provides conclusion and highlights our future work.

II. RELATED WORK

A. MANETs Routing Protocols

Routing protocols in ad hoc networks can be classified into three groups [1]: *Proactive* (*Table Driven*), *Reactive* (*On-Demand*) and *Hybrid*. In proactive routing protocols, each node has a number of tables to maintain routing information to every other node in the network and these tables are updated periodically. Since using periodic update, these protocols incur some significantly overheads and consume bandwidth in the network. Contrarily to proactive protocols, reactive protocols neither maintain nor periodically update their route tables. These routing protocols were designed to overcome the overheads occurred in proactive protocols. Several reactive routing protocols have been proposed such as DSR, AODV, TORA [1, 5] and so on. Hybrid routing protocols combine proactive and reactive routing protocols. These protocols are designed to increase scalability when the wireless networks become larger.

DSR is one of the more commonly accepted routing protocols [5]. In DSR, when a node wishes to send data packets and does not have any routing information, it initiates a route discovery by flooding all of its neighbors with route request (RREQ) packets. Each neighbor broadcasts this RREQ, adding its own address in the header of packet. When the RREQ is received by the destination or by an intermediate node on the route to destination, a route reply (RREP) is generated and sent back to the source node along with the addresses accumulated in the RREQ header. Route maintenance is

978-974-8308-56-2

invoked when the source node detects the broken link and then it attempts any other route or invokes the route discovery again to find the new route for the subsequent packets.

B. QoS Routing

QoS routing is the critical issue in MANETs due to mobility, mobile nature, and resource limitation. Some technical challenges are described in terms of designing optimal routes, providing reliable transmission, robustness, efficiency in resource utilization, adaptability, and unlimited mobility.

In [8] Sinha *et al.* presented the Core-Extraction Distributed Ad Hoc Routing (CEDAR). CEDAR is designed to select routes with sufficient bandwidth resources. The protocol in [7] also works to make a routing decision on accepting an arriving request with a specific bandwidth requirement. While Signal Stability-Based Adaptive Routing tries to discover stronger routes based on signal strength and location stability of the nodes [9]. Most of existing QoS routing protocols merely focus on one application's QoS requirement based on a single QoS constraint which is not usually suitable to the highly unpredictable and dynamic ad hoc environment.

III. QOS-BASED DYNAMIC SOURCE ROUTING PROTOCOL

QDSR is our proposed scheme that separates the data service into two groups. The first group is the non real-time data service which bandwidth is not so sensitive and allows the delay during transmission. Examples of this service are file transfer, web documents and other traditional datagram applications. The second group is the real-time data service which bandwidth and delay are very sensitive such as on-demand multimedia stream, video and audio conference, etc. These kinds of traffic need QoS guarantee on the route. QDSR constitutes an extension of DSR routing protocol, in which QoS features are embedded in the routes selection procedures.

A. QDSR procedures

QDSR relies on three procedures: route discovery, route reservation and route maintenance.

1. Route discovery

QDSR adds a QoS header to an ordinary route request (RREQ) packet and has the following fields: *<packet_type, source_addr, dest_addr, sequence#, route_list, link_info (delay, bandwidth, signal), slot_array_list, qos_enabled, bw_slot_req, qos_function, TTL, data>*. In this research, we consider the bandwidth as the primary importance. It is because bandwidth guarantee is one of the most critical requirements for real time applications. While combination of delay and signal strength are considered as the secondary importance in order to give the mobile nodes a better chance for surviving over a period of time from node movement and path broken.

a) Source node algorithm

When a source node wishes to send the real-time data packets, it first checks its resource availability such as free slots for bandwidth. If there is no resource available, the route discovery is canceled and the upper layer is informed. If the source node has sufficient resource, it will check the route in the route cache whether there is a route to the destination that satisfies the traffic requirement. If no valid route is stored in the route cache, the source node will place the information in RREQ header, initializes the QoS function and broadcasts the RREQ packet to all of its neighbors.

b) Intermediate node algorithm

This algorithm is executed by intermediate nodes on receiving a RREQ packet. From Fig. 1, we can explain the process of intermediate nodes as the following steps:

1. If the pair *<dest_addr, sequence#>* for the RREQ is seen recently, check whether Time to Live (TTL) is zero or not, if TTL counts down to zero, we drop the RREQ and do not process it further. TTL can limit the length of delivery path and control the flooding traffic.
2. Check whether the route request is searching for the path of multimedia traffic or common data traffic. If the traffic is multimedia traffic, the intermediate node computes the link bandwidth and path bandwidth from the sources to this intermediate node. If the result satisfies the QoS requirement, the intermediate node records the state of the data slots to the *slot_array_list*. Otherwise, the RREQ will be drop.
3. Calculates the link delay and signal strength from the sources to this intermediate node and compute the QoS function including the link delay and received signal strength. The QoS function for remaining metrics on path P is given by a heuristic formula (1):

$$f(P) = ((d - D(P))/d) + (S(P)/Rt) \quad (1)$$

where $D(P)$ is the delay at given path, d is the maximum end-to-end delay that can be tolerated, $S(P)$ is the minimum signal strength along the path, and Rt is the received signal threshold. The higher value of QoS function the higher probability of lower delay and more stable link route.
4. Decrement TTL by one and append the address of this node to the *route_list* to track the route which the packet has traversed.
5. Perform selectively re-broadcast mechanism based on the QoS function. In order to reduce the broadcast storm and routing overhead, an intermediate node can re-broadcast a second RREQ packet only if it has higher QoS function than the previous RREQ packet with the same sequence number, otherwise the RREQ will be discarded.

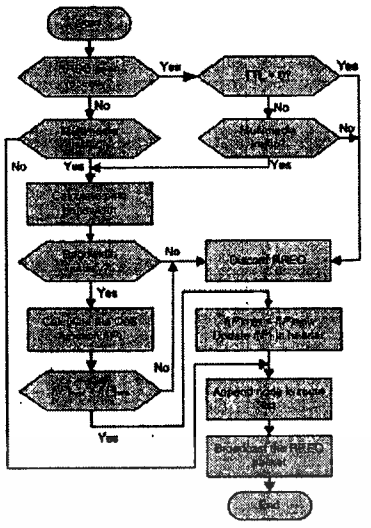


Figure 1. Intermediate node algorithm

c) Destination node algorithm

When the first RREQ packet with QoS enabled reaches the destination node, the node starts the timer interval and during that time the destination node examines the QoS function of every arrived RREQ packet. When the timer interval expired, the destination node selects the RREQ packet that has the highest QoS function and sends the route reply (RREP) packet carrying the route information to the source node. When another RREQ packet with the same sequence number arrives after the time interval, it will not be considered and do not process it further. Figure 2 shows the process of destination node when receive RREQ packets.

2. Resource reservation procedure

On receiving the RREQ packet, the destination node reserves resources using a TDMA-based bandwidth reservation model. As a RREQ packet travels from the source to the destination, it automatically sets up the reverse path from the destination back to the source. From the RREQ packet, we can obtain the state of data slots in according to the information recorded within the RREQ packet. The destination node can set up a QoS route and reserves resources (slots) hop-by-hop backward to the source node. If the reservation operation fails or the resources are not available, a route error (RRER) packet is sent.

3. Route maintenance

If the destination node does not receive the data packets in reserved time slot, it considers that the link is broken and uses its dedicated control time slot to broadcast a route error (RERR) packet with finite TTL. Any node in the QoS route will forward the RERR packet to the source node. On receiving the RERR packet, the source node can re-establish a new RREQ with QoS enabled.

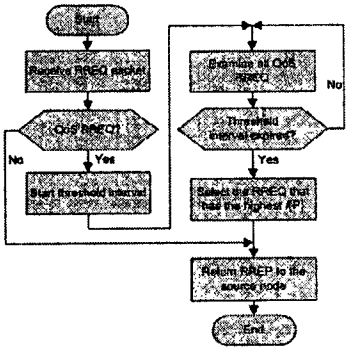


Figure 2. Route decision algorithm on the destination node

IV. SIMULATION AND RESULTS

We simulate and analyze the performance of QDSR by using Network Simulator (NS-2). The performance of QDSR is then compared to DSR. Only the route discovery procedure has been implemented and the maintenance procedure is the one implemented in DSR.

A. Simulation model and parameters

The simulations were operated on an ad hoc network model composed of 8 nodes, whose initial position are chosen from a uniform distribution over a rectangular area 800 x 800m flat space. The nodes' moving speeds are uniformly chosen from 0 to 10 m/s and random waypoint is used as the mobility model. The detail of simulation parameters and constant values for $f(P)$ in Eq. (1) are shown in Table I and II, respectively.

TABLE I. SIMULATION MODEL

Parameter	Value
Simulation area	800 x 800m
Number of nodes	8 nodes
Mobility	0 ~ 10 m/s
Node movement	Random waypoint
Transmission range	250 m
Source / Packet size	CBR / 512 bytes
Sending rate	25 packets/sec
Num of CBR source	3 sources

TABLE II. CONSTANT VALUES

Constant	Value
Max delay (d)	1 sec
R_l	3.65×10^{-10}

B. Simulation results

The result of simulation is shown in Fig. 3. We notice that QDSR improves the average throughput in comparison to DSR. When the mobility is increased, the performance of the network will be significantly degraded. Adding more QoS criteria in the selection of

intermediate nodes along the path leads to a more reliable data transmission which improves the throughput.

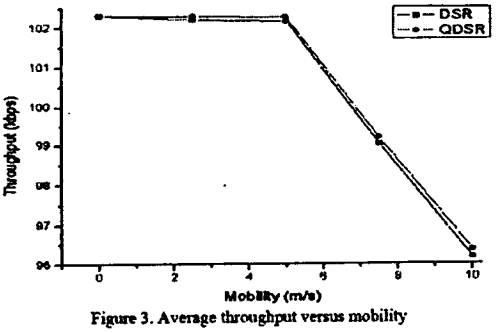


Figure 3. Average throughput versus mobility

From Table III, we can see that on average, QDSR sends about 37.88 % fewer routing packets than those by DSR. Since one of the multiple constraint that considered by QDSR on selecting the route is signal strength, the chosen hop along the path is generally shorter and therefore, it can reduced the probability of route error. Moreover, normalized routing load informs the number of routing packets transmitted per data packet delivered as shown in Fig 4. Since the routing packet on QDSR is sent less often as shown in Table III, lower overall routing overhead for multimedia traffic can be achieved.

TABLE III. NUMBER OF ROUTING PACKETS

Mobility (m/s)	0	2.5	5	7.5	10	Sum
DSR (non-multimedia)	21	76	94	184	190	565
QDSR (multimedia)	14	24	36	65	75	214

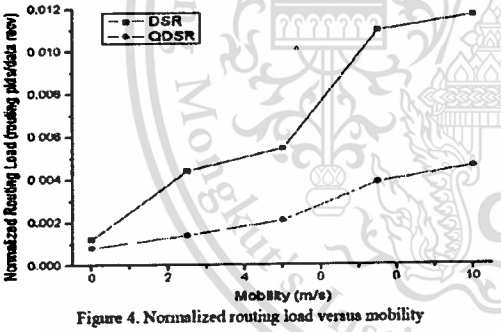


Figure 4. Normalized routing load versus mobility

For the next experiment, we consider the average end-to-end delay of a successful connection between the source node and the destination node. From Fig. 5, we can see that QDSR is appropriate in high mobility model in which its average end-to-end delay is slightly higher but more stable than those on DSR. Since the route chosen by QDSR is frequently more hops and shorter hop than those chosen by DSR, this results in more stable and reliable link from the source node to the destination node.

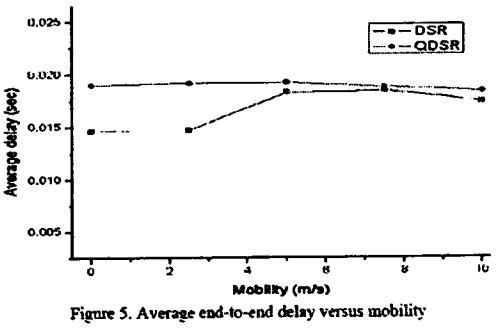


Figure 5. Average end-to-end delay versus mobility

V. CONCLUSION

We propose a new approach for QoS-aware routing to support multimedia applications in mobile ad hoc networks called QDSR. QDSR considers with multiple QoS constraints such as delay, bandwidth, and signal strength to find the most feasible route from the source node to the destination node. It also selects the most stable links which leads to longer-lived routes and reduces route maintenance. We can conclude that our proposed scheme provides the improvement in terms of throughput with lower routing overhead and packet loss rate for real-time traffic and multimedia services in small networks.

ACKNOWLEDGMENT

This work was supported by AUN/SEED-Net JICA Project.

REFERENCES

[1] Mehran Abolhasan, Tadeusz Wysocki, and Eryk Dutkiewicz,"A Review of Routing Protocols for Mobile Ad Hoc Networks." *Ad Hoc Networks*, Volume 2, Issue 1, pp. 1 – 22, January 2004.

[2] P. Mohapatra, J. Li, and C. Gui. "QoS in Mobile Ad Hoc Networks," IEEE Wireless Communications, pp. 44 - 52. June 2003.

[3] B. Zhang and H.T. Mouftah. "QoS Routing for Wireless Ad Hoc Networks: Problems, Algorithms, and Protocols." *IEEE Communications Magazine*, Volume 43, Issue 10, pp. 110 – 117, Oct. 2005.

[4] C.E. Perkins and P. .Bhagwat. "Highly Dynamic Destination-Sequenced Distance-Vector routing (DSDV) for Mobile Computers," *Proceedings of SIGCOMM '94*, pp. 234 - 244, August 1994.

[5] D.B. Johnson, D.A. Maltz, and Y-C Hu, "The Dynamic Source Routing for Mobile Ad-hoc Networks," *IETF Internet draft*, 19 July 2004.

[6] G.S. Ahn, A.T. Campbell, A. Veres, L.H. Sun, "Supporting Service Differentiation for Real-Time and Best-Effort Traffic in Stateless Wireless Ad Hoc Networks (SWAN)". *IEEE Transactions on Mobile Computing*, Vol. 1, No.3, pp. 192-207, 2002.

[7] C.R. Lin and J.-S. Liu, "QoS Routing in Ad Hoc Wireless Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 8, pp. 1426-1438, August 1999.

[8] R. Sivakumar, P. Sinha, V. Bharghavan, "CEDAR : A Core Extraction Distributed Ad Hoc Routing Algorithm ", IEEE JSAC, vol.17, no. 8, pp. 1454-65, Aug. 1999.

[9] R. Dube, C. D. Rais, K-Y Wang, and S. K. Tripathi, "Signal Stability-Based Adaptive Routing (SSA) for Ad Hoc Mobile Networks," *IEEE Personal Communications*, Volume 4, Issue 1, pp. 36 - 45, Feb. 1997.

Appendix B

QDSR Modules

This section contains some important modules that are used in QDSR such as bandwidth calculation, delay and signal strength calculation, and awk script used for analyze the protocol performance.

B.1 Bandwidth Calculation Modules

```
unsigned int DSRAgent::bandwidthCalculation(SRPacket &p, int i)
{
        hdr_sr*srh = hdr_sr::access(p.pkt);
        unsigned int link_BW = 0;
        unsigned int link_bw_i, bandwidth, bandwidth_size;
        unsigned int common_BW = 0;
        unsigned int common_LAST = 0, common_BW_size = 0, remain_BW_size =0;
        unsigned int difference_BW1 = 0, difference_BW2 = 0;
        unsigned int re_diff2 = 0, diff2 = 0, out_bw = 0, bandwidth_re = 0;

        if (i == 1)
        {
                printf("Error!! Never happen\n");
                return srh->req_slot_array_list()[0];
        }
        else if (i == 2)
        {

                link_BW = srh->req_slot_array_list()[i-2] &
                srh >req_slot_array_list()[i-1];
                //printf("link_BW =    %3d: ", link_BW);

                return link_BW;
        }
        else
        {
                //i = i - 1;
                int j = i;

                unsigned int link_bw_i = srh->require_slot_array_list()[i+1] &
                srh->require_slot_array_list()[i];
                bandwidth = bandwidthCalculation(p,i);
                common_BW = bandwidth & link_bw_i;
                common_BW_size = sizebinary(common_BW);

                diff2 = bandwidth ^ common_BW;

                difference_BW1 = sizebinary(link_bw_i ^ common_BW);
                difference_BW2 = sizebinary(bandwidth ^ common_BW);

                if(difference_BW1 <= difference_BW2)
                {
                        bandwidth_size = difference_BW1;
                        remain_BW_size = difference_BW2 - difference_BW1;
                }
                else
                {
                        bandwidth_size = difference_BW2;
                        remain_BW_size = difference_BW1 - difference_BW2;
                }

                if ((remain_BW_size > 0) || (common_BW_size > 0))
                {
                        if (common_BW_size <= remain_BW_size)
                        {
                                bandwidth_size = bandwidth_size + common_BW_size;
                        }
```

```
                        else
                        {
                                bandwidth_size = bandwidth_size + remain_BW_size;
                                common_BW_size = (common_BW_size - remain_BW_size)/2;

                                if (common_BW_size > 0)
                                {
                                        bandwidth_size= bandwidth_size +
                                        common_BW_size;
                                }

                        }
        }

        if ( bandwidth_size >= srh->require_b()) { // compare bandwidth
        if (difference_BW2 >= srh->require_b() ) re_diff2 =
                selectDigit( diff2,srh->require_b());
        else if (difference_BW2 < srh->require_b() ) re_diff2 = diff2 |
                selectDigit(common_BW,fullfil(difference_BW2,srh->require_b()));
        else re_diff2 = selectDigit(common_BW,srh->require_b());
                common_LAST = re_diff2 & link_bw_i;
                out_bw = common_LAST ^ link_bw_i; // might be error here
                bandwidth_re = selectDigit(out_bw,srh->require_b());
        }

        else bandwidth_re = 0; //bn2
                if (p.dest ==net_id) {
                        if (i == 2) srh->require_slot_array_list()[i-1] = re_diff2;
                        srh->require_slot_array_list()[i] = bandwidth_re;
                }
        return bandwidth_re;

        }
}
```

B.2 Slot Reservation Module

```
void DSRAgent::assignedTimeSlotUsed(SRPacket& p,Time todayBandwidth,int nodeLocation)
{
        // if nodelocation is at the first or at the end of a route then just assign
        // else combine incoming and outgoing slot used.
        hdr_sr *srh = hdr_sr::access(p.pkt);
        int location = -1 , remain = -1;
        location = findNodeLocation(p.route.dump());
        if (location == 0) {
                printf("Error assignedTimeSlot %s nodeid %s\n",p.route.dump(),
        net_id.dump());
        goto stop;
        }
        if ( location == 1)
        { // take the first link_bw
        assignedTimeSlotUsedMark(todayBandwidth,stringToInteger(net_id.dump()),
        p.route.BW_path_bandwidth()[location-1]);
        }
        else if ( location == p.route.length() )
        { // take the (location -1 )th link_bw
        assignedTimeSlotUsedMark(todayBandwidth,stringToInteger(net_id.dump()),
        p.route.BW_path_bandwidth()[location-2]);
        }
        else
        { // combine incoming and outgoing slot used.
        unsigned int unionBw = p.route.BW_path_bandwidth()[location-2] |
        p.route.BW_path_bandwidth()[location-1];
        if ((unionBw < 0) || (unionBw >65535 ))
        {
        printf("Dsragent Error union Bw %d node id %s route %s
        ",unionBw,net_id.dump(),
        p.route. dump());
        }
        assignedTimeSlotUsedMark(todayBandwidth,stringToInteger(net_id.dump()),
        unionBw);
        }
        stop: printf("Stop\n");
}
```

B.3 Unsuccessful Reservation

```
void DSRAgent::sendURSV(SRPacket& p){
        hdr_sr *old_srh = hdr_sr::access(p.pkt);
        int location = -1;
        if (p.route.full()) {
                printf("Error sendUrsv \n");
                return;
        }
        SRPacket p_ursv;
        p_ursv.src = net_id;
        p_ursv.dest = p.src;
        p_ursv.pkt = allocpkt() ;
        hdr_ip *new_iph = hdr_ip::access(p_ursv.pkt);
        new_iph->daddr() =
        Address::instance().create_ipaddr(p_ursv.dest.getNSAddr_t(),RT_PORT);
        new_iph->dport() = RT_PORT;
        new_iph->saddr() =
        Address::instance().create_ipaddr(p_ursv.src.getNSAddr_t(),RT_PORT);
        new_iph->sport() = RT_PORT;
        new_iph->ttl() = 255;
        hdr_sr *new_srh = hdr_sr::access(p_ursv.pkt);
        new_srh->init();

        location = findNodeLocation(p.route.dump());
        if ((location == 0 ) || (location > p.route.length()) )
                printf("Error length or node id does not exist in the route path \n");
                p_ursv.route.appendPath_ursv(p.route,0,location);
        for ( int j = 0; j < location ; j++ ) {
                p.route[j].fillSRAddr(new_srh->ursv_addrs()[j]);
                // copy from p.route to a new header
                //printf("dsragent sendURSV that is ok %
                \n",p_ursv.route.path_bandwidth[j]);
        }

        p_ursv.route.length_ursv() = location ; // set length to the route
        new_srh->route_ursv_len() = location; // increament 1
        new_srh->route_ursv_ursv() = 1;
        // this is URSV packet, sent back to release the slots that were assigned
        // propagate the request sequence number in the reply for analysis purposes
        new_srh->rtreq_seq() = old_srh->rtreq_seq();


        hdr_cmn *new_cmnh = hdr_cmn::access(p_ursv.pkt);
        new_cmnh->ptype() = PT_DSR;
        new_cmnh->size() = IP_HDR_LEN;
        p_ursv.route.reverseInPlace();
        p_ursv.route.appendPath_ursv_bw(p.route,0,location);
        for ( int z =1 ; z < location ; z++ ) {
                new_srh->ursv_slot_array_list()[z] = p_ursv.route.path_bandwidth[z-1];
        }

        p_ursv.route.resetIterator();
        p_ursv.route.fillSR(new_srh);
        new_cmnh->size() += new_srh->size();
        for( int m = 0; m < location -1; m++ ) {
                printf("route copy %d route %s\n",p_ursv.route.path_bandwidth[m],
                p_ursv.route.dump());
        }
        double d = 0.0; //Random::uniform(RREQ_JITTER);
        Scheduler::instance().schedule(this,p_ursv.pkt,d);
}
```

B.4 Delay Calculation

```
double DSRAgent::pathDelay(SRPacket &p)
{
        hdr_sr *srh =  hdr_sr::access(p.pkt);

        double path_delay = 0;

        for (int i = 1; i <= srh->num_addrs(); i++)
        {
                path_delay = path_delay + srh->link_info()[i].link_delay_;
        }
```

```
        return path_delay;
}
```

B.5 Signal Strength Calculation

```
void Mac802_11::recv(Packet *p, Handler *h)
{
        hdr_mac *mach = hdr_mac::access(p);
        struct hdr_cmn *hdr = HDR_CMN(p);
        /*
         * Sanity Check
         */
        assert(initialized());
        //printf("%.9f Packet arrive at MAC\n", Scheduler::instance().clock());
        /*
         *  Handle outgoing packets.
         */
        if(hdr->direction() == hdr_cmn::DOWN) {
                //printf("%.9f Send down to physical layer\n",
Scheduler::instance().clock());
                send(p, h);
                return;
        }
        /*
         *  Handle incoming packets.
         *
         *  We just received the 1st bit of a packet on the network
         *  interface.
         *
         */

        /*
         *  If the interface is currently in transmit mode, then
         *  it probably won't even see this packet.  However, the
         *  "air" around me is BUSY so I need to let the packet
         *  proceed.  Just set the error flag in the common header
         *  to that the packet gets thrown away.
         */
        if(tx_active_ && hdr->error() == 0) {
                hdr->error() = 1;
        }

        if(rx_state_ == MAC_IDLE) {
                //printf("%.9f MAC capture packet: Receive power = %e\n",
                //      Scheduler::instance().clock(), p->txinfo_.getRxPr());
                mach->rxPrInfo() =  p->txinfo_.getRxPr();
                setRxState(MAC_RECV);
                pktRx_ = p;
                /*
                 * Schedule the reception of this packet, in
                 * txtime seconds.
                 */
                mhRecv_.start(txtime(p));
        } else {
                /*
                 *  If the power of the incoming packet is smaller than the
                 *  power of the packet currently being received by at least
                 *  the capture threshold, then we ignore the new packet.
                 */
                //printf("MAC received packet: pktRx_->txinfo_.RxPr = %e\n", pktRx_-
>txinfo_.RxPr);
                if(pktRx_->txinfo_.RxPr / p->txinfo_.RxPr >= p->txinfo_.CPThresh) {

                        capture(p);
                } else {
                        collision(p);
                }
        }
}

double DSRAgent::pathSignalMin(SRPacket &p)
{
        hdr_sr *srh =  hdr_sr::access(p.pkt);
        double min_SL = srh->link_info()[1].link_signal_ ;
        //initial signal available = link signal
```

```
        for (int i = 1; i <= srh->num_addrs(); i++)
        {
                if (srh->link_info()[i].link_signal_ < min_SL)
                {
                        min_SL = srh->link_info()[i].link_signal_;
                }
        }
        return min_SL;
}
```

B.6 Goal Function

```
double
DSRAgent::goalFunction(SRPacket &p)
{
        return (((d - pathDelay(p))/d) + ((pathSignalMin(p)/(Rt))));
}
```

B.7 Awk Script

performance_metric.awk

```
BEGIN {
        sends=0;
        recvs=0;
        routing_packets=0.0;
        droppedBytes=0;
        droppedPackets=0;
        highest_packet_id =0;
        sum=0;
        recvnum=0;
        startTime = 1e6;
        stopTime = 0;
     }

  {
  time = $3;
  packet_id = $41;

  # CALCULATE PACKET DELIVERY FRACTION
  if (( $1 == "s") &&  ( $35 == "cbr" ) && ( $19=="AGT" )) {  sends++;
  }

  if (( $1 == "r") &&  ( $35 == "cbr" ) && ( $19=="AGT" ))   {
  recvs++; }

  # STORE START TIME
  if ($19 == "AGT" && ($1 == "+" || $1 == "s") && $37 >= 512)
  {
    if ($3 < startTime)
    {
        startTime = $3
    }
  }

  # STORE STOP TIME
  if ($19 == "AGT" && $1 == "r" && $37 >= 512)
  {
    if ($3 > stopTime)
    {
        stopTime = $3
    }
  }

  # CALCULATE DELAY
  if ( start_time[packet_id] == 0 )  start_time[packet_id] = time;
  if (( $1 == "r") &&  ( $35 == "cbr" ) && ( $19=="AGT" )) {
  end_time[packet_id] = time; }
        else {  end_time[packet_id] = -1; }

  # CALCULATE TOTAL DSR OVERHEAD
```

```
 if (($1 == "s" || $1 == "f") && $19 == "RTR" && $35 =="DSR")
routing_packets++;

 # DROPPED DSR PACKETS
 if (( $1 == "d" ) && ( $35 == "cbr" )  && ( $3 > 0 ))
      {
            droppedBytes=droppedBytes+$37;
            droppedPackets=droppedPackets+1;
      }

      #find the number of packets in the simulation
         if (packet_id > highest_packet_id)
            highest_packet_id = packet_id;

 }

 END {

 for ( i in end_time )
 {
 start = start_time[i];
 end = end_time[i];
 packet_duration = end - start;
 if ( packet_duration > 0 )
 {    sum += packet_duration;
      recvnum++;
 }
 }

    delay=sum/recvnum;
    simTime = stopTime-startTime;
    NRL = routing_packets/recvs;  #normalized routing load
    NRO = routing_packets/simTime;
    PDF = (recvs/sends)*100;  #packet delivery ratio[fraction]
    PDR = recvs/simTime;
    PLR = droppedPackets/simTime;
    printf("send (pkts) = %.2f\n",sends);
    printf("recv (pkts) = %.2f\n",recvs);
    printf("routingpkts = %.2f\n",routing_packets++);
    #printf("data packets  = %d\n",highest_packet_id-(--routing_packets));
    #printf("highest_packet_id = %d\n",highest_packet_id);
    printf("start time = %.2f\n", startTime);
    printf("stop time = %.2f\n", stopTime)
    printf("simulation time = %.2f\n", simTime)
    printf("PDF  = %.2f\n",PDF);
    printf("PDR  = %.2f pkts/sec\n",PDR);
    printf("NRL  = %.5f\n",NRL);
    printf("NRO  = %.5f rt_pkts/sec\n",NRO);
    printf("PLR  = %.5f pkts/sec\n",PLR);
    printf("Average e-e delay(ms)          = %.6f\n",delay*1000);
    printf("No. of dropped data (packets) = %d\n",droppedPackets);
    printf("No. of dropped data (bytes)    = %d\n",droppedBytes);

 }
```

throughput.awk

```
BEGIN {
      recvdSize = 0
      startTime = 1e6
      stopTime = 0
 }

 {
      # Trace line format: normal
      if ($2 != "-t") {
            event = $1
            time = $2
            if (event == "+" || event == "-") node_id = $3
            if (event == "r" || event == "d") node_id = $4
            flow_id = $8
            pkt_id = $12
            pkt_size = $6
            flow_t = $5
            level = "AGT"
      }
      # Trace line format: new
      if ($2 == "-t") {
```

```
            event = $1
            time = $3
            node_id = $5
            flow_id = $39
            pkt_id = $41
            pkt_size = $37
            flow_t = $45
            level = $19
      }

  # Store start time
  if (level == "AGT" && (event == "+" || event == "s") && pkt_size >= 512) {
    if (time < startTime) {
            startTime = time
            }
      }

  # Update total received packets' size and store packets arrival time
  if (level == "AGT" && event == "r" && pkt_size >= 512) {
      if (time > stopTime) {
            stopTime = time
            }
      # Rip off the header
      hdr_size = pkt_size % 512
      pkt_size -= hdr_size
      # Store received packet's size
      recvdSize += pkt_size
      }
  }

  END {
      printf("Average Throughput[kbps] = %.2f\n", (recvdSize/(stopTime-
startTime))*(8/1000));
      printf("StartTime = %.2f\n", startTimet);
      printf("StopTime  = %.2f\n", stopTime);
      printf("RecvdSize = %d bytes\n", recvdSize);
  }
```

Author's Biography

Mr. I Wayan Mustika was born on September 21, 1981 in Denpasar, Indonesia. In 2005, he received B.E. degree (*cum-laude*) in electrical engineering from the department of electrical engineering, faculty of engineering, Gadjah Mada University, Indonesia. From year 2006, he was a lecturer in the department of electrical engineering, Gadjah Mada University, Indonesia. Meanwhile, in the same year, he was awarded AUN/SEED-Net scholarship which is supported by Japan International Cooperation Agency (JICA), to pursue his master degree at King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand. He is currently a member of Communication Network Laboratory at Research Center for Communications and Information Technology (ReCCIT), KMITL. His research interests include wireless local area networks, wireless security and wireless ad hoc networks.