A novel DHT Routing Protocol for MANETs

by

Deepak Chellamani

B.E. (Electrical and Electronics Engineering),

Anna University, Chennai, India

May 2005

Submitted to the graduate degree program in Electrical Engineering and Computer Science and the Graduate Faculty of the University of Kansas School of Engineering in partial fulfillment of the requirements for the degree of Master of Science

Dr. Joseph Evans, Chair

Dr. Arvin Agah, Member

Dr. Gunes Ercal-Ozkaya, Member

Date Defended

The thesis committee for Deepak Chellamani certifies that this is the approved version of the following thesis

A novel DHT Routing Protocol for MANETs

Dr. Joseph Evans, Chair

Dr. Arvin Agah, Member

Dr. Gunes Ercal-Ozkaya, Member

Date Defended

To my family and friends

ACKNOWLEDGEMENTS

This thesis is the result of the help and support of many people.

I would like to thank my advisor Dr Joseph Evans, my committee members Dr Gunes Ercal-Ozkaya and Dr Arvin Agah for providing me with their valuable suggestions and insight. I want to express my sincere gratitude for their invaluable supervision, the opportunity and environment for a very rewarding learning experience.

I would like to thank Dr. Steven Schrock and Dr. Bryan Young for providing me an opportunity to work as Research Assistant in the KDOT MQA project that helped me hone my programming skills.

I am grateful especially to my father Chellamani Ramarathnam, my mother Kalpana Chellamani, and rest of my family members for their love and encouragement throughout my life. I want to thank Muthukumaran Pitchaimani of University of Kansas, for his guidance at various stages of my research experience at KU. I would also like to thank my roommates and friends for their support.

ABSTRACT

The central challenge in Mobile Ad hoc Networks (MANETs) is to provide a stable routing strategy without depending on any central administration. This work presents and examines the working of Radio Ring Routing Protocol (RRRP), a DHT based routing protocol for MANETs inspired from structured overlays in the internet. This design joins effort in answering the fundamental question of efficiency of a DHT substrate [1] compared to conventional routing in ad hoc networks.

Table of Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 Brief History of Wireless Networks

With the advent of portable devices like laptops and handheld PDAs, wireless networks have emerged as the preferred medium of communication in the past decade. With mobile connectivity, these mobile devices also provide a myriad of application services like email and web browsing for the users. Two major classifications of wireless networks are *infrastructured networks* and *infrastructureless networks* or *Mobile Ad-hoc Networks* (MANETs).

In an infrastructured network, the mobile units are connected using bridges known as base stations. Once the mobile units move, a "handoff" occurs as they go out of the range of one base station and into the range of another, thus the mobile node can communicate seamlessly [2]. Typical examples of this type of networks are Wireless Local Area Networks (WLANs) and cellular networks. Currently most wireless connections are infrastructured, which exists between cell phones or laptops connected together by a service provider via access points. Although infrastructured networks provide a great way for mobile communication, the cost and time associated with its setup can be high.

Infrastructureless networks or MANETs are decentralized mobile wireless networks comprised of computing devices that operate without any central administration or an access point. With the advance of technology and vast requirements of communication, research on wireless connectivity is focused on enabling mobile devices to connect with each other in absence of a central administration system [3]. Towards this end, MANETs have gained an increased attention among researchers in recent years.

1.2 Mobile Ad-hoc Networks

Researchers visualize MANETs to be an integral part of 4G architecture and in the next generation networks [3]. MANETs consist of rapidly changing network topology as nodes move in a random manner. They can operate either standalone or may be connected to a larger internet. Due to the absence of fixed infrastructure, nodes setup routes among themselves autonomously. Nodes in a MANET (laptops, handheld PDAs, and so on) move arbitrarily and communicate directly with other nodes sharing the same media (radio, infrared, etc.) within their radio transmission range. Beyond this range, message transfer occurs through hop-by-hop communication. Figure 1.1 shows a typical setup of a MANET.

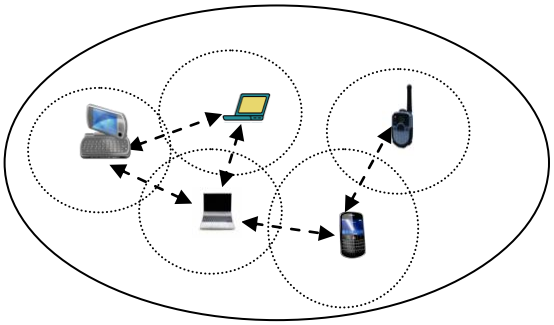


Figure 1.1 Typical MANET setup

Since MANETs are flexible and self-configurable, they enjoy a wide variety of commercial and emergency applications that require a quick deployment of information sharing. Some of the common applications of MANETs include military battlefield communications, disaster-relief scenarios, rural communications, and law enforcement operations.

1.3 Routing in MANETs

Along with the various previously mentioned benefits, MANETs suffer from traditional wireless communication problems like bandwidth optimization, power control, and mobility. Designs involving MANETs must consider its unique problems like link quality variations, network scalability, and network robustness. Since the nodes act as both an end system and a

router, batteries carried by each mobile device have limited processing power thereby limiting services and applications supported by the node. These issues and constraints provide a host of challenges for implementing MANETs. A major group of research has addressed and proposed a variety of solutions for the above-mentioned issues and constraints. These challenges and the critical importance of providing a stable routing strategy makes routing in ad hoc networks the most active research interest in the last few years.

The highly dynamic nature of MANETs causes frequent changes in network topology and makes routing among mobile nodes complex and nontrivial [3]. The primary objective of an ad hoc routing protocol is an efficient and correct route establishment among mobile nodes. The present routing strategies like distance-vector and link-state mechanisms are not suitable for a frequently changing topological environment due to their excessive use of bandwidth and large overhead; thus results in performance degradation. Ad hoc routing protocols must consider their specific needs, especially; mobility and bandwidth constraints. The main design criteria for the routing protocols in MANETs are as follows:

- Simplicity and ease of implementation
- Scalable and reliable
- Distributed and lightweight
- Rapid route convergence

Over the past decade, much research proposes different routing strategies specifically designed for efficient and stable routing in MANETs considering the above-mentioned design issues. The current standardized routing protocols can be broadly classified into three types: proactive, reactive, and hybrid.

1.4 MANETs and Peer-to-Peer

The challenges similar to that of MANETs also arise in peer-to-peer (p2p) overlay networks as they share significant characteristics like decentralization and self-configuration.

The resiliency, redundancy, and dynamic fault tolerance of the p2p networks initially created for file sharing purposes attract researchers to utilize p2p strategies in MANETS. This synergy is possible by their frequently varying network topologies [4]. In MANETs, network topological changes occur due to terminal mobility of nodes; while in p2p overlay networks, it is due to random joining and leaving of clients. P2p applications (like Gnutella[5] and Napster[6]) perform well in serverless architectures that makes them potentially suitable for MANET environments [7].

The synergy and similarities between MANETs and p2p fuel to provide a stable and robust routing strategy for MANETs inspired from structured overlay routing protocols. Existing studies show that MANETs support an effective abstraction of p2p techniques and topologies. Since nodes in MANETs behave as routers and as end hosts, these p2p abstractions can be applied at the network layer as in [8] or above in the application layer as in [9].

1.5 Subsequent Chapters

The structure of rest of the report is as follows. Chapter 2 describes the current types of standardized ad hoc routing protocols. It also explores the motivation and various possible design options in detail. Chapter 3 overviews the theoretical concepts involved in the proposed candidate protocol. Chapter 4 provides the algorithm and the working model for the candidate routing protocol. Chapter 5 describes the results and comparison of various experiments conducted on the protocol.

Chapter 2

RELATED WORK

In a Mobile Ad hoc Network (MANET), due to the wireless nature of the physical connection of the devices, random changes occur in its network topology. The issues present in a MANET are mobility, power constraints, shared broadcast radio channel, bandwidth-constraint, and high error rates. Moreover, hidden terminal problems like packet collisions at a receiving node due to simultaneous transmissions add to the complexity of the routing protocol design for MANETs. A routing protocol must satisfy the requirements like minimum route acquisition delay, quick route reconfiguration, loop-free routing, and minimum control overhead. Its responsibilities include exchanging information, finding a feasible path to a destination, gathering information about path breaks, and repairing those broken paths. Traditional routing protocols used in wired networks perform poorly in MANETs due to the lack of any centralized administration (like base stations) and dynamic environmental changes (like mobility). Researchers in the past decade propose a variety of routing strategies to meet these requirements in MANETs [10]. The current routing protocols in MANETs can be broadly categorized as either table-driven or source initiated.

2.1 Table-driven Routing

Table-driven (or proactive) routing protocols maintain up-to-date routing information about all the nodes in the network; i.e., the nodes maintain one or more routing tables. These protocols are an extension of the wired routing protocols since global topological information is maintained in the form of tables at every node [11]. Optimized Link State Routing protocol (OLSR) [12] is an example of a table-driven routing protocol.

OLSR [12] is an enhancement of traditional link-state mechanisms where each node

maintains the link information to the subset of links to their neighboring nodes. The improvements include removal of loops and faster convergence than normal proactive routing techniques. In OLSR, nodes often exchange their databases with other nodes in order to keep themselves up-to-date with their neighbors' status. Being a proactive protocol, routes are generally readily available. OLSR optimizes the reactivity to topological changes by reducing the maximum time interval for periodic control message transmission [12].

OLSR is designed to perform in a server-less environment. Its control messages contain a sequence number in order to help the destination with out of order arrival of control messages. These situations occur due to collisions and other radio transmission problems in ad hoc wireless networks. Its advantages include reduced overhead compared to the conventional proactive routing protocol, while its drawbacks include a large use of power resources; thus making it unsuitable for sensor networks, which tend to stay idle for longer periods of time and are energy constrained.

2.2 Reactive Routing

The second type of routing strategy employed in MANETs is source initiated on-demand (or reactive) routing. In general, the proactive routing protocols have large control overhead and are not scalable for large networks. An on-demand routing strategy overcomes these drawbacks. Unlike table-driven routing protocols, an on-demand routing protocol executes the path-finding process and exchange routing information only when a path is required. Topological information is not maintained in on-demand routing protocols. The source node discovers the route to the destination whenever the source needs to send data to the destination node. Route discovery in an on-demand routing protocol is an iterative process; i.e., completed when all routing permutations are examined or a route is found. The route is generally maintained until either the destination

node is not reachable or a better route is found. Two examples of source-initiated routing protocols are Ad hoc On-demand Distance Vector routing (AODV) and Dynamic Source Routing (DSR).

AODV is a pure on-demand route acquisition system, since nodes that are not present on a selected path do not maintain the routing information or participate in the routing table exchange [10]. The routes are found using on-demand requests instead of maintaining complete lists. It employs a destination sequence number to identify the most recent path. The source node in AODV discovers the route to a destination by broadcasting route request (RREQ) packets to its neighbors that are then forwarded to their neighbors and so on. Once the RREQ packet reaches the destination, the route reply (RREP) packet is sent back to the source node as a unicast reply through the neighbor which it first received the RREQ packet. A timer deletes the entry in case a RREP packet is not received before the timer expires. Each node may obtain multiple routes to different destinations by using a single RREQ [11].

AODV does not repair link-breaks locally. A link-breakage is identified using periodic beacons from neighboring nodes or through link-level acknowledgements. In case of a link-break the source node will try to reestablish the route to the destination node only if the higher layers require. In other words, the path repair occurs only if there are messages to be transmitted through that path. If the intermediate node learns about a link-break, it sends an unsolicited RREP with a hop-count (distance metric) set as ∞. The main advantage of AODV is its adaptability to dynamic networks; since, its connection setup delay is low and its disadvantage is unnecessary bandwidth consumption due to periodic beacons.

DSR uses source routing; i.e., the sender of the packet specifies the route the packet takes through the network. In DSR, nodes do not require periodic beacons to inform its neighbors of its

presence. Each node maintains a route cache of all the visited nodes. To route a message, each node checks its route cache for an entry of the destination node, and if available, the route in the entry is used. Otherwise, route discovery is initiated by broadcasting a *route request* packet. On reception of the route request, each node checks whether it knows the route to destination using the above-mentioned method. If the intermediate node does not know the route, it forwards the route request through all outgoing links until it reaches a node that knows the route to destination or to the destination node itself. In case the route to destination is available, a *route reply* packet is sent as a unicast message back to source node. Route-error packets and acknowledgements achieve route maintenance in DSR. Route-error packets are generated when a fatal transmission error occurs due to loss of a link. Link operations are verified using acknowledgements. The major difference between AODV and DSR is the data packet in DSR carries the complete path to be traversed, however in ADOV, the source node and the intermediate node store the next-hop information corresponding to each flow for data packet transmission [11]. DSR does not require any symmetry in node links and it saves the bandwidth by not broadcasting the routing advertisements. However, DSR is not scalable for large networks since source-routing mechanism employed in DSR yields considerable overhead.

2.3 Hybrid Routing Protocols

The performance of proactive and reactive routing protocols varies with the network characteristics [13]. There is a fundamental trade-off between proactive routing and reactive route discovery. Although proactive protocols provide low latency and good reliability through readily available routing information, they suffer from poor scalability and high overhead. On the other hand, reactive protocols achieve low routing overhead but suffer from latency due to the on-demand route discovery. The hybrid routing protocols are new generation protocols with

characteristics that tries to find a balance point between proactive and reactive routing by adjusting the degree to which routes are propagated proactively versus the degree to which they are discovered reactively. They improve scalability by maintaining routes to nearby nodes and reactively discovering routes to far away nodes. They can be either zone based (i.e. partitioned) or grouped into trees or clusters. An example of a hybrid routing protocol is Zone Routing Protocol (ZRP).

In ZRP, the network contains two zones, proactive and reactive. The route information for the nodes in the proactive zone are readily available, whereas that of nodes outside the zone are found using any on-demand route discovery method. The subset of the network within which all the nodes can be reached by less than or equal to zone radius hops is known as the routing zone of the node [11]. An intra-zone routing protocol is used in the zone where a node routes messages proactively. The reactive routing protocol used beyond this zone is referred to as the inter-zone routing protocol. It effectively uses the information available at each node's routing zone to discover the route to the destination node. The boundary nodes proactively maintain routes to the destination and send the reply back to the source with sufficient routing information. The main advantage of ZRP is its reduced overhead compared to proactive routing protocols. ZRP requires a relatively small number of query messages, as these messages are routed only to "peripheral" nodes, omitting all the nodes within the routing zones [14]. Its main disadvantage is for large values of the routing zone, the protocol can behave like a pure proactive protocol, while for small values it behaves like a reactive protocol; i.e., its behavior is entirely dependent on the zone radius [10].

2.4 Motivation, Challenges, and General Solutions

MANETs consist of a collection of wireless mobile nodes dynamically forming a

temporary network without the use of any existing network infrastructure or centralized administration. Peer-to-peer (p2p) overlay networks in the Internet also have similar routing challenges like MANETs [4]. P2p networks initially created for file sharing purposes attract the researchers through their resiliency, redundancy, and dynamic fault tolerance in the Internet. This common relationship is possible by their frequently varying network topologies [4]. While the node mobility affects the network topology in MANETs, in p2p overlays the network topology is affected by the membership changes. Thus, there exists a synergy between MANETs and overlay networks in the Internet. Following sections provide the motivating problem and its challenges.

2.4.1 The Motivating Problem and Challenges

The subject of utilizing p2p techniques in MANETs is quite new. The problem of having a scalable p2p overlay network with no central control becomes technically challenging. Since they both share similar characteristics like self-organization, decentralization, hop-by-hop connection establishment, and frequent topological changes, both the types of networks need to solve the fundamental problem of providing a reliable routing strategy in a dynamic environment. This familiarity between the two networks and the popularity of file-sharing applications over the Internet using p2p systems like Napster and Gnutella inspire researchers to apply an abstraction of those strategies in MANETs.

Many fundamental differences exist between the Internet architecture and MANETs that cause various challenges in implementing a p2p overlay abstractions in MANETs. The main problems include bandwidth limitation, multi-access interference, high churn, addressing, and state-efficiency. In addition, topology maintenance of p2p overlays requires periodic monitoring of members for their presence. In the Internet, this operation is feasible because nodes do not change their status rapidly. However, in MANETs it may yield poor results causing a large

routing overhead high cost of route setup.

A p2p overlay model shields the distributed application designers from many difficult issues like fault tolerance, load balancing, and scalability. Similar to those in the Internet, distributed applications and network services in MANETs can potentially benefit from the deployment of a DHT algorithm [8]. A DHT based substrate can efficiently discover and maintain routes based on the physical layer broadcasts. The p2p overlay protocols in the Internet depend upon IP to provide hop-by-hop routing between neighbors. Since p2p overlays are connected over TCP links with physically unlimited range, unlike short-range radio transmission in MANETs, employing a p2p overlay protocol in a MANET environment on top of a multi-hop routing protocol is challenging as it may yield poor results. Moreover, it is difficult to take advantage of the interactions between these protocols. Therefore, a potential research direction in networking is to exploit the synergy between p2p and MANET in order to design better routing protocols for MANETs [4]. The following section discusses the solutions to overcome the difficulties by providing routing indirection.

2.4.2 General Solutions

Conventional ad hoc routing protocols deliver a packet from a source node to a predefined destination node. However, indirect routing differs in that packets are no longer routed based on the destination node's address but on a key [15]. The packet is delivered to the node that is responsible for the packet's key. In other words, the actual address of the final destination node is usually unknown to the sender. The efforts to solve these issues resulted in the emergence of what is known as structured p2p overlay networks built using Distributed Hash Tables (DHTs) [16]. DHTs have proven to be a novel and efficient platform for building a variety of scalable and robust distributed applications like content sharing in the Internet. A DHT layer

abstraction achieves routing indirection in MANETS and provides an efficient way of constructing distributed applications and services. For example, applications such as file-sharing and resource-discovery can benefit from the distributed insert/lookup convergence provided by DHTs [17].

The pros and cons of overlay networks are well recognized. Its advantages are its algorithmic simplicity, proved efficiency in p2p networks, and the ability to diffuse central authority. Overlay networks exist in two types: structured and unstructured. In unstructured overlay networks (like Gnutella), the connections are established arbitrarily. A new node can join the network by broadcasting a join query throughout the network so that it can find as many neighbors as possible. A structured overlay network (like Chord) routes the join queries with the help of a lookup service like a hash table. Originally, it was difficult to maintain due to its larger memory requirements for tracking the topological changes that make it difficult to employ this strategy for querying techniques.

Unstructured overlays are not affected by the above-mentioned constraints, as they flood the network to discover data [18]. However, unstructured overlay networks suffer from poor performance since the neighbor table of the joining node and those of the neighbors share a considerable fraction of nodes. This in turn reduces the flooding mechanism to find routes as the messages travel repeatedly among the same nodes. Though this drawback could be overcome by increasing the number of hops traveled by the node after each visit, this procedure introduces large overhead and thereby affects the robustness and query performance of the system [18].

The following design factors make structured overlay perform efficiently: the decoupling querying mechanism, topology maintenance, local failure detection, and a proximity-neighbor selection algorithm that exploits the heterogeneity, and random walk in structured topologies.

The additional functionality provided by structured overlays has proven important to achieve scalability and efficiency in a wide range of applications [18]. Additionally, structured overlays can eliminate redundant failure detection probes by using structure to partition failure detection responsibility and to locate nodes that need to be informed when a failure is detected.

Nodes in peer-to-peer overlays are heterogeneous [18]; they have different bandwidth, storage, and processing capacities. An overlay that ignores the different node capacities must cap the load at the level that the least capable nodes are able to sustain; otherwise, it risks congestion collapse. It is important to exploit heterogeneity to improve scalability. The additional functionality provided by structured overlays has proven important to achieve scalability and efficiency in a wide range of applications. Thus structured overlays can emulate the functionality of unstructured overlays with comparable or even better performance [18].

From a computer science point of view, this elimination of central control is a very attractive aspect of DHTs [16]. Additionally, it eliminates single points of failure and builds large-scale distributed systems. Current research on MANET integrates several representative protocols like DSR and AODV with DHT based overlay network strategies to provide a scalable substrate for routing in MANETs. A DHT substrate will shield many difficult issues including fault-tolerance, locating objects, scalability, availability, load balancing, and incremental deployment from the distributed application designers. The cons of overlay networks include its scalability issues. Moreover, as mentioned in [8], bandwidth limitations, node mobility, and multi access interference pose unique challenges to deploy such DHTs in MANETs. Since DHTs designed primarily for the Internet-based applications induce high traffic, which is not suitable for the ever-changing network topology and bandwidth limitations of MANETs. There are two options to deploy DHTs in a MANET, a layered or integrated approach.

In the layered approach, a proximity-aware DHT system (like Pastry [19] or Chord [20]) is applied as an overlay on top of MANETs similar to the Internet applications. Pastry maintains its leaf set and the routing table entries without the source routes and DSR maintains the source routes passively as per the demand of the Pastry routing state. Two examples exist for this approach; each in unstructured and structured overlay. The study provided by [9] presents an overlay of Gnutella like structure over MANET environment. On the other hand [21] explores the possibility of overlaying Chord on top of MANET. These strategies are logically similar to their Internet implementations with small modifications made to incorporate the difficulties of MANETs. Though this design is consistent with the International Standard Organization (ISO) model of networking, it prevents adding optimizations. Moreover, the interactions between representative ad hoc routing protocols and an overlay DHT protocol yield poor results due to unnecessary delays.

The second approach integrates the DHT substrate at the network layer along with a representative protocol, or applying a DHT substrate directly on top of the link layer. In the former case, when integrated with an ad hoc routing protocol (like DSR), the interactions between the DHT substrate and the routing protocol provide an optimized solution. MA-Chord defined in [15] is one example of the integrated approach. It combines AODV routing protocol and the Chord overlay routing layer protocol at the network layer to provide an efficient DHT substrate of key-based routing in MANETs. Each node in a MA-Chord network assigns itself a unique overlay ID, which defines its logical position on the virtual overlay ID ring. Furthermore, in MA-Chord, a message's packet header contains a message key. MA-Chord then routes the message to the node in the network that is currently responsible for the message key; i.e., to the node whose overlay ID is currently the numerically closest to the message key among all MA-

Chord nodes in the network.

Ekta [8] is another example of this type of strategy. Ekta implements the DHT abstraction by integrating Pastry and DSR at the network layer, which exploits optimizations made possible from close interactions between the two protocols. In Ekta, a message with a 128-bit key is routed using Pastry's prefix-based routing procedure and delivered to the destination node whose node ID is numerically closest to the message key. When a route lookup for the next logical hop returns a next-hop node from the leaf-set for which a source route does not exist, Ekta initiates a route discovery to find a new source route. On the other hand, if the node selected as the next hop is from the routing table and does not have a route, a prefix-based route-discovery is performed to find the routes to any nodes whose IDs match the prefix for that routing table entry.

Ekta inherits all of the optimizations on route discovery and route maintenance used by the DSR protocol. In addition, Ekta updates its routing table and leaf set using routes snooped while forwarding and overhearing packets, thus constantly discovering fresh and low proximity routes for the leaf-set and the routing table entries. In addition, the Ekta routing structures contain two caches of source routes, the "prefix-based view" of the routing table and the "neighbor-node view" of the leaf set.

The latter case employs a novel DHT routing strategy on top of the link layer without depending on any ad hoc representative protocol. The features of a DHT overlay routing protocol can provide an efficient routing solution for MANETs. There have been many efforts to push these features to lower layers in order to utilize the DHT structure effectively in MANETs. In other words, an underlay routing strategy employed on top of the link layer is a good routing solution for MANETS. Scalable Source Routing protocol is an example of underlay DHT routing protocol.

Scalable Source Routing protocol combines Chord-like structure into a virtual ring formed by the address spaces [22]. It forwards the messages in a greedy manner choosing the physically shortest path between the nodes. The Scalable Source Routing protocol forwards the packet using the virtual distance metric, which is the absolute value of difference between two nodes' addresses A and B [22]. The physical neighbor status is checked by transmitting periodic "*hello*" messages. It requires little infrastructure for its operation and due to its distributed structure it self-organizes quickly, so it is perfectly suited for emergency system applications.

Scalable Source Routing protocol employs indirect packet routing; i.e., it decouples packet address from network nodes. Nodes send packets to abstract destinations that are mapped by the routing protocol to a concrete node. This level of indirection enables data-centric communication where the packet addresses identify data objects instead of nodes [22]. This indirection adapts well with the mobility of the nodes in the MANETs. It assures consistent routing if and only if all nodes have valid routes to their respective virtual neighbors [22]. An iterative process can achieve consistent routing among nodes in the network.

Chapter 3

ARCHITECTURAL OVERVIEW

3.1 Features

Radio ring routing protocol (RRRP) is a DHT based routing protocol for mobile nodes in a MANET. RRRP works directly on top of the link layer using hexadecimal identifiers. RRRP uses a DHT substrate for routing and thus is an example of the structured overlay architecture. It uses its DHT data structure for efficiently initializing the routing tables of the joining nodes to announce the arrival of new nodes. RRRP eliminates redundant failure by a local repair mechanism in which the nodes surrounding the link breakage work around the problem without involving the end nodes. Additionally, delete_key messages ensure that routes are maintained symmetrically across two virtual neighbors. The following sections explain the features of the protocol.

3.1.1 Resiliency and Routing Structure

DHT provides a robust platform for large-scale networks due to its resiliency in the event of node failures. The resiliency of the protocol enables:

- Data replication
- Routing recovery
- Static resilience.

Data replication is generally preserving data in case the node holding the data fails. If the node holding the data fails, the message will not be lost from the whole system. Routing recovery is a mechanism that handles node failures. In case of node failures, routing recovery algorithms repopulate the nodes' routing tables with live nodes. It removes stale routes and replaces them with updated routes. Static resilience in a DHT routing protocol occurs in the event of a node failure before the recovery algorithm takes over. Static resilience is a good

measure of time between a node failure and the start of the recovery algorithm. It also shows how well the protocol adjusts and works around a failure without the aid of any recovery mechanisms.

The geometry of the routing protocol puts a variety of constraints on its design [23] as it affects the resiliency of the protocol. Some of the different geometries are ring (like Chord), tree (in Pastry), and Xor (defined in kademlia [24, 25]). The degree of flexibility offered by the geometry is an important design criterion. It is the amount of freedom offered by the protocol structure to choose neighbors and next-hop paths. In general, a good range of flexibility is to have the ability to achieve O(log n) neighbors with O(1) paths [23]. Therefore, a good geometry must provide flexibility in neighbor selection, which leads to shorter paths and flexibility in route selection, which leads to reliable path selection. Therefore, it is necessary to decide the geometry before considering other design issues. Hypercube, tree, and ring structures are examples of DHT routing geometries. The ring structure provides O(log n) flexibility of route selection and 2^i flexibility in neighbor selection for the *ith* node in the structure. In other words, the distance between the node and its *ith* neighbor is 2^i.

The routing protocol design must provide efficient local route convergence. Local route convergence occurs when the paths of two messages sent from nearby nodes with identical keys tend to converge at a node near the source nodes, in the proximity space [23]. It leads to low latencies and saves bandwidth consumption by providing overlay multicast, caching, and server selection. The ring structure provides better resiliency in the event of node failures. The ring geometry provides better local route convergence. Thus, the ring structure not only provides the greatest flexibility but also provides a good routing performance when compared to other structures like a hypercube or a tree.

3.1.2 RRRP Terminology

This section defines specific terminologies used in RRRP.

Ring neighbor status: The virtual neighbors are maintained in the ring table of each node. There are four possible states of a virtual neighbor: unknown, inactive, pending, and active. The status of a virtual neighbor changes with the different types of messages received by the neighbor (explained in the section 4.).

Path ID: This is a 32-bit integer generated in a random manner by the destination. It differentiates each route entry for a virtual neighbor in the routing table.

Proxy: This is a randomly selected node from the list of active physical neighbors. Each node sends its route setup request to other node through its proxy node. If the request fails then the node selects another proxy, excluding the previously selected proxy.

Next Hop: This is generally the physical neighbor one hop away from the source node towards the destination node.

3.1.3 Packet structure

An RRRP packet consists of a header section and data section. The protocol header structure consists of 256 bits. Figure 3.1 illustrates the breakdown of the protocol header structure. It consists of eight fields of a fixed length of 32 bits each and an options field. The first four fields contain source, destination, proxy, and previous hop addresses of the packet. Both the fields are addresses. The source and destination denotes the sender and receiver of the packet. The proxy denotes the address of the node to which a route setup request will be sent.

The previous hop address contains the address of the node that is one hop before the current node. It is useful in sending back error messages and control packets. The fifth field is the protocol number used in the packet (datagram), which determines the higher layer protocol

employed in the packet. The type field defines the packet type. There are five types of packets in RRRP as explained in section 3.3. The ring-table array is a variable field that contains the list of virtual neighbors' addresses of the sender node. It helps in finding new neighbors for the destination node. The data field is also a variable field containing the higher layer messages. The reserved and options fields are used for error correction and sending extra information in control packets.

Bit offset	0 - 32 bits
0	Source Address
32	Destination Address
64	Proxy Address
96	Previous Hop Address
128	Protocol Number
160	Header length
192	Type
224	Reserved
256	Options
Variable	Ring-table Array
Variable	Data

Figure 3.1 RRRP Header Structure

3.1.4 Neighbor state Transition

Each node in the network maintains two sets of neighbors. First, In RRRP each node maintains a set of physical neighbors. These nodes can communicate through the link layer. Since link layer quality varies quickly in a MANET, it is necessary to monitor the states of the neighboring nodes consistently. This can be achieved by setting up a timer and a two-state transition model. The two states of physical neighbor operations are "linked" and "not-linked." Figure 3.2 illustrates the two-state operation. A node will mark its neighboring node "linked" if a

node receives a hello message from it within the threshold of *t* seconds. Otherwise, the neighboring node is marked "not-linked." In other words, a node's state changes from linked to not-linked when its hello beacon is timed-out or the node is marked unreachable for a threshold. By broadcasting new *hello* beacons, the node can get back to the linked state.

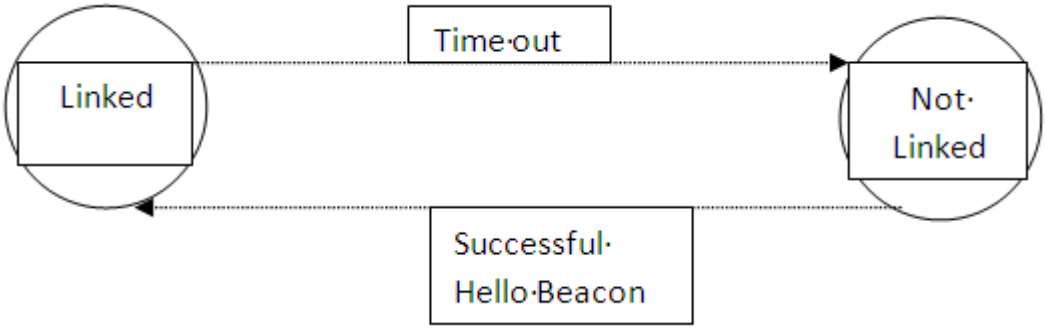


Figure 3.2 Two-State transition model

Since the changes in the underlying layers occur rapidly, the two-state model performs well in an ad hoc environment. It removes the additional toggling of states corresponding to lower layer topological changes. Moreover, a two-state model provides a more stable feature and a cost-effective design for maintaining physical neighbors. Initially, when the network starts settling down; i.e., when the nodes are learning about their neighboring nodes, the set of physical neighbors are inserted as virtual neighbors into the ring table. Thus, the same node could be marked as a ring neighbor and a physical neighbor.

Second, each node maintains the ring structure geometry of its virtual neighbors. RRRP maintains and monitors the routes to this set of nodes. The virtual neighbors are arranged in the ring in ascending order of their identifier space. They are maintained in a ring structure due to its flexibility and better local convergence. RRRP proactively maintains the paths over multiple hops. In addition, the protocol maintains the paths symmetrically; i.e., if node x maintains a route to node y conversely, node y maintains a route to node x. The ring handling is explained in 4.2.

3.2 Route Maintenance

Nodes modify their route entries on receiving these packets, which are different from normal data packets. Route maintenance is an important factor in a DHT proactive routing protocol [26]. In RRRP, fresh updated routes replace the stale routes in regular intervals or in the event of a node failure. Each node maintains a routing table and a ring table. The ring table stores the identifiers of the virtual neighbors in increasing order. The routing table stores the information about the source, destination, next hop after the source, previous hop towards the destination, path ID (a unique identifier to each route), and a validity indicator. The routing information between two nodes is stored in the end nodes; i.e., the source and the destination as well as in the nodes that are along the path. When a virtual neighbor becomes unreachable (due to mobility) or it fails, its entry is removed from the routing table. The intermediate nodes that are maintaining the route also remove the entry. This provides better stability in route maintenance. In the event that a fresh node is available, the source node adds the fresh node in place of the deleted node. The routing information about other nodes in the network is not maintained in RRRP. RRRP routes the messages directed to these nodes using the DHT forwarding algorithm; i.e., forwarding the message to the node whose identifier is “closest” to that of the destination node.

3.2.1 Routing Table Model

Each node in RRRP maintains a routing table. A node maintains route entries to two types of neighbors in its routing table: virtual neighbors and physical neighbors. In addition, each node also maintains the routes that are through the node. This helps in speeding up the routing process and reduces the buffering delay in intermediate nodes. Figure 3.3 illustrates the routing table structure of a node having node ID 4 in RRRP. Each entry contains the values of both endpoints

of the route (left and right neighbors), the next hop towards destination, the path ID of the route, and a validity indicator. The *Left Neighbor* is the address of the node, that initiates the path setup, and the *Right Neighbor* is the destination of the path.

```
Routing table for node 4:
Lt Ngbr     |Rt Ngbr      |NextLeft     |NextRight    |PathId|Validity
3221225476  | 3221225473  | 0000000000  | 3221225473  | ffff |       1|
3221225476  | 3221225474  | 0000000000  | 3221225474  | ffff |       1|
3221225476  | 3221225477  | 0000000000  | 3221225477  | ffff |       1|
3221225476  | 3221225478  | 3221225476  | 3221225477  | 3102 |       0|
3221225480  | 3221225477  | 3221225476  | 3221225477  | 3108 |       1|
```

Figure 3.3 Routing table for a node having node ID 4

These values depend on the type of the route entry:

- If the route entry is a direct route to a virtual neighbor present more than one hop away then the left and right neighbors are source and destination respectively. The *NextLeft* is the address of the node maintaining the routing table. The *NextRight* is the next hop address towards the destination node.
- In case the route entry points to a physical neighbor, which is generally one hop distance away, NextLeft is padded with zeroes and NextRight contains the physical neighbor's address.
- If the node is an intermediate node, then the left and right neighbors are the original source and destination of the route. The NextLeft is the address of the node maintaining the route and NextRight contains the next hop address towards the destination.

The path ID is a random 32-bit integer generated by the destination node. Every route entry for a virtual neighbor has a unique path ID. The right neighbor and path ID combination is unique for a route entry in the routing table, since all entries pointing to a physical neighbor carry the special path ID of FFFF. The validity field contains the value of either 1 or 0 determining whether a route is active or not. A timer checks the validity of a route. If it is marked pending,

RRRP removes the entry from the table if the timer expires before the node holding the entry receives a response from the corresponding node.

The routing table in the example shows Node 4 having five route entries. The first three entries are the routes to its physical neighbors, the fourth entry is a route to its virtual neighbor, and the last entry is a route that flows from Node 8 to Node 5 through Node 4. If the node receives a message directed to a node whose address is not in the routing table, it picks the node with the identifier closest to the destination from the routing table and forwards the message towards that node. RRRP maintains one entry per destination with the help of a route-duplicate prevention function. Upon finding a better route, RRRP updates the old route by deleting the existing path and inserting the new route in its place.

3.2.2 Ring table Structure

A ring table is a circular list data structure with wrap around at zero. It contains the array of virtual neighbors of a node and is the base for maintaining the routing table of a node. A ring table gives the list of addresses that a node maintains a direct path. RRRP ensures that each member in the ring table will have an entry in the routing table of the node. The size of the ring table *r* is a globally defined parameter for all the nodes in the network. The ring table is periodically monitored as nodes constantly join and leave the network. To maintain the integrity of the ring table with link failures and node mobility, each node in RRRP maintains *r* virtual neighbors with closest *r/2* clockwise and *r/*2 counter-clockwise identifiers in the virtual ring. For instance, if the ring size is r = 4, then each node maintains two neighbors of closest identifiers numerically lesser than itself and two neighbors having identifiers greater than itself. If an appropriate member is found by the node to add into its ring set, it deletes the path to a current member whose identifier is greater than that of newly found node. RRRP ensures that the route

information is also removed from the nodes that are along the path, which helps maintaining members symmetrically. Figure 3.4(a) illustrates the above concept. It also shows the ring neighbor set of the node 784.

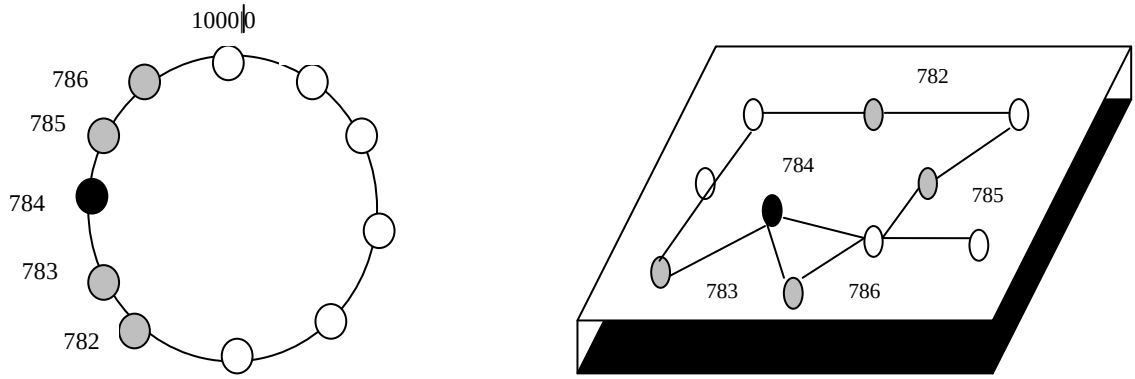


Figure 3.4 Ring structure and physical topology relationship

A ring table has two fields, Endpoint and Status. The Endpoint field contains the address of the virtual neighbor of the source node. Since the identifiers are unique, location-independent, and randomly distributed throughout the network. A virtual neighbor could be a node that is multiple hops away or a physical neighbor identified by a link layer *hello* broadcast. Figure 3.4(b) shows the mapping between a node's ring table and the physical topology of the network. The status field shows the status of the route to the endpoint with respect to the source node. Every node adds its own address in the ring table with a status marked active by default. It simplifies the comparison of its self-address and a newly received address.

```
Virtual Neighbor table for node 4:
Current table size: 5
3221225476  | active |
3221225477  | active |
3221225478  | active |
3221225481  | active |
3221225482  | active |
```

Figure 3.5 Ring Table for node (node ID=4)

Figure 3.5 shows the ring table members of a node whose ID is 4. In this example, the

clockwise neighbors are node 5 and node 6, while the counter clockwise neighbors are node 8 and node 9. The nodes that are in between are not present in its ring table, since they are either unknown to node 4 or due to node failures.

Chapter 4

PROTOCOL OPERATION

4.1 Message Types

A message packet could be either a control packet or data packet. Data packets are higher layer application messages formatted as a packet. It may contain information like voice. Control packets are network layer messages exchanged by a node in order to notify its status to its neighbor. There are five types of control packet messages in Radio Ring Routing Protocol (RRRP): Hello, Setup-request, Setup, Proxy-fail, and Delete_key. Each node processes the above-mentioned messages in different scenarios.

Hello

Every node in the network exchanges hello messages with its physical neighbors at regular intervals. The main purpose of hello messages is to detect whether a node is active or not. A node is marked inactive or broken if its hello message is not received within that interval. On receiving the hello message, each node adds the destination address and a path ID as “FFFF” to its routing table. It indicates that the source and destination are physical neighbors; therefore, they are one hop distance away from each other.

Setup-request

Every node in the network needs to construct and maintain a direct route to all ring table members. A source node sends a setup request to a destination node to setup a direct route. There are two types of setup requests, normal and selective. In the former, the source node sends its request message to itself through local proxy node (a randomly selected physical neighbor). Here, the destination address for the source node is the address of the proxy, which is generally the next hop. Once the message reaches the proxy node, it forwards the request message to the

appropriate destination in the network. In the latter, the source node sends a request to a multi-hop proxy. This multi-hop proxy is a node that informs the source node about new members in the network. In this case, the destination address is multiple hops away. The source node adds the destination address into its ring table and changes its status to *pending*. Each node in RRRP sends its setup-requests at a random time to prevent collision among messages in the network. The normal setup-request is used to join the network initially and the selective setup-request makes the system more intelligent. When a destination node receives a request message it may invoke a setup or proxy-fail message depending on the node receiving the message packet.

Setup

In general, a setup is invoked when a node receives a setup-request. The destination checks the sender address for validity, creates a random unique path ID, and sends the information back to the proxy node. This provides a level of routing indirection in the network. If an intermediate node receives a setup request message, it first checks whether the sender could be its own valid virtual neighbor or not. In case the intermediate node is a virtual neighbor, it sends back a setup reply message as mentioned above; otherwise, it will forward the request to the original destination in the sender's packet. During this process, when the reply arrives at a proxy (either local or multi-hop) it tries to forward the message to the source using the DHT forwarding method; i.e., forwarding the message to the numerically closest identifier of the destination.

On receiving a setup, a node adds the route to the destination in its routing table and changes the status of destination in the ring table to *active*. The route information is also stored in the intermediate nodes of the route. RRRP ensures that the route is setup symmetrically between source and destination. A source node may drop a setup message due to failures and concurrent

route setup-requests. The former case can be controlled by sending a proxy-fail message and the latter by sending a delete_key message to the nodes responsible for concurrent joins.

Proxy-fail

When an incorrect node receives a route setup-request message, it generates a proxy-fail message. In RRRP, a node sends its entire setup requests through its proxy node. When an incorrect node receives a setup-request, it is due to the proxy node forwarding the message to an incorrect destination. This situation can occur due to node mobility or link outage. A proxy-fail message is sent to the source node with the current proxy address in the options field of the packet header.

On receiving a proxy-fail, the source node finds a new proxy from the list of physical neighbors, excluding the previous proxy that resulted in a proxy-fail. Thus, the proxy-fail gives the source node an opportunity to select another proxy and build its ring table with appropriate virtual neighbors. RRRP generates a proxy-fail in case the route request arrives at the source itself (loop formation). Though this case never occurs, it is included as a precautionary measure for future implementations and modifications.

Delete_key

In RRRP, a source node sends a delete_key message to a virtual neighbor in any of the following three cases: Case 1: A hello message is not received from a physical neighbor node within an interval. Physical neighbors communicate with each other by means of link layer radio broadcasts in regular intervals. They are of short range and often used to detect the status of the physical neighbor. MANETs are affected by poor link quality and power constraints. A timer used to monitor the periodic hello beacons expires when the threshold reaches a preconfigured interval limit. In this case, the source node marks the physical neighbor inactive, and it sends a

delete_key message to all the other physical neighbors. This process informs all the neighbors, removes the inactive node, and updates their routing table with current live nodes

Case 2: A node joins or leaves the network. Node mobility is random in a MANET. As nodes constantly move around the network, their physical neighbor set also changes. Here the destination node could be a physical neighbor or a virtual neighbor. In the latter case, the source node sends a delete_key message to the destination through the proxy. The reserved field in the delete_key message packet carries the endpoint address of the destination.

Case 3: The third call to delete_key happens when the source node receives the message from a node that is not in its physical neighbor set or it already has the entry for the path being setup in the routing table [26]. These loops are rare but can occur when virtual neighbor routes are being concurrently setup or torn down. Calling delete_key provides a clean and simple solution to deal with these infrequent loops. On receiving a delete_key messages an intermediate node tries to forward it to the closest identifier of the destination to provide routing indirection.

4.2 Virtual neighbor state-transition model

A four-state transition model is employed for ring-table member maintenance. The four possible virtual neighbor states are: unknown, inactive, pending, and active. The type of packet received classifies the status of a ring-table member. In general, the unknown state of a virtual neighbor is defined as a situation when no node in the network recognizes the source node. This scenario usually occurs during the initial setup of the network, before the nodes in the network start communicating with each other. A node in RRRP maintains its virtual neighbors in any one of the last three states in its ring table. Figure 3.6 shows the state transition model. A periodic hello message or a route-setup message indicates whether the sender is active or not. Whenever a node *x* determines a virtual neighbor *z* active, it inserts a path to z in its routing table. From the

information available in a route-request message from *z*, *x* is able to determine the ring table array of *z* and sends a setup request to any appropriate member *y* present in that array, and the status that of virtual neighbor is marked as pending. If *y* acknowledges with a setup, then a path is setup between *x* and *y*. This allows nodes to exchange their local views of the virtual ring until their views converge and the appropriate paths are setup [26].The edges in the diagram determines *x*'s state in *y*'s request message.

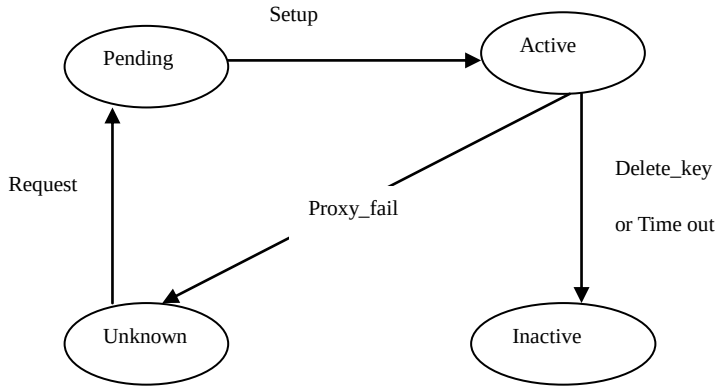


Figure 4.1 Four-state Virtual neighbor transition model

The state transition ensures that a node can send and receive messages from all of its active virtual neighbors. Moreover, the failure detection is symmetric. A timer keeps track of all the request messages. In case a member *y* times out its reply, it is removed from the ring table and marked as inactive. This information is sent across nodes in the path that maintains the route to *y*. Additionally, *x* marks *y* unknown in case of link failure or node mobility. The four-state model provides a robust model for ring table maintanence and failure correction.

4.3 Neighbor handling and failure correction

Nodes move in a random manner in a MANET causing rapid changes in the network topology. Since RRRP is a DHT routing protocol and is table-driven for O(log n), it is necessary for each node to constantly update its ring table. In other words, each node must maintain up-to-

date information on its current virtual neighbors' status and has to check its own ring table for the correct virtual neighbor set. This process can be achieved by exchanging virtual neighbor sets with each other. Failure detection and correction mechanisms provide a efficient method to maintain the consistency of the ring structure.

4.3.1 Exchanging virtual neighbor set

Ring neighbor handling varies with the type of message received. Each node in RRRP sends its virtual neighbor list along with all types of messages except delete_key and hello messages. Virtual neighbor lists are handled only at endpoints and not in intermediate nodes. This allows nodes to exchange their views about the network and helps in quicker stabilization of the network. In RRRP, when a source node receives a list of the destination node's virtual neighbors, it checks for valid neighbors to add to its own ring table set. The source node filters the list of addresses before sending a setup request. Setup requests are not sent to source and destination nodes mentioned in the packet since they are already present in the ring table. Additionally, any physical or virtual neighbor of the node handling the received list is not processed due to the above-mentioned reason. Route setup-requests are sent to any new address if detected as an appropriate virtual neighbor. The endpoint adds the new address to its ring table with the status marked as pending. Once the destination replies, the source adds an entry of the new destination to the ring table with a status marked active and a valid route to the destination in the routing table.

The virtual ring set is made up of numerically closest neighbors on either side of the node ID with wrap around at zero. The nodes whose addresses are numerically less than that of the source are *Left Neighbors* and the nodes that are numerically greater than that of the source are *Right Neighbors.* In general, a node tries to maintain a balance between the number of left and

right neighbors. The ring table size specifies the number of left and right neighbors a node can maintain. The ring size *r* is always maintained such that 2^r is greater than or equal to the number of nodes in the network. This allows the source node to have an equal number of left and right neighbors.

When adding new members, the source node checks the size of the ring table. A node only adds new members when it receives a setup or setup request message from them to prevent convergence problems due to the addition of failed members [26]. The ring table size is specified before the initialization of the network. If the ring table contains less than twice the number specified, then the source adds the new node without removing any of its previous members. In case the ring table contains more than twice the specified size, the source removes a neighbor with the numerically lowest id. If the source node adds the new node as its left neighbor, the member whose address is least in the previous set before the addition is deleted. In case the new node is a potential right neighbor, the member with the numerically greatest id. When nodes remove a member from their virtual neighbor set to make room for a new member, they tear down any virtual neighbor path to the removed member. The source informs that it is no longer in the virtual neighbor set and garbage collects the redundant routes. Delete_key messages provide a clean simple method to remove these nodes.

Ring neighbor sharing does not occur during delete_key and hello messages. Since delete_key messages are used to delete a route entry from a node's routing table, exchanging neighbor views during delete_key will result in incorrect ring table formation. In case of hello messages, a timer in each node ensures that they are periodically received. Piggybacking the hello messages with the neighbor set causes unnecessary delays in the network, as they are short lived; hence, usually not preferred.

During proxy-fail messages, the neighbor-handling function behaves slightly different. Prior to receiving a proxy-fail message, the source assumes that destination is an appropriate virtual neighbor. Generally, a proxy-fail message notifies the source node to get a new proxy. The second proxy chosen could either be a physical neighbor or a virtual neighbor that is multi-hops away. Once it receives a proxy-fail from the destination, the source node tries to find new members to add it to its ring table array. This process is made simple by providing the source node the list of virtual neighbors of the previously assumed destination. The list gives a fresh set of nodes for the source to try as a destination. The source node selects a random physical neighbor as the new proxy and sends a route setup-request to an appropriate node in the list of virtual neighbors received from the destination that sent the proxy-fail message. Once the newly found destination accepts the request, the source selects the successful node as the proxy for future route requests and replies.

4.3.2 Failure detection and correction

RRRP detects a link-breakage or a node failure in timely fashion to ensure ring consistency. It maintains a hard routing state and detects node and path failures using direct communication with its neighbors. As mentioned, RRRP maintains a hard state compared to soft state maintained by many of the other protocols. Symmetric failure detection simplifies hard state maintenance. For instance, if a node *x* marks its neighbor *y* faulty, then *y* also marks *x* faulty. This system guarantees that the routing state is correctly removed from the network on failures. Additionally, it implements reliable node and path failure detection. RRRP also detects node failures using per-hop retransmissions for all types of messages except hellos.

Nodes repair virtual neighbor paths to their virtual neighbors when those paths fail. When a node *x* marks a node *y* failed, it initiates the delete_key of any virtual neighbor path in its

routing table that have *y* as a next hop. This is achieved by sending delete_key messages to all the nodes in that path. After receiving a delete_key message, a source node will try for proper replacement by sending a new route setup-request. If the destination is the appropriate replacement, it replies with a setup reply. Otherwise, it replies with a proxy-fail message that contains the address of the failed proxy through which previous incorrect route setup-request was sent. This also provides a robust mechanism to abort incorrect ring neighbor connections.

Chapter 5

EVALUATION

5.1 Environmental Constraints

The preliminary experimental setup simulates 25 nodes randomly distributed over a 1500m X 1500m square area. The number of nodes was varied from 10 to 100 in equally densed groups. All experiment trials were ran for 30 seconds with RRRP message transfer taking effect from 12th second without node mobility. The initial 12 seconds were used to ensure that the network reached a steady state. For all the other experiments, the plane for simulation was kept at 1500m X 300m up to 50 nodes. For more than 50 nodes an area of 3000m X 600m was used. This method ensures that the node density per square area is constant. All the simulations were run for 1900 seconds. Each simulation was run for three trials and their average is shown. Results were collected from 900 seconds onwards, since the initial time was used for the protocols to reach a steady state. The interval for CBR flow of 180 packets is set between 1000 and 1180 seconds. Each protocol was run for three trials changing the seed of simulation and the average of the three trials are used.

5.2 Collection of Results

RRRP is evaluated using experiments conducted on the Qualnet 4.0 network simulator [27]. A 802.11 (IEEE wireless standard) network is used for the experiments. The preliminary experiment compared the end-to-end delay performance of RRRP and AODV and the next set of experiments compare packet delivery, end-to-end delay, message overhead, and throughput results for RRRP, DSR, OLSR, and ZRP for static and mobile scenarios.

5.2.1 Preliminary Experiment

The preliminary experiments used a variable number of Constant Bit Rate (CBR)

sources. In default configuration, each CBR packet contains 512 bytes of data. Two random nodes in the network were selected as source and destination of the CBR flow of a hundred packets with a one second interval between each packet. Control packets were sent during the entire simulation time; but the results are analyzed for after 12th second; i.e., after the launch of RRRP.

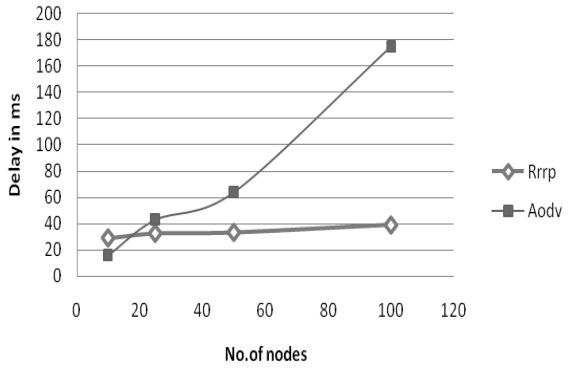
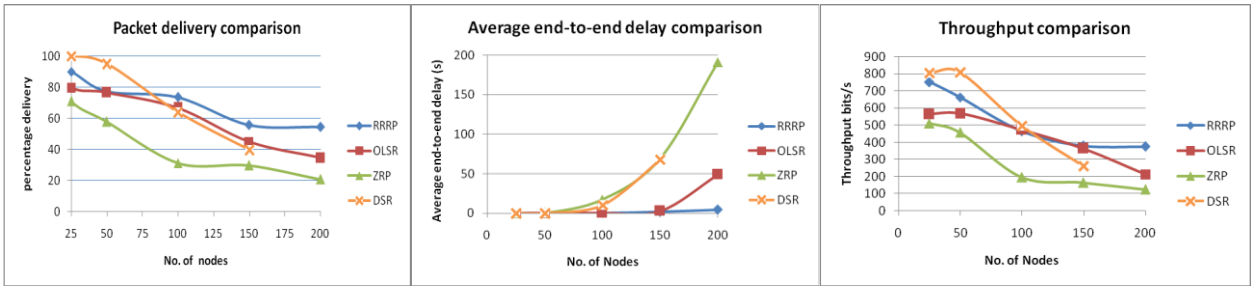


Figure 5.1 End-to-end delay comparison

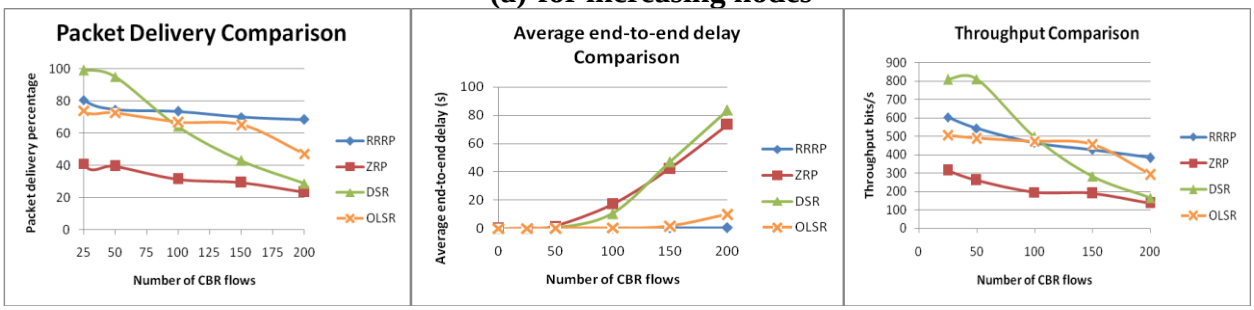
The average end-to-end delay for a CBR flow between two nodes was measured. The experiments were ran for five times by changing the random seed value to test the consistency of the protocol behavior and the average of the five results was used. Figure 5.1 illustrates this comparison when the number of nodes were increased from 10 to 100. As the number of nodes increses, the delivery ratio decreases due to congestion in the network. The delay of AODV increases dramatically because, they queue packets while they repair routes that fail due to congestion. This strategy improves delivery ratios but it results in high delays [26]. Moreover, nodes drop packets that are queued in case of a node failure. The collision avoidance mechanism in lower layers can work around the problem, but it may yield poor results in terms of power consumption. RRRP has low delay across the range of nodes because it never queues packets waiting for routes as each node tries sending the packet to the identifier closest to the destination. Thus, it can achieve a good delivery ratio and quicker delivery.

5.2.2 Static Performance Results Comparison

The second set of experiments compared RRRP with OLSR, DSR, and ZRP. In all experiments, each node sends 180 CBR packets to a random destination in the network. Figure 5.2 illustrates the results for performance comparison for the above-mentioned environmental constraints. The results show that RRRP performs well in all scenarios, as it uniquely does not depend on any other routing protocol's assistance for discovering routes thus provides reliable packet delivery. The performances of the other protocols suffer due to the following reasons.



(a) for increasing nodes



(b) for increasing flows

Figure 5.2 Performance results comparison

Increasing network size: In this experiment, ZRP was used with variable zone size. The protocol initially behaved like a reactive protocol but as the number of neighbors is increases then its characteristics were found similar to a table-driven protocol. In a MANET environment, topological changes affect packet delivery. ZRP tends to perform poorly once the number of nodes was increased to more than 50, thus showing poor scalability. DSR[28], a reactive protocol, also performs well until 50 nodes, but its performance reduces dramatically in all the metrics due to network size increase. As the number of nodes increases, the delivery ratio

decreases due to congestion in the network.

The delay of DSR increases dramatically because, the nodes queue packets while they repair routes that fail due to congestion. Reactive protocols must first determine the route, which may result in considerable delay if the information is not available in caches. The performance of DSR at 200 nodes scenarios was not shown since the simulator frequently broke down during simulations with 200 nodes due to internal problems, such as memory leaks and array overflows.

OLSR behaves like RRRP up to 100 nodes, but it starts to behave poorly beyond since it works on table-driven technology. Although the routes to all destinations are readily available in OLSR, which considerably reduces the delay involved in the route setup process, OLSR suffers from scalability issues as a large amount of bandwidth will be consumed for transmitting routing updates [10].

RRRP has best results across the range of nodes. It achieves best packet delivery by finding efficient routes with help of multi-hop proxy. DHT routing technique mitigates the routing delay even in larger networks that are generally associated with conventional routing protocols.

Increasing flows: The next experiment was conducted to test the behavior of the protocols by varying the number of application flows across the network and keeping the number of nodes constant at one hundred. This process tests for the maximum traffic load each protocol can carry for particular number of nodes. Figure 5.2(b) illustrates the performance comparison for the above-mentioned environmental constraints.

In this static scenario, all protocols except ZRP achieve acceptable results until 50 nodes. With a smaller zone size, it behaves like a reactive protocol and it suffered from delays and congestion in the network. However, with a larger zone size the performance was close to a

table-driven protocol. Moreover, an exact zone size determination for each simulation increases its complexity. The main factors affecting the performance of ZRP are its dependency on the amount of nodes activated. Secondly, the reaction to the traffic demand depends on the gradient of traffic volume [29].

In a purely reactive routing protocol like DSR[10], the nodes neither maintain routing information nor use the network resources when there is no data to be sent; thus, they are ideal for small networks. However, with larger networks with more application flows, if routes containing broken links fail, a new route discovery or route repair must be performed. Until the new route is available, packets are dropped or delayed. Moreover, the reactive route search procedure may involve significant control traffic due to global flooding. In the case of DSR[10], route caches help in reducing the delay marginally, but with increasing flows which cause network congestion and the delay in the delivery inevitably increases exponentially. This, together with the long setup delay, may make pure reactive routing less suitable for real-time traffic.

OLSR[12] uses power and network resources in order to propagate data about possibly unused routes. Since proactive routing maintains information that is immediately available, the delay before sending a packet is minimal. While this is not a problem for wired access points, and laptops, it makes OLSR unsuitable for ad hoc networks that are constrained by energy and bandwidth [10]. Since in OLSR routing information is readily available, there is low delay in the network up to 100 flows. Beyond that scenario, the nodes consume considerable time in transferring routing table updates across the network for synchronization, which hinders the application data flows; hence, the slight increase in end-to-end delay.

RRRP achieves best performance in case of packet delivery across the range of traffic

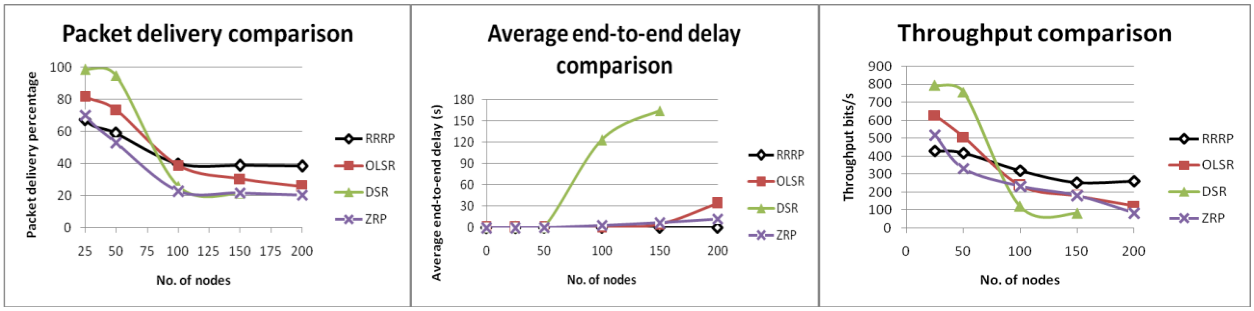
loads by following the DHT overlay routing method; i.e., sending the packet to the nearest identifier of the destination. It has provision for local-repair mechanism that works around the failed link and provides better delivery ratio. Its end-to-end delay is low since, as mentioned before, it routes a packet without queuing them. It also uses less network resources since the nodes maintain only the information about certain nodes in the network unlike a proactive protocol in which nodes maintain complete network information. The better performance of RRRP in larger network is due to its symmetric four-state route maintenance. Unlike a reactive protocol, nodes in RRRP send periodic control messages to their neighbors to measure the link quality, which helps in updating the best route available to the neighboring node.

5.2.3 Mobility Results Comparison

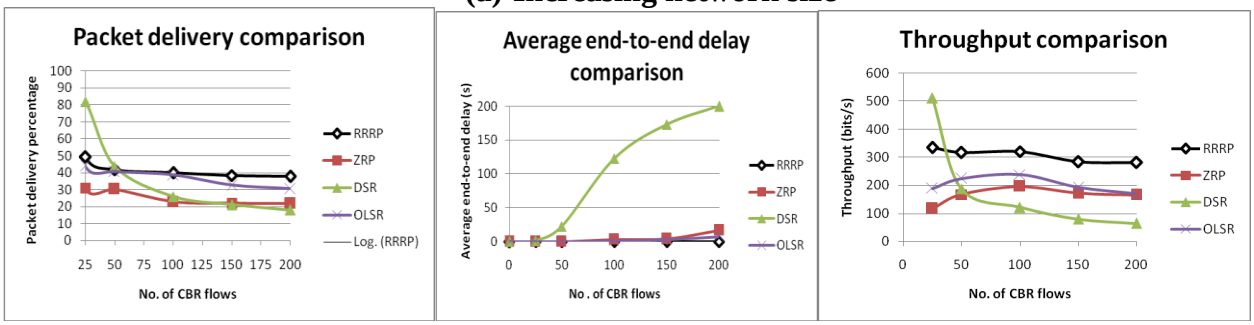
In this set of experiments, mobility was introduced into the environment mentioned in 5.1. Random waypoint mobility model [29], which uses pause time and variations in the node speeds and directions was employed. Pause times are stationary time-periods between random movements of the nodes. For this experiment, a pause time was kept constant at 30 seconds. The speed of the node mobility was varied between 0 to 20 m/s. Figures 5.3 (a) and (b) illustrate the results for the different routing protocols in mobile environment with increasing network size and increasing CBR flows respectively.

Increasing network size: In case of growing network size, the performance of the protocols is similar to the static scenario but there are more node and link failures due to mobility. RRRP achieves good delivery percentage and low delay in all the network sizes. RRRP also has provision for local-repair that can be used to repair routes with low overhead and delay. Other protocols suffer when the network size was increased beyond 100 nodes due to the above-mentioned reasons.

Increasing flows: In case of increasing CBR flows, the performance show similar trend as the static scenario, but incur low delivery percentage due to network congestion. RRRP performs well in all conditions since it does not queue packets thereby reducing the delay. Its packet delivery ratio is also good due to efficient route finding mechanism by multi-hop proxy. Other protocols perform well until 100 flows, beyond that point they incur delay in routing packets. In OLSR, since the routing tables are not updated it induces slight delay.



(a) Increasing network size



(b) Increasing CBR flows

Figure 5.3 Performance comparisons for mobile scenario

The results for TCP throughput in both the experiments are constantly good for RRRP, while other protocols though they are good for smaller network size and lesser loads, their performance degrade rapidly for larger network sizes. This decrease is due to increased link failures owing to mobility of the nodes.

5.2.4 Message Overhead

This section compares the message overhead incurred during both static and mobile scenario. Message overhead is a good measure of bandwidth consumption of routing protocol.

For this discussion the results from increasing network size was used. Figure 5.4 illustrates the comparison for message overhead. In both cases, RRRP performs best as it uses less network resources to route packets by maintaining only ring neighbors and not the entire set of nodes.

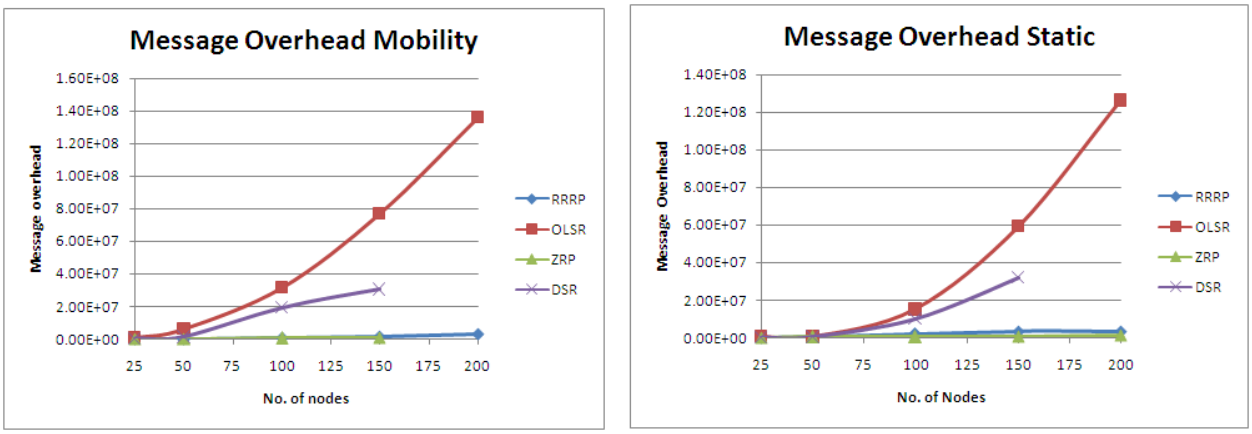


Figure 5.4 Message overhead comparison

OLSR suffers from large overhead due to frequent transfer of routing table updates among the nodes. For DSR, the overhead increase rapidly beyond 100 nodes due to network congestion. In addition, frequent route failures cause redundant control packet flows across the network.

5.3 Conclusions and Future work

MANETs have gained an increased attention among researchers in recent years with the advance of technology and vast requirements of communication and research on wireless connectivity is focused on enabling mobile devices to connect with each other in absence of a central administration system.

This study joins effort in answering the fundamental question of efficiency of a DHT substrate compared to conventional routing in ad hoc networks. The results are shown with help of metrics like throughput, end-to-end delay, message overhead, and packet delivery percentage in both static and mobile conditions. The results show that RRRP performs well in all the scenarios compared to conventional routing protocols. Future research in this direction can be

motivated towards adding more functionality to the protocol and in testing its behavior in adaptive networks. Another avenue for research could be comparing its performance with different DHT structures (like tree and Xor) and evaluate their trade-offs with respect to the ring structure.

REFERENCES:

1. Muthukumaran Pitchaimani and D. Chellamani, *A Candidate protocol for cognitive networks*.
2. Royer, E. and C. Toh, *A review of current routing protocols for ad hoc mobile wirelessnetworks.* IEEE [see also IEEE Wireless Communications] Personal Communications, 1999. **6**(2): p. 46-55.
3. Basagni, S., *Mobile ad hoc networking*. 2004, Hoboken, NJ: John Wiley. xvi, 461 p.
4. Hu, Y., S. Das, and H. Pucha, *Exploiting the synergy between peer-to-peer and mobile ad hoc networks.* In Hot-OS IX, May 2003.
5. Gnutella. *http://wiki.limewire.org/index.php?title=GDF*.
6. Napster. *http://www.napster.com/*.
7. Wu, J., *Handbook on theoretical and algorithmic aspects of sensor, ad hoc wireless, and peer-to-peer networks*. 2006: Auerbach publications.
8. Pucha, H., S. Das, and Y. Hu, *Ekta: An efficient DHT substrate for distributed applications in mobile ad hoc networks.* Proceedings of the 6th IEEE IEEEWorkshop on Mobile Computing Systems and Applications (WMCSA), 2004: p. 163–173.
9. Barbosa e Oliveira, L., I. Guimaraes Siqueira, and A.A. Ferreira Loureiro. *Evaluation of ad-hoc routing protocols under a peer-to-peer application*. in *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*. 2003.
10. Abolhasan, M., T. Wysocki, and E. Dutkiewicz, *A review of routing protocols for mobile ad hoc networks.* Ad Hoc Networks, 2004. **2**(1): p. 1-22.
11. Murthy, C.S.R. and B.S.Manoj, *Ad Hoc Wireless Networks Architecture and Protocols*. 2007, New Delhi: Pearson Education.
12. Clausen, T. and P. Jacquet, *RFC3626: Optimized Link State Routing Protocol (OLSR).* RFC Editor United States, 2003.
13. Ramasubramanian, V., et al., *SHARP: a hybrid adaptive routing protocol for mobile ad hoc networks*, in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*. 2003, ACM: Annapolis, Maryland, USA. p. 303-314.
14. Haas, Z.J. *A new routing protocol for the reconfigurable wireless networks*. in *Universal Personal Communications Record, 1997. Conference Record., 1997 IEEE 6th International Conference on*. 1997.
15. Qi, M. and J. Hong. *MA-Chord: A New Approach for Mobile Ad Hoc Network with DHT Based Unicast Scheme*. in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*. 2007.
16. El-Ansary, S. and S. Haridi, *An Overview of Structured P2P Overlay Networks.* Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks, Auerbach Publications, 2006: p. 665-683.
17. Pucha, H., S. Das, and Y. Hu. *How to implement DHTs in mobile ad hoc networks*. 2004: Citeseer.
18. Castro, M., M. Costa, and A. Rowstron, *Debunking some myths about structured and unstructured overlays*, in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*. 2005, USENIX Association. p. 85-98.
19. Druschel, P. and A. Rowstron. *PASTRY: a large-scale, persistent peer-to-peer storage utility*. in *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*. 2001.
20. Stoica, I., et al., *Chord: a scalable peer-to-peer lookup protocol for internet applications.*

IEEE/ACM Transactions on networking, 2003. **11**(1): p. 17-32.
21. Cramer, C. and T. Fuhrmann, *Performance evaluation of chord in mobile ad hoc networks*, in *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking*. 2006, ACM: Los Angeles, California. p. 48-53.
22. Fuhrmann, T., et al., *Pushing chord into the underlay: Scalable routing for hybrid manets*. 2006, Citeseer: Universität Karlsruhe (TH), Germany.
23. Gummadi, K., et al., *The impact of DHT routing geometry on resilience and proximity*, in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. 2003, ACM: Karlsruhe, Germany. p. 381-394.
24. Zhao, B., Y. Wen, and H. Zhao. *KDSR: An Efficient DHT-based Routing Protocol for Mobile Ad Hoc Networks*. 2009: IEEE Computer Society.
25. Maymounkov, P. and D. Mazieres, *Kademlia: A peer-to-peer information system based on the xor metric.* Peer-to-Peer Systems, 2002: p. 53-65.
26. Caesar, M., et al., *Virtual ring routing: network routing inspired by DHTs.* ACM SIGCOMM Computer Communication Review, 2006. **36**(4): p. 362.
27. Qualnet-4.0. *http://www.scalable-networks.com/products/qualnet/*.
28. Royer, E. and C. Toh, *A review of current routing protocols for ad-hoc mobile wireless networks.* IEEE personal communications, 1999.
29. Camp, T., J. Boleng, and V. Davies, *A survey of mobility models for ad hoc network research.* Wireless Communications and Mobile Computing, 2002. **2**(5): p. 483-502.