

QUALITY OF SERVICE AWARE  
SOURCE INITIATED  
AD-HOC ROUTING

By  
KNUT-HELGE VIK

A thesis submitted in partial fulfillment of  
the requirements for the degree of

MASTER OF SCIENCE

WASHINGTON STATE UNIVERSITY  
Department of Computer Science

MAY 2004

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis  
of KNUT-HELGE VIK find it satisfactory and recommend that it be accepted.

---

Chair

---

---

## ACKNOWLEDGEMENT

I would like to acknowledge my advisor Sirisha Medidi, for all her help and valuable input. I would also like to acknowledge all the members of the Washington State Networking and Security Research Group.

I would not have been able to complete my Master's degree without the help of my fellow students. Thank you all for your support.

Quality of Service aware Source initiated

Ad-hoc Routing

Abstract

by Knut-Helge Vik, M.S.

Washington State University

May 2004

Chair: Sirisha Medidi

Mobil Ad-hoc networks (MANETs) are used in a variety of situations ranging from conference meetings to military operations.

Among the important challenges facing MANETs is the development of suitable routing protocols. A Routing Protocol must be aware of the diversity in the network if the network is going to perform at its best. The routing protocol should be efficient and able to adapt to rapidly changing topologies. A MANET is applicable to any situation requiring network communication in particular short-lived highly mobile networks. The routing protocol needs mechanisms that handle the mobility such that the communication disruption time is minimized. Specifically we desire a routing protocol with quality control mechanisms for every important stage in the lifetime of a route.

This thesis proposes a Quality of Service (QoS) aware source initiated ad-hoc routing protocol (QuaSAR) that adds quality control to all the phases of an on-demand routing protocol. QuaSAR gathers statistical information from the network during route discovery, more specifically battery power, signal strength, bandwidth and latency. In particular we use the signal strength to find stronger connected routes. The metrics are associated to the individual route and later used when a route to the destination is picked. QuaSAR

includes proactive route maintenance features in addition to the reactive maintenance. The proactive mechanism of QuaSAR makes a Mobile Node (MN) on an active route aware of route critical incidents in progress and notifies the appropriate party of the development.

Performance experiments were conducted to test the performance of QuaSAR using ns-2 Network Simulator. Results were compared with Dynamic Source Routing (DSR). The following mobility models were used in the experiments: Random Waypoint, Gauss Markov, Manhattan Grid and Reference Point Group Model.

We have demonstrated that using signal strength and proactive mechanisms as a means of routing will increase the throughput and the packet delivery ratio. The quality control mechanisms of QuaSAR enhance the performance experienced by MNs by giving it more stable paths thus minimizing the communication disruption time.

# Contents

<b>Acknowledgement</b>	<b>iii</b>
<b>Abstract</b>	<b>iv</b>
<b>List of Tables</b>	<b>viii</b>
<b>List of Figures</b>	<b>x</b>
<b>List of Algorithms</b>	<b>xii</b>
<b>Introduction</b>	<b>1</b>
<b>1 Wireless Networks</b>	<b>4</b>
1.1 Routing Protocols . . . . .	5
1.1.1 MANET . . . . .	6
1.1.2 MANET Requirements . . . . .	8
1.2 Quality of Service . . . . .	10
1.2.1 Routing . . . . .	11
1.2.2 MANET . . . . .	14
1.3 Related Work . . . . .	15
1.4 On-Demand Routing for MANETs . . . . .	18
1.4.1 Route Discovery . . . . .	18
1.4.2 Route Choosing . . . . .	20
1.4.3 Route Maintenance . . . . .	22

<b>2</b>	<b>Quality of Service aware Source initiated Ad-hoc Routing</b>	<b>25</b>
2.1	Route Discovery . . . . .	26
2.1.1	QoS Route Request . . . . .	28
2.1.2	QoS Route Reply . . . . .	32
2.2	Route Choosing . . . . .	34
2.3	Route Maintenance . . . . .	38
2.3.1	Proactive . . . . .	38
2.3.2	Reactive . . . . .	44
<b>3</b>	<b>Simulation Experiments</b>	<b>46</b>
3.1	Fine Grain . . . . .	48
3.1.1	Setup . . . . .	48
3.1.2	Results . . . . .	51
3.1.3	Analysis . . . . .	54
3.2	Course Grain . . . . .	56
3.2.1	Setup . . . . .	56
3.2.2	Results . . . . .	61
3.2.3	Analysis . . . . .	74
<b>4</b>	<b>Conclusions and Future Work</b>	<b>77</b>
	<b>Bibliography</b>	<b>80</b>

*I dedicate this thesis to my family,  
who has supported me all these years.*



# List of Tables

3.1	Simulation Metrics . . . . .	47
3.2	Fine Grain Simulation Results . . . . .	54
3.3	Course Grain Mobility Categories . . . . .	56
3.4	Course Grain Mobile Node QoS . . . . .	57
3.5	Course Grain Communication Patterns . . . . .	58

# List of Figures

1.1	Routing in DSR and AODV . . . . .	8
2.1	Transmission range classification . . . . .	28
2.2	Circular problem with Re-Broadcasting . . . . .	31
2.3	Destination Dropping QRREPs that have length $> \{\text{minLength} * 2 + 1\}$ . . . . .	33
3.1	Fine Grain Simulation Environment . . . . .	50
3.2	Fine Grain: Throughput . . . . .	51
3.3	Fine Grain: Mean Latency/Packet . . . . .	52
3.4	Fine Grain: Packets/Route . . . . .	52
3.5	Fine Grain: Packet Delivery Ratio . . . . .	53
3.6	Fine Grain: Protocol Packets/Delivered Packet . . . . .	53
3.7	Fine Grain: Protocol Packets Sent/Delivered Packet . . . . .	54
3.8	The Node density in the Course Grain Simulation is at most 21 neighbors with uniform distribution . . . . .	58
3.9	Random Waypoint: Throughput . . . . .	61
3.10	Random Waypoint: Mean Latency/Packet . . . . .	62
3.11	Random Waypoint: Packets/Route . . . . .	62
3.12	Random Waypoint: Packet Delivery Ratio . . . . .	63
3.13	Random Waypoint: Protocol Packets/Delivered Packet . . . . .	63
3.14	Random Waypoint: Protocol Packets Sent/Delivered Packet . . . . .	64

3.15 Gauss Markov: Throughput . . . . .	64
3.16 Gauss Markov: Mean Latency/Packet . . . . .	65
3.18 Gauss Markov: Packet Delivery Ratio . . . . .	65
3.17 Gauss Markov: Packets/Route . . . . .	66
3.19 Gauss Markov: Protocol Packets/Delivered Packet . . . . .	66
3.20 Gauss Markov: Protocol Packets Sent/Delivered Packet . . . . .	67
3.21 Manhattan Grid: Throughput . . . . .	67
3.22 Manhattan Grid: Mean Latency/Packet . . . . .	68
3.23 Manhattan Grid: Packets/Route . . . . .	68
3.24 Manhattan Grid: Packet Delivery Ratio . . . . .	69
3.25 Manhattan Grid: Protocol Packets/Delivered Packet . . . . .	69
3.26 Manhattan Grid: Protocol Packets Sent/Delivered Packet . . . . .	70
3.27 Reference Point Group: Throughput . . . . .	71
3.28 Reference Point Group: Mean Latency/Packet . . . . .	71
3.29 Reference Point Group: Packets/Route . . . . .	72
3.30 Reference Point Group: Packet Delivery Ratio . . . . .	72
3.31 Reference Point Group: Protocol Packets/Delivered Packet . . . . .	73
3.32 Reference Point Group: Protocol Packets Sent/Delivered Packet . . . . .	73

# List of Algorithms

1	QRREQ-Source Nodes: Broadcasting QRREQ . . . . .	29
2	QRREQ-Intermediate Nodes: Receiving QRREQ . . . . .	31
3	QRREP-Destination Nodes: Receiving QRREQ . . . . .	33
4	Route Choosing: QoS Route . . . . .	36
5	Route Choosing: QoS Route, Accept Lowest QoS . . . . .	36
6	Route Choosing: Best-Fit QoS Route, Accept Lowest QoS . . . . .	37
7	RCR: Originating Nodes . . . . .	40
8	RCR: Preempt Path . . . . .	43
9	RCR: Intermediate Nodes . . . . .	43
10	RCR: Source/Destination Nodes . . . . .	45

# Introduction

The first wireless LAN came together in 1971 when networking technologies met radio communications at the University of Hawaii as a research project called ALOHAnet [2]. However it wasn't until the introduction of the Internet that wireless technology started to become a serious alternative to wired Local Area Networks (LAN) [28]. The technology had been unreliable and expensive but started to gain trust. With the increasing popularity came also the importance of the development of Wireless Routing Protocols. The introduction of wireless networking introduced a new set of opportunities since users can move around without a physical network cable attached to the Mobile Node. This mobility is desirable but must be handled by appropriate protocols.

The diversity of Mobile Nodes available today creates opportunities and challenges. The opportunities seem to be limitless as even the weaker mobile nodes are powerful enough to handle heavy computational processing. However as the weaker are getting stronger the stronger are getting even more powerful. The day when all mobile nodes are created equal may happen but until then we must be aware of the diversity and the challenges they pose.

Applications have different features facilitating them to perform their mission as a computer processor. These features have requirements to their surroundings such as hardware, operating system and network performance. Hypertext Transfer Protocol (HTTP) [19], File Transfer Protocol (FTP) [40] and real-time protocols [16] all have different requirements in terms of reliability and speed. To be able to offer these application protocols the different

features they demand the lower layers need Quality of Service control [6].

Quality of Service is not a new term in Computer Science but it wasn't given much attention before the major offspring of the Internet began in the early nineties [55]. Before the Internet era began Quality of Service hadn't been a serious issue in computer networks and had typically been handled simply by adding more resources when the service became unsatisfactory. However this was not a viable solution for handling Quality of Service degradation in the increasingly popular Internet. Following the network boom the Wireless technology became cheaper and subsequently popular. The network protocols were not designed with Quality of Service in mind and needed upgrading.

The main goal of any communication whether it is lingual or binary is to diminish the communication disruption time. Two persons talking to each other don't want to repeat every other sentence as they couldn't understand everything the other said because of external events like noise. We can apply this analogy to two computers communicating with each other in a network. They don't want to waste energy on resending information because of events like bit errors or packet drops. There are solutions to every problem, not necessarily perfect ones but solutions that will diminish the hitches. The two persons can for example talk louder, move indoors, or communicate by using their hands. The two computers can get a better connection by moving closer, but unlike humans the computers can't think and therefore needs carefully designed protocols that handle the faults that eventually will happen.

The chapters of this thesis are organized as follows. In Chapter 1 we introduce wireless networks, in particular ad-hoc networks. Next the concept of Quality of Service is explained and applied to routing protocols. The focus of this thesis is Quality of Service aware Mobile Ad-hoc Network (MANET) Routing Protocols and in the related work section we present work within this area of research. The basis for our proposed MANET routing protocol is on-demand source routing, which has particular implications that are explained in detail.

In Chapter 2 our proposed routing protocol is presented with illustrating pictures and clarifying algorithms. Then in Chapter 3 we give results from simulation experiments we performed, and an analysis that highlights important aspects of the results. Chapter 4 comprises conclusions to our work and highlights the challenges we face within the Quality of Service aware MANET Routing Protocols research area.

# Chapter 1

## Wireless Networks

Wireless networks can be applied to a wide variety of situations. The military has become dependant on wireless communication. A lost backpacker is happy if there is a wireless connection. Other wireless networks can be there as a service for example at a coffee shop. There are however in essence two types of wireless networks:

- **Infrastructure based:** Includes designated communication points that are interconnected with each other.
- **Infrastructure free:** Has no designated communication points.

Infrastructure based wireless networks [24] have Base Stations that connects the Mobile User to a network and as in wired networks the routing is done by designated stationary routers. The problems occur when a Mobile Node (MN) moves out of range of the Base Station it originally connected to. In Mobile IP [38] this Base Station is referred to as the Home Agent. The Home Agent will try to detect that a MN is moving out of range and tries to hand it off to a different Base Station. It is important that a handoff happens seamless for the client operating the MN. This is not possible in all cases i.e. if a MN is moving too fast for the Base Station to detect the signal strength degradation. The infrastructure offers a certainty that a MN will be able to communicate as long as there are available



Base Stations within signal range. It is the communication disruption time that has to be diminished.

Infrastructure free networks do not have designated stationary routers. Mobile Ad-hoc Networks (MANET) is a type of wireless network without an infrastructure [29]. The MNs in a MANET operate as both a client and a router, which immediately assumes that every MN has this capability. Instead of having Base Stations with only one important task MANETs introduces every MN as a potential router. Packets sent through the network will be forwarded to the destination by MNs attached to the MANET. It is very likely that communication disruption will happen in both infrastructure based and infrastructure free networks. However unlike infrastructure networks the mobility of the MNs becomes an important factor in MANETs. Route discovery and choosing is of high importance and must handle a number of situations i.e. high mobility. A MANET is an appealing network but it needs protocols that are able to handle the events that will occur.

## 1.1 Routing Protocols

The opportunities in mobile networks have introduced a new set of issues that needs to be handled by carefully developed protocols. The most important network specific protocols in the Open Systems Interconnection (OSI) Model [1] are the Media Access Control (MAC) Layer and the Network Layer.

The main task for the MAC Layer is to govern the air and ensure that as few as possible collisions occur. The de facto standard is for the initiator to send a Request-To-Send (RTS) packet to the MN it wants to communicate with, this MN responds with Clear-To-Send (CTS) packet. To avoid collisions the CTS includes a back-off time, which holds off the neighbors from transmitting in this timeslot. There are known issues the MAC Layer has to deal with for example the hidden terminal problem [21].

The Network Layer is often referred to as the Routing Protocol and we will continue to do this. The Routing Protocol handles routing related actions. It may for example discover and choose a route to start sending packets through. Once a route is found the communication can start. When a route breaks it is the routing protocol that handles the chain of events following for example notifying the source. The Routing Protocol in Infrastructure based mobile networks and MANETs are very different. A MANET places additional responsibility on each MN since they need the ability to function as a router. The Router hierarchy in infrastructure based mobile networks i.e. Mobile IP handles all the routing and adds complexity to the infrastructure rather than the MNs.

### 1.1.1 MANET

The two main categories of MANET Routing Protocols are as follows [44]:

- Table driven Routing Protocols
- Source initiated on demand driven Routing Protocols (SIODR-protocols)

Table driven Routing Protocols attempt to keep the routing information on each node in the network up to date. This indicates that every change detected in the network is advertised and propagated throughout the network to maintain a consistent network view [39, 33]. The main disadvantage of table driven protocols is the overhead inflicted to the network in order to keep the tables up to date. During high mobility the topology will change very fast in terms of communication range. A table driven protocol seeks to keep the current view of the network and typically broadcasts updates throughout the network. When the network size increases this becomes a problem. The network is potentially swamped with Routing Protocol packets decreasing the network performance. The overhead is apparent as all MNs, active and inactive, must update their routing table and store routes to every MN in the network even though it might never use any of the routes. MNs typically have limited

battery power and will not be able to enter sleep mode if the routing protocol constantly needs to perform Routing Protocol administration i.e. table updates.

SIODR-protocols do not attempt to maintain information about the entire network. Instead routes are found when a Mobile Node wants to communicate with a node in the network. This system increases the route establishment time but it will save the network from Routing Protocol packets when Mobile Nodes are dormant in terms of network activity. SIODR-protocols are more scalable than table driven routing protocols. The most popular SIODR-protocols are Dynamic Source Routing (DSR) [25], Ad-hoc On-Demand Distance Vector Routing (AODV) [37], Temporally-Ordered Routing Algorithm (TORA) [36], Associativity-Based Routing (ABR) [48] and Signal Stability Adaptive Routing (SSA) [18].

DSR uses source routing, which means the source knows the entire route to the destination. AODV uses hop-by-hop routing where the nodes only save the previous and next hop information on any route. TORA's routing is based on the assumption of a global positioning system (GPS) [35] present. ABR and SSA use signal strength as the basis for routing.

Variants of table driven- and SIODR-protocols exist i.e. Cluster based Routing Protocol (CBRP) [30], which is a hybrid between the two. Typically a Cluster Routing Protocol divides the network into groups and appoints one MN as the Master and the others as Slaves for each group. It is very similar to the idea of the routing in Bluetooth networks [23] with Pico nets and Master/Slave. The Master is only connected to other Masters and is in charge of forwarding all packets within and between clusters. There are several variants and optimizations of this outline. The idea is to avoid the scalability problem of table driven protocols and the route establishment time issue of SIODR-protocols. However CBRP make fairly strong assumptions for this to work in practice. There must be MNs that are powerful enough to be in charge of forwarding all the packets. Every Master is

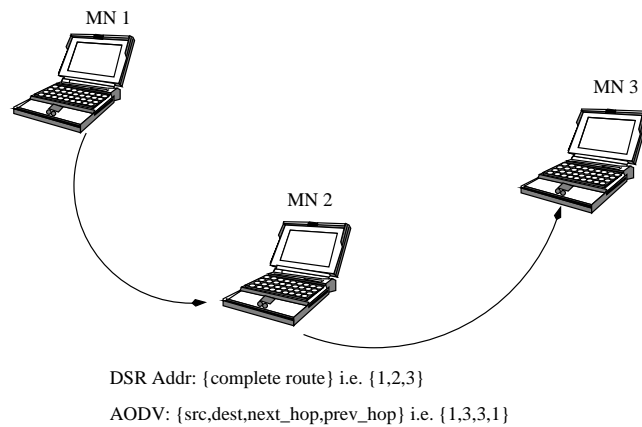


Figure 1.1: Routing in DSR and AODV

within transmission range with at least one other Master. Because of the diversity of MNs these assumptions do not apply all the time. Adding clusters and appointing different tasks transforms the MANET into a new type of network that is semi infrastructure free. This approach deals with the problems of routing differently than a totally infrastructure free environment.

A MN should not have to store large amounts of routing information it will never use as in table driven protocols. The assumptions made in CBRP are strong and limits the applicability. Instead the individualism in SIODR-protocols prevents the scalability issues table driven protocols have, moreover SIODR does not assign specific tasks to any MN in the ad-hoc network.

### 1.1.2 MANET Requirements

The infrastructure free nature of MANETs implies that it is more susceptible to route errors, packet corruption and packet drops. It is therefore vital to develop routing protocols that preserves the deployment capabilities and at the same time diminishes the route and packet errors. The design of routing protocols requires the routing protocol to be efficient and

adaptive to rapidly changing topologies. The Routing Protocol must have quality control during all the important phases of route establishment. For this to be worth the effort the quality control should not increase the route establishment time significantly. The quality control design must be done carefully with tradeoffs between quality control overhead and efficiency. In addition to route establishment quality control there has to be extensive route maintenance. The lifetime of routes in MANETs decreases as the mobility of MNs increases making efficient Route Error handling important. MNs must notify the source of a broken link. This is important for route cache purposes and vital when the transmission protocol has no flow control i.e. User Datagram Protocol (UDP) [46]. There should be a quality control that helps disregard routes that are highly unreliable. Before a route error occurs there may be events that can be used in order to predict route breaks. The Routing Protocol should aim to preempt route breaks when route critical incidents happen by using proactive messaging.

The Network Layer stores the routes and the information related to each route in its route cache. Since the routes are stored locally it is vital that the information is updated when incidents are detected. For table based Routing Protocols the cache is updated whenever a topology change is detected. There must also be mechanisms in SIOBR-protocols that invalidate routes that contain dead links. If there are outdated routes they must be updated and possibly completely removed from the cache.

In [10] a performance comparison between DSDV, TORA, DSR and AODV was presented. The results showed that DSR had a higher packet delivery rate than the other protocols. They also tested the protocol overhead to the network and once again DSR outperformed the other protocols. DSDV that uses a table driven routing protocol had constant protocol overhead but the packet delivery decreased as the mobility increased. The results in [10] suggest that table driven protocols have to be redesigned in order to work well in MANETs. TORA and AODV are SIOBR-protocols and had fairly high delivery

rates but during when the mobility increased the routing protocol overhead became very high.

## 1.2 Quality of Service

Quality of Service (QoS) is the collective term used when talking about quality control of any system. In computer networks QoS involves adding mechanisms to control the network activity and a prediction of how routes will perform based on previously gathered statistics. Important principles, specifications and mechanisms need to be addressed during system design for it to have QoS that works satisfactory. The QoS needs to be implemented carefully. Some key QoS principles and concepts are listed below [6]:

**Transparency principle:** Applications should be shielded from the complexity of underlying QoS management.

**Flow performance Specification:** Categorizes the flow performance requirements of the user i.e. throughput rate, latency, jitter and loss rate.

**Level of service:** Specifies the degree of end-to-end resource commitment required i.e. deterministic, predictive and best effort [45].

**Cost of service:** Addresses what the user is willing to pay to get the QoS it demands. If the service comes with no cost the worst-case scenario is that everybody will ask for the best possible service.

**QoS Mapping:** This mechanism performs the translation of the QoS between the system layers, i.e. Application Layer, Transport Layer and Network Layer.

**Flow Construction:** The flow discovery based on the flow performance metrics.

In addition to concepts and principles there has to be mechanisms that enforces and maintain the initial frame that were sketched. The mechanisms have to be chosen depending

on the QoS system that is currently being developed. Important QoS mechanisms are listed here:

**QoS monitoring:** The levels of a system may track the QoS achieved by lower levels [11].

**QoS maintenance:** Compares the monitored QoS against the expected performance and then exerts tuning operations on resource modules to sustain the delivered QoS [13].

**QoS degradation:** Indicates to the upper layer that the lower layers have failed to acquire or maintain the demanded QoS for the flow. The application can try to adapt to the QoS or reduce the QoS [13] i.e. renegotiate the QoS with the destination.

**QoS availability:** Allows an application to specify how often and which QoS parameters it wants feedback about from the QoS monitoring mechanism.

We refer the reader to [6] for further study of QoS Architecture.

### 1.2.1 Routing

Quality of Service routing [52] has been a serious issue for some time but got more attention when the Internet grew to unimaginable sizes in the mid-nineties. With the rise of the Internet came applications that took advantage of this network and had different QoS requirements that needed to be dealt with. QoS aware communication is a necessity if these applications are to be given what they require. Without QoS information there cannot be route prioritizing and the service the application gets is random. QoS cannot be handled on one level alone; it has to be incorporated into several layers where each layer has specific QoS related tasks. The Network Layer plays a vital part in QoS routing since it is on this layer the route information is stored for later to be used in the route choosing algorithm.

Each layer in the OSI model [1] has very different but at the same time specific tasks that facilitates the collaboration between them. The QoS should therefore not try to create an

entirely new layer but rather be an extension to the logic of the existing layers. For example the MAC Layer should handle all media access related QoS issues and the Network Layer should handle route QoS.

Several levels of QoS guarantees can be given but in general they can be summarized to guaranteed service and best effort [6]. Guaranteed service protocols aim to give the source complete control of the offered QoS it is getting. These protocols can give end-to-end guarantees to flows and the source will be notified if the QoS needs to be renegotiated. Best effort protocols do not offer end-to-end guarantees but provides routes with no commitment to bandwidth and latency.

The MAC Layer should take care of signal and link related calculations. Bandwidth and signal strength are two QoS metrics that are important in terms of speed but also route reliability. The Network Layer is thus interested in knowing how the one-hop situation is from the MAC Layer. The Network Layer takes care of the routing of packets to the destination and the MAC Layer considers its neighbors within transmission range. The delegation of tasks in the OSI model helps us to identify which QoS issues should be handled where. The Network Layer must receive route sensitive information from lower layers in order to be fully able to choose better routes.

Integrated Service (IntServ) [53] is a state full Routing Protocol that requires routers to manage per flow states and per flow operations. IntServ provides end-to-end guarantees or controlled load service on a per flow basis. It is not scalable, as it requires substantial memory from the routers. Differentiated Service (DiffServ) is a stateless approach that generalizes the QoS demands to a small amount of traffic classes. DiffServ also gives end-to-end guarantees but they are limited to the traffic class the flow fits in. IntServ uses a signaling protocol like Resource Reservation Protocol (RSVP) to establish resource reservation state on all the routers in the path. DiffServ uses the notion of edge and core routers. Edge



routers process packets based on finer traffic granularity and core routers process packets based on a small number of Per Hop Behaviors (PHB) encoded by bit patterns in the packet header using a priority-like scheduling and buffering mechanism.

The Internet uses best effort routing as a default. Due to the size of the Internet and the technology currently available the QoS guarantees can't be stronger. Offering all Internet applications end-to-end guarantees would require that the Internet had the available resources for this. However the diversity of the nodes attached and the bandwidth capabilities these have makes this a very hard problem to solve.

The QoS constraints were divided into two categories in [6] : path constraints and tree constraints. Path constraints needs to be satisfied from the sender to the receiver while tree constraints must be satisfied over the entire multicast distribution tree created by the multicast Routing Protocol from the sender(s) to the receivers.

Next we introduce the core QoS metrics:

- **Latency:** Per packet delay that is required
- **Bandwidth:** Link speed requirements
- **Jitter:** Variation in latency that can be tolerated
- **Packet loss:** Minimum number of packet drops accepted

The composition rules that are used for QoS metrics depend on the nature of each metric. Additive metrics are i.e. latency, jitter, hop-count, logarithm of successful transmission probability and cost. Computational metrics can be 1-loss probability (successful transmission probability). Concave/minmax are i.e. bandwidth and in mobile networks battery and signal strength. Additive and computational metrics are calculated and added to on each hop on the route. Concave/minmax metrics are the minimum/maximum of the metric on

the route and is a guideline to the strength of the route. In other words a route is only as strong as its weakest link, i.e. if a route has one hop with significantly lower bandwidth than the bit rate it will cause packet loss. This is often referred to as bottleneck bandwidth [27].

It was proven by Wang and Crowcroft that finding the best path subject to two or more additive/multiplicative metrics is an NP-Complete [52] problem. Using any two or more of delay, delay jitter, cost or loss probability is NP-Complete. The only feasible combination is bandwidth and one of the additive/multiplicative metrics. Of the additive/multiplicative metrics delay is the most important one. Even though delay jitter, cost and loss probability are suited for certain environments, most applications will benefit more from using the delay metric.

### **1.2.2 MANET**

Depending on the nature of the network the level of QoS guarantee has to be chosen carefully. LAN is a network that can handle end-to-end QoS guarantees since the resource management is restricted to giving a limited amount of users their QoS demands. MANET on the other hand is not an easy network to administer. The network nature implies that a wide variety of MNs will connect and disconnect throughout the lifetime of the MANET. The mobility and the diversity of MNs make resource reservation in MANETs a hard problem. MNs are both clients and routers and have to be able to store all the routing information locally. When a MANET grows this may become a problem for the weaker nodes in the network. Instead a best effort approach fits better for MANETs. MNs should have the ability to choose between QoS attributed routes. The statistics associated to each route is not a guarantee but an image of the route from the previous update. They represent a prediction to how the service is. This approach increases the responsibility of the Routing Protocol and its ability to keep the MNs route cache up to date.

### 1.3 Related Work

There have been many attempts to incorporate QoS to an existing MANET Routing Protocol. To our knowledge none of the existing approaches have been implemented. When we mention test results we are only referring to preliminary results. Some of the more important work will be presented in this section. A brief description of each is given.

**A Flexible QoS Model for MANETs (FQMM):** FQMM [54] adopted the idea of DiffServ to MANETs. It is designed for small to medium sized MANETs, with fewer than 50 nodes and using a flat non-hierarchical topology. FQMM defines three types of nodes as in DiffServ; Ingress, interior and egress nodes. An ingress node is the source sending data. Interior nodes are the forwarding nodes. An egress node is the destination. As in DiffServ the QoS is mapped to Per Hop Behaviors (PHB) from the ingress node and forwarded according to these by the interior nodes. FQMM propose a hybrid between per-class and per-flow provisioning. The highest priority traffic is given per-flow provisioning while the other traffic types are given per-class provisioning. FQMM is a good framework but assumes that the topology is known to everybody and does not consider mobility to a great extent. The identified problem of scalability is also a negative.

**A Distributed Quality-of-Service Routing in Ad-Hoc Networks:** In [14] they propose a distributed QoS routing scheme that selects a network path with sufficient resources to satisfy a certain delay or bandwidth requirement in a dynamic multi-hop mobile environment. They use multi path parallel route discovery instead of flooding the network and assumes that distance-vector routing is used [33]. They introduce fault tolerance mechanisms that shifts traffic to neighbor nodes when the QoS degrades and reconfigures the path around the breaking point rather than using an entirely new path. Since [14] uses distance-vector Routing Protocol it is not a very

scalable approach and with high mobility it will lead to massive overhead.

**Ad hoc QoS on-demand routing (AQOR) in MANETs:** AQOR [56] is a resource reservation and signaling algorithm. AQOR provides end-to-end QoS support in terms of bandwidth and end-to-end delay in MANETs. They introduce detailed computation algorithms for available bandwidth calculation and end-to-end delay in an unsynchronized wireless environment. The wireless channel is a shared medium and can only be used one at a time by the nodes within transmission range. The bandwidth calculation is based on the aggregate traffic of the neighborhood and is performed on the MAC Layer. AQOR proposes to use HELLO-packets [32] to keep an updated view of the neighborhood. It reserves bandwidth on each node along a path that is being used by the source. The reservation has been done in the route discovery phase but doesn't actually take place until the first packet has been forwarded at a node. AQOR proposes an adaptive route recovery model when a QoS violation has been detected. This model makes the destination do a reverse route exploration. The bandwidth calculation and resource reservation model in AQOR showed promising results.

**Optimized Link State Routing (OLSR) for MANETs:** OLSR is a proactive QoS routing protocol [5]. It uses a table driven link state routing protocol and thus inherits the stability it exhibits and the advantage of having routes available when needed. OLSR exchanges topology information with other nodes in the network regularly. They propose to use Multi Point Relays (MPR), which are selected nodes that forward broadcast messages during the flooding of topology information. MPR nodes are selected based on its position to the neighbors and transmission range. The idea is to use nodes that are within transmission range with the most neighbors and this way avoid that all the nodes in the network take part in the forwarding of topology updates. They add QoS extensions to the messages used during neighbor discovery.

OLSR uses an end-to-end bandwidth calculation proposed in [58] in order to find the minimum bandwidth on a route. Each node stores the minimum bandwidth and maximum delay in its routing table. They use one-way-delay and assume that they have a global timing structure that makes them able to use this delay information with a degree of certainty. OLSR applies an admission control to the incoming traffic in each MPR node. The admission control analyzes the available bandwidth to allow the selection of an MPR by a new node. They use HELLO Message format [32] with a willingness field, which indicates how willing a MN is to be a MPR point. OLSR uses QoS calculations that are well suited and optimizes Link State Routing Protocol with MPR. OLSR will as many other MANET Routing Protocols perform well with low mobility. However using table driven Routing Protocols in MANETs will cause substantial overhead. The MPR selection is susceptible to failure when the mobility increases. It suggests that the MPRs will have neighbors that are just inside transmission range, which makes it a very fragile approach.

**On-demand Link State multi-path QoS routing in a Wireless MANET:** In [15] they propose a protocol that reactively collects link-state information from source to destination in order to dynamically construct a partial network topology. They use the CDMA/TDMA channel model [42] to find routes that satisfies the QoS in terms of bandwidth specified by the source and gives it end-to-end QoS guarantees. They assume that a mobile node knows the available bandwidth to all its neighbors. When a source needs a QoS route to the destination it sends out Route Request (RREQ) packets to the neighbors who can satisfy the bandwidth constraint. The RREQ is propagated to the destination. While propagated each RREQ packet records the link-state information from source to destination. The destination will take as many RREQs it can and perform calculations to find the best multi path that satisfies the

bandwidth back to the source. It then sends a reply back and the resources are reserved on the way back to the source. The protocol provides interesting ideas in terms of bandwidth calculation and a multi-path route to the destination. Using a link-state algorithm adds protocol overhead but they have optimized it by using an on-demand approach.

## 1.4 On-Demand Routing for MANETs

In this chapter the major phases for routing protocols and the challenges that face on-demand routing protocols will be introduced. The particular challenges are linked to research that has been done following which a short presentation of the main ideas. The general phases of on-demand routing for MANETs are:

- **Route Discovery:** Initially the source has no routes and has to initiate a route discovery.
- **Route Choosing/Settling:** Once a route is in the cache the route choosing can start
- **Route Maintenance:** Involves maintaining the route cache and for example notifying MNs of route errors.

### 1.4.1 Route Discovery

The Route Discovery of source initiated on demand routing protocols consists of broadcasting route request (RREQ) packets from the source. These packets are in turn re-broadcast until they reach the destination. Before re-broadcasting each receiver will update the packet and/or its route cache depending on which Routing Protocol is being used. DSR will have the entire source route in the RREQ packet when it reaches the destination while AODV that uses per hop routing will only have the {src, dest} in the RREQ when it reaches the

destination. In AODV nodes on the route only store the  $\{\text{prev\_hop}, \text{next\_hop}, \text{src}, \text{dest}\}$  in their route cache which will lower the memory overhead. However, AODV augments the complexity of route error handling because of this, in addition the route cache hit ratio [4] will decrease more than with source routing schemes. When the destination receives a RREQ it will respond by unicasting a route reply back to the source using the reversed route. This approach assumes bidirectional links. However, the de facto standard on the MAC Layer uses RTS/CTS packets, which requires bi-directional links.

There are more sophisticated routing protocols that during RREQ broadcasting stores route statistical data in the packet. The statistics are used later at the source when choosing the path to send data through. We will highlight some implications and approaches next. Known problems involving Route Discovery of on-demand Routing Protocols in MANETs are:

- **Broadcast-storm problem:** With high mobility more routes will break and hence the route discovery phase has to be initiated several times. RREQs will flood the network and will potentially cause a severe performance drop in the network.
- **Route-reply-storm problem:** A continuance/effect of the broadcast-storm problem. In high connectivity MANETs the destination will receive a growing number of RREQs as the number of MNs in the MANETs increase. A number of the RREPs contain routes that will never be used and will fill up the sources route cache increasing the overhead and the possibility of a stale cache.

In [34] they present an interesting statistical analysis of the broadcast-storm problem in MANETs. The actual new ground covered by a propagating RREQ is on the average 41 percent and at most 61 percent. When receiving the broadcast the third time the probability of covering new ground drops to 19 percent, and with more than three propagations the probability drops to 5 percent. This suggests that using a *one-propagating broadcasting*

scheme there is a high probability of not getting all the routes possible. Although this is not a problem in high connectivity networks it can be in sparsely connected MANETs. However, it also suggests that a lot of the re-propagated broadcast messages will cover little ground and gain nobody. Statistical and geometrical broadcast schemes for MANETs have been proposed [7, 47] with varying degree of success. The route-reply-storm problem has not been researched extensively since it is an effect of the broadcast-storm problem. Simple threshold schemes can be applied to solve it.

A well adjusted on-demand MANET routing protocol must have mechanisms to control the broadcast-storm problem in order to discover the routes that will suit the source. A selective re-broadcasting scheme should be applied that carefully handles the issues involved. The route-reply-storm problem is not a severe problem but should be taken into account. The source must be given a wide variety of routes but the destination should be able to distinguish between statistically useful routes and useless routes.

#### **1.4.2 Route Choosing**

When receiving RREPs the source now has to choose between the routes. There are a number of route choosing algorithms from naive shortest path algorithms to complex quality controlled algorithms. DSR uses a shortest path algorithm and AODV uses the route with lowest delay. Link-stability based algorithms are used in Signal Stability Adaptive Routing (SSA) [18], Associativity Based Routing (ABR) [48] and Route lifetime Assessment Based Routing (RABR) [3].

SSA uses the notion of strongly connected routes and weakly connected routes. All the paths are established exclusively along stable links where the stability measure is based on the signal strength of the received RREQ. The route connectivity is based on how long the route has been active. The route-choosing algorithm chooses routes based on the link connectivity metric.



ABR uses the idea of choosing stable links over transient links. In ABR a link is considered stable if it exists for at least  $\{A_{thresh} = 2r_{tx}/v$ , where  $r_{tx}$  is the transmission range and  $v$  denotes the relative speed of two devices}. The justification is that after the connection time  $A_{thresh}$  there is a good chance that the nodes are moving with similar speed and direction, and will probably stay together for a while. The methods of ABR involve predicting when the signal will drop below a critical threshold using the signal strength changes from previously received packets. They assume by this that the direction of the nodes is fairly stable. There are other stability based routing protocols but many assume an available geo-satellite system for example GPS to be able to work.

DSR and AODV were tested against more sophisticated link stability based route-choosing algorithms in [10]. The results were based on link lifetimes and didn't consider throughput at all. The results showed that the simulation setup has a major impact on the link lifetimes. The simulation setup involves choosing movement patterns, shape of the simulation area, transmission ranges and grid size among others. With certain setups shortest path route choosing performed about as good as link stability-based algorithms. The results highlight the importance of choosing a simulation setup that captures the stability feature. For example Random Waypoint [8] with low node density and high mobility is not a good way of testing link-stability based algorithms. This is because the probability of finding a good path increases for link stability based algorithms but also for protocols without stability features. When this is said the results in [10] also showed that having link-stability will increase link lifetimes in most scenarios and is worthwhile. For a thorough assessment of the simulation setup and the complete simulation results we refer the reader to the paper.

In addition to link-stability a routing protocol should consider the QoS of a path. In the route-choosing algorithm there must be mechanisms that consider the QoS demands and decide upon a route accordingly. It is vital that the QoS demands are prioritized according to importance. The performance of a route is useless if it breaks the next second, this

indicates that link-stability is more important than other performance metrics in the route choosing. However as the route length increases the probability of a route error will increase accordingly and the importance of link-stability measures diminishes. Proper mechanisms to handle different scenarios must be present in the route-choosing algorithm [31].

### 1.4.3 Route Maintenance

MANETs are prone to link failures which make route maintenance a vital phase. Route Maintenance for on demand routing protocols can be divided into two sub-phases:

- **Proactive Route Maintenance:** Proactive mechanisms aim to predict and preempt route failures of any kind.
- **Reactive Route Maintenance:** Reactive mechanisms are not initiated before the failure has occurred.

The definition of a route failure is routing protocol dependant but can be summarized to link failure and QoS violations. Proactive link failure protocols are proposed in [22, 9].

In [22] they introduce a preemptive region where the received signal strength has dropped below a threshold. When a MN determines that the signal strength has dropped below the threshold a message is sent back to the source with a request to initiate route discovery. They use the received signal strength to calculate the relative speed between two nodes. From the speed they estimate when the route is going to break and when the source has to be notified in order to have time to complete a new route discovery. Radio signals are subject to channel fading and transient interference, which can lead to erroneous calculations and unnecessary route discoveries. Continuous route discoveries will reduce the network performance. In [22] they propose to use established mechanisms in the cellular telephony field to mitigate this problem. When the signal strength drops below a preemptive threshold the MN starts to ping the previous-hop node with a maximum of 3 times. If the previous-hop does not

respond with a ping (called pong) within a ping-timeout a route warning is sent back to the source. Also, if 3 pong responses are received having received signal strength below the preemptive threshold a route warning is sent. The number 3 was found after an empirical study.

Instead of using the signal strength the transfer time of a packet on each link is used in [9] to estimate the distance between two nodes. When a MN discovers that the transfer time is above a threshold a ping packet with a timestamp is sent to the previous hop, which in turn responds with a new time-stamped pong packet. If two pong packets are received with a transfer time above a threshold the MN will send a warning to the source which checks if the route is still active. If it is still active a rediscovery routine is invoked otherwise a pure route discovery is initiated. For more details we refer the reader to [9].

Proactive QoS routing for MANETs [57] is based on OLSR [5] described in section 1.3. They change the multipoint relay (MPR) selection algorithm. Instead of choosing MPRs purely based on transmission range and the nodes that cover the most neighbors they introduce bandwidth as a selection criterion. They propose three algorithms that use these factors with varying priority in the MPR decision-making. The results show that using the bandwidth will increase the performance significantly, and they prove that two of their algorithms will find the maximum bandwidth path. As most of the QoS approaches so far they too fail to consider mobility. We refer the reader to [57] for further study of the preliminary results.

Reactive route failure mechanisms should be present in any route protocol for MANETs. Examples of reactive on-demand route protocols are DSR and AODV. For these protocols a reactive mechanism must update the route caches of the involved parties. Link failure must trigger a message to notify the source of a dead link. A QoS route protocol that has end-to-end guarantees must have a messaging system that notifies the source when the QoS

can't be met and need to be renegotiated. Keeping route caches up to date is vital and is the prime assignment for reactive route failure mechanisms.

The overall goal for route maintenance is to diminish the communication disruption time. Proactive and reactive failure mechanisms should provide the routing protocol with means that enhances the performance and not the Achilles heel that slows down the network.

## Chapter 2

# Quality of Service aware Source initiated Ad-hoc Routing

Quality of Service aware Source initiated Ad-hoc Routing (QuaSAR) adds quality control to all the important phases of a routing protocol. In this chapter we first identify the phases in QuaSAR and then describe the algorithms for the protocol. QuaSAR has the following three phases:

- **Route Discovery:** Collects QoS statistics and associates them to the route. QuaSAR is an on-demand routing protocol and uses QoS Route Requests and QoS Route Replies in the route discovery phase.
- **Route Choosing:** Uses the QoS route attributes to distinguish better routes.
- **Route Maintenance:** Has proactive mechanisms that aim to preempt route breaks based on battery level and signal strength estimation. In addition it has reactive mechanisms that prevent stale route caches.

The routing in QuaSAR is based on on-demand source routing which is what DSR uses. However, QuaSAR has additional quality control features in all the phases of routing whereas DSR is a purely reactive routing protocol. On-demand suggests that the source needs to discover the route to the destination when it needs the route. Source routing

implies that the source of every data flow stores the entire route to the destination in its cache. Although source routing adds memory overhead it was found in [10] that source routing is more resilient to mobility than AODV, TORA and DSDV that uses other routing schemes. The following sections will explain the algorithms QuaSAR uses.

## 2.1 Route Discovery

QuaSAR is an on-demand routing protocol and finds a route to the destination by flooding the network with a QoS route request (QRREQ). Upon reception the destination sends a QoS route reply (QRREP) back to the source with the entire path. Among the issues with on-demand routing protocols is the broadcast-storm problem as previously described in Section 1.4. QuaSAR adds quality control to the re-broadcasting of QRREQs. We have adopted the idea of selective re-broadcasting and link the idea to the current QoS of the QRREQ and the state of the receiving MN. The route discovery algorithm is given in Algorithm 1.

QuaSAR gives applications the opportunity to provide the Network Layer with QoS demands that are used during the route discovery. The QoS metrics include performance metrics and link metrics. A description is given here:

- **Latency:** The end-to-end latency the application can tolerate.
- **Bandwidth:** The lowest bandwidth the application can tolerate on the route.
- **Signal Strength:** The signal strength is used to measure the distance between the hops. The metric defines the maximum distance that can be tolerated between any hop in terms of percentage of total transmission range.
- **Battery Power:** The Application Layer estimates the length of the transmission or data to be transmitted. The Network Layer maps the estimations to a battery power demand.

It is important to classify QoS metrics in order to spare the network for unnecessary overhead during route discovery. The concept of QoS mapping is applied to all the QoS metrics QuaSAR is using and the standard classification is the higher class is better. Latency and bandwidth have 8 classes where currently 4 Mbps is the best bandwidth and with the latency accordingly small. Battery power and signal strength have three classes. If the Application Layer has any preferences the battery power can be one of the deciding factors in the route discovery. It is based on the length of the transmission and/or size of data to be transmitted. The lowest battery power class is critical and means the node has the capability of a threshold packet forwards before it runs out of battery power. The application can also use signal strength in route discovery, which will help to find a stronger linked route that has a higher probability of survival. Class 3 signal strength means the MNs on the route are all within 80 percent of transmission range, class 2 is within 90 percent, and class 1 is outside 90 percent of transmission range. Shown in Figure 2.1 The signal strength calculations will be explained in more detail in Section 2.3.

Radio signals operate in a shared medium and are susceptible to fluctuations caused by geographical and atmospheric circumstances. The deviation from the actual distance between two MNs might trigger unwanted reactions, however during route discovery a source is interested in a strong link and a strongly fluctuating link is not desirable.

The Route discovery in QuaSAR can be divided into two subphases:

- **QoS Route Request:** Broadcast on the way to the destination
- **QoS Route Reply:** Unicast going back to the source

The sections that follow explain these subphases in detail.

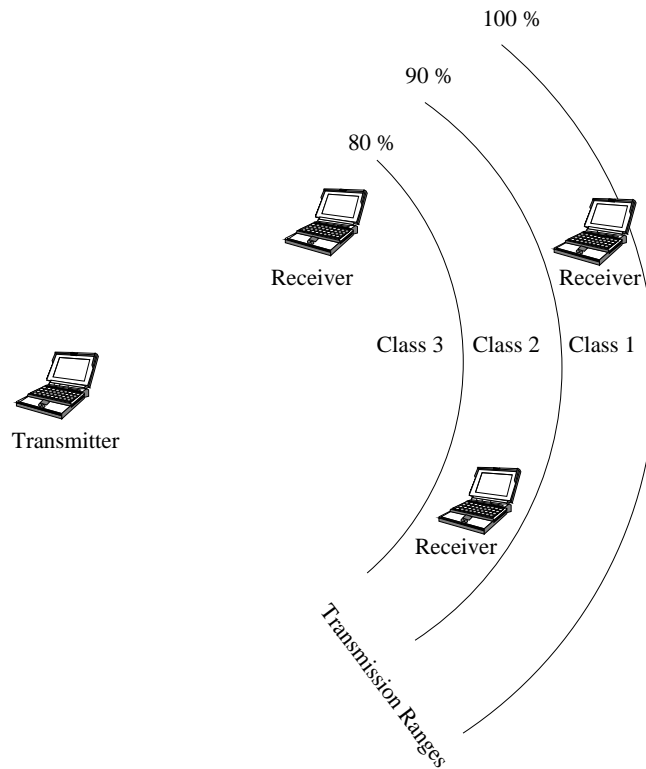


Figure 2.1: Transmission range classification

### 2.1.1 QoS Route Request

QuaSAR adds a QoS header to an ordinary route request (RREQ) packet. In the QoS Route Request (QRREQ) we have added the following in order to store the route statistics for later use:

- **QoS Demands:** Contains the QoS the current application seeks
- **QoS Available:** Contains the current QoS image of the route

Broadcasting is in essence a means for notifying all participants of a network about something. Using broadcasting and flooding the network with route requests does not promise anything but this. QuaSAR addresses this problem by adding a status to the



route request such that nodes that have previously propagated a QRREQ can rebroadcast a second QRREQ if the QoS of it is better.

Before broadcasting a QRREQ the QoS header must be initialized. The QoS demands are initialized to the applications requirements but QoS available is slightly different. Battery, signal strength and bandwidth are min/max metrics, and are initialized appropriately to the highest class. We use end-to-end latency thus we don't include this metric in the QRREQ but record the latency when the QRREQ returns. It is possible to use one-way latency but that will require time synchronization [26]. Shown in Algorithm 1.

---

**Algorithm 1** QRREQ-Source Nodes: Broadcasting QRREQ

---

```

1: QoSdemands = applicationLayer.getQoSdemands()
2: qosClassifier.map(QoSdemands)
   {(signal, battery) three classes, (bandwidth) eight classes}
3: if QoSdemands.QoSvalid() then
4:   QRREQ.QoSdem = QoSdemands
5: else
6:   QRREQ.QoSdem = QoSMinimum
7: end if
8: initialize(QRREQ.QoSav)
   {(signal, battery, bandwidth) initialized to highest}
9: Broadcast(QRREQ)

```

---

When an intermediate node receives the QRREQ it records the QRREQ id and updates the available QoS in the QRREQ QoS header. An intermediate node will in addition remember the minimum length of the route contained in a QRREQ, the best QoS mapped to a number according to the QoS metric precedence rules and the current service class of the QRREQ. The QoS metric precedence rules are presented in Section 2.2. The QRREQ statistics are stored for each route discovery session, and are designed to address some of the problems that will occur by using broadcast as the means of finding routes. We assume that we have a MAC Layer that can calculate the available bandwidth taking into consideration the aggregate traffic of the neighborhood. AQOR [56] presented a protocol that takes this

into consideration.

Before a MN rebroadcasts a QRREQ it invokes a routine to find the service class and the service level of the QRREQ. If all the QoS demands are met the service class is set to two, however if any of them were not met the service class is set to one. If the battery on the MN is about to run out the service class is set to zero and the QRREQ is dropped. The service level is a statistical number describing the QoS of the QRREQ using the QoS metric precedence. The current QoS available are mapped to classes as described in Section 1.2 and the service level is calculated from them.

QRREQs are now selectively re-broadcasted based on the service class and the service level of the QRREQ compared to the MNs QRREQ re-broadcasting history. Shown in Algorithm 2. A MN will rebroadcast a QRREQ if it has not previously processed a QRREQ with better service class. If the MN has processed a QRREQ with the highest service class any following QRREQ must have a better service level and the  $QRREQ.route.length() < \{2 * minQRREQLength + 1\}$ . The route length is used to avoid QRREQ living to long finding routes that are statistically of no use. Shown in Figure 2.2.

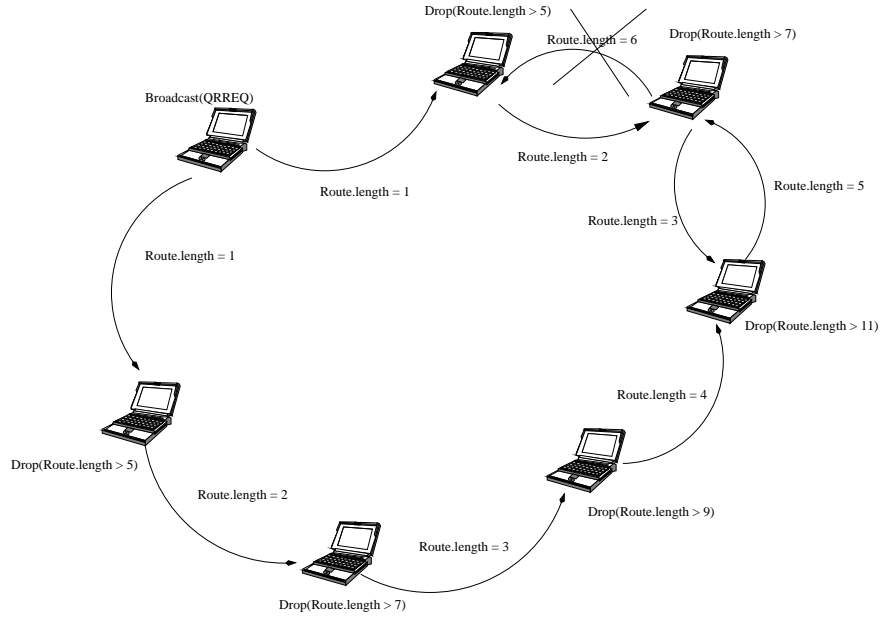


Figure 2.2: Circular problem with Re-Broadcasting

---

**Algorithm 2** QRREQ-Intermediate Nodes: Receiving QRREQ

---

```

1: ServiceClass = ServiceLevel = processQrreq = 0
2: if NewQRREQSession(QRREQ) then
3:   ResetQRREQState(QRREQ)
4: end if
5: UpdateQoSav(QRREQ)
6: ServiceClass = GetQoSClass(QRREQ.QoSav, QRREQ.QoSdem)
7: ServiceLevel = GetQoSLevel(QRREQ.QoSav, QRREQ.QoSdem)
8: if qrreqServiceClass < ServiceClass then
9:   processQrreq = true
10: else if qrreqServiceLevel < ServiceLevel  $\wedge$  QRREQ.route.length() < ((qrreqMinLen *
    2) + 1) then
11:   processQrreq = true
12: end if
13: if processQrreq == true then
14:   qrreqServiceLevel = max(ServiceLevel, qrreqServiceLevel)
15:   qrreqServiceClass = max(ServiceClass, qrreqServiceClass)
16:   qrreqMinLen = min(QRREQ.route.length(), qrreqMinLen)
17:   QRREQ.route.AddToRoute(this→IP)
18:   Broadcast(QRREQ)
19: end if

```

---

### 2.1.2 QoS Route Reply

In Section 1.4 we identified and introduced a problem caused by the broadcast-storm problem that is the route-reply-storm. With a high density network and a big number of MNs the number of routes that potentially are sent back to the source is very high. Most of these routes are never used and only waste memory. QuaSAR uses a selective route-reply algorithm that aims to give the source a wide range of routes instead of all the routes.

The destination automatically sends a QRREP to the first threshold QRREQs that are received, but after the threshold has been exceeded only selective requests are responded to. The MN stores the best QoS metrics of the current QRREQ session, the previous hops and the minimum length route. These variables are then used in the selective route-reply algorithm. Only QRREQs who have a length  $\leq \{\text{minLength} * 2 + 1\}$  are considered. The formula was found after an empirical study of the simulation results. After this a QRREQ is only responded to if the previous hop hasn't been processed or if it has one better QoS metric. Shown in Figure 2.3.

If the minimum route length is one hop QuaSAR uses this as a route length of 2. In the case of length 2 any QRREQ route with more than 5 hops aren't considered. There are issues with doing this i.e. with different antenna strengths in the network. However the selective route reply phase does not execute before a threshold of QRREQs has been responded to. Shown in Algorithm 3.

Once a QRREQ is accepted and statistics have been noted a QRREP is unicast back to the source. QuaSAR does not update the QoS of a QRREP since the QoS will not have had time to change significantly. Updating the QoS both ways will consume MN battery power, steal CPU cycles and make the source wait longer for a QRREP.

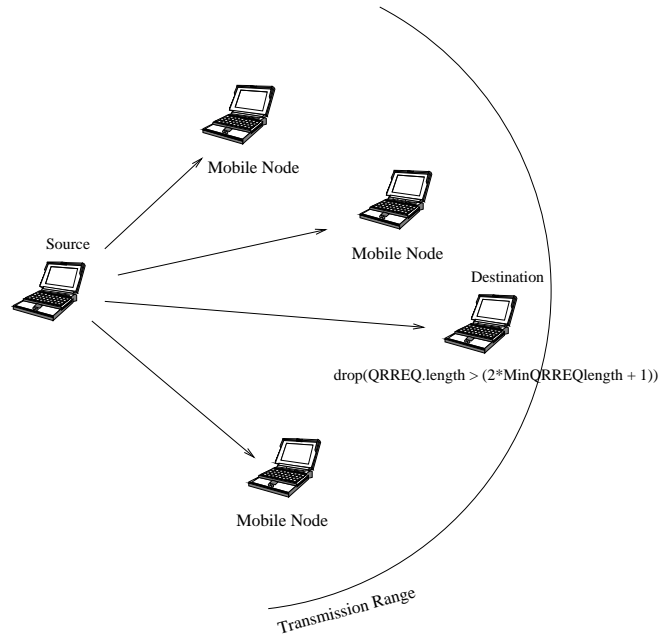


Figure 2.3: Destination Dropping QRREPs that have length  $> \{\text{minLength} * 2 + 1\}$

---

**Algorithm 3** QRREP-Destination Nodes: Receiving QRREQ

---

```

1: ServiceClass = ServiceLevel = processQrrep = 0
2: if NewQRREQSession(QRREQ) then
3:   ResetQRREQState(QRREQ)
4: end if
5: if qrrepCounter < qrrepThreshold then
6:   if newQoSFeatures(QRREQ) then
7:     processQrrep = true
8:   else if QRREQPrevHopNew(QRREQ)  $\wedge$  QRREQ.route.length() < ((qrrepMinLen
   * 2) + 1) then
9:     processQrrep = true
10:  end if
11: else if qrrepCounter < absoluteThreshold then
12:   SaveBestQoS(QRREQ)
13:   saveQRREQPrevHop(QRREQ)
14:   qrrepMinLen = min(qrrepMinLen, QRREQ.route.length())
15:   processQrrep = true
16: end if
17: if processQrrep == true then
18:   qrrepCounter++
19:   createQRREP(QRREQ)
20:   Unicast(QRREP)
21: else
22:   drop(QRREQ)
23: end if

```

---

## 2.2 Route Choosing

QuaSAR employs a route discovery phase that collects route statistics. These statistics are used in the Route Choosing phase to find a route that is better according to the combination of these numbers. Once the route discovery phase of any routing protocol collects statistical route information there must be a carefully designed Route Choosing algorithm that takes them into account. Choosing the best path in the route cache is an impossible task since any of them potentially can be the best choice in the long run. The route may look inferior after route discovery was completed but may in fact have been better with the right statistics at hand. Even having the statistics will make us choose routes that weren't the best. Instead we have to focus on trying to choose better routes using the collected statistics and the QoS demands from the application.

QuaSAR records available bandwidth, latency, signal-strength and battery power for each route during route discovery. The route-choosing algorithm has to be able to interpret and convert these statistics to be able to distinguish the routes efficiently. QuaSAR uses QoS metric precedence to be able to choose between routes, and the application has the opportunity to choose the ranking of the metrics. This is done because applications have very different needs in terms of QoS and should be able to influence the route choosing all the way. However, if the application does not have any preferences the default metric precedence in terms of route importance is as follows:

- **Battery Power:** Is the most route critical metric. If the MN is too weak there is no point in considering the route at all, since it will break unless the operator of the MN gives it more power.
- **Signal Strength:** If the route has one hop with lowest signal strength class other routes should be considered.

- **Bandwidth:** Bottleneck bandwidth may cause massive packet drops.
- **Latency:** Is the least route critical metric but must be present nonetheless. Streaming applications needs an estimate of the end-to-end latency.

Several route choosing algorithms are possible from these metrics and it proves to be very hard finding an optimal algorithm. Applications should be able to decide how they want the routes chosen. Real Time applications needs fast routes but not necessarily the most stable routes, FTP applications has the opposite requirements. Longer routes are often slower than shorter routes, but shorter routes may be more susceptible to route breaks because of weak connectivity. QuaSAR has three route choosing algorithms that treat the QoS differently and consequently will result in different routes:

1. QoS Route (Algorithm 4)
2. QoS Route, Accept lowest QoS (Algorithm 5)
3. Best-fit QoS Route, Accept lower QoS (Algorithm 6)

Algorithm 4 searches for a route that qualifies according to the applications demands and chooses the shortest of them if it finds one. If there are no routes satisfying the demands a new route discovery is initiated. Although flawed in the sense of the possibility of waiting forever, there might be certain transfers that are no use starting until a route satisfying the demands have been found. It should be up to the application to decide.

Most applications will accept a lower QoS than requested and Algorithm 5 will offer a route as long as it has one. Algorithm 5 will find the best path according to the prioritization of the metrics illustrated above. QuaSAR uses this algorithm as a default currently.

---

**Algorithm 4** Route Choosing: QoS Route

---

```
1: currShortest = MAX
2: currServiceLevel = MIN
3: for  $i = 0$  to cache.size()  $\wedge$  route.contains(dest) do
4:   ServiceClass = GetQoSClass(route.QoSav, QoSdem)
5:   ServiceLevel = GetQoSLevel(route.QoSav, QoSdem)
6:   if ServiceClass == serviceOK  $\wedge$  currServiceLevel  $\leq$  ServiceLevel  $\wedge$  currShortest  $>$ 
     route.length() then
7:     currServiceLevel = ServiceLevel
8:     currShortest = route.length()
9:     QoSRoute = route
10:  end if
11: end for
12: return QoSRoute
```

---

---

**Algorithm 5** Route Choosing: QoS Route, Accept Lowest QoS

---

```
1: currShortest = MAX
2: currServiceLevel = currServiceClass = MIN
3: for  $i = 0$  to cache.size()  $\wedge$  route.contains(dest) do
4:   ServiceClass = GetQoSClass(route.QoSav, QoSdem)
5:   ServiceLevel = GetQoSLevel(route.QoSav, QoSdem)
6:   if currServiceClass  $>$  ServiceClass then
7:     continue
8:   else if currServiceClass  $<$  ServiceClass then
9:     currServiceClass = ServiceClass
10:    currServiceLevel = ServiceLevel
11:    currShortest = route.length()
12:    QoSRoute = route
13:   else if currServiceLevel  $<$  ServiceLevel  $\wedge$  currShortest  $\leq$  route.length() then
14:     currServiceLevel = ServiceLevel
15:     currShortest = route.length()
16:     QoSRoute = route
17:   end if
18: end for
19: return QoSRoute
```

---



Always giving the best route to an application violates the QoS principle cost of service mentioned in Section 1.2. There should be a penalty involved if an application is always using the best possible route. Algorithm 6 takes this into consideration and tries to find the path that comes closest to the demanded QoS. However it will always prefer a route that qualifies over a route that nearly qualifies.

---

**Algorithm 6** Route Choosing: Best-Fit QoS Route, Accept Lowest QoS

---

```

1: currShortest = MAX
2: currServiceLevel = currServiceClass = MIN
3: for  $i = 0$  to cache.size()  $\wedge$  route.contains(dest) do
4:   ServiceClass = GetQoSClass(route.QoSav, QoSdem)
5:   ServiceLevel = GetQoSLevel(route.QoSav, QoSdem)
6:   if currServiceClass > ServiceClass then
7:     continue
8:   else if currServiceClass < ServiceClass then
9:     currServiceClass = ServiceClass
10:    currServiceLevel = ServiceLevel
11:    currShortest = route.length()
12:    QoSRoute = route
13:   else
14:     RouteQoSFit = getQoSFit(route.QoSav)
15:     {if RouteQoSFit > 0  $\rightarrow$  route QoS is better than demanded}
16:     {if RouteQoSFit == 0  $\rightarrow$  route QoS is equal to demanded}
17:     {if RouteQoSFit < 0  $\rightarrow$  route QoS is worse than demanded}
18:     {i.e. closestToZero() prioritizes currRouteQoSFit > 0 when RouteQoSFit < 0}
19:     if currShortest  $\geq$  route.length()  $\wedge$  closestToZero(currRouteQoSFit, RouteQoSFit)
20:       then
21:         currServiceLevel = ServiceLevel
22:         currShortest = route.length()
23:         currRouteQoSFit = RouteQoSFit
24:         QoSRoute = route
25:       end if
26:     end if
27:   end for
28: return QoSRoute

```

---

## 2.3 Route Maintenance

MANETs will experience times of mobility and it is therefore important to handle the mobility and the issues it brings. One of the major challenges of wireless networks is to diminish the communication disruption time caused by the mobility. Mobile IP have developed complex handoff schemes [51] between Base Stations and it wasn't until recently that research has been done to apply this feature to MANETs [22, 9]. QuaSAR has both proactive and reactive route maintenance mechanisms. The mechanisms are explained in detail in the next subsections but can be summarized to:

- **Proactive Route Maintenance:** QuaSAR introduces a Route Change Request (RCR) designed to catch a route critical incident and react before the route breaks by notifying the sending party.
- **Reactive Route Maintenance:** A Reactive Route Error (RERR) message is sent to the source once a link is dead. Reactive Route Rescue (RRES) is initiated when a link dies trying to send a packet containing data. RRES tries to find an alternate route to the destination.

### 2.3.1 Proactive

QuaSAR has proactive mechanisms that aim to preempt route breaks based on battery power and signal strength estimations. Route critical incidents in MANETs may be caused by:

- **Signal strength weakening:** MN is moving out of range
- **Battery power depletion:** MN Probably disconnects soon
- **Memory shortage:** Becomes selfish and drops packets

If a MN discovers that route critical incidents most likely will happen QuaSAR sends a Route Change Request (RCR) back to the source, using the reverse route, informing it about the current problem. Depending on the current problem the node(s) involved are flagged by inserting them into a RCR-table and the route choosing algorithm will give the routes containing the element(s) less of a priority than other routes. Flagging nodes as route critical is faulty if there aren't any update mechanisms. It is possible that nodes experiencing battery power problems may receive more power if the operator changes the battery. And node movement will cause a critical signal hop to become stronger thus invalidating RCR-table entries. These changes are handled in QuaSAR when the route discovery phase is initiated. As the QRREQs are propagated the routes are checked for RCR-nodes and the intermediate MN's RCR tables are updated if i.e. a link has changed from critical to better. When the source receives QRREPs it also updates the RCR-table. It will happen that routes are wrongly flagged, but these routes will contain MNs that took part in causing a route critical incident and it is probably wise to disregard it.

A Route Change Request packet includes these fields:

- **Reason [ ]** : The reasons for an RCR in QuaSAR may be low signal or low battery
- **Route**: The route with the RCR reason
- **Originator**: The source of the RCR
- **PrevNode**: If a low signal RCR is sent the low signal link must include two MNs, the previous node and the originator
- **NewRoute**: As the RCR traverses through MNs they look for new routes to the destination or source depending on where the RCR is sent first. This is explained in detail later

An RCR will originate from a MN that operates as a router for a currently active flow. In the case of weak link detection the destination can also send an RCR back to the source. A weak battery power message can originate from the destination but only as a notifying mechanism.

---

**Algorithm 7** RCR: Originating Nodes

---

```

1: issueRCR = false
2: reason[2] = false
3: if PowerLevel == Critical then
4:   issueRCR = true
5:   reason[lowBattery] = true
6: end if
7: if StatusOfFlowValid(pkt) then
8:   transToBreak = EstimateTrans(pkt.signalStrength)
9:   timeToBreak = EstimateTime(pkt.signalStrength)
10:  if (transToBreak < Threshold)  $\wedge$  (timeToBreak > MinProcessTime) then
11:    issueRCR = true
12:    reason[LowSignal] = true
13:  end if
14: end if
15: if issueRCR == true  $\wedge$  TimeForNewRCR() then
16:   RCR.Route = pkt.route
17:   RCR.Originator = this $\rightarrow$ IP
18:   RCR.reason[] = reason[2]
19:   if reason[LowSignal] == true then
20:     RCR.prevNode = pkt.route[prevNode]
21:   end if
22:   if transToBreak < Threshold || SentToDestPrev(pkt) then
23:     RCRTimeout = RCRToSourceTimeout {the timeout prevents RCR flooding}
24:     RCR.dest = pkt.source
25:   else
26:     RCRTimeout = CalcRCRToDestTimeout(transToBreak,timeToBreak)
27:     {an estimate for how long it will take to process the RCR}
28:     RCR.dest = pkt.dest
29:   end if
30:   initQoSheader(RCR) {RCR.QoSdem = MIN, RCR.QoSav = initvals()}
31:   Unicast(RCR)
32: end if

```

---

When a MN receives a data packet it will check the signal strength by which the packet was received with and the current power level of the MN. If any of these checks show that a route break is likely to happen a proactive mechanism is started. QuaSAR defines a threshold for the battery power where it has less then a number of packet forwards before the MN will die with the current battery level and battery usage. The received signal strength is used to estimate when the route will break and issue an RCR at an appropriate time based on the bit rate of the data flow. Shown in Algorithm 7. A thorough description of how and when to issue an RCR follows.

QuaSAR uses the same approach proposed in [22] (refer to Section 1.4.3) to preempt route breaks caused by a weak signal. In [22] they use a received signal strength threshold to decide when to start a proactive mechanism. In our opinion it is an incomplete approach to only consider a proactive mechanism once the preemptive threshold has been exceeded. QuaSAR does not have a preemptive region and bases the execution of the proactive mechanism on an estimation of how many transmissions there are left before the route will break. The estimation assumes that the current trend in the received signal strength will continue. We use the previous and current signal strength, and previous and current receive time to estimate how many transmissions the link between the previous hop and the receiving MN has left before it dies.

We assume we are using the wireless channel model TwoRayGround [50] implemented in ns-2 Network Simulator [49]. TwoRayGround predicts a received power at distance  $d$  by considering both the direct path and a ground reflection path. The generalized formula for received power (watt) looks like this,

$$P_{received} = \frac{P_0}{d^n} \quad (2.1)$$

where  $P_0$  is a constant for each transmitter/receiver pair, based on antenna gain and height.  $d$  is the distance between the MNs.  $n$  will be 2 until a crossover distance  $d_{crossover}$

where  $n$  becomes 4. TwoRayGround has a faster power loss than for example Free Space equation [20] but it was shown in [43] that it predicts more accurately at a long distance.

A path must be preempted such that the source has time to receive the RCR and complete a route discovery before the path breaks. The time for preemption  $T_{preempt_{src}}$  by notifying the source must satisfy the following constraint

$$T_{preempt_{src}} \geq Lt_{MN,src} + T_{routediscovery} \quad (2.2)$$

where  $Lt_{MN,src}$  is the latency between the MN and the source.  $T_{routediscovery}$  is the time it takes to complete a Route Discovery. If the RCR is sent to the destination there must be more time

$$T_{preempt_{dest}} = Lt_{MN,dest} + Lt_{dest,src} + T_{preempt_{src}} \quad (2.3)$$

Finding  $T_{preempt_{src}}$  or  $T_{preempt_{dest}}$  dynamically is a hard problem and is subject to future studies. We assume that we have this time and use it to find the optimal number of transmissions  $\gamma$  a path must be preempted

$$\gamma = T_{preempt} \times CBR, \quad \{\gamma > 0, \gamma \in N\} \quad (2.4)$$

where  $\gamma$  is rounded up to the closest natural number. QuaSAR uses the same ping/pong messaging approach as [22] in order to further verify the authenticity of the signal strength weakening. In our implementation we didn't use the ping/pong verification since TwoRayGround is a deterministic wireless channel model. Algorithm 8 presents the path preemption estimation.

---

**Algorithm 8** RCR: Preempt Path

---

```
1: deltaDistance = prevDistance - currDistance
2: if deltaDistance < 0 then
3:   ChangeSign(deltaDistance) {make the integer positive}
4:   for transToBreak = 1 to (DestPreempt() + 1) do
5:     d = currDistance + (deltaDistance × transToBreak)
6:     Pr = calcPower(d, P0)
7:     if Pr < ReceiveThreshold() then
8:       break
9:     end if
10:  end for
11:  if transToBreak ≤ DestPreempt() then
12:    InitiateVerification(transToBreak) {Ping/Pong based}
13:  end if
14: end if
```

---

---

**Algorithm 9** RCR: Intermediate Nodes

---

```
1: InsertRCRTable(RCR)
2: if Valid(RCR.newRoute) then
3:   RCR.newRoute.add(this→IP)
4:   UpdateQoSav(RCR.QoSav)
5: end if
6: if route = RouteChooseAlg(pkt.dest) ∧ ComopareQoS(route, RCR.newRoute) then
7:   RCR.newRoute = route
8: end if{RouteChoose Algorithm will check if route contains RCR mobile nodes}
9: Forward(RCR)
```

---

A weak link is categorized to how many estimated transmissions the data flow has left before the route breaks. If it is below a threshold an RCR will be sent. Based on the estimations the algorithm decides whether there is time to send an RCR to the destination in order to trigger a RCR/QRREP or if it has to send an RCR directly to the source. As previously mentioned the source must be able to complete a route discovery before the route breaks. Once proactive mechanisms are used they must be exploited to the fullest, thus QuaSAR uses RCRs both as a notifying mechanism and as a unicast route discovery. The RCR is unicast through the active route and forwarding MNs checks whether it has a new route to the destination, as shown in Algorithm 9. A part of the RCRs responsibility is to

work as a route discovery avoidance mechanism. If an RCR is first sent to the destination, it means that the results from the estimation routine indicated there is enough time to try to provoke a QRREP from the destination. The RCR-reasons and the routes are prioritized during route choosing as follows (most critical first):

1. Low battery powered node and a low signal hop
2. Low battery powered node
3. Low Signal hop

If the RCR was sent to the destination it checks if the RCR contains any route suggestions and compares it to routes it has in its own cache. If a route is found a RCR/QRREP is unicast back to the source otherwise if no route is found the RCR is dropped. If the problem persists at the MN it will now send an RCR back to the source applying the same algorithm only now the MNs tries to find a route to the destination. The source checks for routes and initiates a route discovery if it fails to find a route that satisfies the QoS demands. To avoid continuous route discoveries QuaSAR restricts the interval to every threshold seconds. Although the route cache does not contain routes that satisfy the QoS demands the probability that one will be found if a route discovery just finished is low. The goal is to enhance the network performance and at the same time give the user what it requires. However this is not always possible and therefore tradeoffs must be in place that limits performance-degrading actions from happening too often. Shown in Algorithm 10.

### **2.3.2 Reactive**

QuaSAR offers reactive route maintenance mechanisms to keep route caches up to date. When any packet fails to transmit a Route Error (RERR) message is sent back to the sender of the packet. The MN will update its cache excluding the dead link from any route. A stale route cache is unwanted and will lead to more packet drops and it is therefore of



the utmost importance that dead links are notified to the sending MNs. Moreover, MNs forwarding the RERR will update its route cache.

When a route error occurs a Route Rescue (RRES) mechanism tries to find an alternate route without regarding the QoS. A packet about to be dropped doesn't need to consider the QoS in terms of network performance anymore since it has been violated in any case.

---

**Algorithm 10** RCR: Source/Destination Nodes

---

```

1: InsertRCRTable(RCR)
2: if Valid(RCR.newRoute) then
3:   RCR.newRoute.add(this→IP)
4:   UpdateQoSav(RCR.QoSav)
5:   if QoS(RCR.newRoute) > serviceNotOK then
6:     AddRouteCache(RCR.newRoute)
7:   end if
8: end if
9: if route = RouteChooseAlg(pkt.dest) ∧ CompareQoS(route, RCR.newRoute) then
10:  RCR.newRoute = route
11: end if
12: ServiceClass = GetQoSClass(RCR.QoSav, QoSdem)
13: ServiceLevel = GetQoSLevel(RCR.QoSav, QoSdem)
14: if RCR.ToSource then
15:   if (ServiceClass ≠ serviceOK ∧ TimeForNewQRREQ()) then
16:     SendOutQRREQ()
17:   end if
18: else if RCR.ToDest then
19:   if (ServiceClass ≠ serviceNotOK then
20:     CreateQRREP(RCR)
21:     Unicast(QRREP) {QRREP/RCR with best route back to source}
22:   end if
23: end if
24: drop(RCR)

```

---

## Chapter 3

# Simulation Experiments

QuaSAR was implemented in ns-2 Network Simulator [49] version 2.1b9a. Ns-2 is a discrete event simulator that can simulate a wide variety of networks including MANETs. It is used for research problems all over the world and is a respected tool for network simulations. QuaSAR uses source routing thus we used the already implemented source-routing protocol that DSR uses. From this protocol we added the features we mentioned in Chapter 2. We were not provided any results from other QoS Ad-hoc Routing Protocols and had to test QuaSAR against an ad-hoc routing protocol without QoS support. DSR was in [10] found to be the overall best routing protocol that is currently available for ad-hoc networks. Hence we tested QuaSAR against DSR, where DSR is a simpler source routing protocol without the extensive quality control QuaSAR has. For further introduction to DSR we refer the reader to [25].

The simulations can be categorized in two. A presentation follows.

- **Fine Grain Simulation:** Designed an experiment that captures the features of QuaSAR
- **Course Grain Simulations:** Used tools produced by CMU [41] and University of Bonn (bonnMotion) [17] to test protocol performance with different mobility models comparing QuaSAR and DSR.

When performing simulations we have to specify an environment. This environment has to include the most important factors that recognize real life use of the tested protocols. For our protocols we needed a network with MNs that had different levels of QoS. Movement had to present to test QuaSAR and DSR properly. Ultimately our environment was distinguished by:

- **Mobile Nodes:** The quality and amount of MNs.
- **Environment Size:** The size of the grid/playground
- **Communication patterns:** How many MNs are active
- **Movement patterns:** The speed and pattern MNs move about or the positions MNs are placed in the grid.

To our knowledge QoS aware Routing Protocols have not been tested in terms of throughput before. QuaSAR bases the routing on the QoS demands from the Application Layer and the signal strength is taken into account if it has any demands. For the sake of research we tested QuaSAR against DSR when the application had the strongest signal strength demand. That is class 3 signal strength, which means the MN is within 80 percent of the total transmission radius. The metrics that QuaSAR and DSR were tested on is presented in Table 3.1.

Table 3.1: Simulation Metrics

Category	Value
Throughput	packets/second
Packet Delivery Rate	(packets received)/(packets sent)
Latency	(mean latency)/packet
Protocol Overhead	(protocol packets)/(packets received)
	(protocol packets sent)/(packets received)
Route	(Average number of packets)/route

The throughput, packet delivery ratio and latency metrics are all standard measurements. We also wanted to test the protocol overhead in the network in terms of the number of protocol packets received by any MN per data packet received by the destination. Further we tested the protocol overhead in terms of the number of protocol packets that were actually sent by a MN per data packet received by the destination. The reason is that the selective re-broadcasting feature that QuaSAR uses will create some overhead in the network, but the number of protocol packets that were sent may in fact be less in QuaSAR than in DSR. Additionally we wanted to test the tendency the protocols have to choosing routes and added packets/route as a testing metric. If the number of packets/route is lower it indicates that the routing protocol uses more routes during the simulation.

In the following subchapters we present the fine grain and the course grain simulations that we performed. An analysis of the results is given for both.

## **3.1 Fine Grain**

In a fine grain simulation the operator should have control over every aspect. We wanted to show the weakness of DSR compared to QuaSAR in a given scenario. To do that we had to create an environment that was realistic and at the same time captured all the features we have implemented in QuaSAR and exposed the problems DSR has. In the following we present an introduction to the simulation setup, we then provide the results and an analysis of them.

### **3.1.1 Setup**

In the next subsections we are justifying the choices we made during the environment setup of the fine grain simulation.

## Environment Setup

The purpose of the fine grain simulation is to show a small environment and from that expose the weaknesses in the tested protocols. We had 8 MNs with different QoS features in the network in order to take advantage of the QoS support in QuaSAR. Ultimately the goal was to place a number of MNs within reach of the destination but it was important that each of them would loose connection to the destination during the simulation. This way both QuaSAR and DSR is tested properly on all the phases of a routing protocol. Simultaneously there had to be at least one MN within reach of the destination at all times. We ended up with this setup for the simulation:

**Weak MNs:** We placed three weak MNs with a link capacity of 128 Kbps in the grid.

They are placed such that they are one hop away from the destination. However they all have a weak connection to the destination and will loose connection when the destination moves. The weak MNs do not move during the simulation.

**Strong Antennas:** There are four static antennas with a link capacity of 4 Mbps placed in such a way that  $\{1,2\}$  and  $\{3,4\}$  can communicate, in addition 2 and 4 has a link to the destination but will loose connection when it moves.

**Source Node:** The source was placed such that it is within range of the two closest antennas and all the weak MNs. It has a weak link to the three weak MNs and a strong link to the antennas. The source sends at a constant bit rate (CBR) of 50 packets/second, where one packet is 512 bytes. The source will need a minimum bandwidth of 200 Kbps on the route to avoid bottleneck packet drops. The QoS demands of the destination include a class 3 signal, a bandwidth of 2 Mbps and according end-to-end latency.

**Destination Node:** The destination moves back and forth such that it will loose connection to all but the closest strong antenna at the end points. This fulfills the goal of having at least one link to the destination at all times. The speed of the node is 10 m/s and it moves 200 meters in east and west direction. The starting point of the destination node is {450, 470} and it moves between {350, 470} and {550, 470}. It will complete one patrol/round in 40 seconds.

The transmission range is 250 meters for all the involved nodes. The simulation lasted 80 seconds, which was enough for the destination to perform two patrol rounds.

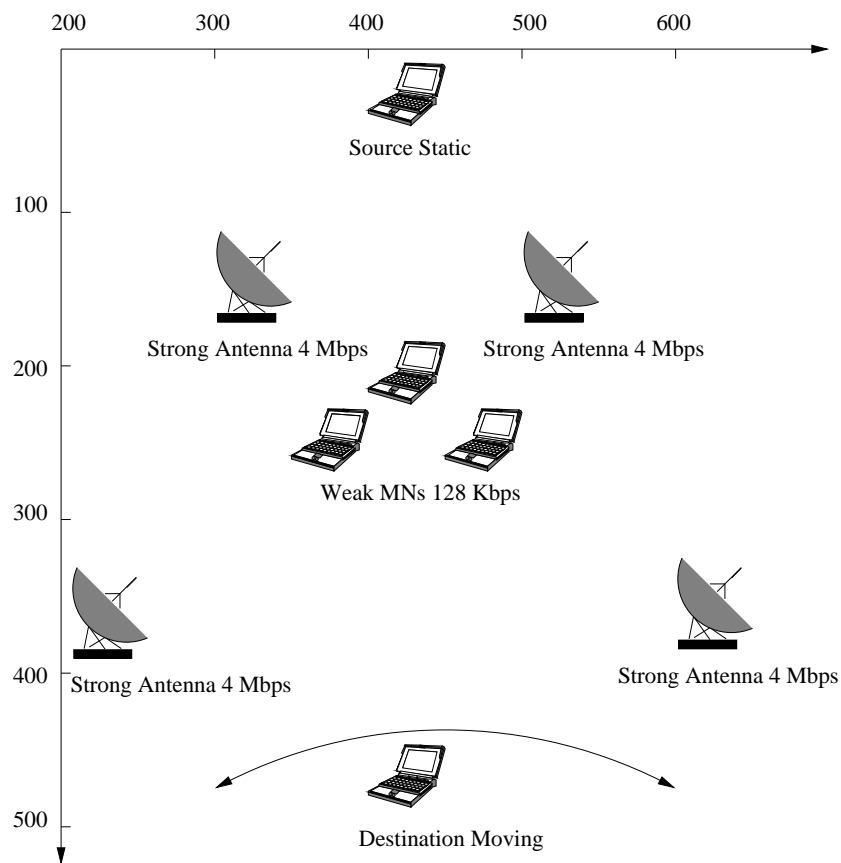


Figure 3.1: Fine Grain Simulation Environment

## Applied to Real World

The simulation environment can for example be from a military setting where the source is on a lookout point receiving data from central sources needing to pass the new operation details. The source needs to stay in touch with the destination for example an armored vehicle, which is patrolling an area up ahead in case of an attack. The antennas scattered around are tents with stronger communication equipments. The weak MNs are soldiers with standard communication equipment for each team. See Figure 3.1.

### 3.1.2 Results

The results of the fine grain simulation are presented next. They were tested on the metrics previously introduced in Table 3.1.

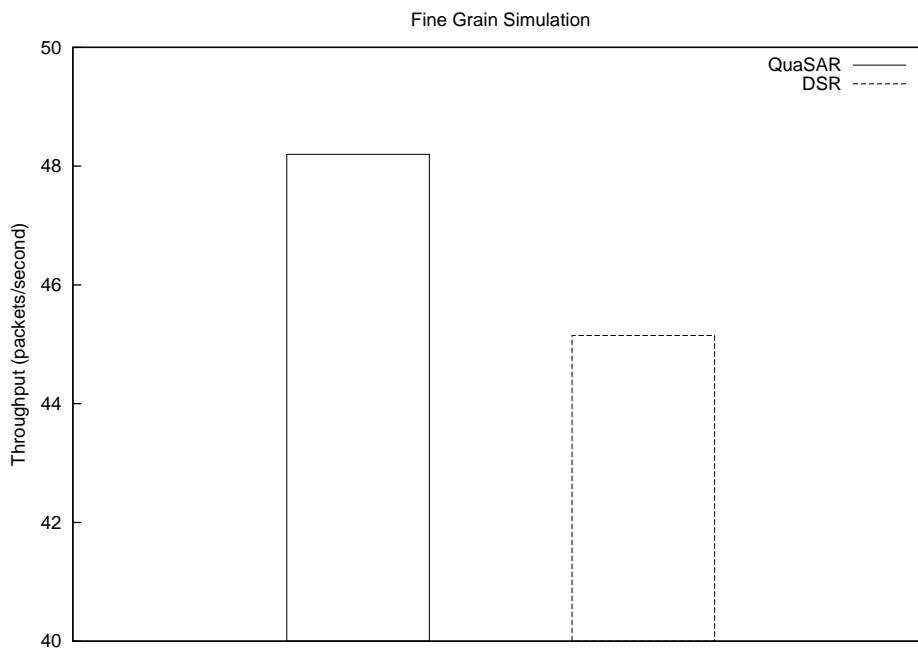


Figure 3.2: Fine Grain: Throughput

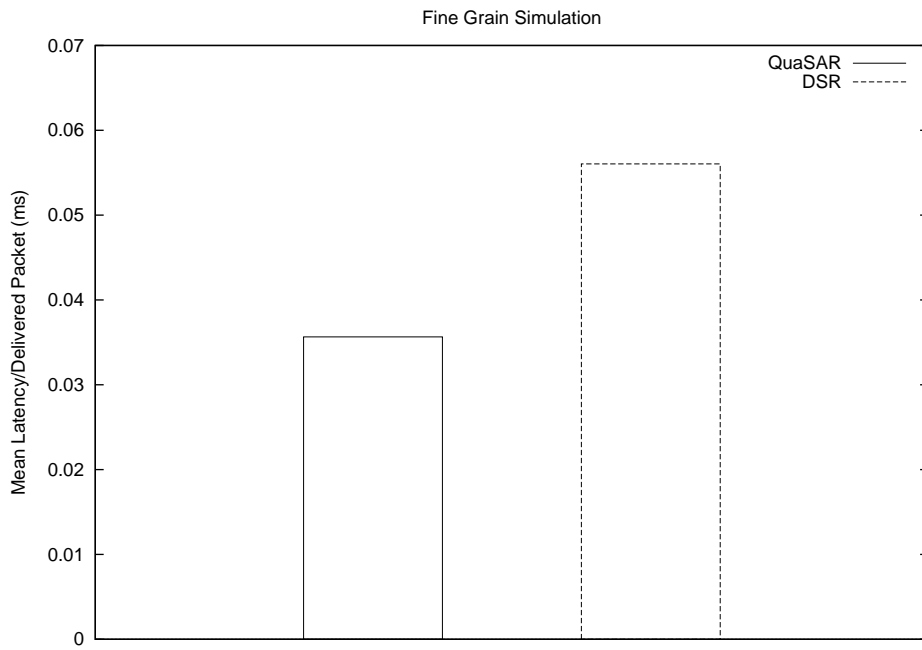


Figure 3.3: Fine Grain: Mean Latency/Packet

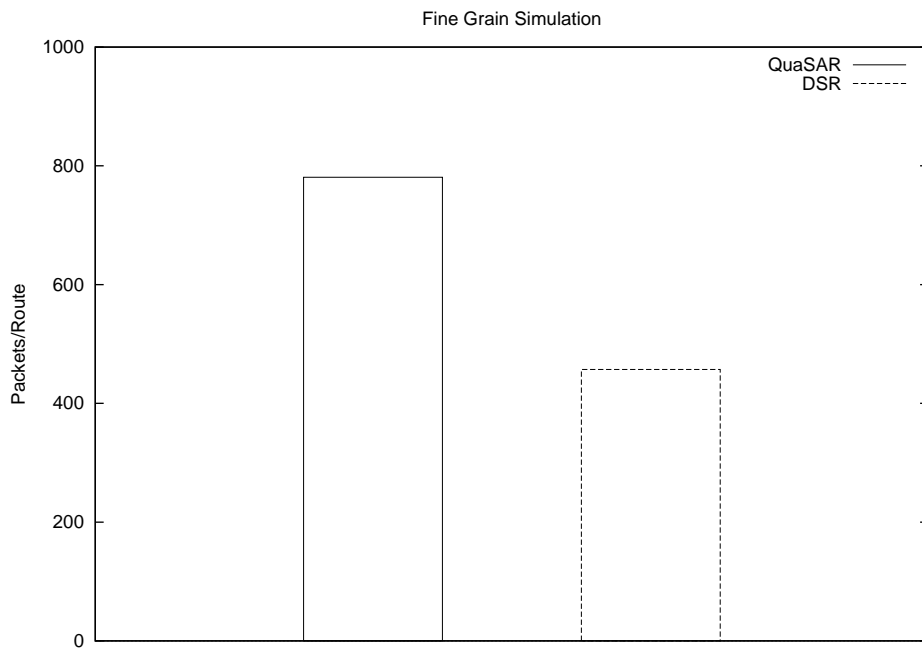


Figure 3.4: Fine Grain: Packets/Route



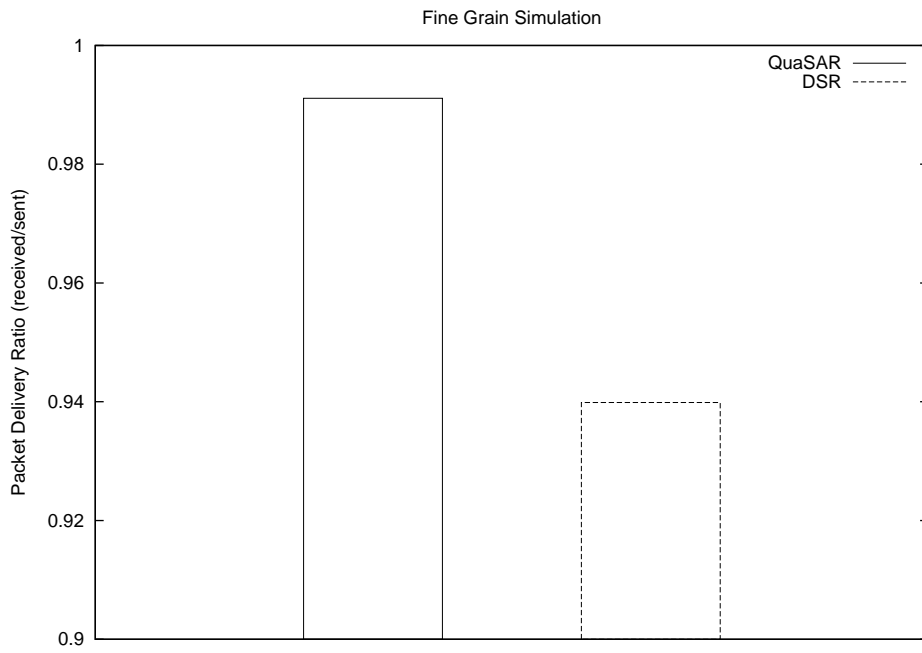


Figure 3.5: Fine Grain: Packet Delivery Ratio

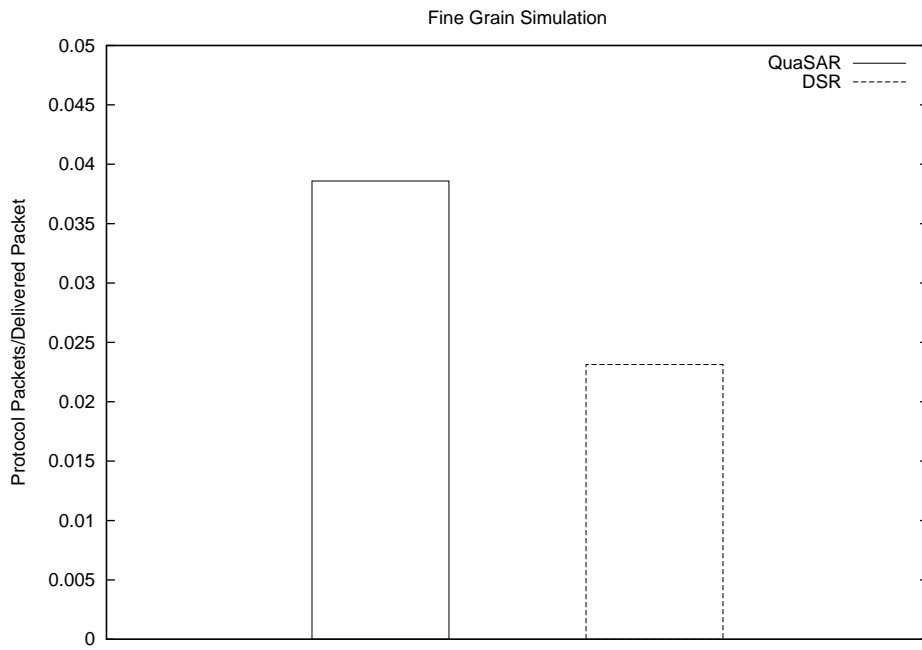


Figure 3.6: Fine Grain: Protocol Packets/Delivered Packet

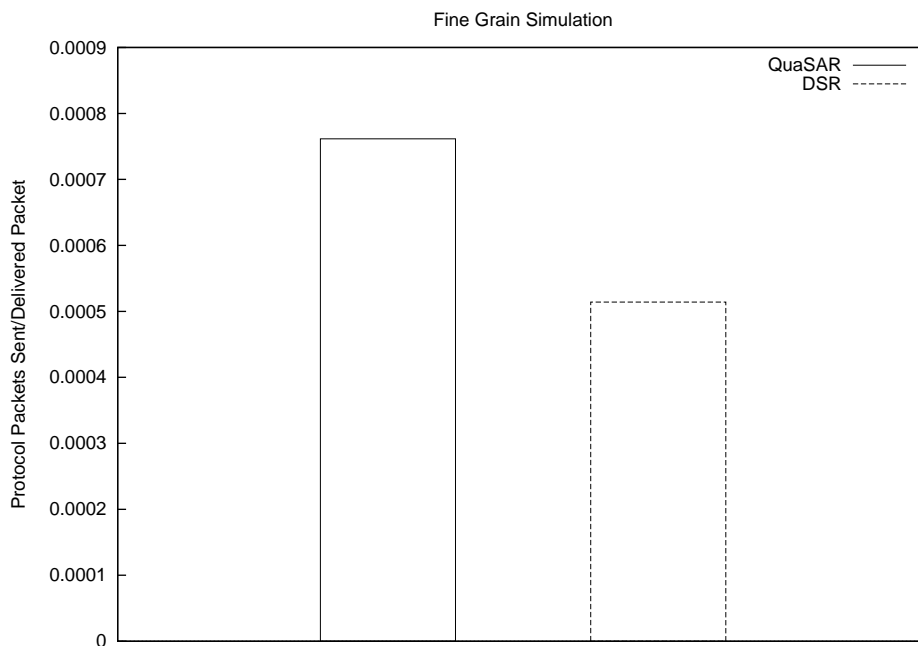


Figure 3.7: Fine Grain: Protocol Packets Sent/Delivered Packet

Table 3.2: Fine Grain Simulation Results

Protocol	Pk/Sec	Del. Ratio	(Pr. Pk)/Pk	(Pr. Pk Snt)/Pk	Pk/Rt	Latency
QuaSAR	48.2	99.1	0.038	0.0007	780.8	0.035
DSR	45.1	93.9	0.023	0.0005	457.1	0.056

### 3.1.3 Analysis

An analysis of the fine grain simulation and results follows.

#### Simulation Analysis

The simulation captured the problems DSR has when a network consists of MNs with diverse QoS. QuaSAR is aware of the battery power, signal strength, bandwidth and end-to-end latency of a route and will choose longer routes instead of shorter routes if the QoS demands are not met with the shorter routes. In this case the weak MNs are within one hop of the destination but they all have a weak link both in terms of bandwidth and signal

strength. The signal strength between the weak MNs and the destination is class 1, which is outside 90 percent of transmission range.

DSR is based on shortest route and will exhaust all the routes starting with the shortest, which in this scenario consist of the weak MNs routes before trying any longer routes. The two best routes in terms of QoS are the strong antenna routes though every link will die these routes will stay up longer than any other routes in the scenario. Since none of the routes will stay up throughout the simulation a route discovery must be initiated on a regular basis thus DSR will rediscover the weak MN links and the same problem takes place all over again. The DSR protocol is stuck in a situation it is not handling well at all.

QuaSAR on the other hand will switch between the two antenna routes as the source receives a Route Change Request (RCR) before the link is about to break. This switching will diminish the communication disruption time and the packet loss ratio.

### **Results Analysis**

The throughput increased with 3 packets/second. A packet size of 512 bytes will result in an increased throughput of 12 Kbps, a considerable improvement in such a scenario. The throughput increase is a result of the quality control in QuaSAR.

It can be seen from the packet delivery ratio that DSR has a significantly higher number of dropped packets than QuaSAR. QuaSAR has a delivery ratio of 99.1 percent whereas DSR is on 93.9 percent. The proactive route maintenance along with the QoS metrics in QuaSAR diminishes the packet loss by choosing more reliable routes and notifying the source when route critical incidents are in progress. These mechanisms will lead to an increased number of protocol packets in the network. The reason is that whenever the source receives an RCR packet and fails to find a QoS route it will start a route discovery. Since the routes with the weak MNs do not meet the signal strength demands the source will initiate a new route discovery when a strong antenna route breaks. The selective re-broadcasting feature

QuaSAR has is not an important factor in the overhead as the node density is low.

The simulation shows that using the QoS awareness and proactive route maintenance we have added in QuaSAR, will increase the throughput and the packet delivery ratio. To our knowledge this has not been shown before.

## 3.2 Course Grain

The course grain simulation is based on tools that create an environment based on certain distinctive scenario types. A randomly generated environment is a good way of testing whether a new protocol performs well compared to an already existing scheme. Since we are using tools to randomly generate a network it is not very reliable to give results based on only one simulation. We performed ten simulations where we for each created a new randomly generated scenario for the simulation. After ten simulations the results were averaged out. This will provide stronger and trustworthier results. In the next sections we present the simulation setup, results and an analysis.

### 3.2.1 Setup

Here follows a presentation of the scenario setup. All of the simulations were run for 100 seconds. We wanted to test the mobility models during different levels of mobility and varied the mobility. See Table 3.3.

Table 3.3: Course Grain Mobility Categories

Mobility	Max Speed (m/s)	Pause Time (seconds)
Low	4	30
Medium	10	20
High	20	10

## Mobile Nodes

For our course grain experiments we used 50 wireless nodes. These nodes were generated with randomized Quality of Service. However, all the active nodes in the network were given the same QoS. The Quality of Service of an inactive node in our network is presented in Table 3.4.

Table 3.4: Course Grain Mobile Node QoS

Category	Value
Bandwidth	Minimum 0.5 Mbps, Maximum 4.5 Mbps
Battery Power	2 percent of the MNs run out of battery
Transmission Power	Statically 250 meters

The QoS demands were in terms of end-to-end latency, bandwidth, signal strength and battery power. See Chapter 2.1 for a detailed introduction. In order to test QuaSAR properly we felt it was important that every node had the same QoS demands and QoS capacity for each experiment. This was to capture the trends in the network better as we changed the mobility. Thus all the 50 nodes were given the same QoS demands for each simulation.

## Environment Size

It was vital to pick an environment size that would give our implementation a fair chance of success. To do this we wanted to have an environment where the node density was reasonably high. The size of the grid was chosen to be 700 meters by 700 meters. This makes the node density in a perfectly distributed environment approximately 1 MN for each 100 meter by 100 meter. The transmission range was statically set to 250 meters, which means that every MN will have no less than 7 neighbors and at most 21 neighbors with a uniform distribution. See Figure 3.8.

A QoS oriented routing protocol must have a fair amount of routes to choose from in

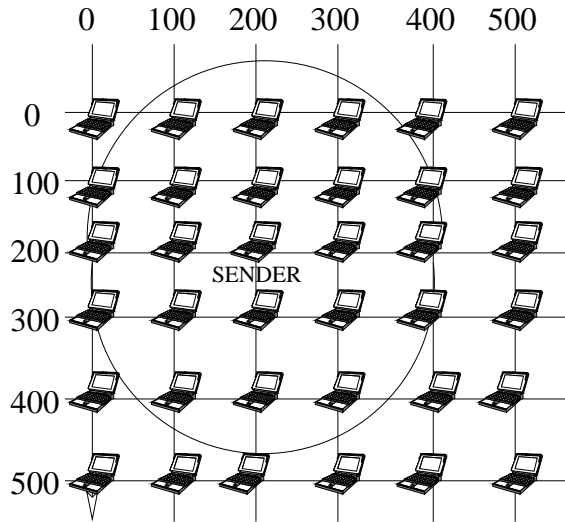


Figure 3.8: The Node density in the Course Grain Simulation is at most 21 neighbors with uniform distribution

a randomly generated environment, which we are going to test. The node density we get when picking 50 MNs and a 700-meter by 700-meter grid should provide a fair playground for QuaSAR and DSR.

### Communication and Movement Patterns

We used CMU’s traffic-pattern-generator [41] to randomly produce the communication patterns for our experiments. The parameters we used to create the communication patterns are presented in Table 3.5.

Category	Value
Number of Nodes	50 MNs
Maximum Number of Active Nodes	10 MNs
Packet Size	512 bytes
Constant Bit Rate	3 packets/second

Choosing a maximum of 10 active nodes in the network translates to maximum 20

percent of the nodes being active at the same time. This number indicates that the network is reasonably active. The packet size is set to 512 bytes and choosing 3 packets/second means that one node can produce 12 Kbps. We primarily wanted to test signal strength routing and the proactive route maintenance features of QuaSAR and by choosing a CBR this low we lessened the importance of choosing a high link capacity route.

QuaSAR has mechanisms that are triggered by mobility, thus we needed to simulate node movements as well. Mobility models can be classified into entity models and group models. In the entity models the actions of the MNs are completely independent. On the other hand in a group model there are several MNs that are moving together for example a group of soldiers. Group mobility has a cooperative nature.

We used a tool bonnMotion [17] for the movement pattern generation. BonnMotion is Java software that creates and analyses mobility scenarios. It is developed within the Communication Systems group at the Institute of Computer Science IV at the University of Bonn, Germany, where it serves as a tool for the investigation of mobile ad hoc network characteristics. BonnMotion supports these mobility models:

- Random Waypoint Model
- Gauss Markov Model
- Manhattan Grid Model
- Reference Point Group Mobility Model

Random Waypoint, Gauss Markov and Manhattan Grid model are all entity models. The Reference Point Group Mobility Model is a group model. A brief description of them is given.

**Random Waypoint Model** makes a node pick a random destination (waypoint) in the grid. The MN then starts moving towards this point in a straight line and at a

constant speed. When it gets there it waits for a predefined time before it starts the process again. The destination points are uniformly randomly distributed on the system area. For further reading about this model we refer to the reader to [8].

**Gauss Markov Model** was designed to adapt to different levels of randomness via one tuning parameter [12]. Initially every MN is assigned a current speed and direction. Then at fixed intervals the speed and direction is updated and movement takes place. The new speed and direction is calculated based upon the previous speed and direction, and a variable alpha that decides the randomness of the new values. The goal of this model is to avoid the sharp turns and the straight walk for example in Random Waypoint. For a thorough description of the model we refer the reader to [12].

**Manhattan Grid Model** is a city model that emulates the movement based on the streets and blocks of a city where you are restricted to walk along predefined paths and behavioral guidelines [12]. Very rarely can anybody walk completely random and this model aims to emulate MN movement in a city.

**Reference Point Group Mobility Model** is a group entity model. In this model there are  $n$  groups that move about randomly on the system area. In addition the MNs within the group also move randomly. Each group has a logical center that is used to calculate group movement. The movement of this center characterizes the speed and direction of the MNs belonging to the group. Each MN moves around their predefined reference point, which is completely dependant on the logical center of the group [12].

### **Wireless Channel Model**

As mentioned in section 2.3 we are assuming a TwoRayGroundModel [50] to be the platform for initiating the proactive maintenance based on signal strength weakening. We use this model in all of our simulations.



### 3.2.2 Results

We present the results from each of the mobility models mentioned above. They were tested on the metrics introduced in Table 3.1.

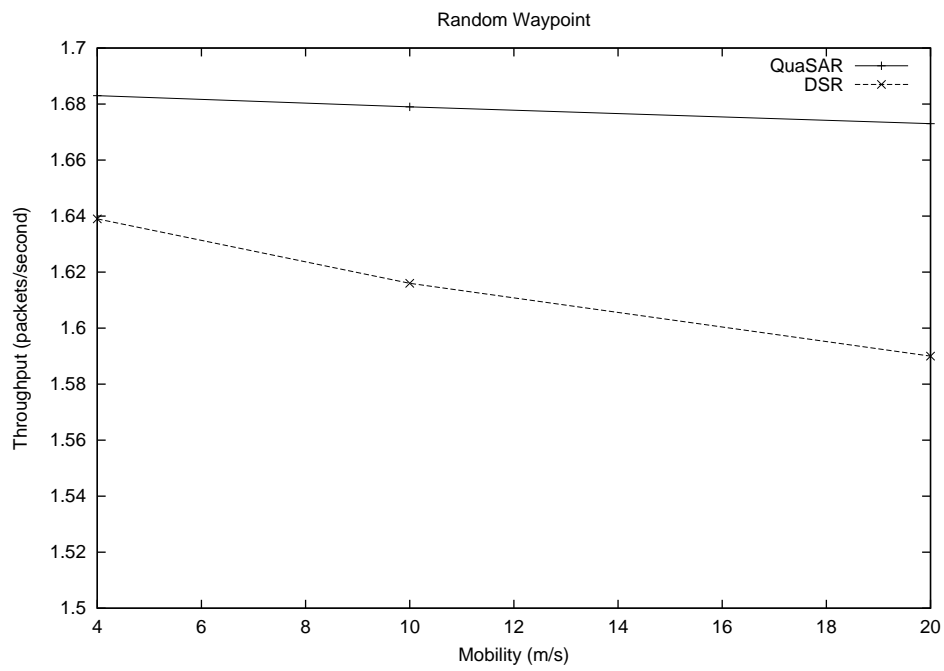


Figure 3.9: Random Waypoint: Throughput

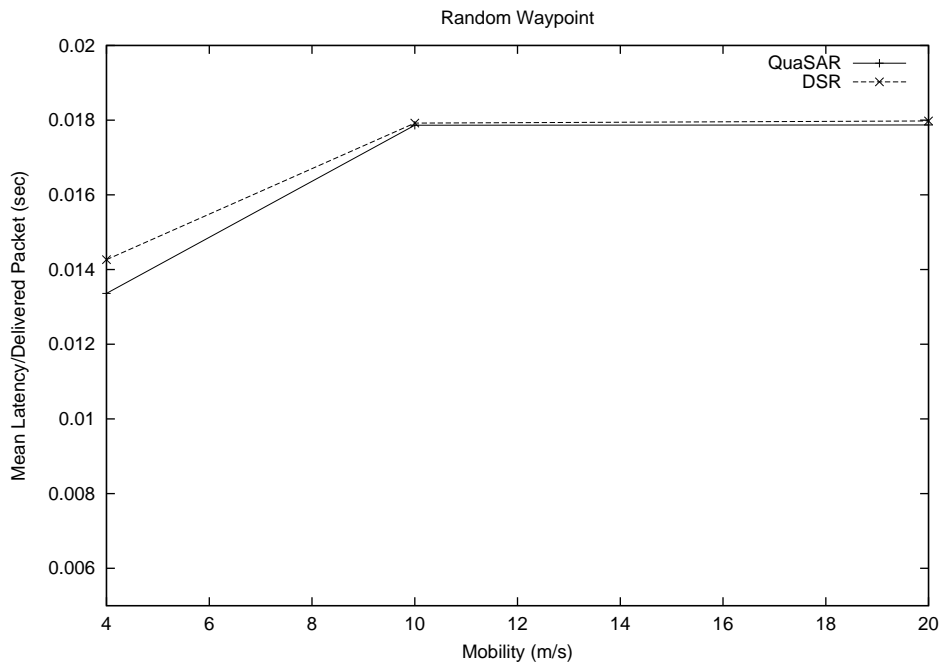


Figure 3.10: Random Waypoint: Mean Latency/Packet

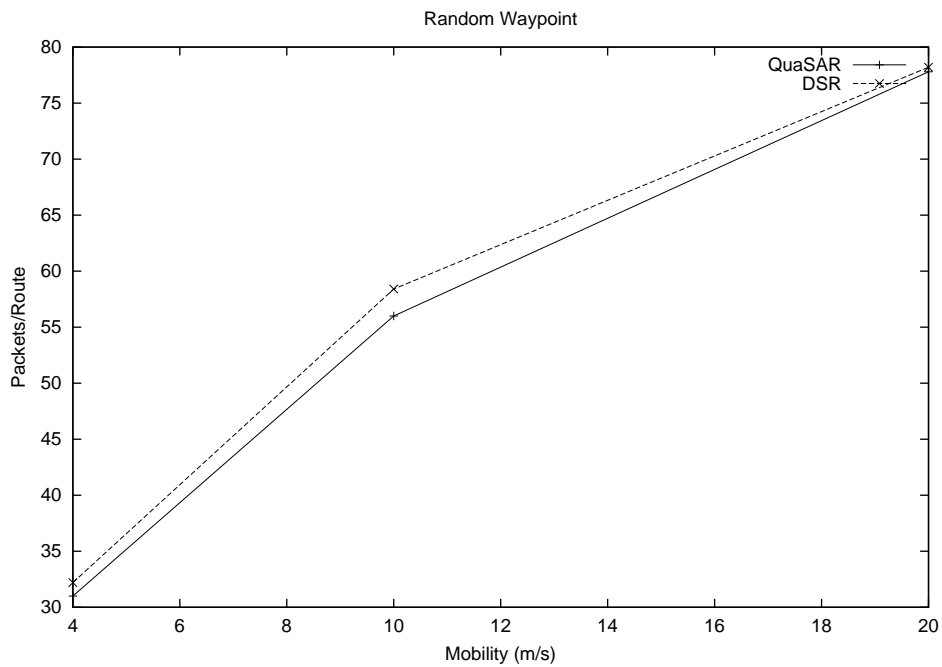


Figure 3.11: Random Waypoint: Packets/Route

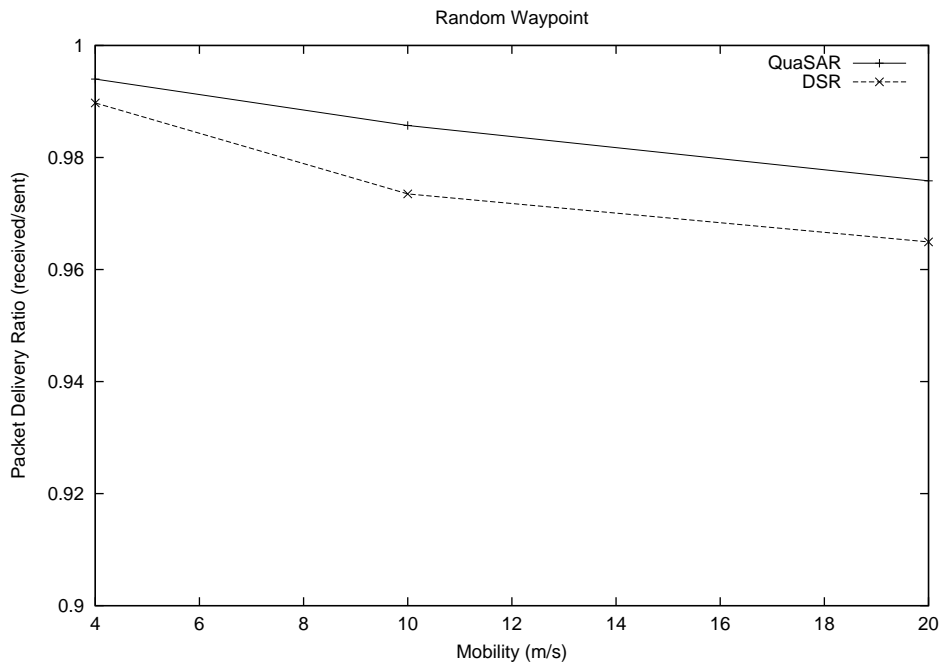


Figure 3.12: Random Waypoint: Packet Delivery Ratio

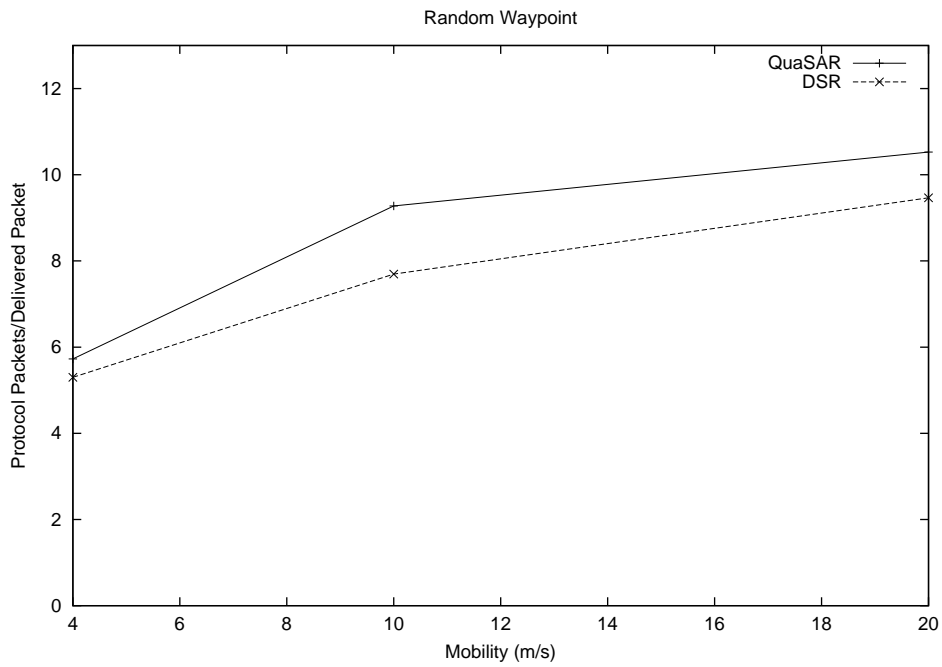


Figure 3.13: Random Waypoint: Protocol Packets/Delivered Packet

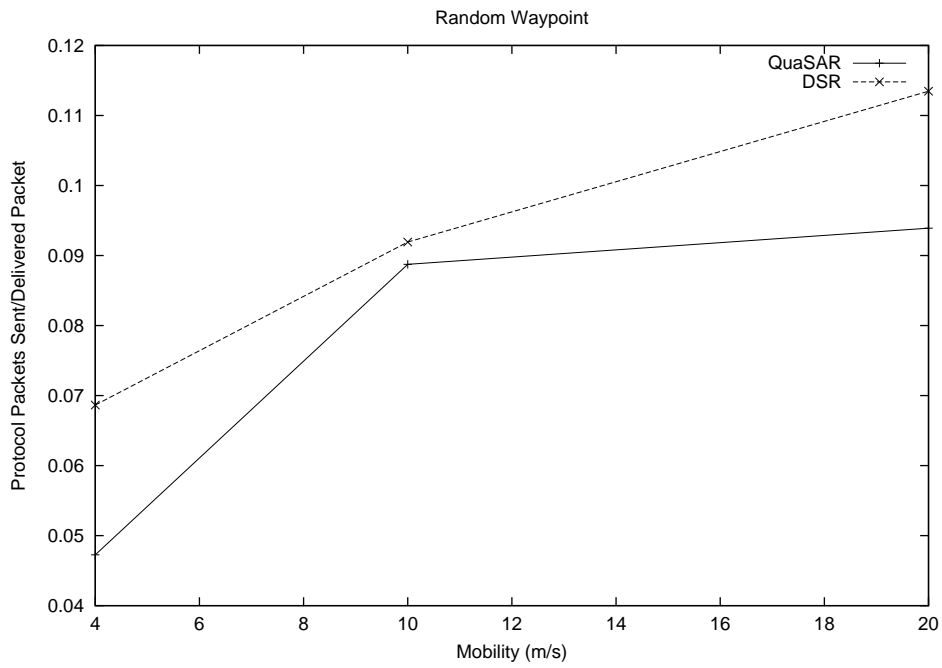


Figure 3.14: Random Waypoint: Protocol Packets Sent/Delivered Packet

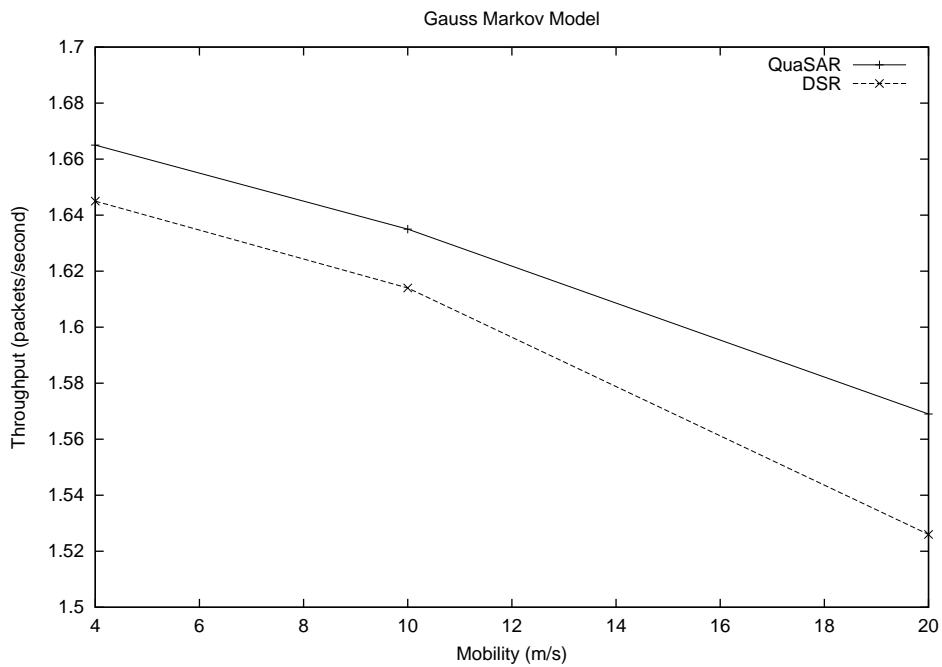


Figure 3.15: Gauss Markov: Throughput

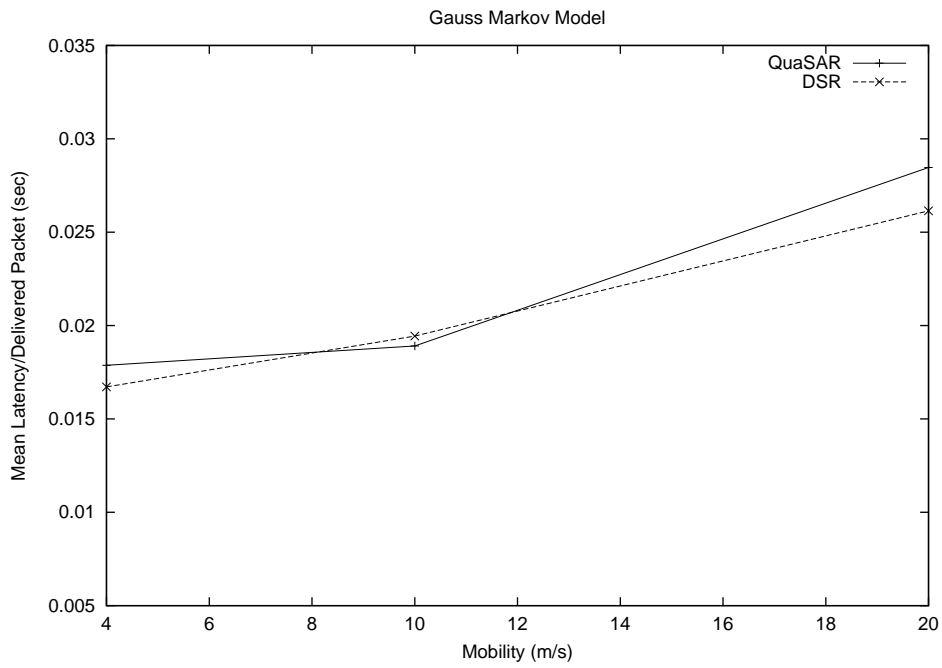


Figure 3.16: Gauss Markov: Mean Latency/Packet

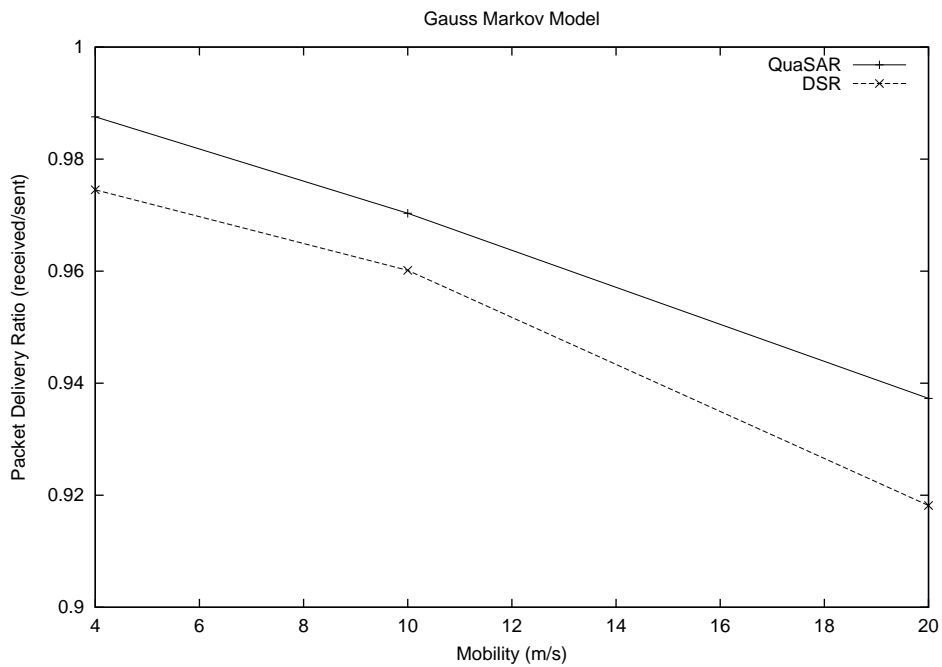


Figure 3.18: Gauss Markov: Packet Delivery Ratio

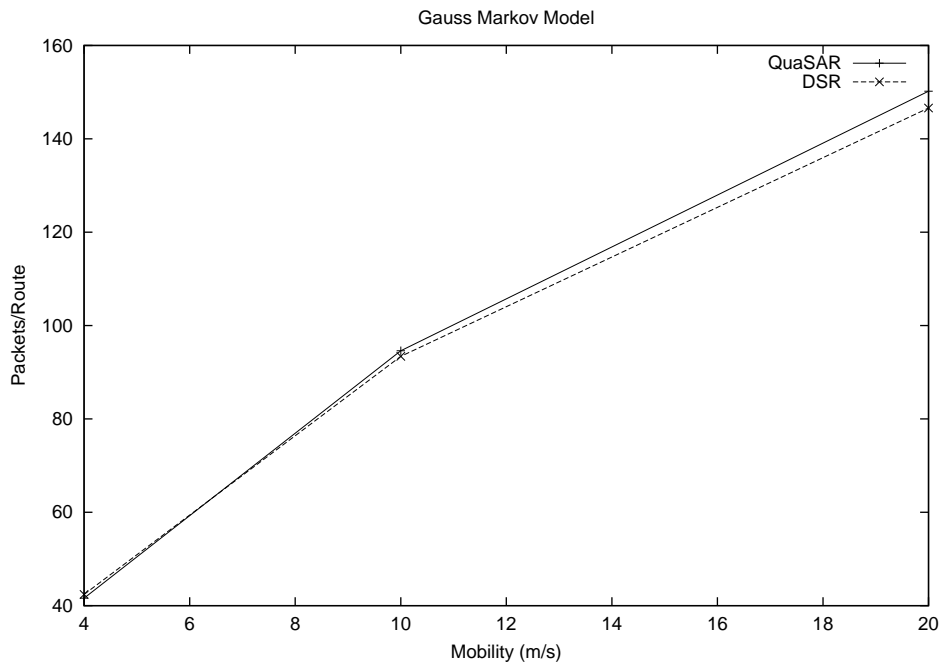


Figure 3.17: Gauss Markov: Packets/Route

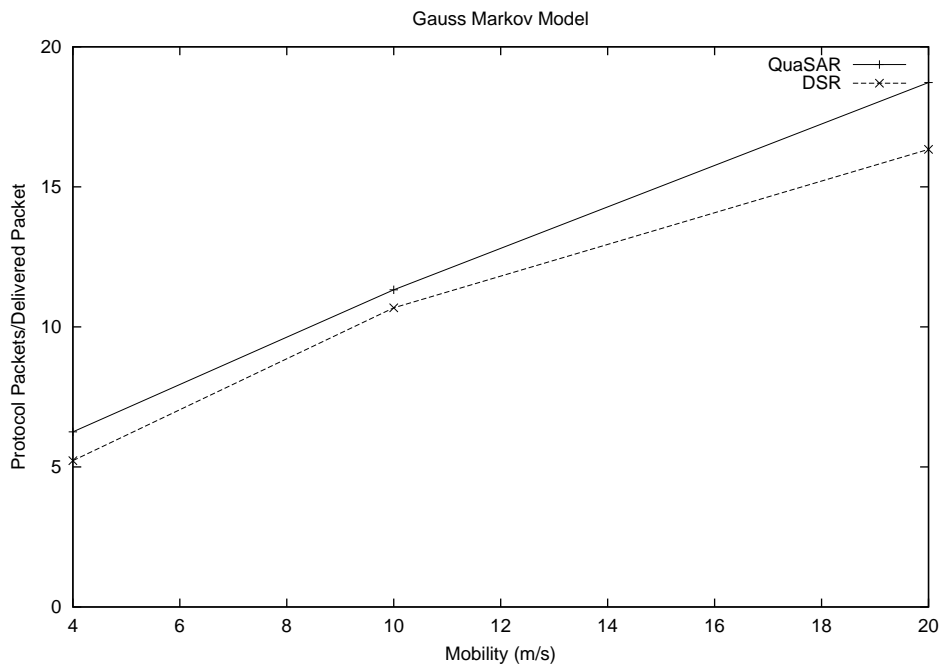


Figure 3.19: Gauss Markov: Protocol Packets/Delivered Packet

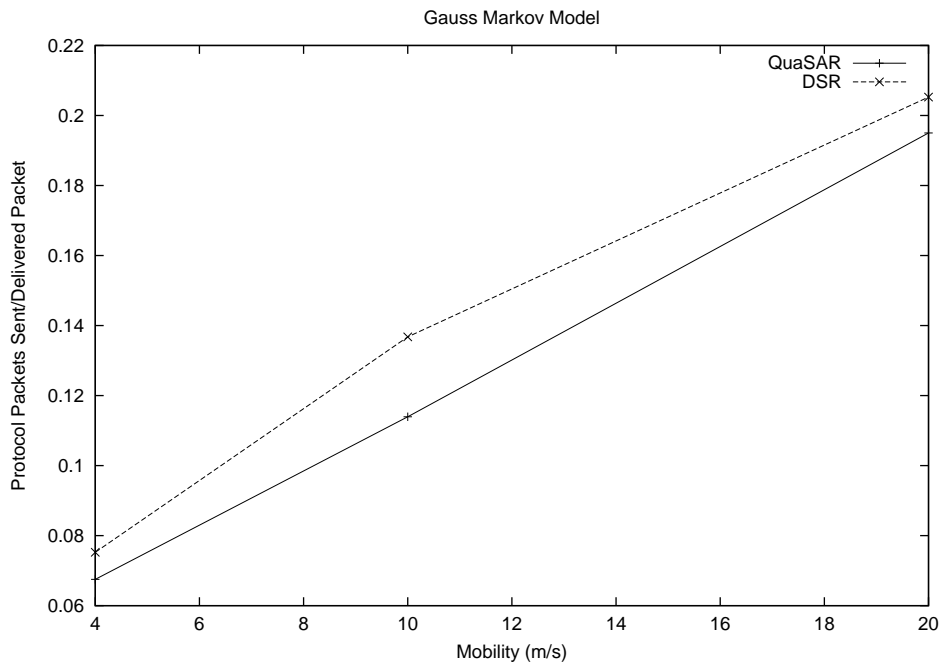


Figure 3.20: Gauss Markov: Protocol Packets Sent/Delivered Packet

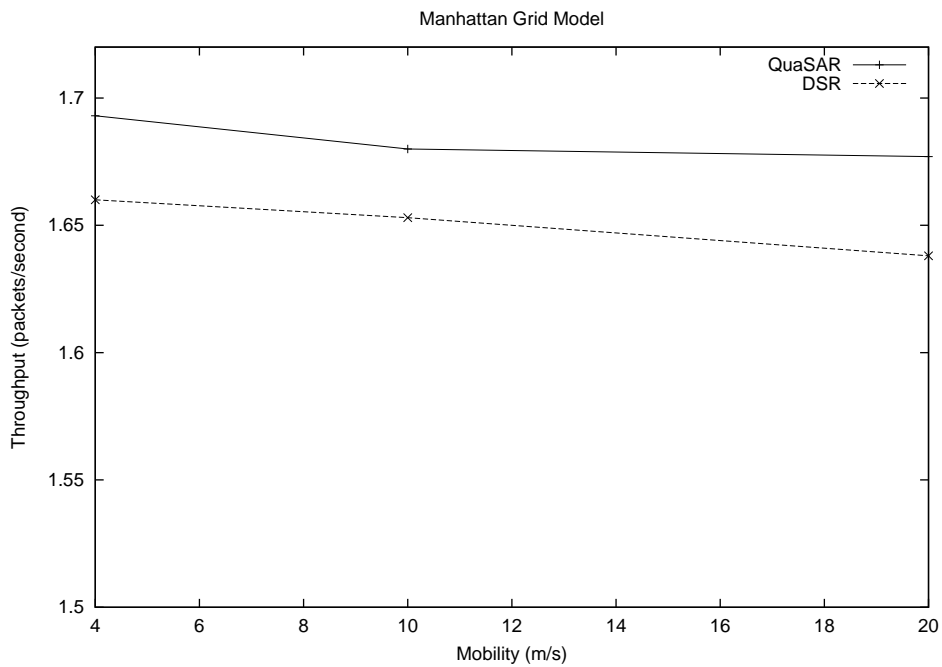


Figure 3.21: Manhattan Grid: Throughput

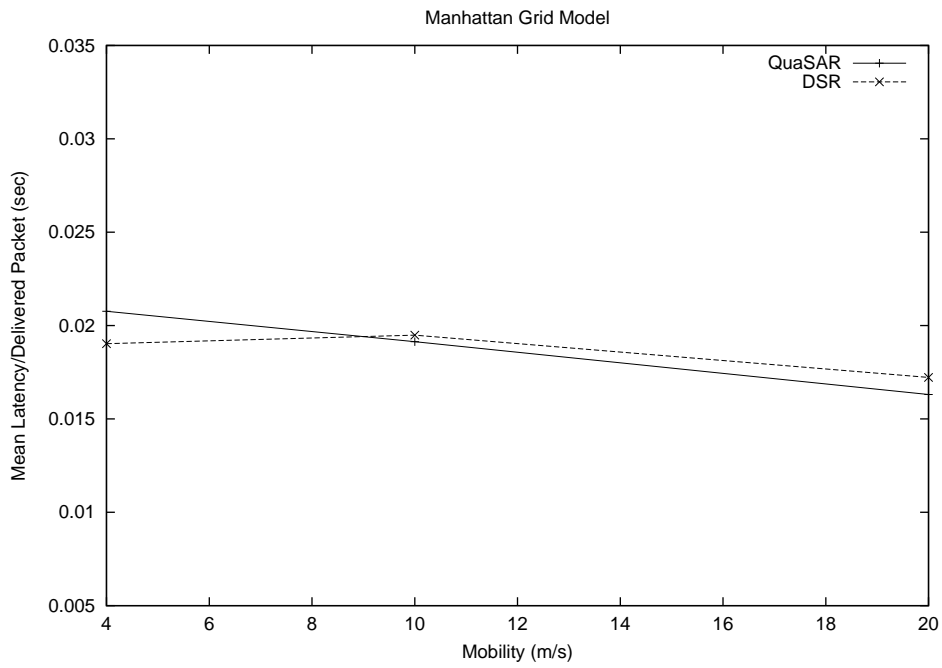


Figure 3.22: Manhattan Grid: Mean Latency/Package

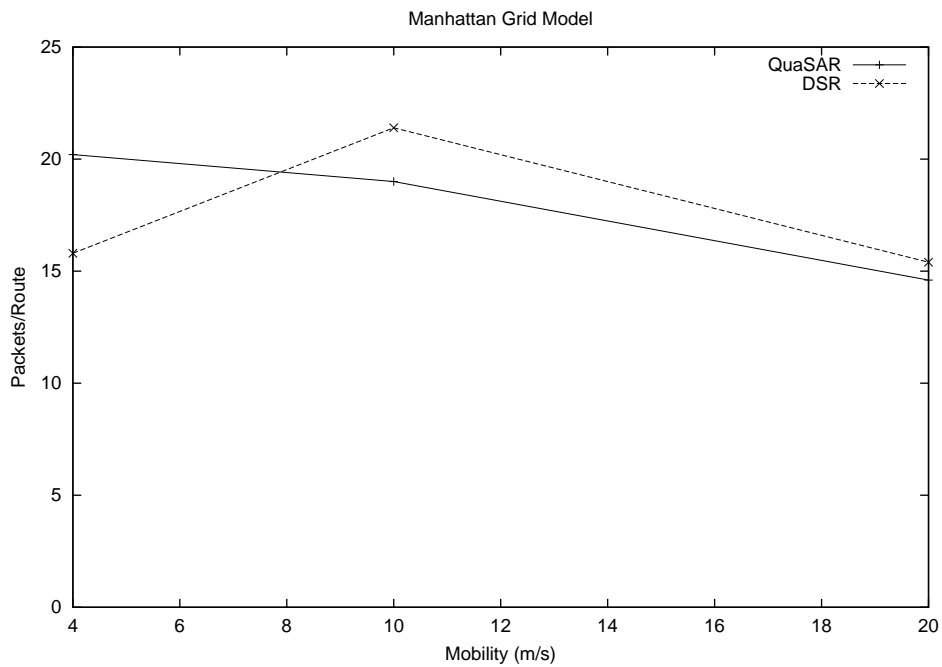


Figure 3.23: Manhattan Grid: Packets/Route



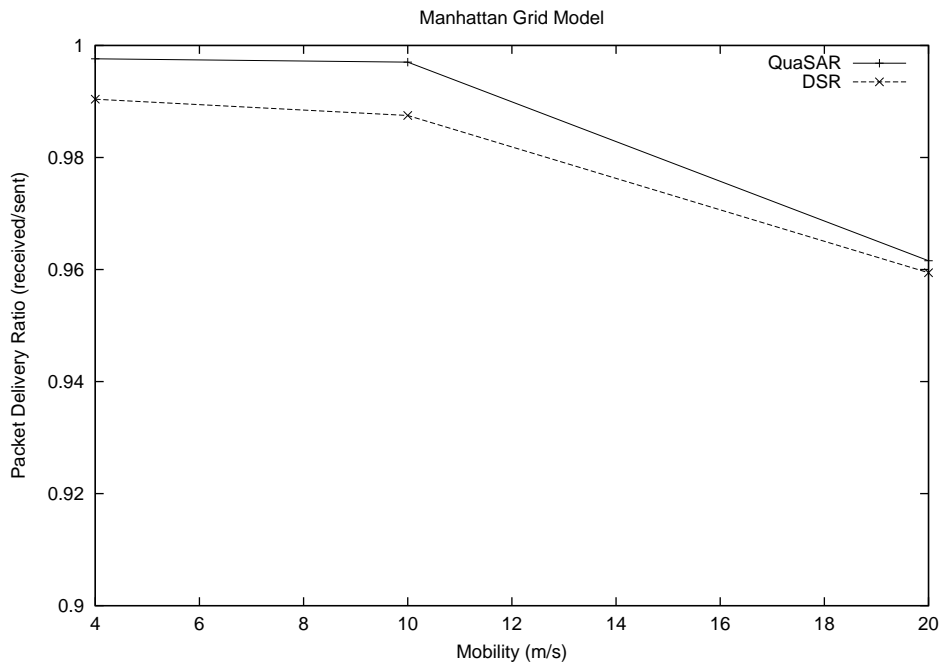


Figure 3.24: Manhattan Grid: Packet Delivery Ratio

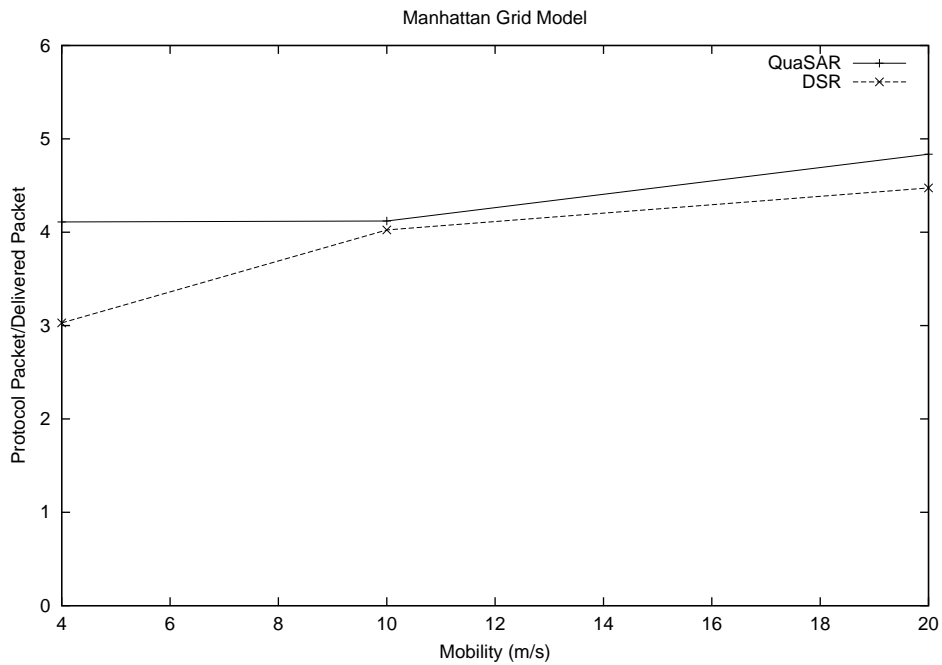


Figure 3.25: Manhattan Grid: Protocol Packets/Delivered Packet

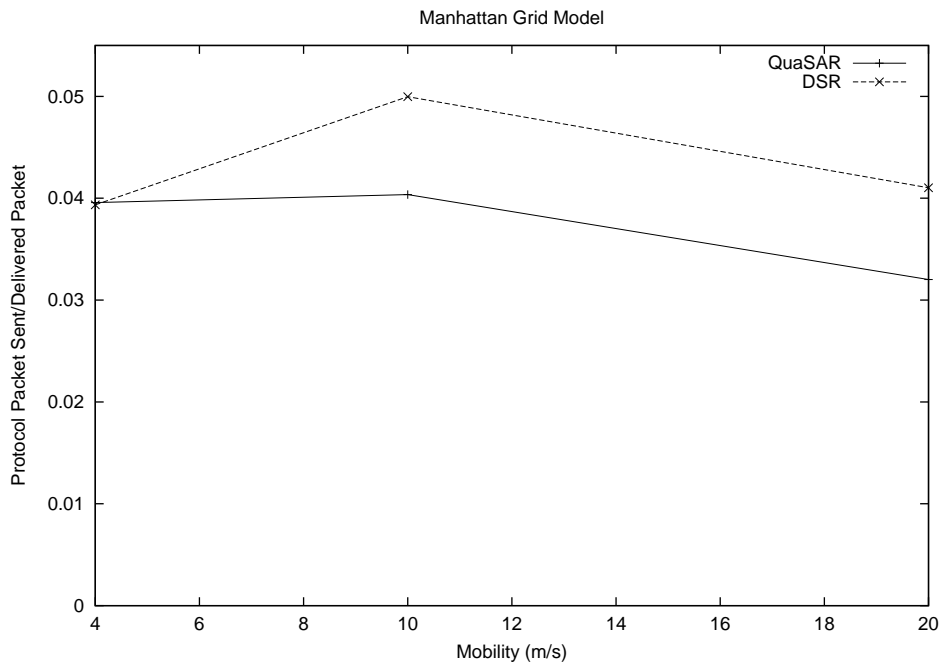


Figure 3.26: Manhattan Grid: Protocol Packets Sent/Delivered Packet

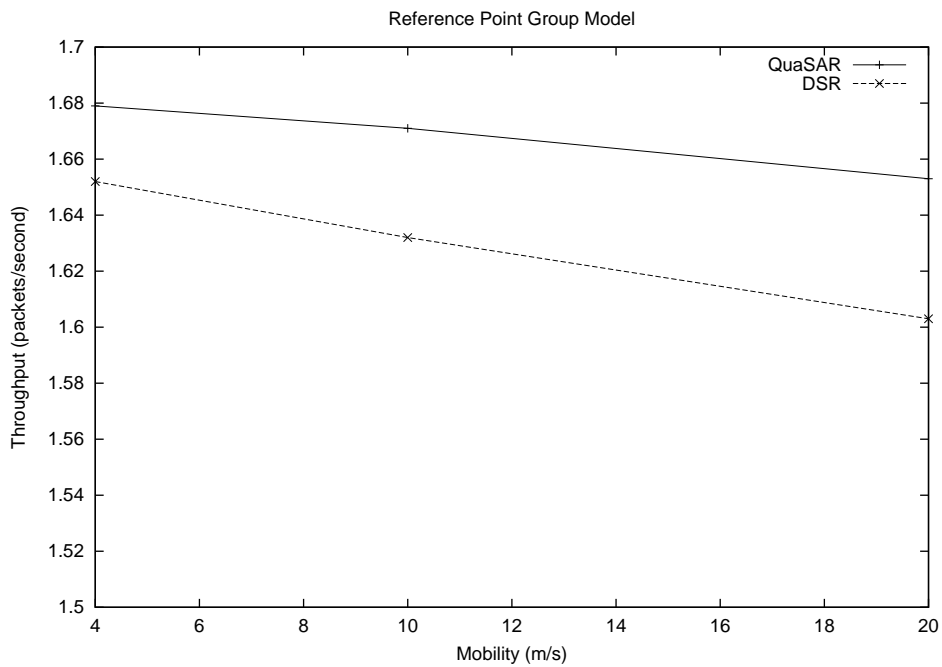


Figure 3.27: Reference Point Group: Throughput

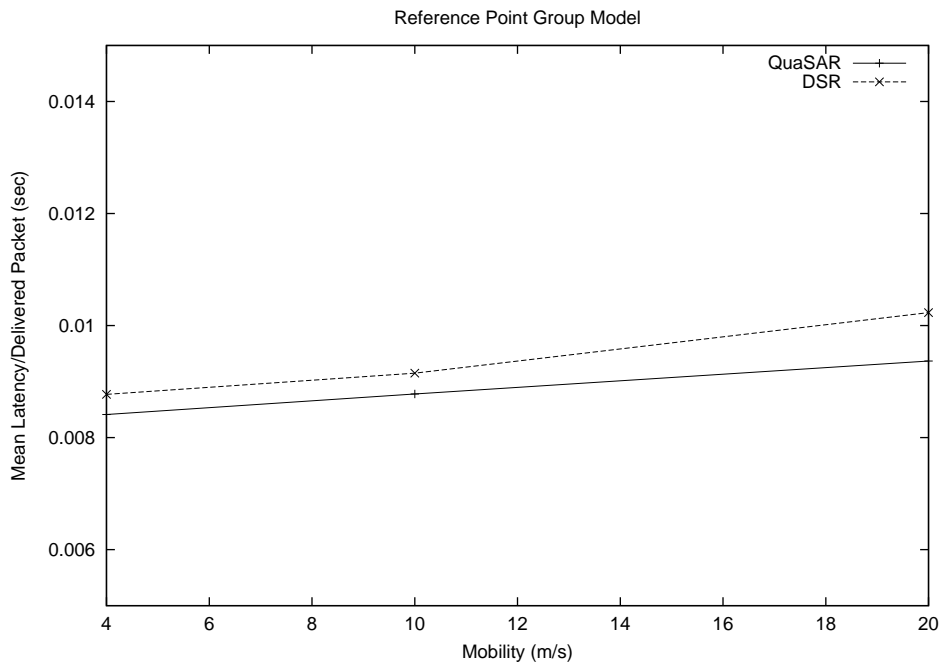


Figure 3.28: Reference Point Group: Mean Latency/Packet

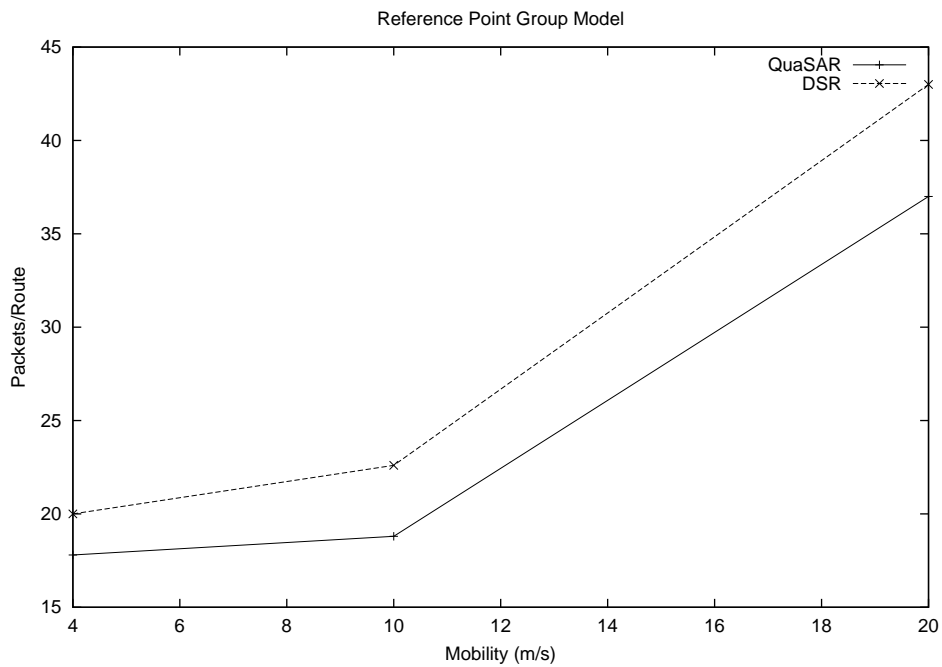


Figure 3.29: Reference Point Group: Packets/Route

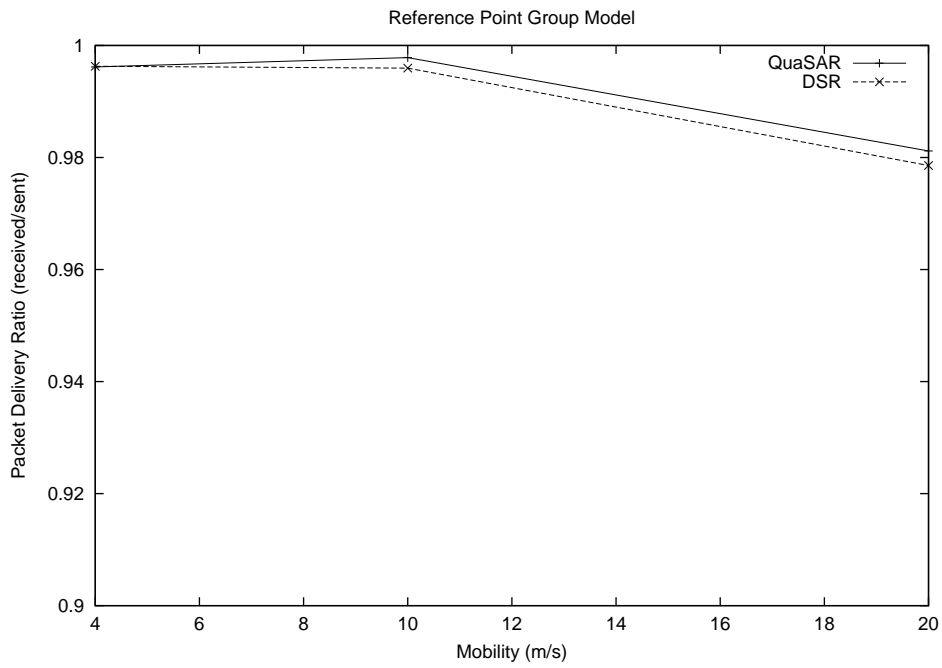


Figure 3.30: Reference Point Group: Packet Delivery Ratio

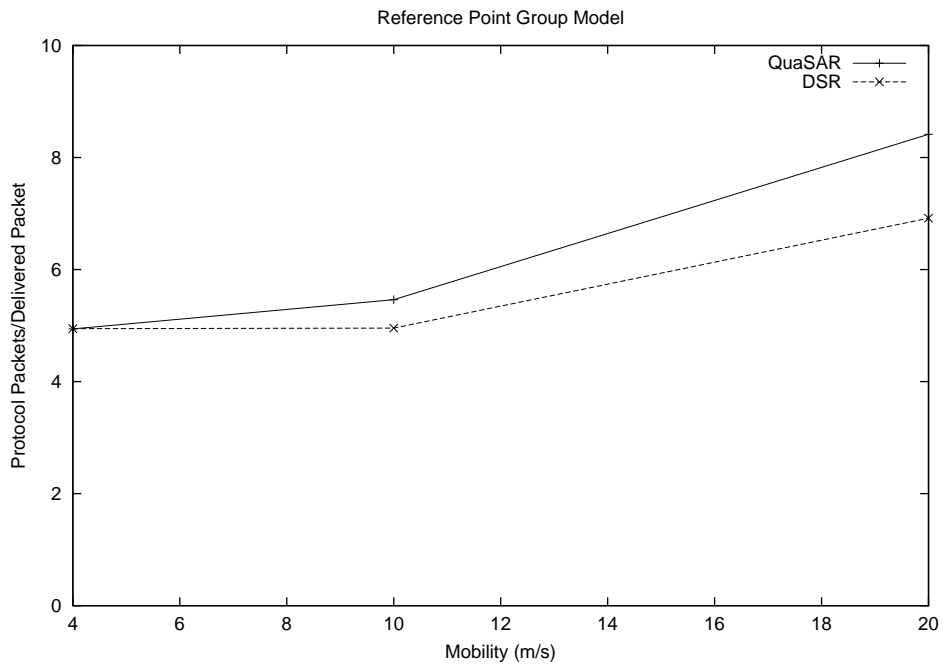


Figure 3.31: Reference Point Group: Protocol Packets/Delivered Packet

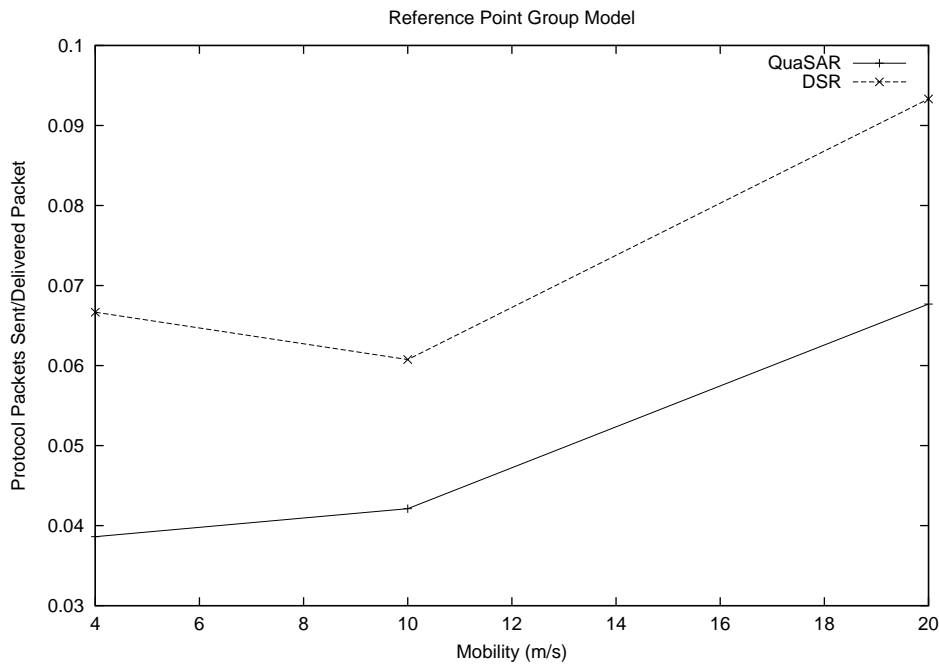


Figure 3.32: Reference Point Group: Protocol Packets Sent/Delivered Packet

### 3.2.3 Analysis

We analyze the results from the mobility models previously presented. Since the simulations are randomly generated we focus more on results analysis with a justification to how we believe the results came about with a basis in the mobility model. Random Waypoint and Gauss Markov Model will be analyzed together as we regard them similar models. Manhattan Grid and Reference Point Group Model are analyzed separately.

#### Random Waypoint and Gauss Markov Model

QuaSAR performs better than DSR in terms of throughput and packet delivery ratio. With Random Waypoint QuaSAR doesn't drop much at all in terms of throughput. For Gauss Markov however the throughput seems to be dropping at a similar rate for QuaSAR and DSR with QuaSAR a point better. This is probably due to the randomness of a walk in the Gauss Markov Model where MNs don't walk in a straight line and the speed varies.

The packet delivery ratio is close to 100 percent since the grid is pretty small and the node density is high, but the tendency is clearly in favor of QuaSAR. One of the goals of having the quality control in QuaSAR was to make it choose better routes yielding a higher probability of a successful delivery.

QuaSAR has a higher number of (protocol packets)/(packets received) than DSR. The proactive route maintenance of QuaSAR and the QoS demands makes the source look for routes more often than DSR. When a route critical incident is happening and an RCR packet is received by the source it results in a route discovery phase being initiated. That is if the source doesn't have a route that satisfies the QoS demands and a route discovery hasn't been performed in the last ten seconds where the ten seconds was chosen after an empirical study. In addition using selective re-broadcasting with high node density will cause a growth in QRREQs in the network. If the average number of neighbors is 21 the selective re-broadcasting overhead is a function of  $(x * 21)$  where  $x$  is the number of re-broadcasts. This is the main reason why QuaSAR has a higher number of (protocol packets)/(packets received). The number of (protocol packet sent)/(packets received) is actually lower for QuaSAR than DSR. This is probably due to the high number of route replies DSR sends back to the source when the node density is high.

The (mean latency)/packet is about the same. However, we had expected QuaSAR to have a slightly higher latency than DSR. The explanation may be that QuaSAR selects routes with better QoS for example bandwidth and although they are longer the latency is still low. One would think that choosing longer routes with stronger links would make the average packets per route increase, but QuaSAR and DSR seems to be performing about the same. One cause may be the usage of RCR packets where a route is preempted before it breaks and a route discovery is initiated. The route discovery will lead to new routes in the cache that the route choosing algorithm will pick accordingly.

Although the increase in throughput is very slight we believe that in a tougher environment with sources demanding more bandwidth the tendency will be stronger. We already see that the packet delivery rate is noticeably higher, which is evidently because QuaSAR chooses more robust routes than DSR.

### **Manhattan Grid Model**

The throughput statistics show that the Manhattan Grid Model has a higher throughput on average both for QuaSAR and DSR than Random Waypoint and Gauss Markov when the mobility increases. Having a mobility of 20 m/s in a city model is not very realistic, but for research purposes we chose to include this high mobility in the results.

QuaSAR has a slightly higher throughput than DSR in this model as well. The same tendencies as with Random Waypoint and Gauss Markov can be seen with the other statistics. The mean latency is about the same,  $(\text{protocol packets})/(\text{packets received})$  is higher with QuaSAR than with DSR but still  $(\text{protocol packets sent})/(\text{packets received})$  is lower. The delivery ratio shows that QuaSAR on the average picks route that have a higher chance of successfully delivering a packet.

The result that stands out is the packets/route where QuaSAR has significantly more packets/route during low mobility. However DSR has more packets/route when the mobility increases. This might suggest that QuaSAR is suited for city models since the average speed of a MN in a city will rarely be more than 10 m/s.

### **Reference Point Group Model**

The same tendencies was seen with this group model in terms of throughput,  $(\text{protocol packets})/(\text{packets received})$ ,  $(\text{protocol packets sent})/(\text{packets received})$  and delivery ratio. However, QuaSAR has lower  $(\text{mean latency})/\text{packet}$  than DSR for this group model. This may be because RPGM is a group model and routes are fairly stable. The number of packets

per route is not what was expected since it increased considerably with the mobility. It might be an implementation choice we are not aware of.



## Chapter 4

# Conclusions and Future Work

We have shown that introducing signal strength as a means of choosing and maintaining routes yields higher throughput and delivery ratio. In the fine grain simulation we aimed to find the best case scenario and found that the throughput increased significantly in QuaSAR compared to DSR. The fine grain simulation is in our opinion a realistic scenario and may occur a number of times during the lifetime of a MANET. It is impossible to find every possible case and certainly handle it. We must instead try to handle the scenarios that are bound to happen from time to time in the best possible fashion.

The results from the Mobility Models show that even with the totally random movement of Random Waypoint and Gauss Markov Model QuaSAR has a better throughput and delivery rate. In the Manhattan Grid Model and Reference Point Group Model the same tendencies continued.

QuaSAR uses selective re-broadcasting and though this means the protocol is more likely to find better routes it also causes more overhead to the network. Using selective route replies reduces the network activity sparing the source for useless routes but not enough to equal the re-broadcasting overhead. There is room for improvements in the route discovery phase of QuaSAR. Future work must include testing whether using selective re-broadcasting is worth the overhead and in fact proves to find better routes. An enhancement

of the selective re-broadcasting scheme is to include selective flooding where a number of nodes are chosen to rebroadcast a route request.

The QoS awareness and the proactive route maintenance used in QuaSAR proved to help the throughput and decrease the packet loss ratio in both the fine and course grain simulations we performed. To our knowledge this has not been shown before. Future work should include testing how the signal strength QoS demands will change the throughput. We also want to perform tests with higher constant bit rate and also varying bit rate to see how QuaSAR handles different situations. TCP is a natural choice of transmission protocol to include in this test. During mobility the routes will be changed continuously as links will die. But when there exist routes that stay active for a longer time the QoS of this route may change. Future work should include testing a QoS update mechanism where the QoS of a route is updated on a regular basis.

There are different metrics used to estimate a path break. Some of them were introduced in section 1.4.3. The common denominator is the use of ping/pong messages to verify the path break. We proposed using the number of transmissions left before a route break as the main part of the estimation metric. Using an estimation based preemption technique will not always work, but we believe that our scheme should be the standard approach. Future work must include testing the approaches against each other.

The QoS in QuaSAR does not include any reputation based metrics other than excluding weak routes. Adding more metrics will increase the overhead and complicate the route choosing even more. There should be research that tests whether more or less statistics is better and in what scenario the statistics are especially useful. There are scenarios of high mobility when having a lot of metrics yields no advantage at all.

We have not handled the route cache issues involved when having QoS Routing. The QoS of a route is source-to-destination and does not regard the intermediate hops. To increase

the route cache effect there should be mechanisms that saves the per hop QoS, such that the number of route discoveries decreases. This assumes that an intermediate MN can reply to a QRREQ when it has a route to the destination satisfying the QoS. Future work should include testing how route caches can be stored efficiently and still preserve the QoS for each route. In addition there should be tests to how the cache can increase the performance of an ad-hoc network.

# Bibliography

- [1] Open System Interconnection Model, OSI 7498.
- [2] Norman Abramson. ALOHAnet (c), 1970.
- [3] Sulabh Agarwal, Ashish Ahuja, Jatinder Pal Singh, and Rajeev Shorey. Route-Lifetime Assessment Based Routing (RABR) Protocol for Mobile Ad-Hoc Networks. In *ICC (3)*, pages 1697–1701, 2000.
- [4] Ben Liang And. Optimizing Route-Cache Lifetime in Ad Hoc Networks.
- [5] Khaldoun Al Agha Anelise Munaretto, Hakim Badis and Guy Pujolle. A Link-state QoS Routing Protocol for Ad Hoc Networks. Technical report, 2002.
- [6] Christina Aurrecochea, Andrew T. Campbell, and Linda Hauw. A Survey of QoS Architectures. *Multimedia Systems*, 6(3):138–151, 1998.
- [7] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A distance routing effect algorithm for mobility (dream). In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 76–84. ACM Press, 1998.
- [8] Christian Bettstetter, Hannes Hartenstein, and Xavier P&#233;rez-Costa. Stochastic properties of the random waypoint mobility model: epoch length, direction distribution, and cell change rate. In *Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, pages 7–14. ACM Press, 2002.

- [9] Azzedine Boukerche and Liqin Zhang. A preemptive on-demand distance vector routing protocol for mobile and wireless ad hoc networks. In *Proceedings of the 36th Annual Simulation Symposium*, page 73. IEEE Computer Society, 2003.
- [10] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Mobile Computing and Networking*, pages 85–97, 1998.
- [11] I. Busse, B. Deffner, and H. Schulzrinne. Dynamic QoS control of multimedia applications based on RTP, Jan 1996.
- [12] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.
- [13] A. Campbell and G. Coulson. A Quality of Service Architecture, Apr 1994.
- [14] Shigang Chen and Klara Nahrstedt. A Distributed Quality-of-Service Routing in Ad-Hoc Networks. *IEEE Journal on Selected Areas in Communications*, 17(8), August 1999.
- [15] Yuh-Shyan Chen, Yu-Chee Tseng, Jang-Ping Sheu, and Po-Hsuen Kuo. On-Demand, Link-State, Multi-Path QoS Routing in a Wireless Mobile Ad-Hoc Network.
- [16] Z. Chen, S. Tan, R. Campbell, and Y. Li. Real time video and audio in the World Wide Web. *Proc. Fourth International World Wide Web Conference*, 1995.
- [17] Michael Gerharz Christian de Waal. BonnMotion (c), 2002-2003.
- [18] R. Dube, C. Rais, K. Wang, and S. Tripathi. Signal stability based adaptive routing (SSA) for ad hoc mobile networks, Feb 1997.
- [19] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Technical report, 1998.
- [20] H.T. Friis. A note on a simple transmission formula, May 1946.

- [21] Chane L. Fullmer and J. J. Garcia-Luna-Aceves. Solutions to hidden terminal problems in wireless networks. In *Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 39–49. ACM Press, 1997.
- [22] Tom Goff, Nael Abu-Ghazaleh, Dhananjay Phatak, and Ridvan Kahvecioglu. Preemptive routing in ad hoc networks. *J. Parallel Distrib. Comput.*, 63(2):123–140, 2001.
- [23] J Haartsen. The Bluetooth Radio System. *IEEE Personal Communications*, pages 28–36, Feb 2000.
- [24] Tomasz Imielinski and B. R. Badrinath. Mobile Wireless Computing: Challenges in Data Management. *Communications of the ACM*, 37(10):18–28, 1994.
- [25] David B Johnson and David A Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.
- [26] R. Kay. Time synchronization in ad hoc networks. In *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 173–182. ACM Press, 2001.
- [27] Kevin Lai and Mary Baker. Measuring Bandwidth. In *INFOCOM (1)*, pages 235–245, 1999.
- [28] Will E. Leland, Murad S. Taqq, Walter Willinger, and Daniel V. Wilson. On the self-similar nature of Ethernet traffic. In Deepinder P. Sidhu, editor, *ACM SIGCOMM*, pages 183–193, San Francisco, California, 1993.
- [29] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of Ad Hoc wireless networks. In *Mobile Computing and Networking*, pages 61–69, 2001.
- [30] J. Li M. Jiang and Y.-C. Tay. Cluster based routing protocol (CBRP) functional specification (Internet-Draft), Aug 1998.

- [31] Peter Martini Michael Gerharz, Christian de Waal and Paul James. Strategies for Finding Stable Paths in Mobile Wireless Ad Hoc Networks. *Proc. of the 28th Annual IEEE Conference on Local Computer Networks (LCN)*, pages 130–139, October 2003.
- [32] J. Moy. The OSPF Specification. Technical Report 1131, 1989.
- [33] Shree Murthy and J. J. Garcia-Luna-Aceves. A Routing Protocol for Packet Radio Networks. In *Mobile Computing and Networking*, pages 86–95, 1995.
- [34] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162. ACM Press, 1999.
- [35] U.S. Department of Defense and Ivan Getting. Global Positioning System (GPS).
- [36] V. Park and S. Corson. Temporally ordered routing algorithm (TORA) version 1, functional specification, 1998.
- [37] C. Perkins. Ad-hoc on-demand distance vector routing, Nov 1997.
- [38] C. E. Perkins and A. Myles. Mobile IP. *Proceedings of International Telecommunications Symposium*, pages 415–419, 1994.
- [39] Charles Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.
- [40] J. Postel and J. Reynolds. File Transfer Protocol (FTP), Oct 1985.
- [41] The Rice Monarch Project. Traffic Pattern Generator for Network Simulator (ns) (version 2).
- [42] S. Ramakrishna and J. Holtzman. A Scheme for Throughput Maximization in a Dual-Class CDMA System, Oct 1997.

- [43] T. S. Rappaport. *Wireless communications, principles and practice*, 1996.
- [44] E. Royer and C. Toh. A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks, Apr 1999.
- [45] Martin Sanfridson. Problem Formulations for QoS Management in Automatic Control. Technical report, S-100 44 Stockholm Sweden, 2000.
- [46] Dheeraj Sanghi, Ashok K. Agrawala, Olafur Gudmundsson, and Bijendra N. Jain. Experimental Assessment of End-to-End Behavior on Internet. In *INFOCOM (2)*, pages 867–874, 1993.
- [47] W. Su, S. Lee, and M. Gerla. *Mobility Prediction and Routing in Ad Hoc Wireless Networks*, 2000.
- [48] Chai-Keong Toh. Associativity-based routing for ad hoc mobile networks. *Wirel. Pers. Commun.*, 4(2):103–139, 1997.
- [49] UCB/LBNL/VINT. *Network Simulator (ns) (version 2)*.
- [50] UCB/LBNL/VINT. *Network Simulator (ns) (version 2) Documentation*.
- [51] Yu Wang, Weidong Chen, and Joseph Ho. Performance Analysis of Adaptive Location Management for Mobile IP.
- [52] Zheng Wang and Jon Crowcroft. Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE Journal of Selected Areas in Communications*, 14(7):1228–1234, 1996.
- [53] P. White. *RSVP and integrated services in the internet: A tutorial*, May 1997.
- [54] H. Xiao, W. Seah, A. Lo, and K. Chua. A Flexible Quality of Service Model for Mobile Ad-Hoc Networks.
- [55] X. Xiao and L. M. Ni. Internet QoS: A Big Picture. *IEEE Network*, 13(2):8–18, March 1999.



- [56] Qi Xue and Aura Ganz. Ad hoc qos on-demand routing (AQOR) in mobile ad hoc networks. *J. Parallel Distrib. Comput.*, 63(2):154–165, 2003.
- [57] Louise Lamont Ying Ge, Thomas Kunz. Proactive QoS Routing in Ad Hoc Networks. In *Ad-Hoc, Mobile, and Wireless Networks, Second International Conference ADHOC-NOW*, pages 60–71, October 2003.
- [58] C. Zhu and M. Corson. QoS routing for mobile ad hoc networks, June 2001.