Performance Evaluation of Multimedia Traffic
Over Ad Hoc Wireless Networks

Yung-Ching Tseng

Coordinated Science Laboratory
1308 West Main Street, Urbana, IL 61801
Performance Evaluation of Multimedia Traffic Over Ad Hoc Wireless Networks

Tseng, Y-C.

Coordinated Science Laboratory
1308 W. Main Street
Urbana, IL 61801

Army Research Office
Scientific Systems Company
Boston Massachusetts

Approved for public release; distribution unlimited.

Our goal is to build a simulator using NS to analyze the performance of mobile ad-hoc networks. We assume that the mobile hosts move relatively slowly with respect to each other. The simulations are done under a fixed routing protocol. The physical layer characteristics are the main issues we focus on to compute packet errors. A link layer retransmission module was built to deal with errors due to multiple access and thermal noise that cannot be corrected by a Reed-Solomon coding scheme. Using this simulator, we then studied the performance of bursty large-range and short-range dependent traffic sources.

Wireless networks, MANET, Error-correcting codes, FH-CDMA, simulation

UNCLASSIFIED

UNCLASSIFIED

UNCLASSIFIED

UL
PERFORMANCE EVALUATION OF MULTIMEDIA TRAFFIC OVER AD HOC WIRELESS NETWORKS

BY
YUNG-CHING TSENG
B.S., Cheng-Kung University, Tainan, 1995

THESIS
Submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2000

Urbana, Illinois
ABSTRACT

Our goal is to build a simulator using NS to analyze the performance of mobile ad-hoc networks. We assume that the mobile hosts move relatively slowly with respect to each other. The simulations are done under a fixed routing protocol. The physical layer characteristics are the main issues we focus on to compute packet errors. A link layer retransmission module was built to deal with errors due to multiple access and thermal noise that cannot be corrected by a Reed-Solomon coding scheme. Using this simulator, we then studied the performance of bursty large-range and short-range dependent traffic sources.
ACKNOWLEDGMENTS

It is with great pleasure that I express gratitude to my thesis supervisor, Professor Rayadurgam Srikant. His guidance, encouragement, and expert technical help were essential for the completion of this work.

Finally I thank all the fellow students in our office for the kindly assistance during the course of this thesis.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2 ROUTING PROTOCOLS</td>
<td>9</td>
</tr>
<tr>
<td>2.1 Dynamic Source Routing (DSR) Algorithm</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Destination-sequenced Distance-Vector (DSDV) Routing</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Ad-Hoc On-Demand Distance-Vector (AODV)</td>
<td>16</td>
</tr>
<tr>
<td>2.4 Wireless Routing Protocol (WRP)</td>
<td>17</td>
</tr>
<tr>
<td>2.5 Zone Routing Protocol (ZRP)</td>
<td>19</td>
</tr>
<tr>
<td>2.6 Temporally-Ordered Routing Algorithm (TORA)</td>
<td>20</td>
</tr>
<tr>
<td>2.7 System- and Traffic-Dependent Adaptive Routing (STARA) Algorithm</td>
<td>30</td>
</tr>
<tr>
<td>3 MODEL</td>
<td>34</td>
</tr>
<tr>
<td>3.1 Mobility Model</td>
<td>34</td>
</tr>
<tr>
<td>3.2 Routing Scheme</td>
<td>35</td>
</tr>
<tr>
<td>3.3 FH-CDMA and Link-layer Retransmission</td>
<td>35</td>
</tr>
<tr>
<td>3.4 Measurements</td>
<td>36</td>
</tr>
<tr>
<td>4 PHYSICAL LAYER</td>
<td>37</td>
</tr>
<tr>
<td>4.1 Packet Error Probability Under FH-CDMA</td>
<td>37</td>
</tr>
<tr>
<td>4.2 Physical Layer Simulation Parameters</td>
<td>44</td>
</tr>
<tr>
<td>4.3 Probability of Correct Decoding versus Number of Test Symbols</td>
<td>45</td>
</tr>
<tr>
<td>5 SIMULATION AND RESULTS</td>
<td>49</td>
</tr>
<tr>
<td>5.1 Simulation Environment</td>
<td>49</td>
</tr>
<tr>
<td>5.2 Simulator Modifications</td>
<td>50</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>45</td>
</tr>
<tr>
<td>4.2</td>
<td>46</td>
</tr>
<tr>
<td>5.1</td>
<td>53</td>
</tr>
<tr>
<td>5.2</td>
<td>54</td>
</tr>
<tr>
<td>5.3</td>
<td>55</td>
</tr>
<tr>
<td>5.4</td>
<td>58</td>
</tr>
<tr>
<td>5.5</td>
<td>59</td>
</tr>
<tr>
<td>5.6</td>
<td>60</td>
</tr>
<tr>
<td>5.7</td>
<td>61</td>
</tr>
<tr>
<td>5.8</td>
<td>62</td>
</tr>
<tr>
<td>5.9</td>
<td>63</td>
</tr>
<tr>
<td>5.10</td>
<td>64</td>
</tr>
<tr>
<td>5.11</td>
<td>65</td>
</tr>
</tbody>
</table>

4.1 Redundancy ratio

4.2 Probability of correctly decoding $P(d)$ w.r.t. different SNR

5.1 Flow

5.2 Constants used for exponential on/off source

5.3 Constants used for Pareto on/off source

5.4 Pareto on/off source with 100,000 packets (rate = 120 kbps, idle = 300 ms, burst = 500 ms, shape = 1.5)

5.5 Exponential on/off source with 100,000 packets (rate = 300 kbps, idle = 300 ms, burst = 500 ms)

5.6 Pareto on/off source with 75,000 packets (rate = 120 kbps, idle = 300 ms, burst = 500 ms, shape = 1.5)

5.7 Exponential on/off source with 75,000 packets (rate = 300 kbps, idle = 300 ms, burst = 500 ms)

5.8 Flow for topology in [1].

5.9 Pareto on/off source with 40,000 packets for SNR = 12 dB (rate = 5 kbps, idle = 300 ms, burst = 500 ms, shape = 1.5)

5.10 Pareto on/off source with 40,000 packets for SNR = 14 dB (rate = 5 kbps, idle = 300 ms, burst = 500 ms, shape = 1.5)

5.11 Pareto on/off source with 40,000 packets for SNR = 16 dB (rate = 5 kbps, idle = 300 ms, burst = 500 ms, shape = 1.5)
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A simple wireless ad-hoc network.</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>Count-to-infinity problem. Above each node, its routing table entry is shown.</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>AODV implementation detail.</td>
<td>17</td>
</tr>
<tr>
<td>2.3</td>
<td>Predecessor information is used to reduce temporary loops. Below each node is path length and the predecessor. (a) The topology with stable routing tables. (b) Link (J, K) fails. (c) Node K sends messages to avoid temporary loops. (d) Stable routes are maintained in each node.</td>
<td>18</td>
</tr>
<tr>
<td>2.4</td>
<td>Zone notation.</td>
<td>20</td>
</tr>
<tr>
<td>2.5</td>
<td>ZRP IERP.</td>
<td>20</td>
</tr>
<tr>
<td>2.6</td>
<td>Full reversal. (a) shows a link failure connecting the destination node; (b) to (d) illustrate the propagation of link reversal effect; stable routes are maintained as shown in (e).</td>
<td>23</td>
</tr>
<tr>
<td>2.7</td>
<td>Partial reversal. (a) shows a link failure connecting the destination node; (b) to (d) illustrate the propagation of link reversal effect; stable routes are maintained as shown in (e).</td>
<td>25</td>
</tr>
<tr>
<td>2.8</td>
<td>Route creation in TORA.</td>
<td>28</td>
</tr>
<tr>
<td>2.9</td>
<td>Link failure with no reaction.</td>
<td>28</td>
</tr>
<tr>
<td>2.10</td>
<td>Re-establishing routes after failure of last DN link.</td>
<td>29</td>
</tr>
<tr>
<td>2.11</td>
<td>Erasing invalid routes after a failure which partitions the network.</td>
<td>30</td>
</tr>
<tr>
<td>2.12</td>
<td>Delay equalization of STARA.</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>FH-CDMA transmission.</td>
<td>38</td>
</tr>
<tr>
<td>4.2</td>
<td>Symbol numbers in a dwell interval.</td>
<td>40</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

A mobile ad-hoc network (MANET) consists of a set of nodes which are free to move arbitrarily in a shared wireless channel. For example, the node may be located in a taxi, a notebook in a classroom, or on a person. Due to the localized nature of wireless environments, nodes that are within a certain node's transmission range are said to be its "neighbors." The commutative property holds only when assumption of "bi-directional" links is made. In ad-hoc wireless networks, some hosts wishing to communicate may be out of wireless transmission range of each other, but may be able to communicate if other hosts in the network are able to forward packets to the destination. Thus, mobile hosts in the network are considered to function as a router, and are able to establish a route to transmit packets by exchanging neighborhood information messages. For example, as shown in Figure 1.1, mobile host A is out of the transmitter range to mobile host C, but with the forwarding of mobile host B, packets can still get through. Thus, mobile host A is called the "hidden terminal" to host C. The forwarding is done by wireless routing protocols which dynamically establish multihop paths from packet sender to packet destination.
As mentioned before, the main difference between a MANET host and a wireline terminal is that every host is not only a receiving device, but also has the ability to forward messages to others and to listen to messages sent from other nodes. Each of the nodes functions as a router combined with a wireless platform in a host. Since a node functioning in a wireless environment has a limit on its transmission and receiving range, routing and multiple access schemes used in a MANET are different from those in a wireline network. Several design considerations unique to MANET are:

1. *Dynamic topologies*: Since a node in a MANET is mobile, it is possible for a node to get disconnected or set up a new connection with others upon moving. Hence, the topology may change with time or when nodes adjust their transmission or reception parameters.

2. *Bandwidth-constrained, variable capacity links*: Owing to the hardware design, wireless links possess less capacity than the wireline counterparts. Moreover, taking into account the effects of fading, noise and interference, and multiple access, the overall throughput of wireless links is far less than the transmission rate.
3. **Power-constrained operation**: Mobile devices usually rely on batteries instead of a constant power supply. Therefore, power conservation is the major design criterion.

4. **Limited security**: MANETs are generally more prone to eavesdropping and spoofing than wireline networks.

Route information in each mobile node is maintained in the form of a routing table. This routing table is built using *route discovery* which can be done either in a centralized or distributed manner. A centralized routing algorithm chooses a specific node to do the routing computation and maintenance. This route information is then conveyed to each node in the network. The node that performs route computation must have full knowledge of the network. The main drawback of centralized algorithms is the risk that the computing node may fail to function and thus other nodes would have no way of recovering routing information. Therefore, a distributed route discovery is essential for wireless network routing algorithm design.

In the routing table, each node maintains the route to a destination by storing the next hop and distance for that destination. To accomplish this in a distributed manner, each node has to do its computation based on the connectivity with its neighbors and information provided by neighboring nodes. The operation of a distributed routing scheme is briefly described below:

1. **Route discovery**: Suppose that a source node wishes to transmit to a particular destination. The source node first checks its own routing table for the next hop to the destination. If the source’s routing table does not contain an entry for the
destination of the packet, it will then initiate route discovery to its neighboring nodes requesting a valid route. Generally, route discovery from a source node for some destination node is done by sending out some message called “route request” to the source’s neighborhood. Those neighboring nodes will then send “route reply” to the sending node if they know a route to that particular destination, or forward that “route request” to their neighborhood if no route is available in the routing table. If this network is connected, eventually a “route reply” will be sent back to the source node via a valid route, and nodes along that route will also save an entry to the destination for future use. Transmissions to destinations that already have an entry in the routing table will be done right away according the routing table afterwards.

2. Route maintenance: Once a valid route entry has been recorded into the routing table of a source node, the source node will keep using this entry if it has any traffic to send to that destination. As network topology changes over time, the recorded route (or next hop) may not still be the best one. To maintain an efficient routing entry, the source node adjusts the routing entry with update messages sent by its neighbors. If any change has been made to a routing entry, the source node propagates the change through neighbors. Thus, better routes are always maintained dynamically through message passing.

3. Link failure operation: If a node stops receiving HELLO messages from one of its neighboring nodes, it assumes the link to this neighbor has been disconnected,
i.e., the link fails. As soon as the link fails, the node at the other side is no longer a neighboring node, and the routing table entry corresponding to that node will be deleted. Some schemes reset the route length to $\infty$. This is a significant event which triggers immediate update messages to all neighbor nodes and forces those nodes to erase the old route which uses the failed link. For proactive algorithms, a new route discovery to the node at the other side of that failed link will then start. For reactive ones, there will be no entry still using that failed link, and some destinations may become unreachable until new route discovery is addressed.

A MANET algorithm should possess the following features:

- **Distributed computation:** This is an essential property in dynamic topologies to achieve effective and valid route computation, because centralized route computation requires a small subset of mobile hosts to have knowledge of the whole network. In general, this is difficult although some hybrid partially centralized routing schemes have been studied [2].

- **Loop freedom:** A distributed protocol may suffer from looping due to invalid route information. It is important to avoid looping and guarantee finite-time convergence when designing a routing algorithm.

- **Rapid convergence to loop-free route(s):** This design goal also results from the dynamic topology. The routes need to be discovered and used before the topology changes. If the route discovery delay is larger than the topology changing rate, the routing table would be recording and propagating stale entries.
• **Quick response upon link/node failures:** To ensure successful transmission, a quick recovery from failure is required.

• **Demand-based operation:** This is to avoid unnecessary periodic message passing overhead.

• **Efficiency:** This can be measured by:

  1. Average data bits transmitted
  2. Average control bits transmitted

• **Asymmetric link support:** Because of different transmission parameters among mobile hosts, a link is not always bi-directional. Moreover, it is possible that links may have different capacity in different directions.

• **Scalability:** Some specific networks such as military networks or highway networks may be extremely large compared to classroom networks. Therefore, a good routing protocol must perform well when the network size scales up in terms of number of nodes, network connectivity (i.e., average number of neighbor nodes), rate of topological change, or link capacity.

• **Metrics support:** Different routing metrics may result in different delay and throughput characteristics. Some possible metrics are:

  1. Reliable path
  2. Stable path
3. Minimum delay path

4. Minimum hop count path

5. Minimum total power path

- "Sleep" period (no data transmission) operation: If a mobile hosts is not active, some message passing may be avoided for power conservation.

The main goal of developing a simulator is to build a tool to study the impact of routing schemes, physical layer characteristics, multiple access algorithms, etc. on the performance of a MANET. We started with the well-known Network Simulator (NS) software [3] and added the following modules for the above purpose:

1. Physical layer characteristics

2. Multiple access and thermal noise corruption

3. Routing protocols

4. Performance metrics

5. Mobility

Physical-layer parameters have been added in order to calculate the packet error rate. The routing scheme we have been using through this thesis is a basic shortest-path algorithm for a static topology to measure packet delay and throughput. The mobility module has been constructed, but has not yet been merged into the simulator.

The rest of this thesis is organized as follows. Chapter 2 gives the basic ideas of recent routing protocols. Chapter 3 gives detailed description of the mobility model and
routing scheme we use and the measurements we take. Chapter 4 explains FH-CDMA parameters in the physical layer. Chapter 5 presents assumption and functionality of the modules we add to the widely-used NS simulator for our simulation purposes and also gives the result of our simulation. Finally, Chapter 6 concludes the thesis and suggests future work.
CHAPTER 2
ROUTING PROTOCOLS

Several distributed ad-hoc network routing protocols have been proposed recently. They can be broadly categorized as follows:

1. Proactive: A proactive algorithm performs continuous route updates in each node. An example is the Destination-Sequenced Distance-Vector (DSDV) routing [4]. Such an algorithm has little delay in getting a valid route, but transmits lots of control messages.

2. Reactive: A “reactive” algorithm operates only when a route request is made. This request searches for a route globally by flooding the network. It then gets a route which may be optimal but after a large delay. Ad-hoc On-Demand Distance-Vector Routing Protocol (AODV) [5] and Dynamic Source Routing (DSR) algorithm [6] are two typical reactive algorithms.

3. Hybrid: A hybrid routing algorithm combines both proactive and reactive schemes to reduce the route discovery delay and the update messaging overhead. An example is the Zone Routing Protocol (ZRP) [7].
2.1 Dynamic Source Routing (DSR) Algorithm

Under the Dynamic Source Routing (DSR) algorithm [6] a source node performs route discovery by broadcasting a Route Request packet with a recorded source route listing only itself. The routing table is stored in a route cache. A node which hears the request forwards the packet and adds itself to the recorded source route if the node is not either the target of the request or already in the source route list. The Route Reply will be sent back with the whole recorded source route to the source node after the request reaches that destination. Any further request will then be discarded. Therefore, DSR does not guarantee to find the optimal route.

If a reply has not been received after some time interval, the source node sends the request again. This may occur when the source and destination nodes are disconnected. If this is the case, repeated queries only flood the network. But it may also be the case that the request or reply messages get lost; therefore, sending a request again avoids marking some reachable destinations as isolated.

As mobile hosts move, some source routes may become invalid. DSR does lazy update on updating stale routes. That is, unless an error message is received due to an unreachable destination, the source node will keep using the recorded source route. The nodes with the knowledge of topology change do not send out update messages without receiving a request.

In addition to the basic algorithm above, several optimization features are incorporated to conserve bandwidth and improve the speed of route discovery:
- Cache route: If an intermediate node has route in cache, it replies with this route, and further propagation of route request is aborted.

- Piggybacking: Small data packets may be piggybacked with route request, route reply, or even error messages to achieve better efficiency. But then the “return mail problem” arises when a data packet that is piggybacked with a route request is received by an intermediate node with a valid route to the destination; the node immediately discards the request packet and replies with a Route Reply packet. In this case, the data packet is also lost. This problem can be fixed before discarding the Route Request packet. The node sending the reply examines the packet, and reconstructs a data packet which will be sent towards the destination.

- Modification of replies on reverse path to reflect shorter routes: When a data packet is received as the result of operating in promiscuous receive mode, the node checks if the Routing Header packet contains its address in the unprocessed portion of the source route. If so, the node knows that packet could bypass the unprocessed hops preceding it in the source route. The node then sends what is called a gratuitous Route Reply message to the packet’s source, giving it the shorter route without these hops.

- Increased time interval to retransmit route discovery request: This avoids querying too frequently if source and destination are disconnected.

- Avoid looping: If a source route is received with repeated nodes, it is to be discarded. This avoids looping and unnecessary processing of all the packets.
• *Avoid reply collision:* If several nodes are replying to the same node, after sensing the collision, each of the sending nodes waits for a random time before sending reply again.

• "Eavesdrop" on neighbor's link error messages to update route cache: If a link error message is overheard, nodes update their own route cache right away. This operation bypasses the lazy update for nodes that can hear a link failure and improves the delay.

Since the source route is recorded in the replying packet, this algorithm does not require bidirectional links. Also, once a route is discovered, it will be continuously used until it becomes invalid. There is no guarantee on the optimality and delay for the routes in the cache.

2.2 Destination-Sequenced Distance-Vector (DSDV) Routing

The Destination-Sequenced Distance-Vector (DSDV) [4] routing algorithm is based on the Distributed Bellman-Ford (DBF) algorithm with hop count as a metric. To illustrate how DBF works, define $D_i$ as the shortest distance from node $i$ to the destination, and $d_{ij}$ as link distance between node $i$ and node $j$. The Bellman's equation is then

$$D_i = \min_j \{d_{ij} + D_j\}.$$ (2.1)
For a particular destination, node \( i \) stores a node \( j \) and the hop count to destination through node \( j \) if the route through node \( j \) is the shortest one.

The drawback of the Bellman-Ford algorithm is that it suffers from the "count-to-
infinity" problem. In Figure 2.1, to begin with, node \( B \) stores a routing table entry for node \( A \) with next hop as node \( A \), and a hop count of 1; node \( C \) stores a routing table entry for node \( A \) with next hop as node \( B \) and a hop count of 2; and node \( D \) stores a routing table entry for node \( A \) with next hop as node \( C \) and a hop count of 3. After link \((A, B)\) has been disconnected, node \( B \) believes its neighbor \( C \) has a route to the destination, and hence updates its routing table accordingly. Node \( C \) also takes the information given by node \( B \) as the route to the destination. This iterates until every node’s path length to the isolated destination grows to \( \infty \) and never converges.

Under DSDV, a modification has been made to avoid this problem using the destination sequence numbers. This sequence number is a nondecreasing integer initiated by the destination. Each destination has its own set of sequence numbers. For implementation purposes, valid sequence numbers are even numbers while the odd sequence numbers indicate invalid routes. When a node receives several updates regarding the same destination, it selects one as its routing table entry. The route selection criterion for a certain destination is to choose a newer route, i.e., a larger valid sequence number. If several routes with the same valid sequence numbers are present, the one with smaller route metric is chosen.

The route discovery operation of DSDV works in an entirely distributed fashion with the assumption of bidirectional and symmetric links. Each mobile host’s routing table
Figure 2.1 Count-to-infinity problem. Above each node, its routing table entry is shown. Each entry to each possible destination node contains the destination address, a sequence number for that destination, hop count to destination node, next hop to destination node, and time since last update. Initially, the unreachable destination leaves a routing table entry with \( \infty \) as hop count. Routes to a certain destination are propagated from the destination to every node in the same connected graph. Each time that destination node sends out an update message to its neighbors, it increases the sequence number associated with itself by “2”. This number is always an even number because it is initiated by the destination node and therefore it is a valid route. As the route information propagates, a node which receives this information only increases its next hop field and hop count to the destination.

When node \( i \) detects the link to a neighbor node \( j \) fails, it updates its routing table as follows:
1. Mobile hosts that detect the link failure build information describing the link as $\infty$, and the sequence number is increased by "1". Advertising broken link information is the only case that sequence numbers are changed by mobile hosts other than the destination node.

2. Nodes that receive an $\infty$ metrics with equal or larger sequence number must trigger a route update broadcast to disseminate this significant change about that particular destination.

Route maintenance is done by sending update messages. Update messages are advertised in two forms:

1. **Full dump:** This is the message carrying the whole routing table to neighbors. It is usually scheduled to transmit when the mobile host is in the "sleep" mode in order to save network bandwidth. Because each mobile host agrees to forward packets, if full dump is scheduled too often, the network bandwidth will be wasted on transmitting control packets. When mobile host movements become frequent, the full dump can be used to reduce the size of an incremental update.

2. **Incremental update:** This carries only significant information changes since the last full dump. If a new sequence number with the same metrics is received, it will not be put into the next incremental update. When a stable route shows a different metric for some destination, it is considered to be a significant change, and must be broadcast right away.
A common problem when nodes update their routing table by message passing is the "fluctuating route problem." This is caused by broadcasting a temporary longer route to neighbors. If nodes advertise upon receiving an update, the route length goes up and down alternately before it finally gets to a stable value. DSDV conquers this problem by postponing advertising update messages. When a mobile host can determine somehow that a better route is likely to show up soon, the previous message can be held not to be broadcast unless the destination is previously unreachable. This requires mobile hosts to keep a history on previous routes' weighted average fluctuation times.

Under DSDV, a source node obtains a route to a destination for a transmission by looking up its routing table. The route is ready when the source node wishes to send. This is done through update messages initiated by destination nodes. Routing table entries are kept even for destination nodes that are not reachable. Therefore, DSDV consumes a large amount of network bandwidth for control messages, and maintains large routing tables.

2.3 Ad-Hoc On-Demand Distance-Vector (AODV)

Ad-Hoc On-Demand Distance-Vector (AODV) [5] operates in a similar way as DSDV with two major differences. The first is that a routing entry can be marked as inactive if no packet refers to it within a time interval called ACTIVE_TIME_OUT, and can be purged out if after BAD_LINK_LIFETIME since it is marked inactive. The second difference is that a mobile host maintains an "active_list" which contains nodes that are actively using the route. AODV is a so called "on-demand" protocol because a Route
Request (RREQ) packet is generated only when a node determines it needs a route to a destination which is not cached in its routing table either because the destination is previously unknown or because the route expires. Upon receiving RREQ, a Route Reply (RREP) is unicast back to the source node (see Figure 2.2). More implementation details are given in [5]. For example, a link breakage is detected by HELLO packets. If the number of HELLO packets lost is greater than ALLOWED HELLO LOSS, then it is assumed there is no longer a directed link to that neighboring node.

2.4 Wireless Routing Protocol (WRP)

Wireless Routing Protocol (WRP) [8] assumes bidirectional links and uses retransmission to ensure reliable transmissions. It is based on the path-finding algorithm (PFA) which utilizes information regarding the length and second-to-last hop (predecessor) of the shortest path to each destination to avoid the count-to-infinity problem of the DBF algorithm. However, temporary loops are still incurred by PFA. With WRP, the count-to-infinity problem is eliminated, and temporary loops are reduced after link failures, thus leading to faster convergence.
Figure 2.3 Predecessor information is used to reduce temporary loops. Below each node is path length and the predecessor. (a) The topology with stable routing tables. (b) Link $(J, K)$ fails. (c) Node $K$ sends messages to avoid temporary loops. (d) Stable routes are maintained in each node.

Three different message types are defined in WRP for route maintenance purposes: Update, Hello, and ACK. A Hello message is a null update message to inform the other party that this sending node is alive. An ACK message indicates that there is good connectivity, and should be sent after receiving an Update or Hello message. An Update message exchanges some information between nodes, including node ID, sequence number (assigned by sending node), update list, and a response list.

A node checks “all” information reported from neighbors each time it processes an event involving a neighbor, and a positive ACK is then replied for the “entire” Update message, not to each entry. This results in faster convergence after a single source failure because temporary loops are reduced. To understand how WRP avoids temporary loops,
assume a topology with stable routes as shown in Figure 2.3(a). Node $K$ is the predecessor for routes $I$ to $J$ and $B$ to $J$. Figure 2.3(b) to (d) show updating procedure after link between nodes $K$ and $J$ fails. In Figure 2.3(b), node $K$ sends updates to nodes $I$ and $B$ indicating that node $K$ itself has no valid route to the destination node $J$. In Figure 2.3(c), nodes $I$ and $B$ start the route discovery query by sending out *Hello* packets to their neighbors. Notice that owing to the predecessor information stored along with the path length, nodes $I$ and $B$ now purge out the looping routes through node $K$, and thus faster convergence is achieved as shown in Figure 2.3(d).

### 2.5 Zone Routing Protocol (ZRP)

Zone Routing Protocol (ZRP) [7] serves as a hybrid routing algorithm by defining a "zone" in the topology. For a particular node $A$ (as shown in Figure 2.4), its "zone" is the set of nodes which are less than or equal to $r$ hops away, where $r$ is the zone radius. Nodes exactly $r$ hops away from node $A$ are called "peripheral nodes" of node $A$. ZRP has a reactive property between zones, but intrazone, a proactive algorithm is preferred. The route discovery algorithm within zones depends on which algorithm is chosen as the Intrazone Routing Protocol (IARP); this algorithm between zones uses Interzone Routing Protocol (IERP). In IERP the source first checks if the desired destination is within its routing zone. If yes, then the route is obtained immediately; if not, the source "bordercasts" a route request to all its peripheral nodes. All the nodes repeat the same procedure until the destination node is reached. The route to that is then sent back to
the source node. In Figure 2.5 node A is the source node that wishes to find a route to node L. The zone radius is now set to 2.

To optimize the algorithm for route discovery time, route accumulation is applied, i.e., a node receiving a query knows the path back to the query source. Hence, with the definition of zones, ZRP ensures that topology changes only affect local nodes, and updates are locally propagated.

2.6 Temporally-Ordered Routing Algorithm (TORA)

The Temporally-Ordered Routing Algorithm (TORA)[9] is based on the Gafni and Bertsekas (GB) algorithm [10]. The GB algorithm does link reversal to obtain a secondary
route so that there is no need to respond to topological change immediately. The main goal of link reversal is to maintain at least one outgoing arc for each node.

Each node $i$ (other than the destination) maintains the link status with an entry for each link $(i, k)$, where node $k$ is a neighbor of node $i$. The status of the links is determined by the "height" of the node, and its height entry for the neighbor. The link is directed from the higher node to the lower node. If a neighbor $k$ is higher than node $i$, the link is marked upstream (UP). If a neighbor $k$ is lower than node $i$, the link is marked downstream (DN). If the neighbor's height entry is NULL, the link is marked undirected (UN). Finally, if the height of node $i$ is NULL, then any neighbor’s height that is not NULL is considered lower, and the corresponding link is marked downstream (DN). Initially, nodes obtain their own position according to the information propagated from the destination node. The destination node holds lowest “height” in the topology.

Link reversal is triggered when a node detects that all its DN links are lost, which might be due to some link or node failure. The response is to reverse some or all of the UP links to DN depending on different types of algorithms. After the reversal, the node gains at least one outgoing link to which it sends all its traffic. If every node other than the destination maintains at least one outgoing link, the packets sent into the network will eventually reach the destination. As long as a node has outgoing links, it continues to use the path until the next link reversal is triggered. This algorithm guarantees that at least one path is maintained from any node to the destination, but the path might not be an optimal one. The good point is that topology change now does not affect a node’s transmission, i.e., topology change is decoupled from control message generation.
Unlike algorithms based on the DBF algorithm, TORA does not flood updating messages upon topological change. Control messages are generated only when nodes lose their last outgoing link.

As mentioned above, link reversal is done in either of two ways:

1. Full reversal:

   At each iteration, each node, other than the destination, with no outgoing links reverses the directions of “all” its incoming links. Each node \( i \) maintains a pair \((\alpha_i, i)\) where \( i \) is node ID, \( \alpha_i \in \mathcal{N} \), which is used to order these node pairs. Those pairs can be ordered lexicographically with \((\alpha_i, i) < (\alpha_j, j)\) if

   - \( \alpha_i < \alpha_j \), or
   - \( \alpha_i = \alpha_j \) and \( i > j \)

   Let \( N_i \) be a set of neighbor nodes of node \( i \). When node \( i \)'s height becomes a local minimum of those of nodes in \( N_i \), it does the link reversal by generating a local maximum. Therefore, links will then become outgoing with respect to node \( i \). The reversal procedure can be explained as follows:

   \( \forall j, j \in N_i, \) if \( (\alpha_i^k, i) < (\alpha_j^k, j) \) at the \( k \)th iteration and \( i \neq \) destination ID,

   then node \( i \) increases \( \alpha_i^k \) to

\[
\alpha_i^{k+1} = \max\{\alpha_j^k \mid j \in N_i\} + 1 \tag{2.2}
\]
Figure 2.6 Full reversal. (a) shows a link failure connecting the destination node; (b) to (d) illustrate the propagation of link reversal effect; stable routes are maintained as shown in (e).

Equation (2.2) sets the value of $\alpha_i^{k+1}$ in the $k+1$th iteration to one more than the greatest $\alpha_j^k$ among all the neighbors of node $i$. As in Figure 2.6(a), if one link to the destination fails, this triggers nodes (those indicated with “R”) without any outgoing links to reverse all their incoming links. It is proved that eventually each node will maintain at least one outgoing link except the destination node. This implies that each node will have at least one path to the destination.

2. Partial reversal:

A reversed list is kept by each node other than the destination to record the nodes which have reversed the direction of a particular link. Therefore, nodes will not reverse the direction of a link if some node listed on the reversed list has already reversed this link before. If every link of this node has been reversed, and there is no way to get at least one outgoing link-by-link reversal, the reversed list will be reset and all the links are reversed. Each node $i$ keeps a triple $(\alpha_i, \beta_i, i)$ where $\alpha_i$ and $\beta_i$ are integers. The set of triples is ordered lexicographically. Unlike the full link reversal, an additional integer is stored to maintain a reference level of nodes. Within the reference level every operation is limited to be done locally. Only when a
node loses all its outgoing links due to adjacent link failure, it can then initiate a new reference level which entails the node to be a global maximum, i.e., the highest one. This reference level is then propagated out through neighbors. The neighboring nodes do not always become local maxima during this propagation, but they can reverse the links which are higher than their height if all their outgoing links are lost due to that link failure. The information stored in each node is as follows:

- α_i as the reference level, β_i as the integer for local comparison, and i as the node ID. The initial set of (α_i, β_i, i) is such that α_i^0 = 0 ∀i, and for any link (i, j) have (α_i^0, β_i^0, i) > (α_j^0, β_j^0, j) which means link (i, j) goes from i to j. At the kth iteration, a node i other than the destination for which (α_i, β_i, i) < (α_j, β_j, j) ∀j ∈ N_i increases α_i^k to

\[ \alpha_i^{k+1} = \min\{\alpha_j^k \mid j \in N_i\} + 1 \]  

(2.3)

and sets β_i^k to

\[ \beta_i^{k+1} = \begin{cases} 
\min\{\beta_j^k \mid j \in N_i \text{ with } \alpha_i^{k+1} = \alpha_j^k\} - 1, & \exists j \text{ with } \alpha_i^{k+1} = \alpha_j^k \\
\beta_j^k, & \text{otherwise}
\end{cases} \]  

(2.4)

All other nodes j maintain the same integers α_j and β_j. It can be shown as in Figure 2.7 that the difference between partial reversal and full reversal is that in partial reversal, once the link has been reversed by some node, it will not be reversed again by other nodes.
Figure 2.7 Partial reversal. (a) shows a link failure connecting the destination node; (b) to (d) illustrate the propagation of link reversal effect; stable routes are maintained as shown in (e).

It has been shown that both of these algorithms maintain at least one outgoing link, and eventually get to the destination within finite iterations. However, the GB algorithm is designed to work in a connected network. TORA incorporates partial link reversal with its own route erasing mechanism to accomplish the routing operation when the network is partitioned.

Routing optimality, i.e., finding shortest path, is of less importance to TORA. Also, if a route for some source-destination pair is maintained in the routing table but not used before network topology changes, it is considered a waste of network bandwidth. Therefore, TORA is designed to decouple the link-state sensing mechanism from the routing part. The former is done by Internet MANET Encapsulation Protocol (IMEP).

IMEP takes care of control message delivery and link-status sensing. It assures that each node is always aware of its neighboring nodes and maintains a correct set of neighbors. When a node decides to transmit, the message is guaranteed by IMEP to be delivered to its neighbors and to be received correctly and in order of transmission.

With the knowledge of the GB algorithm, we now summarize the routing operation of TORA as follows:
- At any time, a quintuple is associated with each neighbor node of node $i$. This is to order each node lexicographically.

$$
\text{Height } H_i = (\tau_i, oid_i, r_i, \overset{\text{delta w.r.t. ref. level}}{\delta_i}, i) 
$$

(2.5)

where

- $\tau_i$: reference level, the time tag set to the “time” of the link failure
- $oid_i$: the originator ID, i.e., ID of the node which defined the new reference level
- $r_i$: single bit to divide the ref. level into sublevels
- $\delta_i$: an integer used to order nodes w.r.t a common reference level
- $i$: the unique ID of the node

- Each node maintains a link-status table which records the status of link $(i, k)$, $\forall k \in \text{neighborset of node } i$.

- Based on partial link reversal, all nodes are ordered lexicographically.

- The basic concepts used to minimize communication overhead are as follows:

  - Routes are established only when necessary by constructing a directed acyclic graph (DAG) rooted at the destination.

  - A “query/reply” process is used.

  - Link activation triggers no reaction.

  - Reaction to link failure is triggered only when necessary (a node lost its DN link).
– Scope of failure reactions is minimized.

TORA has three main operation conditions based on three distinct message types:

1. Query (QRY) packet: QRY packets are used for creating routes.

2. Update (UPD): When a node receives this packet, it updates its *height* accordingly. UPD packets are used for both creating and maintaining routes.

3. Clear (CLR): This packet is used for erasing routes.

The three operation conditions are:

1. *Route creation:* Creating routes requires use of the QRY and UPD packets (Figure 2.8). This refers to the mechanism of checking if at least one outgoing link exists.

2. *Route maintenance:* This is only performed for nodes that have a height other than NULL. Furthermore, any neighbor's height that is NULL is not used for the computations (Figure 2.9 and Figure 2.10).

3. *Route erasure:* If CLR is received and the node erases a route, it must re-examine if there is any downstream link left. If all downstream links are lost due to *Route Erasure*, it defines a new reference level (Figure 2.11).

Compared to other protocols, TORA has more computation overhead but supports sub-optimal routes. The following summarizes the characteristics of TORA.

- TORA is a source-initiated algorithm based on the *link-reversal* (GB) algorithm.
Figure 2.8 Route creation in TORA.

Figure 2.9 Link failure with no reaction.

- It is a reactive distributed algorithm, that is, it only maintains information about adjacent nodes.

- It decouples the generation of potentially far-reaching control message propagation from the rate of topological changes by dividing the protocol into “link status sensing mechanism” (use IMEP) and “routing mechanism” to minimize reaction to topological changes.
Figure 2.10 Re-establishing routes after failure of last DN link.

- The objective of TORA is not to find an optimal route, but to provide information about several routes.

- It has “multipath” support to minimize topology change effects and alleviate congestion.

- The operation of TORA is loop-free.

- TORA assumes all nodes have synchronized clocks (by GPS or NTP (Network Time Protocol)).

- TORA also has good scalability because internode communication is minimized.

- TORA minimizes communication overhead, but increases node processing overhead.
Figure 2.11 Erasing invalid routes after a failure which partitions the network.

2.7 System- and Traffic-Dependent Adaptive Routing (STARA) Algorithm

The System- and Traffic-Dependent Adaptive Routing Algorithm (STARA)\cite{11} uses "mean delay" as the link metric without the assumption of bidirectional links. The objective of STARA is to determine the "best" next hop node $k$ which lies on a minimum-delay path from source node $s$ to destination node $d$ in order to avoid congestion resulting from hop-based algorithms.
The first step of the algorithm is for each node to discover its own receiving and transmitting neighborhood sets in the network by broadcasting a probing packet. All nodes that hear the broadcast from node \( i \) include node \( i \) in their transmitting neighborhood \( N_i^T(t) \) at time instant \( t \). In the meantime, they construct an ACK back to node \( i \) by constrained-flooding, which refers to the fact that packets will be discarded after going over a certain number of hops during the process of broadcasting. After node \( i \) receives ACK packets, it includes the senders of those packets into its receiving neighborhood \( N_i^R(t) \). \( D_{sk}^d(t) \) is defined as the expected delay of a data packet routed from source node \( s \) to destination node \( d \) through next hop node \( k \) at time instant \( t \), where \( k \in N_i^R(t) \). \( D_{sk}^d(t) \) is initiated to zero, and as a result, all the paths from node \( s \) to \( d \) will be explored before any particular path is chosen. STARA chooses a path to send a packet based on the criteria of “path delay.” To get as accurate an estimation of the delay as possible, it is assumed that queueing delay is the dominant part over propagation delays. Each node is assumed to have its own accurate clock, but no synchronization between nodes is needed.

The way to estimate the delay is for the destination node to send an ACK upon receiving a packet. This requires the ACK to have higher priority than any other packets. If the source node does not send any traffic, probing packets are sent to obtain new path delay estimation. For previous estimation, STARA uses an “exponential-forgetting factor” \( \lambda \in [0, 1) \) as a weight:

\[
D_{sk}^d(t) = (1 - \lambda) \sum_{l=0}^{\infty} \lambda^l \times D_{sk}^d(t - l)
\]

(2.6)
past and present values of the parameter, giving more weight to more recent values. Changes of $\lambda$ will be useful in adapting to changes in network topology. However, if no traffic is sent over a path, the new delay will never be estimated again. Therefore, dummy packets are sent for probing purposes. A destination sends an ACK after receiving the probing packet. To get a more accurate delay estimation, a probing packet is given the same priority as a data packet.

To avoid a large reordering buffer in the receiver side, STARA equalizes "all" utilized delays over paths, instead of using pure delay minimization to route the packets to neighbor $k^*$, where

$$D^d_{sk^*}(t) = \min_{k \in N^R(t)} D^d_{sk}(t) \quad \text{(2.7)}$$

Node $s$ allocates its traffic over its neighbors $\{p^d_{sk} : k \in N^R_s(t)\}$ as follows:

$$p^d_{sk}(t) = p^d_{sk}(t - 1) + \alpha(t) \times (D^d_s(t) - D^d_{sk}(t)) \quad \text{(2.8)}$$

where $\alpha(t)$ is the adaption step size and average delay

$$D^d_s(t) = \sum_{k \in N^R_s(t)} p^d_{sk}(t) \times D^d_{sk}(t) \quad \text{(2.9)}$$

Packets will be routed over a small subset among all available paths between any source-destination pair (Figure 2.12). Those paths that have not been used are considered to have large delays at the time the source obtains the delay estimation. Since packets are sent along a chosen path, the delay on this particular path will be greater than
Previously estimated. This avoids later traffic having to be sent on the same path and, hence, delay on every path is equalized. Shortest-path algorithms explicitly force traffic to be sent on the same path with the least hop count because hop counts are the path metric. Chances are some links may be chosen as critical ones because many shortest paths are using them; therefore, congestion along some links may increase packet delay.

In the contrary, mean delay is used as path metrics in STARA. Under the assumption that queueing delay is the dominant part over propagation delay, the delay estimation reflects the condition of paths. However, if the network congests and causes packets to be dropped and retransmitted, the goal of avoiding a large reorder buffer cannot be fulfilled. Therefore, the delay equalization among paths can only be done with the requirement of less congestion existing in the network.

\[
D_{11} + D_{12} = D_{21} + D_{22}
\]

Figure 2.12 Delay equalization of STARA.
CHAPTER 3
MODEL

The objective of this simulator is to generate an environment that could be used as a platform for further studies on mobile networks. To emulate the CDMA physical layer characteristics, the simulator’s possession of link-layer retransmission is essential.

Section 3.1 gives the detail of our mobility model. Section 3.2 describes the routing scheme we used in this thesis. Section 3.3 describes our retransmission scheme. Section 3.4 presents several measurements we take to analyze the performance.

3.1 Mobility Model

The mobility model is for mobile hosts to route packets to the designated destination correctly even if the destination node is moving. To achieve this goal, the networking protocols must be able to optimize the use of the network connection and provide information on location change to the higher layer. Owing to the transmission and reception range constraint of a MANET, a mobile host cannot possibly know the identity of all nodes in the network, and thus communications are done by packet forwarding and broadcasting. Therefore, the operation is done with the routing protocols.
Original routing in ad-hoc networks is done with static recorded routes which are constructed manually. A later advancement is the use of either distance vector or link state routing. After much tune up, new protocols are presented recently.

In this thesis, the routing strategy is chosen to follow a precomputed route which is constructed based on the shortest path algorithm using hop count as link metric. We use a fixed topology for different traffic patterns in order to compare the packet error rates.

3.2 Routing Scheme

The routing in this thesis is based on the shortest-path algorithm. Since a static topology is used as shown in Figure 5.2 on page 53, static routes are specified in the Tcl file.

3.3 FH-CDMA and Link-layer Retransmission

The model of frequency-hopping code-division multiple access (FH-CDMA) is to interleave a "frame" of codewords into several packets. At the beginning and end of a packet, a number of $N_{ts}$ test symbols are added as a flag of packet corruption. Since packets contain encoded codewords, estimation is the best way to tell if the codewords in the packets can be decoded correctly. The detail computation is described later in Chapter 4, and the module implementation is in Sections 5.2.2 and 5.2.3.
3.4 Measurements

We use an semistatic topology in the entire simulation. Throughput and packet delay are our main concerns in comparing different traffic sources applied.

The throughput of a MANET is difficult to compute due to the complex function of topology, channel access protocols, traffic type, traffic matrix, power control, routing algorithm, and so on. In this thesis, we measure the proportion of packets which are received and decoded correctly on an end-to-end basis.

End-to-end delay is measured as the time interval between when then the packet first leaves the source node and when it is successfully decoded at the destination node. If a packet is lost, either because of multiple access or thermal noise, then after a certain time interval, normally one round-trip time, the sending node will retransmit that packet again if the retransmission count is not violated.
CHAPTER 4
PHYSICAL LAYER

In Section 4.1 we summarize the computation of packet error probability under slow-hopping FH-CDMA as presented in [1]. This would allow us to perform packet-level simulations while taking physical layer characteristics into account. In regard to the parameters, Section 4.2 explains the parameters used in [1], and Section 4.3 presents some numerical examples.

4.1 Packet Error Probability Under FH-CDMA

Packets consist of codewords (Figure 4.1). Let \( L \) be the number of codewords in a packet. Typically, the number of codewords is chosen to be equal to the number of symbols in each dwell interval of the frequency-hopping scheme. We are considering slow-hopping, thus there are multiple symbols per dwell interval (i.e., the interval during which a single frequency is used). Slow-hopping seems to be a more widely used FH scheme. Let us suppose that the frequency spectrum is divided into \( q \) bands and during each dwell interval, one particular frequency is chosen with probability \( 1/q \). Some terminology:

- **Slotted Transmission**: We assume that packet is transmitted in fixed slots, i.e., the begin and end times for a packet transmission are fixed. The advantages of slotted over unslotted transmission are well known in the context of the ALOHA
protocol. Slotted transmission requires clock synchronization of the order of the packet length between radios. This is possible because packet lengths are large enough to permit this synchronization. Depending on the distances between the radios, a transmission from radios A and B would reach radio C at different times due to propagation delays. Thus, a slot cannot start and end at the time for radios A, B and C unless additional care is taken. To account for propagation delays, guard slots are added at the beginning and at the end of a packet slot. Thus, a slot includes the time taken to transmit a packet plus the guard slots.

- **Interleaving**: During each dwell interval, one symbol per codeword is transmitted. Thus, if two transmitters happen to transmit at the same frequency during a dwell interval, instead of corrupting an entire codeword, a maximum of one symbol per codeword would be corrupted. Depending upon the error-correcting capability of the code, this may allow the receiver to decode all the codewords.
• *Hit:* We say that a hit has occurred if two transmissions take place at the same frequency at the same time. When a hit occurs, binary symbols in the slot are in error with some probability which we will calculate.

• *Asynchronous hopping:* If each receiver-transmitter pair is time-synchronized to the beginning and end of dwell intervals, then such a scheme is called synchronous hopping. The synchronization needed for packet-slotting is easier since a packet slot is much larger than a dwell interval. In practice, synchronous hopping is not possible. Thus, we assume that hopping is asynchronous. However, it is very hard to compute packet error probabilities for asynchronous hopping exactly. We will make certain approximations (which are actually upper bounds) to compute this probability.

• *Erasures:* In each dwell interval, $2N_t$ test binary symbols are transmitted (Figure 4.2). Half of the test symbols are located at the beginning of the dwell interval and half are located at the end. If more than $\gamma$ test symbols in either group are erased, then the dwell interval is said to be erased. Code symbols in erased dwell intervals are not used in the Reed-Solomon decoding. The idea behind doing this is that it is better to have a symbol erased than to have an errored symbol. When the test binary symbols indicate a large number of errors, the entire interval is assumed to be erased.

• *(n, k) extended Reed-Solomon code:* Each codeword consists of $n$ symbols from an extended Reed-Solomon code. The code rate (i.e., the ratio of information bits to
Figure 4.2 Symbol numbers in a dwell interval.

total number of bits in the codeword) is $k/n$. Each codeword is decoded correctly if $e + 2t \leq n - k$, where $e$ is the number of symbol erasures and $t$ is the number of symbol errors. (Note that $t$ here refers to code symbol errors, not binary symbol errors.) There are $\log_2 n$ bits in each code symbol.

- **Thermal noise:** Even in the absence of a hit, symbols can be received in error or a whole dwell interval may be erased due to the presence of thermal noise.

For the purpose of computing the packet error probability, we assume that the only source of a hit is multiple-access interference. Specifically, we ignore partial-band interference. This interference can be treated without much difficulty.

Let $d$ denote the event that a packet is decoded correctly. Then, due to the properties of the Reed-Solomon code,

$$P(d) = \sum_{t=0}^{(n-k)/2} \sum_{e=0}^{n-k-2t} P(e, t)$$

(4.1)

where we ignore the probability of incorrect decoding when $e + 2t > n - k$ since this probability is typically very small. Let $m$ denote the number of dwell intervals with
multiple access interference. Then,

\[ P(e, t) = \sum_{m=0}^{n} P(e, t|m)P(m) \]  \hspace{1cm} (4.2)

where \( P(m) \) is the probability that the number of dwell intervals hit by multiple-access interference is \( m \). Thus,

\[ P(m) = \binom{n}{m} P_h^m (1 - P_h)^{n-m} \]  \hspace{1cm} (4.3)

where \( P_h \) is the probability that a dwell interval is hit by multiple-access interference. Let \( em \) be the number of dwell intervals erased due to multiple-access interference and \( en \) be the number of dwell intervals erased due to thermal noise. Define \( tm \) and \( tn \) in a similar manner for code symbol errors. Then,

\[ P(e, t|m) = \sum_{em=0}^{e} \sum_{en=0}^{e-em} P(em|m)P(en|en, n) \sum_{tm=0}^{t} \sum_{tn=0}^{t-tm} P(tm|m, em)P(tn|m, em) \]  \hspace{1cm} (4.4)

Suppose that there are \( K \) other users attempting to transmit in a particular packet slot, then \( P_h \) is given by

\[ P(h) = 1 - (1 - 2/q)^K \]  \hspace{1cm} (4.5)

One would use \( 1/q \), instead of \( 2/q \), in the above equation if hopping is synchronous. But since we are assuming that the hopping is asynchronous, \( 2/q \) gives an upper bound.
Let \( p_t \) be the probability that a binary symbol is received in error when thermal noise alone is present. Then,

\[
p_t = \frac{1}{2} e^{-\frac{r E_b}{2N_0}}
\]  

(4.6)

where \( r \) is the rate of the code, \( E_b \) is the energy per information bit and \( N_0 \) is the one-sided density of the thermal noise. Thus, the probability of a code symbol being in error when thermal noise alone is present is given by

\[
p_{tt} = 1 - (1 - p_t)^{\log_2 n}
\]  

(4.7)

since there are \( \log_2 n \) binary symbols in each code symbol. The probability that a dwell interval is erased by thermal noise is given by

\[
p_{et} = 1 - \left( \sum_{i=0}^{\gamma} \binom{N_{ts}}{i} p_t^i (1 - p_t)^{N_{ts} - i} \right)^2
\]  

(4.8)

Recall that the test symbols are on both ends of a dwell interval. Thus, multiple access interference will only hit one of the two sets of test symbols since hopping is asynchronous. Further, we assume that each test symbol that is hit by multiple access interference is in error with probability 1/2. Thus, the probability of erasure, when multiple access interference is present, is given by

\[
p_{em} = 1 - \sum_{i=1}^{\gamma} \binom{N_{ts}}{i} (1/2)^i (1/2)^{N_{ts} - i} \sum_{i=1}^{\gamma} \binom{N_{ts}}{i} p_t^i (1 - p_t)^{N_{ts} - i}
\]  

(4.9)
We will assume that all code symbols in dwell intervals that are hit by multiple-access interference, but not erased, are in error with probability 1. Now, we can compute \( P(e, t|m) \) as follows:

\[
P(e, t|m) = \sum_{e_1=0}^{e} \binom{m}{e_1} p_{em}^{e_1} (1 - p_{em})^{m-e_1} \binom{n-m}{e-e_1} p_{et}^{e-e_1} (1 - p_{et})^{n-m-e+e_1} \\
\times \binom{n-m-e+e_1}{t-m+e_1} p_{tt}^{t-m+e_1} (1 - p_{tt})^{n-e-t}
\]

(4.10)

where we assume \( \binom{a}{b} = 0 \) when \( b < 0 \).

Below we summarize the assumptions that lead to the above computations.

- When a dwell interval is hit by multiple access interference, we assume that one of the two sets of binary test symbols (at the two ends of this dwell interval) is hit.

- When a binary test symbol is hit by multiple access interference, we assume that it is in error with probability 1/2.

- If a dwell interval is not erased by multiple access interference, we assume that all the code symbols in this dwell interval are in error with probability 1.

- If a dwell interval is not hit by multiple access interference, then errors and erasures can occur due to the presence of thermal noise.

- If more than \( \gamma \) binary test symbols are in error, we assume that all the code symbols in that dwell interval are erased.

- We assume that the probability of correct decoding is zero if \( e + 2t > n-k \).
4.2 Physical Layer Simulation Parameters

Physical layer parameters hold an important role in determining the packet error rate at the decoder side in a packet radio network. Packet errors may come from thermal noise of the channel or collision due to multiple access. To model the physical layer impact on the routing scheme, frequency-hopping code-division multiple access (FH-CDMA) is primarily used.

The parameters in [1] are as follows:

- $n = 32$ is the total number of symbols in a codeword

- $k = 16$ is the number of message symbols in a codeword

- $L = 250$ is the number of codewords in a packet

- $q = 100$ is number of frequency bands

- $P[\text{test symbol in error} \mid \text{test symbol hit}] = 0.5$

- $K = \#\text{MobileHosts} - 1$ is the number of packets in flight other than the packet under consideration

- $N_{ts} = 5$ is the number of test symbols in each test interval

Therefore, packet size $= L \times n$ symbols $= L \times n \times 8$ bits and the redundancy added to detect erasures and errors is then

$$\frac{\text{test symbols added}}{\text{original packet size}} = \frac{2N_{ts} \times n}{L \times n} = \frac{2N_{ts}}{n}$$ (4.11)
<table>
<thead>
<tr>
<th>$L$</th>
<th>$n$</th>
<th>$N_{ts}$</th>
<th>Original Packet size</th>
<th>Redundancy Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>32</td>
<td>5</td>
<td>960</td>
<td>0.3333</td>
</tr>
<tr>
<td>100</td>
<td>32</td>
<td>5</td>
<td>3200</td>
<td>0.1</td>
</tr>
<tr>
<td>100</td>
<td>128</td>
<td>5</td>
<td>12800</td>
<td>0.0782</td>
</tr>
</tbody>
</table>

Table 4.1 shows different packet redundancy ratio by different packet sizes. Whether the dwell interval is considered erased or erroneous depends on how many symbols in the test interval are erased.

Expected traffic parameters are a traffic matrix, thermal noise, frame parameters, mobile hosts’ transmission parameters, and frequency hopping parameters. Most of them follow the paper, but traffic matrix is totally different. It specifies the packets sources and destinations. Combining mobile hosts’ transmission parameters, the retransmission is decided by a random generator according to the retransmission probability.

### 4.3 Probability of Correct Decoding versus Number of Test Symbols

Table 4.2 and Figure 4.3 show the result of the C program with various values of $\gamma$ and $E_b/N_0$ running on 10 or 20 static stations. Figures 4.4, 4.5, and 4.6 are the results of probability of correct decoding.
Figure 4.3 Plot of probability of correctly decoding $P(d)$.

Table 4.2 Probability of correctly decoding $P(d)$ w.r.t. different SNR

<table>
<thead>
<tr>
<th>$E_b/N_0$</th>
<th>$\gamma$</th>
<th>$P(d)$ for 10 mobile hosts</th>
<th>$P(d)$ for 20 mobile hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1</td>
<td>0.5536152</td>
<td>0.2612048</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>0.5674986</td>
<td>0.252795</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>0.5404270</td>
<td>0.2150608</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>0.5262660</td>
<td>0.1989448</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>0.5234452</td>
<td>0.1956592</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0.9280606</td>
<td>0.6718476</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>0.9254442</td>
<td>0.6499515</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>0.9167010</td>
<td>0.6148458</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>0.9120865</td>
<td>0.5973606</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>0.9111490</td>
<td>0.5938826</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0.9958762</td>
<td>0.9052194</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>0.9954206</td>
<td>0.8956068</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>0.9947650</td>
<td>0.8838951</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>0.9944140</td>
<td>0.8778713</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>0.9943420</td>
<td>0.8756543</td>
</tr>
</tbody>
</table>
Figure 4.4 Plot of probability of correctly decoding $P(d)$ at SNR = 10 dB.

Figure 4.5 Plot of probability of correctly decoding $P(d)$ at SNR = 11 dB.
Figure 4.6 Plot of probability of correctly decoding $P(d)$ at SNR = 12 dB.
CHAPTER 5
SIMULATION AND RESULTS

5.1 Simulation Environment

The simulator we have used is the Network Simulator\(^1\) (NS) [3] from Berkeley.

The simulator is written in C++ and oTcl (Object Tool Command Language). The oTcl interpreter is used to define the network characteristics, for example, nodes, links, traffic in the network, and protocol types. The parameters for the network are implemented in C++ modules and “bound” to Tcl objects in order to be reconfigurable through Tcl scripts, which are used by NS during the simulations. The packets can be recorded into a trace file with indication of sequence numbers, protocols, source and destination nodes, and time stamps. In regard to the visualization, the simulation result is recorded for the Network Animator (NAM). NAM is able to trace link bandwidth usage, link loss and packet flow. Besides, owing to the recorded simulation process, it is possible to replay the simulation with an adjustable rate. The simulation flow is illustrated in Figure 5.1. For more detail about NAM, refer to Appendix A.

\(^1\)NS version 2.1b4
5.2 Simulator Modifications

A typical NS simulation is designed for end-to-end transmission with TCP (transmission control protocol) or UDP (user datagram protocol). In addition, to simulate a wireless network, we need models of packet loss and retransmission.

5.2.1 Multiple access detection

A packet may be corrupted due to multiple access hits. The computation of the probability of correctly decoding a packet was described in Section 4.1. The NS implementation is done by adding a separate packet queue in each node before packets are accepted. NS has a module called classifier as the interface for dealing with incoming and outgoing packets (and ACKs). Above this classifier, the packets are sent to different agents in order to realize different transport-layer protocols. Each agent can only access packets which belong to itself and send packets to the corresponding end hosts. Therefore, the multiple access packet queue must be implemented at the first interface that sees all connections towards this node in order to determine if a collision occurs.

To distinguish colliding packets, the multiple access packet queue does not actually store the whole packet, but store the arrival time and sequence number instead. Each
packet is held for one packet transmission time before it is sent to the next module. Within this extra time interval, a counter is maintained to accumulate the number of packet arrivals. After the timer goes off, i.e., the extra packet transmission time is reached, the count of arrivals is sent to the FH-CDMA routine as the multiple access parameter.

If this packet gets through the classifier module, it is then sent to the next hop; otherwise, it is simply discarded. The sender node will wait for a certain time to do the retransmission.

5.2.2 Link-layer retransmission

NS does not allow a packet to be dropped at any hop due to wireless link error. To do this, we implemented a hop-by-hop link layer transmission. In other words, the agent is modified to do the forwarding for the packets received from previous hop. Also, a sink module is created to receive packets and forward them to the next hop specified in the Tcl file instead of redirecting them through dynamically found routes. This module can be modified later to follow the route by some routing algorithm.

Two major simulation parameters in this module are:

1. Timeout for retransmission: This is the time interval the node should wait for the ACK before this same packet is retransmitted.
2. *Number of retransmissions*: This parameter is the maximum retransmissions a node should do before eventually giving up a packet. If this number is reached, the packet is discarded and will be recorded as a lost packet.

### 5.2.3 Probability of correct decoded packets

This subroutine calculates the probability of a correct decoded packet according to Section 4.1.

The number of colliding hosts is provided by the *classifier* module. Together with thermal noise factor, the subroutine returns a probability of decodable packet $P_d$. This determines if this packet should be dropped with probability $(1 - P_d)$. Even if there is no collision, there might be some corruption due to thermal noise.

### 5.3 A Random Topology

The simulation scenario we use is the topology as shown in Figure 5.2 which has 15 nodes and 26 links. Ten flows are sent from different source nodes with different path lengths. Table 5.1 lists the intermediate nodes each flow goes through from the source node to the destination node. The traffic generator we use for each flow is either all exponential on/off or Pareto on/off. The traffic starting time for each flow is randomized.
Figure 5.2 A random generated topology.

<table>
<thead>
<tr>
<th>flow ID</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>10</td>
<td>0</td>
<td>8</td>
<td>11</td>
<td>6</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>12</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1 Flow

5.3.1 Exponential on/off source

Exponential on/off source generates traffic according to an exponential on/off distribution. Packets are sent at a fixed rate with constant size during “on” periods, and no packets are sent during the “off” periods. Both the lengths of “on” and “off” periods are taken from an exponential distribution. The parameters adopted in the simulations are shown in Table 5.2.
Table 5.2 Constants used for exponential on/off source

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean burst time</td>
<td>500 ms</td>
</tr>
<tr>
<td>Mean idle time</td>
<td>300 ms</td>
</tr>
<tr>
<td>Packet sending rate</td>
<td>120 Kbps</td>
</tr>
<tr>
<td>Packet size</td>
<td>1000 Bytes</td>
</tr>
</tbody>
</table>

5.3.2 Pareto on/off source

Pareto on/off source generates traffic according to a Pareto on/off distribution. The “on” and “off” periods are taken from a Pareto distribution.

\[
P(X_{burst} < x) = 1 - \left( \frac{\eta_{burst}}{\eta_{burst} + x} \right)^{\beta_{burst}}
\]  
\[P(X_{idle} < x) = 1 - \left( \frac{\eta_{idle}}{\eta_{idle} + x} \right)^{\beta_{idle}}
\]

When the shape parameter $\beta$ is between 1 and 2, Pareto distribution is a good source for bursty traffic with variance of $\infty$. The means of a Pareto traffic are

\[
\mu_{burst} = \frac{\eta_{burst}}{\beta_{burst} - 1}
\]
\[
\mu_{idle} = \frac{\eta_{idle}}{\beta_{idle} - 1}
\]

In NS, the parameters are specified in Table 5.3.

5.3.3 Result

Using $\text{SNR} = 12$ dB, each traffic source of the ten flows generates 75,000 or 100,000 packets. After discarding the first 1,000 packets, the result of 100,000-packet simulation of
Table 5.3 Constants used for Pareto on/off source

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean burst time</td>
<td>500 ms</td>
</tr>
<tr>
<td>((\mu_{\text{burst}}))</td>
<td></td>
</tr>
<tr>
<td>Mean idle time</td>
<td>300 ms</td>
</tr>
<tr>
<td>((\mu_{\text{idle}}))</td>
<td></td>
</tr>
<tr>
<td>Shape ((\beta_{\text{burst}} = \beta_{\text{idle}}))</td>
<td>1.5</td>
</tr>
<tr>
<td>Packet sending rate</td>
<td>120 Kbps</td>
</tr>
<tr>
<td>Packet size</td>
<td>1000 Bytes</td>
</tr>
</tbody>
</table>

the Pareto on/off source is shown in Table 5.4, and exponential on/off source in Table 5.5. Results of 10,000-packet simulation are shown in Tables 5.6 and 5.7.

For the topology shown in Figure 5.2, both exponential on/off source and Pareto on/off source achieve 0% dropping rate with SNR \(\geq 14\) dB regardless of \(\gamma\) and the number of maximum retransmissions. For Pareto on/off source, with a total of 94567.50 packets sent, the average delay is 5040.5759 s; for exponential on/off source, 90479.00 packets are sent with delay of 4989.1523 s.

5.4 Topology in [1]

The simulation scenario we use is the topology as shown in Figure 5.3, which has 12 nodes and 22 links. Sixteen flows are sent from different source nodes with different path lengths. Table 5.8 lists the intermediate nodes each flow goes through from the source node to the destination node. In this subnetwork, only four source-destination pairs are examined: (4, 5), (5, 4), (6, 7) and (7, 6). All packets generated at source nodes 4 through 7 are for the corresponding destinations only. The traffic generator we use for each flow is the Pareto on/off source. Furthermore, the traffic starting time for each flow is randomized.
Figure 5.3 The topology in [1].

5.4.1 Result

After discarding the first 1,000 packets, i.e., transient state, the result of 40,000-packet simulation of the Pareto on/off source is shown in Table 5.9 for SNR = 12 dB, Table 5.10 for SNR = 14 dB, and Table 5.11 for SNR = 16 dB. The simulation result shows that the value of $\gamma$ has little effect on average packet delay (in seconds). The average delay is smaller when $\gamma$ is small. However, the increment of maximum number of retransmissions actually reduces the packet loss rate (Figure 5.4). The packet loss rate decreases dramatically when more than one retransmission is allowed. Results of simulations of Pareto on/off source are shown in Tables 5.9, 5.10, and 5.11.
Figure 5.4 Packet loss rate versus $\gamma$ with SNR = 12 dB.
Table 5.4 Pareto on/off source with 100,000 packets (rate = 120 kbps, idle = 300 ms, burst = 500 ms, shape = 1.5)

<table>
<thead>
<tr>
<th>Max num of ReTX</th>
<th>SNR (dB)</th>
<th>$\gamma$</th>
<th>total sent (pkt)</th>
<th>loss rate (%)</th>
<th>Avg Delay (s) (no transient)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
<td>5</td>
<td>94846.20</td>
<td>0.815712</td>
<td>4978.5468</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>5</td>
<td>95430.80</td>
<td>0.015127</td>
<td>4994.9410</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>5</td>
<td>92967.70</td>
<td>0.000642</td>
<td>4981.7738</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>5</td>
<td>93744.30</td>
<td>0.000000</td>
<td>4988.7857</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>5</td>
<td>93744.30</td>
<td>0.000000</td>
<td>4992.9929</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>5</td>
<td>93744.30</td>
<td>0.000000</td>
<td>5009.8215</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>4</td>
<td>94846.20</td>
<td>0.815712</td>
<td>4978.5468</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>4</td>
<td>95432.50</td>
<td>0.015326</td>
<td>5011.3831</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>4</td>
<td>92967.70</td>
<td>0.000642</td>
<td>4955.4393</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>4</td>
<td>93744.30</td>
<td>0.000000</td>
<td>5009.8215</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>4</td>
<td>93744.30</td>
<td>0.000000</td>
<td>5009.8215</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>4</td>
<td>93744.30</td>
<td>0.000000</td>
<td>5009.8215</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>3</td>
<td>94999.70</td>
<td>0.830195</td>
<td>4985.4653</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>3</td>
<td>95432.50</td>
<td>0.015326</td>
<td>5011.3831</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>3</td>
<td>92967.70</td>
<td>0.000642</td>
<td>4955.4393</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>3</td>
<td>93744.30</td>
<td>0.000000</td>
<td>5009.8215</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>3</td>
<td>93744.30</td>
<td>0.000000</td>
<td>5009.8215</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>93744.30</td>
<td>0.000000</td>
<td>5009.8215</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>2</td>
<td>95989.90</td>
<td>0.861862</td>
<td>5028.8496</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>2</td>
<td>94321.30</td>
<td>0.015817</td>
<td>4987.0207</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>2</td>
<td>92967.70</td>
<td>0.000642</td>
<td>4955.4393</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>2</td>
<td>93304.20</td>
<td>0.000000</td>
<td>4991.3926</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>2</td>
<td>93304.20</td>
<td>0.000000</td>
<td>4991.3926</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>2</td>
<td>93304.20</td>
<td>0.000000</td>
<td>4991.3926</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>1</td>
<td>94605.80</td>
<td>0.838115</td>
<td>4991.2561</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>1</td>
<td>93179.20</td>
<td>0.013213</td>
<td>4968.2011</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>1</td>
<td>93404.20</td>
<td>0.000324</td>
<td>5001.5033</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>1</td>
<td>92747.50</td>
<td>0.000000</td>
<td>4954.9713</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>1</td>
<td>92747.50</td>
<td>0.000000</td>
<td>4954.9713</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>1</td>
<td>92747.50</td>
<td>0.000000</td>
<td>4954.9713</td>
</tr>
</tbody>
</table>
Table 5.5 Exponential on/off source with 100,000 packets (rate = 300 kbps, idle = 300 ms, burst = 500 ms)

<table>
<thead>
<tr>
<th>Max num of ReTX (dB)</th>
<th>SNR (dB)</th>
<th>$\gamma$</th>
<th>total sent (pkt)</th>
<th>loss rate (%)</th>
<th>Avg Delay (s) (no transient)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
<td>5</td>
<td>93813.90</td>
<td>0.828402</td>
<td>5004.0417</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>5</td>
<td>94082.50</td>
<td>0.015108</td>
<td>5000.1515</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>5</td>
<td>93985.00</td>
<td>0.000426</td>
<td>5001.9944</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>5</td>
<td>94269.80</td>
<td>0.000000</td>
<td>5003.6880</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>5</td>
<td>94269.80</td>
<td>0.000000</td>
<td>5004.7042</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>5</td>
<td>94269.80</td>
<td>0.000000</td>
<td>5008.7689</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>4</td>
<td>93925.50</td>
<td>0.823014</td>
<td>5007.4721</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>4</td>
<td>94082.50</td>
<td>0.015108</td>
<td>4996.2613</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>4</td>
<td>93884.30</td>
<td>0.000319</td>
<td>5001.5645</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>4</td>
<td>94269.80</td>
<td>0.000000</td>
<td>5008.7689</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>4</td>
<td>94269.80</td>
<td>0.000000</td>
<td>5008.7689</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>4</td>
<td>94269.80</td>
<td>0.000000</td>
<td>5008.7689</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>3</td>
<td>93778.40</td>
<td>0.820801</td>
<td>4996.2809</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>3</td>
<td>94082.50</td>
<td>0.015108</td>
<td>4996.2613</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>3</td>
<td>94001.10</td>
<td>0.000212</td>
<td>4998.8435</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>3</td>
<td>94057.60</td>
<td>0.000000</td>
<td>5000.1458</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>3</td>
<td>94057.60</td>
<td>0.000000</td>
<td>5000.1458</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>94057.60</td>
<td>0.000000</td>
<td>5000.1458</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>2</td>
<td>93971.50</td>
<td>0.801416</td>
<td>4999.8479</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>2</td>
<td>94183.90</td>
<td>0.011260</td>
<td>4993.3538</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>2</td>
<td>94219.50</td>
<td>0.000211</td>
<td>4999.4224</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>2</td>
<td>93893.00</td>
<td>0.000000</td>
<td>5002.0651</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>2</td>
<td>93931.00</td>
<td>0.000000</td>
<td>5002.0651</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>2</td>
<td>93931.00</td>
<td>0.000000</td>
<td>5002.0651</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>1</td>
<td>94008.10</td>
<td>0.812574</td>
<td>5002.6134</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>1</td>
<td>94036.60</td>
<td>0.012324</td>
<td>4999.3098</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>1</td>
<td>93948.40</td>
<td>0.000213</td>
<td>4992.9282</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>1</td>
<td>94175.40</td>
<td>0.000000</td>
<td>5004.1395</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>1</td>
<td>94175.40</td>
<td>0.000000</td>
<td>5004.1395</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>1</td>
<td>92747.50</td>
<td>0.000000</td>
<td>4954.9713</td>
</tr>
</tbody>
</table>
Table 5.6 Pareto on/off source with 75,000 packets (rate = 120 kbps, idle = 300 ms, burst = 500 ms, shape = 1.5)

<table>
<thead>
<tr>
<th>Max num of ReTX</th>
<th>SNR (dB)</th>
<th>$\gamma$</th>
<th>total sent (pkt)</th>
<th>loss rate (%)</th>
<th>Avg Delay (s) (no transient)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
<td>5</td>
<td>75919.50</td>
<td>0.832303</td>
<td>3979.1697</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>5</td>
<td>75918.20</td>
<td>0.015967</td>
<td>3985.8248</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>5</td>
<td>75148.10</td>
<td>0.000667</td>
<td>4011.0077</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>5</td>
<td>74846.60</td>
<td>0.000000</td>
<td>4004.0564</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>5</td>
<td>74846.60</td>
<td>0.000000</td>
<td>4004.0564</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>5</td>
<td>74846.60</td>
<td>0.000000</td>
<td>4004.0564</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>4</td>
<td>75919.50</td>
<td>0.832303</td>
<td>3979.1697</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>4</td>
<td>75918.20</td>
<td>0.015967</td>
<td>3985.8248</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>4</td>
<td>75148.10</td>
<td>0.000667</td>
<td>4011.0077</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>4</td>
<td>74846.60</td>
<td>0.000000</td>
<td>4004.0564</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>4</td>
<td>74846.60</td>
<td>0.000000</td>
<td>4004.0564</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>4</td>
<td>74846.60</td>
<td>0.000000</td>
<td>4004.0564</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>3</td>
<td>75919.50</td>
<td>0.832170</td>
<td>3979.1672</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>3</td>
<td>75918.20</td>
<td>0.015967</td>
<td>3985.8248</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>3</td>
<td>75148.10</td>
<td>0.000667</td>
<td>4011.0077</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>3</td>
<td>74846.60</td>
<td>0.000000</td>
<td>4004.0564</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>3</td>
<td>74846.60</td>
<td>0.000000</td>
<td>4004.0564</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>74846.60</td>
<td>0.000000</td>
<td>4004.0564</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>2</td>
<td>75933.20</td>
<td>0.829213</td>
<td>3979.8719</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>2</td>
<td>75534.90</td>
<td>0.016435</td>
<td>3992.9479</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>2</td>
<td>75148.10</td>
<td>0.000667</td>
<td>4011.0077</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>2</td>
<td>74754.00</td>
<td>0.000000</td>
<td>4000.6683</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>2</td>
<td>74754.00</td>
<td>0.000000</td>
<td>4000.6683</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>2</td>
<td>74754.00</td>
<td>0.000000</td>
<td>4000.6683</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>1</td>
<td>75719.80</td>
<td>0.829252</td>
<td>3990.4715</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>1</td>
<td>74466.30</td>
<td>0.013435</td>
<td>3955.7066</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>1</td>
<td>74843.40</td>
<td>0.000397</td>
<td>4004.3595</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>1</td>
<td>74846.60</td>
<td>0.000000</td>
<td>4003.4054</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>1</td>
<td>74846.60</td>
<td>0.000000</td>
<td>4003.4054</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>1</td>
<td>74846.60</td>
<td>0.000000</td>
<td>4003.4054</td>
</tr>
</tbody>
</table>
Table 5.7 Exponential on/off source with 75,000 packets (rate = 300 kbps, idle = 300 ms, burst = 500 ms)

<table>
<thead>
<tr>
<th>Max num of ReTX</th>
<th>SNR (dB)</th>
<th>$\gamma$</th>
<th>total sent (pkt)</th>
<th>loss rate (%)</th>
<th>Avg Delay (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
<td>5</td>
<td>75006.00</td>
<td>0.822940</td>
<td>4002.8114</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>5</td>
<td>75307.40</td>
<td>0.014214</td>
<td>3998.6355</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>5</td>
<td>75104.40</td>
<td>0.000267</td>
<td>4001.8415</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>5</td>
<td>75376.60</td>
<td>0.000000</td>
<td>4008.7789</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>5</td>
<td>75376.60</td>
<td>0.000000</td>
<td>4008.7789</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>5</td>
<td>75376.60</td>
<td>0.000000</td>
<td>4008.7789</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>4</td>
<td>75089.50</td>
<td>0.833248</td>
<td>4005.0919</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>4</td>
<td>75307.40</td>
<td>0.014214</td>
<td>3998.6355</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>4</td>
<td>75071.10</td>
<td>0.000399</td>
<td>3999.8958</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>4</td>
<td>75376.60</td>
<td>0.000000</td>
<td>4008.7789</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>4</td>
<td>75376.60</td>
<td>0.000000</td>
<td>4008.7789</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>4</td>
<td>75376.60</td>
<td>0.000000</td>
<td>4008.7789</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>3</td>
<td>75027.70</td>
<td>0.809950</td>
<td>3997.5580</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>3</td>
<td>75307.40</td>
<td>0.014214</td>
<td>3998.6355</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>3</td>
<td>75202.40</td>
<td>0.000133</td>
<td>3998.8395</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>3</td>
<td>75389.10</td>
<td>0.000000</td>
<td>4009.4436</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>3</td>
<td>75389.10</td>
<td>0.000000</td>
<td>4009.4436</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>75389.10</td>
<td>0.000000</td>
<td>4009.4436</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>2</td>
<td>75188.80</td>
<td>0.804988</td>
<td>4000.7890</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>2</td>
<td>75403.80</td>
<td>0.011275</td>
<td>3995.5231</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>2</td>
<td>75370.80</td>
<td>0.000264</td>
<td>3999.4356</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>2</td>
<td>75078.20</td>
<td>0.000000</td>
<td>3997.6329</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>2</td>
<td>75078.20</td>
<td>0.000000</td>
<td>3997.6329</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>2</td>
<td>75078.20</td>
<td>0.000000</td>
<td>3997.6329</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>1</td>
<td>75172.50</td>
<td>0.848442</td>
<td>4001.5147</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>1</td>
<td>75228.60</td>
<td>0.013007</td>
<td>4000.6319</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>1</td>
<td>75270.40</td>
<td>0.000267</td>
<td>3998.5961</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>1</td>
<td>75338.30</td>
<td>0.000000</td>
<td>4005.6263</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>1</td>
<td>75338.30</td>
<td>0.000000</td>
<td>4005.6263</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>1</td>
<td>75338.30</td>
<td>0.000000</td>
<td>4005.6263</td>
</tr>
</tbody>
</table>
Table 5.8 Flow for topology in [1].

<table>
<thead>
<tr>
<th>Flow ID</th>
<th>4 $\Rightarrow$ 0 $\Rightarrow$ 1 $\Rightarrow$ 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4 $\Rightarrow$ 0 $\Rightarrow$ 3 $\Rightarrow$ 5</td>
</tr>
<tr>
<td>2</td>
<td>4 $\Rightarrow$ 2 $\Rightarrow$ 1 $\Rightarrow$ 5</td>
</tr>
<tr>
<td>3</td>
<td>4 $\Rightarrow$ 2 $\Rightarrow$ 3 $\Rightarrow$ 5</td>
</tr>
<tr>
<td>4</td>
<td>5 $\Rightarrow$ 1 $\Rightarrow$ 0 $\Rightarrow$ 4</td>
</tr>
<tr>
<td>5</td>
<td>5 $\Rightarrow$ 1 $\Rightarrow$ 2 $\Rightarrow$ 4</td>
</tr>
<tr>
<td>7</td>
<td>5 $\Rightarrow$ 3 $\Rightarrow$ 0 $\Rightarrow$ 4</td>
</tr>
<tr>
<td>8</td>
<td>5 $\Rightarrow$ 3 $\Rightarrow$ 2 $\Rightarrow$ 4</td>
</tr>
<tr>
<td>8</td>
<td>6 $\Rightarrow$ 0 $\Rightarrow$ 2 $\Rightarrow$ 7</td>
</tr>
<tr>
<td>8</td>
<td>6 $\Rightarrow$ 0 $\Rightarrow$ 3 $\Rightarrow$ 7</td>
</tr>
<tr>
<td>8</td>
<td>6 $\Rightarrow$ 1 $\Rightarrow$ 2 $\Rightarrow$ 7</td>
</tr>
<tr>
<td>8</td>
<td>6 $\Rightarrow$ 1 $\Rightarrow$ 3 $\Rightarrow$ 7</td>
</tr>
<tr>
<td>8</td>
<td>7 $\Rightarrow$ 2 $\Rightarrow$ 0 $\Rightarrow$ 6</td>
</tr>
<tr>
<td>8</td>
<td>7 $\Rightarrow$ 2 $\Rightarrow$ 1 $\Rightarrow$ 6</td>
</tr>
<tr>
<td>8</td>
<td>7 $\Rightarrow$ 3 $\Rightarrow$ 0 $\Rightarrow$ 6</td>
</tr>
<tr>
<td>8</td>
<td>7 $\Rightarrow$ 3 $\Rightarrow$ 1 $\Rightarrow$ 6</td>
</tr>
</tbody>
</table>
Table 5.9 Pareto on/off source with 40,000 packets for SNR = 12 dB (rate = 5 kbps, idle = 300 ms, burst = 500 ms, shape = 1.5)

<table>
<thead>
<tr>
<th>Max num of ReTX</th>
<th>SNR (dB)</th>
<th>$\gamma$</th>
<th>total sent (pkt)</th>
<th>loss rate (%)</th>
<th>Avg Delay (s) (no transient)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>12</td>
<td>5</td>
<td>38571.19</td>
<td>14.995576</td>
<td>4976.8153</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>5</td>
<td>39053.94</td>
<td>0.696426</td>
<td>4968.9048</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>5</td>
<td>38568.38</td>
<td>0.69307</td>
<td>4982.4112</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>5</td>
<td>38855.38</td>
<td>0.004938</td>
<td>4998.4810</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>5</td>
<td>39027.75</td>
<td>0.000480</td>
<td>4994.0442</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>5</td>
<td>38831.88</td>
<td>0.000000</td>
<td>4986.5081</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>4</td>
<td>38480.75</td>
<td>14.987718</td>
<td>4996.0796</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>4</td>
<td>39123.50</td>
<td>0.702823</td>
<td>4974.4945</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>4</td>
<td>38568.38</td>
<td>0.69307</td>
<td>4982.4112</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>4</td>
<td>38855.38</td>
<td>0.004938</td>
<td>4998.4810</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>4</td>
<td>39027.75</td>
<td>0.000480</td>
<td>4994.0442</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>4</td>
<td>38831.88</td>
<td>0.000000</td>
<td>4986.5081</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>3</td>
<td>38480.75</td>
<td>14.987718</td>
<td>4996.0796</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>3</td>
<td>39123.50</td>
<td>0.702823</td>
<td>4974.4945</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>3</td>
<td>38936.94</td>
<td>0.625210</td>
<td>5030.2217</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>3</td>
<td>38855.38</td>
<td>0.004938</td>
<td>4980.4810</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>3</td>
<td>39027.75</td>
<td>0.000480</td>
<td>4994.0442</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>38831.88</td>
<td>0.000000</td>
<td>4986.5081</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>2</td>
<td>38480.75</td>
<td>14.987719</td>
<td>4996.0795</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>2</td>
<td>38798.50</td>
<td>0.639475</td>
<td>4951.7818</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>2</td>
<td>39148.31</td>
<td>0.692151</td>
<td>5005.1171</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>2</td>
<td>38855.38</td>
<td>0.004938</td>
<td>4998.4810</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>2</td>
<td>39027.75</td>
<td>0.000480</td>
<td>4994.0442</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>2</td>
<td>38218.81</td>
<td>0.000000</td>
<td>5006.0598</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>1</td>
<td>38796.69</td>
<td>15.007272</td>
<td>4989.4843</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>1</td>
<td>38993.75</td>
<td>0.666016</td>
<td>4965.1786</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>1</td>
<td>39422.50</td>
<td>0.657787</td>
<td>5051.1774</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>1</td>
<td>38618.50</td>
<td>0.002750</td>
<td>4969.6236</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>1</td>
<td>38998.19</td>
<td>0.000479</td>
<td>4990.5082</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>1</td>
<td>38866.38</td>
<td>0.000000</td>
<td>4976.6494</td>
</tr>
</tbody>
</table>
Table 5.10 Pareto on/off source with 40,000 packets for SNR = 14 dB (rate = 5 kbps, idle = 300 ms, burst = 500 ms, shape = 1.5)

<table>
<thead>
<tr>
<th>Max num of ReTX</th>
<th>SNR (dB)</th>
<th>(\gamma)</th>
<th>total sent (pkt)</th>
<th>loss rate (%)</th>
<th>Avg Delay (s) (no transient)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>14</td>
<td>5</td>
<td>37941.38</td>
<td>14.907746</td>
<td>5022.4886</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>5</td>
<td>38811.00</td>
<td>0.664337</td>
<td>4992.4812</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>5</td>
<td>38896.44</td>
<td>0.053259</td>
<td>4963.7723</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>5</td>
<td>39575.75</td>
<td>0.002079</td>
<td>4988.3548</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>5</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>5</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>0</td>
<td>14</td>
<td>4</td>
<td>37941.38</td>
<td>14.907746</td>
<td>5022.4886</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>4</td>
<td>38811.00</td>
<td>0.664337</td>
<td>4992.4812</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>4</td>
<td>38896.44</td>
<td>0.053259</td>
<td>4963.7723</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>4</td>
<td>39575.75</td>
<td>0.002079</td>
<td>4988.3548</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>4</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>4</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>0</td>
<td>14</td>
<td>3</td>
<td>37941.38</td>
<td>14.907746</td>
<td>5022.4886</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>3</td>
<td>38811.00</td>
<td>0.664337</td>
<td>4992.4812</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>3</td>
<td>38896.44</td>
<td>0.053259</td>
<td>4963.7723</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>3</td>
<td>39575.75</td>
<td>0.002079</td>
<td>4988.3548</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>3</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>3</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>0</td>
<td>14</td>
<td>2</td>
<td>37941.38</td>
<td>14.907746</td>
<td>5022.4886</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>2</td>
<td>38811.00</td>
<td>0.664337</td>
<td>4992.4812</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>2</td>
<td>38896.44</td>
<td>0.053259</td>
<td>4963.7723</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>2</td>
<td>39575.75</td>
<td>0.002079</td>
<td>4988.3548</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>2</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>2</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>0</td>
<td>14</td>
<td>1</td>
<td>37941.38</td>
<td>14.907746</td>
<td>5022.4886</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>1</td>
<td>38811.00</td>
<td>0.664337</td>
<td>4992.4812</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>1</td>
<td>38896.44</td>
<td>0.053259</td>
<td>4963.7723</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1</td>
<td>39575.75</td>
<td>0.002079</td>
<td>4988.3548</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>1</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>1</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
</tbody>
</table>
Table 5.11 Pareto on/off source with 40,000 packets for SNR = 16 dB (rate = 5 kbps, idle = 300 ms, burst = 500 ms, shape = 1.5)

<table>
<thead>
<tr>
<th>Max num of ReTX</th>
<th>SNR (dB)</th>
<th>$\gamma$</th>
<th>total sent (pkt)</th>
<th>loss rate (%)</th>
<th>Avg Delay (s) (no transient)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16</td>
<td>5</td>
<td>37941.38</td>
<td>14.907746</td>
<td>5022.4886</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>5</td>
<td>38811.00</td>
<td>0.664337</td>
<td>4992.4812</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>5</td>
<td>38896.44</td>
<td>0.053259</td>
<td>4963.7723</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>5</td>
<td>39575.75</td>
<td>0.002079</td>
<td>4988.3548</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>5</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>5</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>0</td>
<td>16</td>
<td>4</td>
<td>37941.38</td>
<td>14.907746</td>
<td>5022.4886</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>4</td>
<td>38811.00</td>
<td>0.664337</td>
<td>4992.4812</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>4</td>
<td>38896.44</td>
<td>0.053259</td>
<td>4963.7723</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>4</td>
<td>39575.75</td>
<td>0.002079</td>
<td>4988.3548</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>4</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>4</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>0</td>
<td>16</td>
<td>3</td>
<td>37941.38</td>
<td>14.907746</td>
<td>5022.4886</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>3</td>
<td>38811.00</td>
<td>0.664337</td>
<td>4992.4812</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>3</td>
<td>38896.44</td>
<td>0.053259</td>
<td>4963.7723</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>3</td>
<td>39575.75</td>
<td>0.002079</td>
<td>4988.3548</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>3</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>3</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>0</td>
<td>16</td>
<td>2</td>
<td>37941.38</td>
<td>14.907746</td>
<td>5022.4886</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>2</td>
<td>38811.00</td>
<td>0.664337</td>
<td>4992.4812</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>2</td>
<td>38896.44</td>
<td>0.053259</td>
<td>4963.7723</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>2</td>
<td>39575.75</td>
<td>0.002079</td>
<td>4988.3548</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>2</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>2</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>0</td>
<td>16</td>
<td>1</td>
<td>37941.38</td>
<td>14.907746</td>
<td>5022.4886</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>1</td>
<td>38811.00</td>
<td>0.664337</td>
<td>4992.4812</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>1</td>
<td>38896.44</td>
<td>0.053259</td>
<td>4963.7723</td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>1</td>
<td>39575.75</td>
<td>0.002079</td>
<td>4988.3548</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>1</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>1</td>
<td>38831.69</td>
<td>0.000000</td>
<td>4977.2089</td>
</tr>
</tbody>
</table>
CHAPTER 6
CONCLUSION

In this thesis we have developed a simulation tool to study the effect of the link-layer retransmission on delay and packet loss rates under FH-CDMA in all-wireless networks. The traffic sources analyzed here are exponential on/off source and Pareto on/off source under different SNR values.

One of the control parameters is $\gamma$, which is the maximum allowance of erroneous test symbols in each of the dwell intervals of an interleaved packet. If more than gamma test symbols are corrupted in a given dwell interval, this dwell interval is said to be erased, and consequently the corresponding symbols in each codeword are erased. In Reed-Solomon code, if a code symbol is erased, it is not used in the decoding. The other control parameter is the "maximum number of retransmission" for link-layer retransmission. Up to this number of retransmissions can be done in each forwarding node. A packet is said to be dropped if it has been determined undecodable after each retransmission. Based on those parameters, we can calculate the end-to-end packet loss rate and average packet delay.

The NS simulator is applied to model the system. Using modules in NS, a scalable topology can be created by Tcl scripts. New modules are created in order to operate properly under a hop-to-hop basis for collision calculation: the original TCP modules
are modified into source, sink, and forwarding agents. The classifier module is extended to take care of the physical layer calculation which corresponds to the correct decoding probability of a packet. Since end-to-end traffic has been broken up to be a hop-by-hop, traffic is then in the unit of "flow."

The results of the simulation of a random topology and the topology in [1] show that increasing the maximum number of retransmissions reduces the packet loss rate, but increases the average packet delay. The γ value can affect the correct decoding probability of a packet and hence the packet dropping rate, but the effect is smaller than that of the maximum number of retransmissions. Allowing to retransmit can dramatically alleviate the packet being dropped due to collision or noise.

Since each traffic flow and the physical channel characteristics can be modeled in a hop-to-hop basis, the current simulation routing scheme can be further scaled up to a dynamic one without changing the physical layer model. In an all-wireless ad-hoc network environment, the route for each source-destination pair may change dynamically under some routing algorithm according to some specific link metrics. As long as each mobile host is able to detect collision, the model is still valid. Therefore, a future research interest will be to incorporate the physical layer properties with dynamic topologies in order to evaluate different routing schemes in a mobile ad-hoc network. Furthermore, path loss characteristics should be modeled to accurately simulate networks over large areas.
APPENDIX A

NETWORK ANIMATOR (NAM)

Figure A.1 shows a screen shot of Network Animator (NAM). NAM is able to trace the link bandwidth used and link loss during the simulation. The screen shot of tracing a packet is shown in Figure A.2. A simulation file written in Tcl is run by NS. An output file for NAM is then generated after the simulation is done. NAM simply displays the node location and packet flows according to the NAM file. Therefore, NAM has the good feature of being able to play back the simulation with adjustable display rate and different topology layout. Incorporating with mobile node module, NAM is capable of displaying moving topology dynamically.
Figure A.1 NAM display.
Figure A.2 The graphical interface of NAM with link and packet trace.
REFERENCES


