Evaluation of Wireless Ad Hoc Protocols Using Random Motion

A prototype was used to run simulation based on different scenarios to obtain significant results in order to highlight the strengths and pitfalls of the protocols under random motion

A Thesis
Submitted to the Department of Computer Science and Engineering
Of
BRAC University
By

Nabil Imani
Student ID: 05210022
Malik Asif Russd
Student ID: 06110023

In Partial Fulfillment of the
Requirements for the Degree
of
Bachelor of Science in Electronics of Communication Engineering
Summer 2009

DECLARATION

I hereby declare that this thesis is based on the results found by ourselves. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted for any degree.

Signature of
Supervisor

____
Sadia Hamid Kazi

Students' names & Signatures:

____
1. Nabil Imani

____
2. Malik Asif Russd

ABSTRACT

The evolution of wireless communication and circuit technology has enabled the development of an infrastructure which consists of sensing, computation and communication units that makes administrator capable to observe and react to a phenomenon in a particular environment. The building block of such an infrastructure is comprised of hundreds or thousands of small, low cost, multifunctional devices which have the ability to sense compute and communicate using short range transceivers known as sensor nodes. The interconnection of these nodes forming a network called wireless sensor network.

Ad hoc networks are characterized by a lack of infrastructure, and by a random and quickly changing network topology. Thus the need is there for a tough dynamic routing protocol that can accommodate such an environment. The report consists of elaborate description of three different Ad Hoc routing protocols– DSDV, DSR and AODV.

As there are numerous studies that have already taken place for basic routing protocols we focused on basing our study for random motion. This allows us to obtain a more realistic data as not all nodes stays stationary. A prototype was used to run simulation based on different scenarios under random functions that obtained significant results in order to highlight the strength and weakness of the protocols through analyzing performance indicators, namely throughput and goodput.

Contents

INTRODUCTION

Ad hoc Sensor Networks

During the past few years, advances and breakthroughs in micro electro mechanical systems, which consist of the integration of mechanical elements, sensors and electronics on a common silicon substrate by using cutting edge micro fabrication technology, have enabled low cost and low power wireless communication capabilities to arise. These new breakthroughs enable tiny sensor nodes to communicate in short distances. Thus a small sensor node can be either completely self contained or integrated into other equipment, and a large number of such nodes can be deployed in an area of interest. Each sensor node is capable of a limited amount of processing, but when coordinated with the information from other nodes, they have the ability to measure the given physical environment in great details or to execute a task with complex functions.

Thus, a sensor network can be described as a group of sensor nodes that synchronizes with each other to perform some specific actions. Since each sensor node is fitted with an on-board processor, sensor nodes use their processing abilities to find out simple computations and transmit only the required data. These features allow the Sensor networks to have a wide potential use for both military and commercial applications.

For military purposes distributed sensor networks could be used to gather information regarding potential threats such as, enemy personnel or vehicles, biological or chemical agents, or radioactive fallout. Sensor networks could also be used to monitor environmental conditions, gather information in high risk areas, or maintain physical grounds security. In such applications, running wires or cabling is usually impractical. A sensor network is required that is fast and easy to install and maintain.

Requirements for Ad hoc networks

We are hoping to focus our research on providing information for disaster management. In order for ad hoc sensor networks to work in those environment and condition they are required to include the following:

- Large number of sensors: Apart from the sensors on the ocean floor or the use of mobile, robotic sensors etc, most of the nodes in a smart sensor network are stationary. Networks of 10,000 or even 100,000 nodes are envisioned, so scalability is a major issue.

- Low energy use: Since in many applications the sensor nodes will be placed in a remote area, service of a node may not be possible. In this case, the lifetime of a node may be determined by the battery life, thereby requiring the minimization of energy expenditure.

- Network self-organization: Given the large number of nodes and their potential placement in hostile locations, it is essential that the network be able to self-organize; manual configuration is not feasible.

- Querying ability: A user may want to query an individual node or a group of nodes for information collected in the region. Depending on the amount of data fusion performed, it may not be feasible to transmit a large amount of the data across the network. Instead, various local sink nodes will collect the data from a given area and create summary messages. A query may be directed to the sink node nearest to the desired location. [2]

With the coming availability of low cost, short range radios along with advances in wireless networking, it is expected that wireless ad hoc sensor networks will become commonly deployed. In these networks, each node may be equipped with a variety of sensors, such as acoustic, seismic, infrared, still/motion video camera, etc. These nodes may be organized in clusters such that a locally occurring event can be detected by most of, if not all, the nodes in a cluster. Each node may have sufficient processing power to make a decision, and it will be able to broadcast this decision to the other nodes in the cluster. One node may act as the cluster master, and it may also contain a longer range radio using a protocol such as IEEE 802.11 or Bluetooth. [2]

Proactive (Table-driven routing protocols):

Proactive routing protocols maintain routes to all destinations, regardless of whether or not these routes are needed. In order to maintain correct route information, a node must periodically send control messages. Therefore, proactive routing protocols may waste bandwidth since control messages are sent out unnecessarily when there is no data traffic. The main advantage of this category of protocols is that hosts can quickly obtain route information and quickly establish a session.

Features of Proactive Routing Protocol:

- Tries to keep routing-information to all nodes every time up-to-date.
- Proactive protocols are based on periodic exchange of control messages and maintaining routing tables.
- Each node maintains complete information about the network topology locally.
- This information is collected through proactive exchange of partial routing tables stored at each node.
- Since each node knows the complete topology, a node can immediately find the best route to a destination.
- However, a proactive protocol generates large volume of control messages and this may take up a large part of the available bandwidth.

- The control messages may consume almost the entire bandwidth with a large number of nodes and increased mobility. [3,4,5]

DSDV

Destination-Sequenced Distance Vector routing protocol is a classic routing protocol which is based on the Distributed Bellman-Ford algorithm. It is quite suitable for creating ad hoc networks with small number of nodes.

In DSDV, each route is tagged with a sequence number indicating how old the route is, this is originated by the destination. The main contribution of the algorithm was to solve the Routing Loop problem. Each node manages its own sequence number by assigning it two greater than the old one every time. When a route update with a higher sequence number is received, the old route is replaced. In case of different routes with the same sequence number, the route with better metric is used.

The updates can be detected in two ways, it could be transmitted periodically or immediately when any topology change is detected. There are also two ways of performing routing update, “Full Dump” allows a node to transmit the complete routing table, while in “incremental update” only the entries that have changed after the last update. In DSDV a “Settling Time” data is employed which is used to predict the time when route becomes stable in order to avoid variations in route updates. [7]

In DSDV, broken link may be detected by the layer-2 protocol, or it may instead be inferred if no broadcasts have been received for a while from a former neighboring node.

DSDV requires a regular update of its routing tables, which uses up battery power and a small amount of bandwidth even when the network is idle. Whenever the topology of the network changes, a new sequence number is necessary before the network reconverges. Thus, DSDV is not suitable for highly dynamic networks. [6]

DSDV in operation

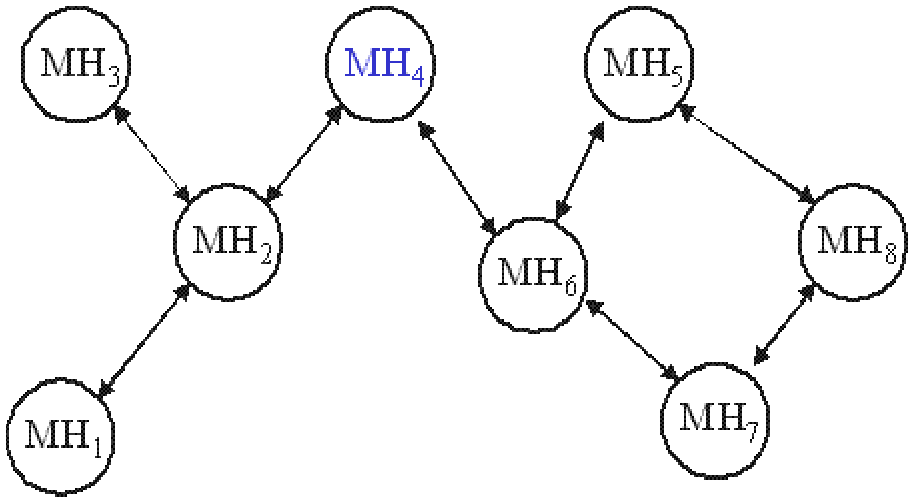


Fig 2.3.b.ii) Operation of DSDV (a)

Destination	**Next Hop**	**Metric**	**Seq. No**
MH4	MH4	0	S406_MH4
MH1	MH2	2	S128_MH1
MH2	MH2	1	S564_MH2
MH3	MH2	2	S710_MH3
MH5	MH6	2	S392_MH5
MH6	MH6	1	S076_MH6
MH7	MH6	2	S128_MH7
MH8	MH6	3	S050_MH8

After MH1 moves in the network from the neighboring of MH2 to MH7 and MH8.

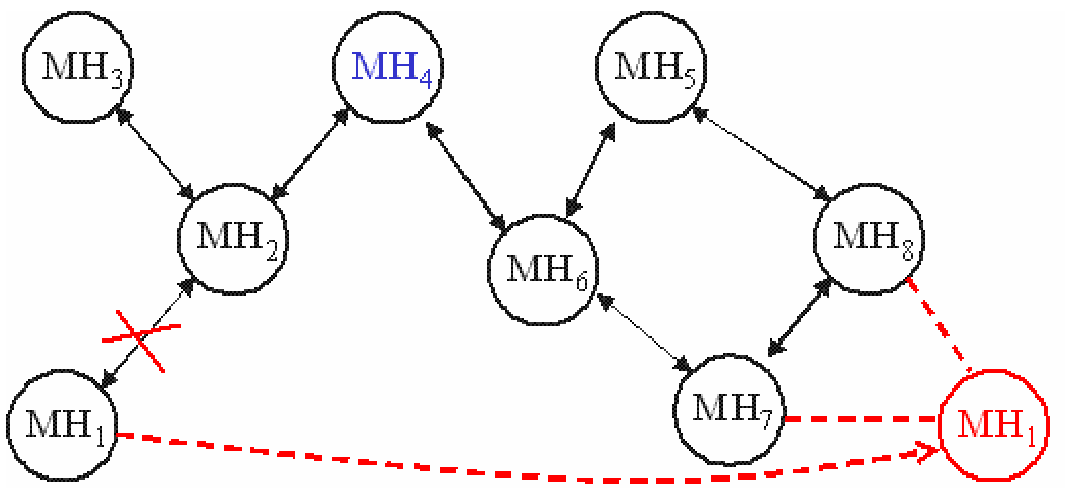


Fig 2.3.b.iii) Operation of DSDV (b)

1. We get a triggered update by MH1, broadcasted to MH7 and MH8.
2. On detection of broken link: Immediate incremental update triggered by MH2 with odd sequence number and infinite metric.
3. Updates are propagated through the network. [9]

Destination	**Next Hop**	**Metric**	**Seq. No**
MH4	MH4	0	S516_MH4
MH1	MH6	3	S238_MH1
MH2	MH2	1	S674_MH2
MH3	MH2	2	S820_MH3
MH5	MH6	2	S502_MH5
MH6	MH6	1	S186_MH6

MH7	MH6	2	S238_MH7
MH8	MH6	3	S160_MH8

Advantages of DSDV:

- DSDV is an efficient protocol for route discovery. Whenever a route to a new destination is required, it already exists at the source.
- Simplicity
- Hence, latency for route discovery is very low.
- DSDV also guarantees loop-free paths. [7,8]

Disadvantages of DSDV:

- However, DSDV needs to send a lot of control messages. These messages are important for maintaining the network topology at each node.
- This may generate high volume of traffic for high-density and highly mobile networks.
- Special care should be taken to reduce the number of control messages.
- Overhead grows in quadratic to the number of nodes in the network
- The protocol requires selection of the following parameters: periodic update interval, maximum value of the "settling time" for a destination and the number of update intervals, which may transpire before a route, is considered stale.[7,8,9]

Reactive or on-demand routing protocols:

In a reactive routing protocol a route is only being established when a node needs to communicate with a particular node. When a node requires a route to destination, it initiates route discovery process within the network. This process completes once one route is found or all possible route permutations are examined. Once a route is discovered and established, it is maintained by route maintenance procedure until either destination becomes inaccessible along every path from source or route is no longer desired. [5]

Features of Reactive Routing Protocol:

- In a reactive protocol, a route is discovered only when it is necessary.
- In other words, the protocol tries to discover a route only on-demand, when it is necessary.
- These protocols generate much less control traffic at the cost of latency, i.e., it usually takes more time to find a route compared to a proactive protocol.
- No periodic updates
- Use localized route discovery

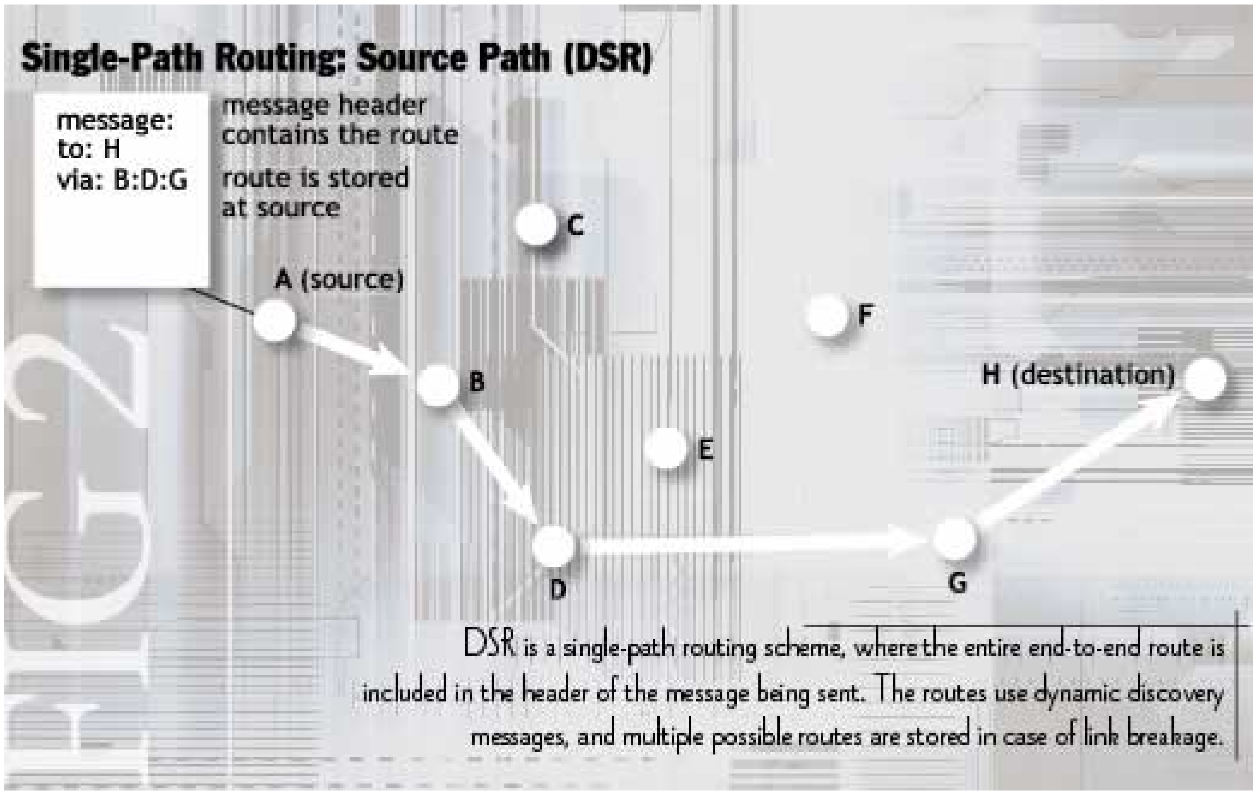
Signaling traffic grows with the increasing mobility of active nodes. [5]

DSR

The *Dynamic Source Routing* protocol (DSR) [Johnson 1994, Johnson 1996a, Broch 1999a] is a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. Using DSR, the network is completely self-organizing and self-configuring, requiring no existing network infrastructure or administration. Network nodes (computers) cooperate to forward packets for each other to allow communication over multiple "hops" between nodes not directly within wireless transmission range of one another. As nodes in the network move about or join or leave the network, and as wireless transmission conditions such as sources of interference change, all routing is automatically determined and maintained by the DSR routing protocol. Since the number or sequence of intermediate hops needed to reach any destination may change at any time, the resulting network topology may be quite rich and rapidly changing.

[12]

The DSR protocol allows nodes to dynamically discover a *source route* across multiple network hops to any destination in the ad hoc network. The protocol is composed of the two main mechanisms of "Route Discovery" and "Route Maintenance", which work together to allow nodes to discover and maintain routes to arbitrary destinations in the ad hoc network. All aspects of the protocol operate entirely on-demand, allowing the routing packet overhead of DSR to scale automatically to only that needed to react to changes in the routes currently in use.
Each data packet sent then carries in its header the complete, ordered list of nodes through which the packet must pass, allowing packet routing to be trivially loop-free and avoiding the need for up-to-date routing information in the intermediate nodes through which the packet is forwarded. By including this source route in the header of each data packet, other nodes forwarding or overhearing any of these packets may also easily cache this routing information for future use.


Single-Path Routing: Source Path (DSR)
message:
to: H
via: B:D:G
message header contains the route
route is stored at source
A (source)
B
C
D
E
F
G
H (destination)
FIG 2
DSR is a single-path routing scheme, where the entire end-to-end route is included in the header of the message being sent. The routes use dynamic discovery messages, and multiple possible routes are stored in case of link breakage.

Example of DSR in operation

Route Discovery

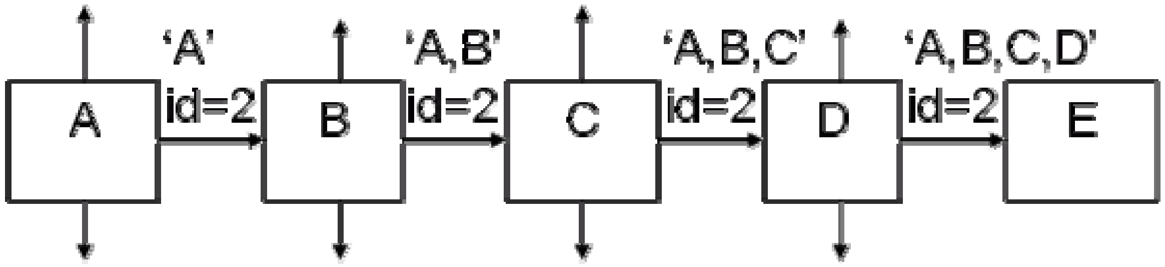


Fig 2.4.b.i) DSR Route Discovery (a)

If node A has in his Route Cache a route to the destination E, this route is immediately used. If not, the Route Discovery protocol is started:

1. Node A (initiator) sends a Route Request packet by flooding the network

2. If node B has recently seen another Route Request from the same target or if the address of node B is already listed in the Route Record, Then node B discards the request.
3. If node B is the target of the Route Discovery, it returns a Route Reply to the initiator. The Route Reply contains a list of the “best” path from the initiator to the target. When the initiator receives this Route Reply, it caches this route in its Route Cache for use in sending subsequent packets to this destination.

Otherwise node B isn’t the target and it forwards the Route Request to his neighbors (except to the initiator).[11]

Advantages of DSR:

- This protocol used a reactive approach which eliminates the need to periodically flood the network with table update messages which are in table-driven approach
- The intermediate nodes also utilize the route cache information efficiently to reduce the control overhead.
- Current and bandwidth saving because there are no hello messages needed (beacon-less).
- Less Power Consumption.

Disadvantages of DSR:

- Intermediate nodes can lead to inconsistent routes if the source sequence number is very old and the intermediate nodes have a higher but not the latest destination sequence number, thereby having stale entries.
- Multiple Route Request packets in response to a single Route Request packet can lead to heavy control overhead.
- The Route Maintenance protocol does not locally repair a broken link. The broken link is only communicated to the initiator.
- The DSR protocol is only efficient in MANETs with less then 200 nodes.
- Problems appear by fast moving of more hosts, so that the nodes can only move around in this case with a moderate speed.

- Flooding the network can cause collusions between the packets.
- There is always a small time delay at the beginning of a new connection because the initiator must first find the route to the target. [11]

AODV

Ad-hoc On-demand Distance Vector is a loop-free routing protocol for ad-hoc networks. AODV is a reactive protocol which means that the routes are created only when they are needed. It is capable of withstanding network diversity behaviors for e.g. node mobility, packet losses and failures of link. This allows them to act independently in an environment of mobile nodes. Traditional routing tables are used which uses one entry per destination and sequence numbers in order to verify whether routing information is up to date as well as to prevent routing loops.

Operation in AODV

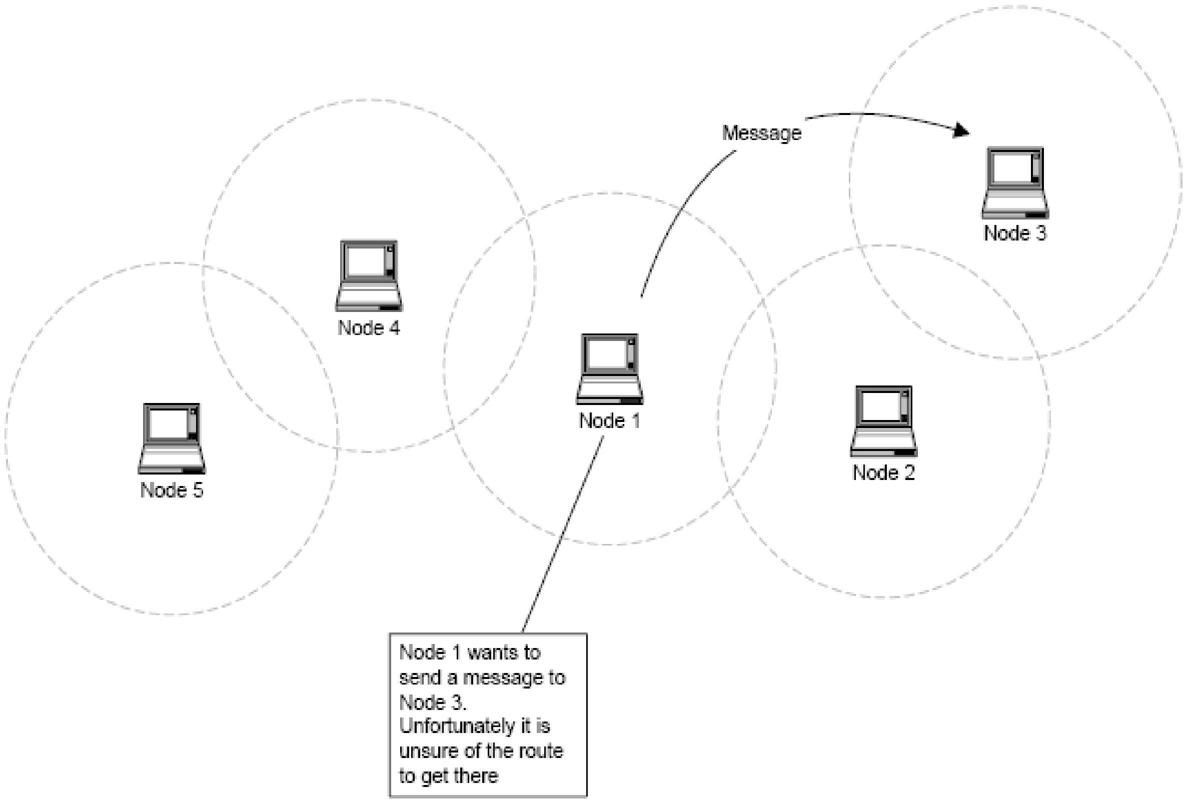
The routing table entry for a destination contains three important fields:

- A next hop node - All packets intended for the destination are sent to the next hop node. The sequence number acts as a measure of the freshness of a route.

- A sequence number - The sequence number acts as a measure of the freshness of a route.

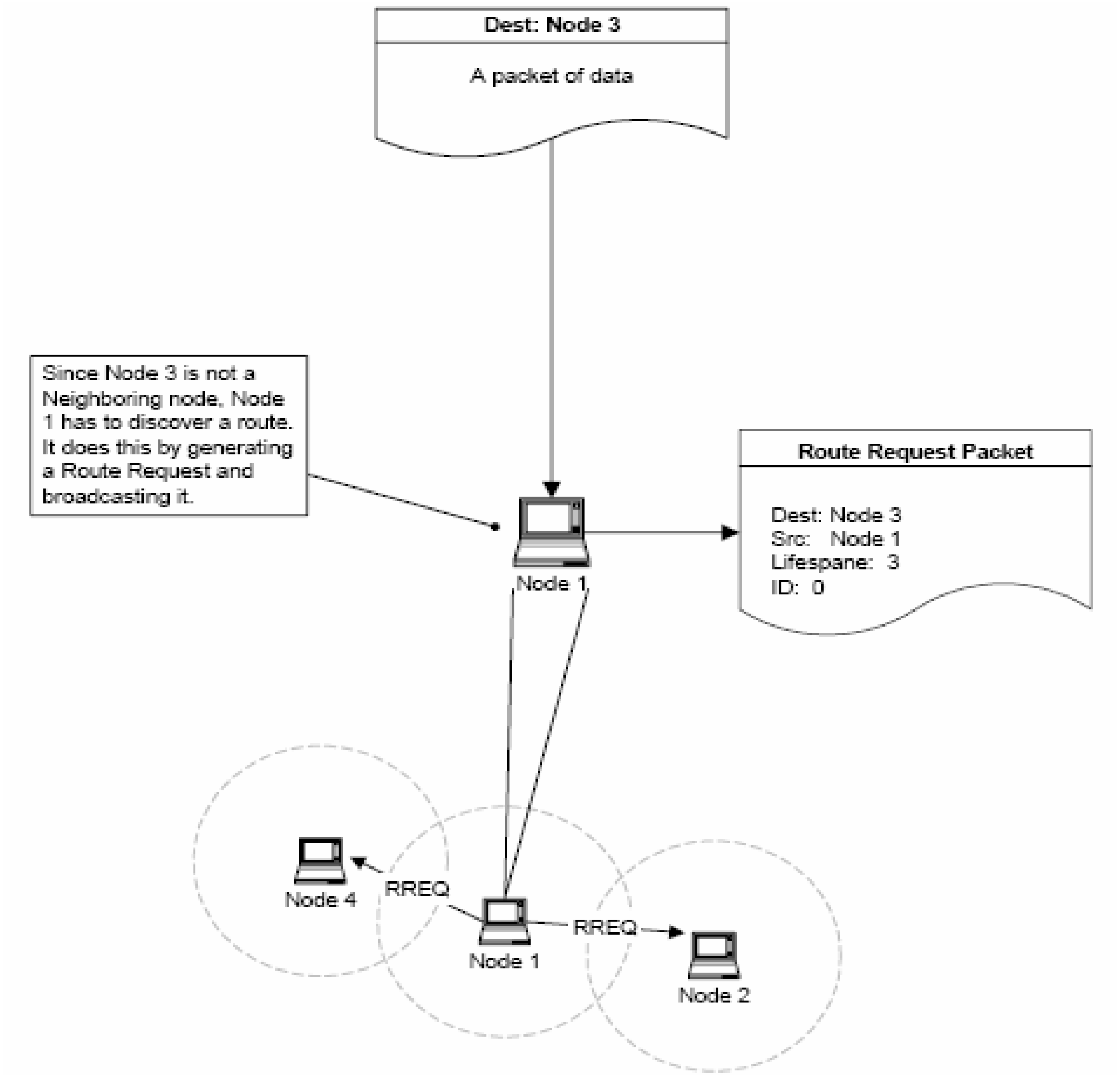
- A hop count - The hop count represents the current distance to the destination node.

AODV maintains time-based states in each available node meaning a routing entry not recently used is expired and neighboring nodes are notified if a route is broken.


Message
Node 3
Node 4
Node 1
Node 2
Node 5
Node 1 wants to send a message to Node 3. Unfortunately it is unsure of the route to get there

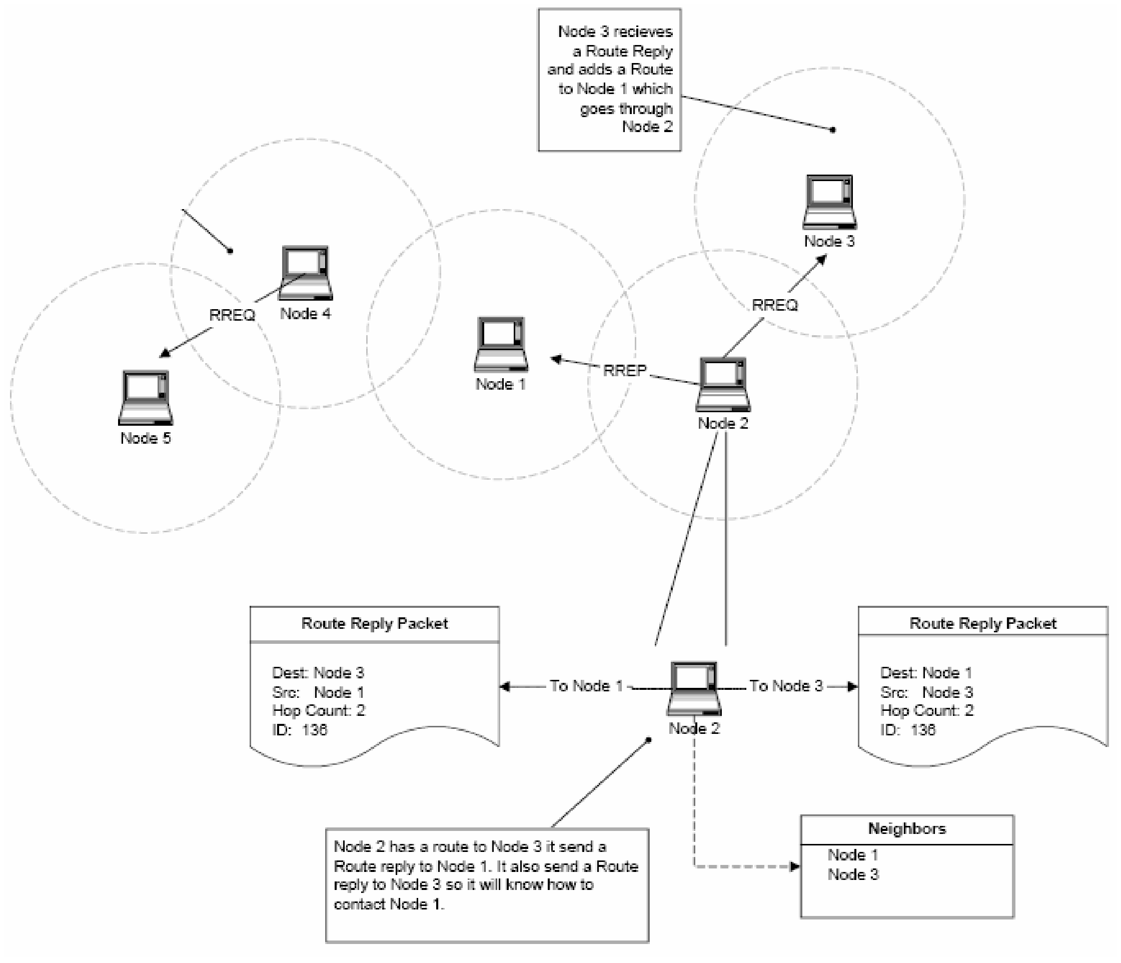
AODV mechanism (i)

Nodes in AODV discover routes in request-response cycles. Request for a route is sent by a node to a destination by an RREQ message broadcast to all its neighbors. When an RREQ message is received by a node but does not possess the route to the requested destination, the node in turn broadcasts the RREQ message. A memory of the reverse route to the requesting node is saved which is used to forward future responses to this RREQ. The process is repeated until the RREQ reaches a node containing a route to the desired destination. This node responds with an RREP message which is unicast along the reverse-routes of the neighboring nodes until it the node which originated the request is reached. Thus, at the end of this request-response cycle a *bidirectional* route is established between the requesting node and the destination.


Dest: Node 3
A packet of data
Since Node 3 is not a Neighboring node, Node 1 has to discover a route. It does this by generating a Route Request and broadcasting it.
Route Request Packet
Dest: Node 3
Src: Node 1
Lifespane: 3
ID: 0
Node 1
Node 4
RREQ
Node 1
RREQ
Node 2

AODV mechanism (ii)

When a node faces a problem like connectivity failure, the node invalidates its route by sending an *RERR* to all nodes that received its *RREP*.


Node 3 recieves a Route Reply and adds a Route to Node 1 which goes through Node 2
Node 3
RREQ
Node 4
RREQ
Node 5
Node 1
RREP
Node 2
Route Reply Packet
Dest: Node 3
Src: Node 1
Hop Count: 2
ID: 136
To Node 1
Node 2
To Node 3
Route Reply Packet
Dest: Node 1
Src: Node 3
Hop Count: 2
ID: 136
Node 2 has a route to Node 3 it send a Route reply to Node 1. It also send a Route reply to Node 3 so it will know how to contact Node 1.
Neighbors
Node 1
Node 3

AODV mechanism (iii)

After receiving the three AODV messages: RREQ, RREP and RERR, the nodes update the next hop, sequence number and the hop counts of their routes.

Expiration time, also known as lifetime, is reset each time the route has been used. Active route timeout is the time after which the route is considered as invalid, and so the nodes delete their reverse entries. If active route timeout is big enough route repairs will maintain routes. RFC 3561 defines it to 3 seconds. [10]

Advantages of AODV:

- Nodes store only the routes that are needed

- Need for broadcast is minimized
- Reduces memory requirements and needless duplications
- Quick response to link breakage in active routes
- Loop-free routes maintained by use of destination sequence numbers
- Scalable to large populations of nodes
- Lesser connection delay. [1]

Disadvantages of AODV:

- One disadvantage is that intermediate nodes can lead to inconsistent routes if the source sequence number is very old and the intermediate nodes have a higher but not the latest destination sequence number, thereby having stale entries.
- Multiple Route Request packets in response to a single Route Request packet can lead to heavy control overhead. [14]

METHODOLOGY

Simulator

For our thesis project, we choose NS-2 as our simulation environment since it is the most widely used network simulator. NS-2 has several advantages which suits our simulation project:

1. It provides extensive support for simulating TCP/IP, Routing and multicast protocols over wired and wireless network.
2. It provides a lot of standard modules to be used in sensor network.
3. It is an object oriented design which provides a lot of documents.
4. It provides a random function which is critical for our project

Simulation Environment

Creating random traffic-pattern for wireless scenarios:

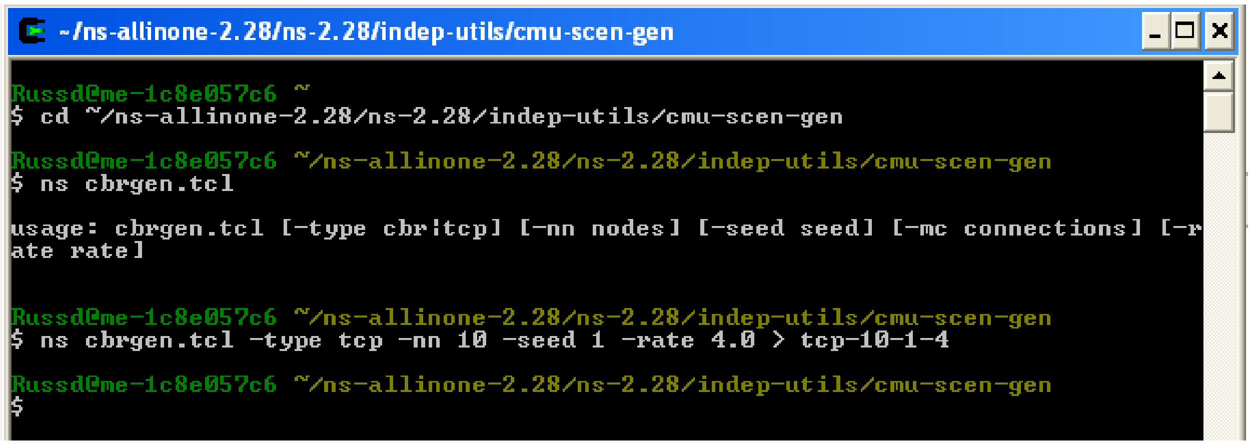
Random traffic connections of TCP and CBR can be setup between mobile nodes using a traffic-scenario generator script. This traffic generator script is available under ~ns/indep-utils/cmu-scen-gen and is called cbrgen.tcl. It can be used to create CBR and TCP traffics connections between wireless mobile nodes. In order to create a traffic-connection file, we need to define the type of traffic connection (CBR or TCP), the number of nodes and maximum number of connections to be setup between them, a random seed and incase of CBR connections, a rate whose inverse value is used to compute the interval time between the CBR packets. So the command line looks like the following:

```
ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-mc connections]
[-rate rate]
```

The start times for the TCP/CBR connections are randomly generated with a maximum value set at 180.0s.

For example, let us try to create a TCP connection file between 10 nodes, having maximum of 8 connections, with a seed value of 1.0 and a rate of 4.0. So at the prompt type:

```
ns cbrgen.tcl -type tcp -nn 10 -seed 1.0 -mc 1 -rate 4.0 > tcp-10-1-4
```



```
Russd@me-1c8e057c6 ~
$ cd ~/ns-allinone-2.28/ns-2.28/indep-utils/cmu-scen-gen

Russd@me-1c8e057c6 ~/ns-allinone-2.28/ns-2.28/indep-utils/cmu-scen-gen
$ ns cbrgen.tcl

usage: cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-mc connections] [-r
ate rate]


Russd@me-1c8e057c6 ~/ns-allinone-2.28/ns-2.28/indep-utils/cmu-scen-gen
$ ns cbrgen.tcl -type tcp -nn 10 -seed 1 -rate 4.0 > tcp-10-1-4

Russd@me-1c8e057c6 ~/ns-allinone-2.28/ns-2.28/indep-utils/cmu-scen-gen
$
```

From tcp-10-1-4 file (into which the output of the generator is redirected) thus created, one of the tcp connections looks like the following:

```
#
# 1 connecting to 2 at time 2.5568388786897245
#
set tcp_(0) [$ns_ create-connection  TCP $node_(1) TCPSink $node_(2) 0]

$tcp_(0) set window_ 32

$tcp_(0) set packetSize_ 1024

set ftp_(0) [$tcp_(0) attach-source FTP]

$ns_ at 2.5568388786897245 "$ftp_(0) start"
#
```

Thus a TCP connection is setup between node 1 and 2.

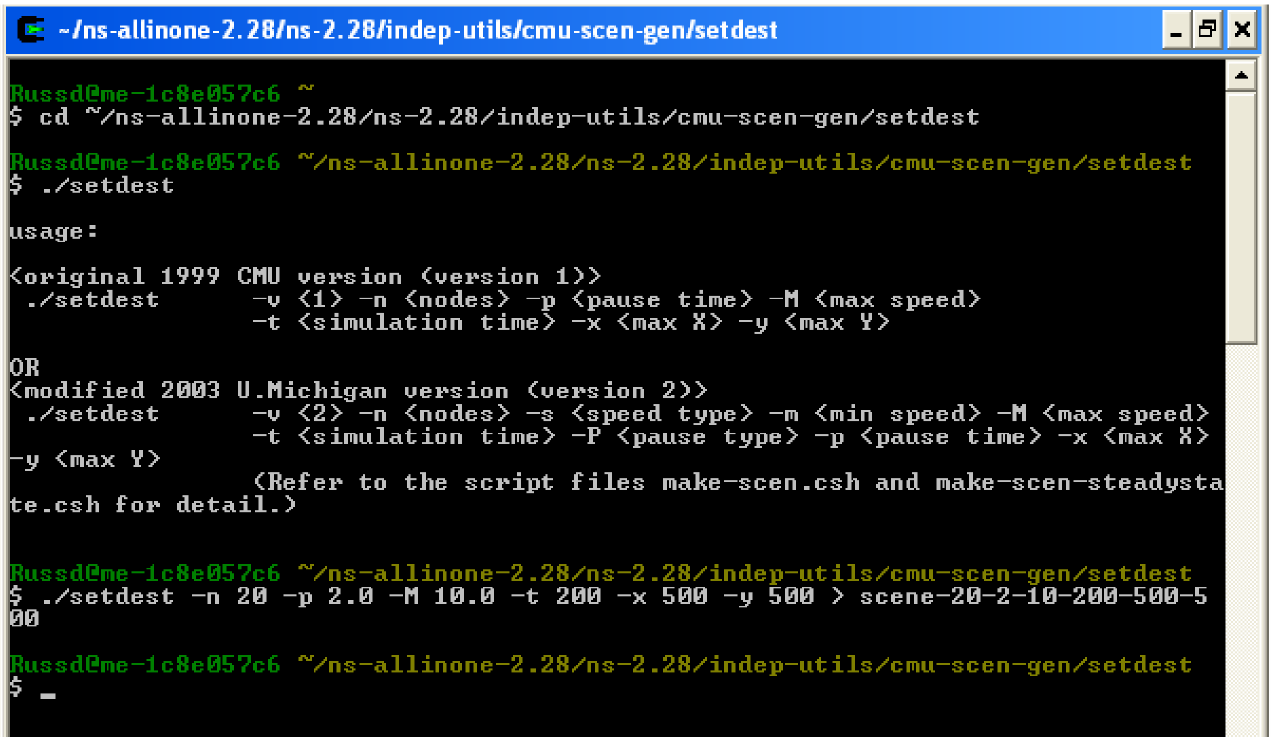
Creating node-movements for wireless scenarios:

The node-movement generator is available under ~ns/indep-utils/cmu-scen-gen/setdest directory and consists of setdest{.cc,.h} and Makefile. Setdest with arguments as shown below:

```
./setdest [-n num_of_nodes] [-p pausetime] [-M maxspeed] [-t simtime]

[-x maxx] [-y maxy] > [outdir/movement-file]
```

Let's say we want to create a node-movement scenario consisting of 20 nodes moving with maximum speed of 10.0m/s with an average pause between movements being 2s. We want the simulation to stop after 200s and the topology boundary is defined as 500 X 500. So our command line will look like:

```
./setdest -n 20 -p 2.0 -M 10.0 -t 200 -x 500 -y 500 > scene-20-2-10-200-500-500
```



```
Russd@me-1c8e057c6 ~
$ cd ~/ns-allinone-2.28/ns-2.28/indep-utils/cmu-scen-gen/setdest

Russd@me-1c8e057c6 ~/ns-allinone-2.28/ns-2.28/indep-utils/cmu-scen-gen/setdest
$ ./setdest

usage:

<original 1999 CMU version (version 1)>
 ./setdest      -v <1> -n <nodes> -p <pause time> -M <max speed>
                -t <simulation time> -x <max X> -y <max Y>

OR
<modified 2003 U.Michigan version (version 2)>
 ./setdest      -v <2> -n <nodes> -s <speed type> -m <min speed> -M <max speed>
                -t <simulation time> -P <pause type> -p <pause time> -x <max X>
-y <max Y>
                (Refer to the script files make-scen.csh and make-scen-steadysta
te.csh for detail.)


Russd@me-1c8e057c6 ~/ns-allinone-2.28/ns-2.28/indep-utils/cmu-scen-gen/setdest
$ ./setdest -n 20 -p 2.0 -M 10.0 -t 200 -x 500 -y 500 > scene-20-2-10-200-500-5
00

Russd@me-1c8e057c6 ~/ns-allinone-2.28/ns-2.28/indep-utils/cmu-scen-gen/setdest
$ _
```

The output is written to stdout by default. We redirect the output to file scene-20-2-10-200-500-500. The file begins with the initial position of the nodes and goes on to define the node movements.

```
$ns_ at 2.000000000000 "$node_(10) setdest 449.033027142023 166.293197448595
0.354983963500"
```

This line from scene-20-2-10-200-500-500 defines that node_(10) at time 2.0s starts to move toward destination (449.03, 166.29) at a speed of 0.35m/s. These command lines can be used to change direction and speed of movement of mobilenodes.

The Simulated Scenarios:

Adhoc1:

a) Traffic Pattern:

In this scenario, traffic models were generated for 10 nodes with TCP traffic sources, with maximum connections of 1 at a rate of 8kbps.
(-rate 2.0: in one second, 2 packets are generated. The packet size is 1024 byte. Therefore the rate is 2*1024*8=16kbps)

b) Scenario Pattern:
Mobility models are created for the simulations using 10 nodes, with pause time of 10 seconds, maximum speed of 20m/s, topology boundary of 500x400 and simulation time of 150 secs.

Adhoc2:

a) Traffic Pattern:

In this scenario, traffic models were generated for 10 nodes with TCP traffic sources, with maximum connections of 1 at a rate of 16kbps.
(-rate 4.0: in one second, 4 packets are generated. The packet size is 1024 byte. Therefore the rate is 4*1024*8=32kbps)

b) Scenario Pattern:
Mobility models are created for the simulations using 10 nodes, with pause time of 8 seconds, maximum speed of 30m/s, topology boundary of 500x400 and simulation time of 150 secs.

Adhoc3:

a) Traffic Pattern:

In this scenario, traffic models were generated for 10 nodes with TCP traffic sources, with maximum connections of 4 at a rate of 24kbps.
(-rate 6.0: in one second, 6 packets are generated. The packet size is 1024 byte. Therefore the rate is 6*1024*8=49kbps)

b) Scenario Pattern:

In this simulation, mobility models are created for the simulations using 10 nodes, with pause time of 6 seconds, maximum speed of 40m/s, topology boundary of 500x400 and simulation time of 150 secs.

Simulation scripts to compare the performance of DSDV, AODV, and DSR:

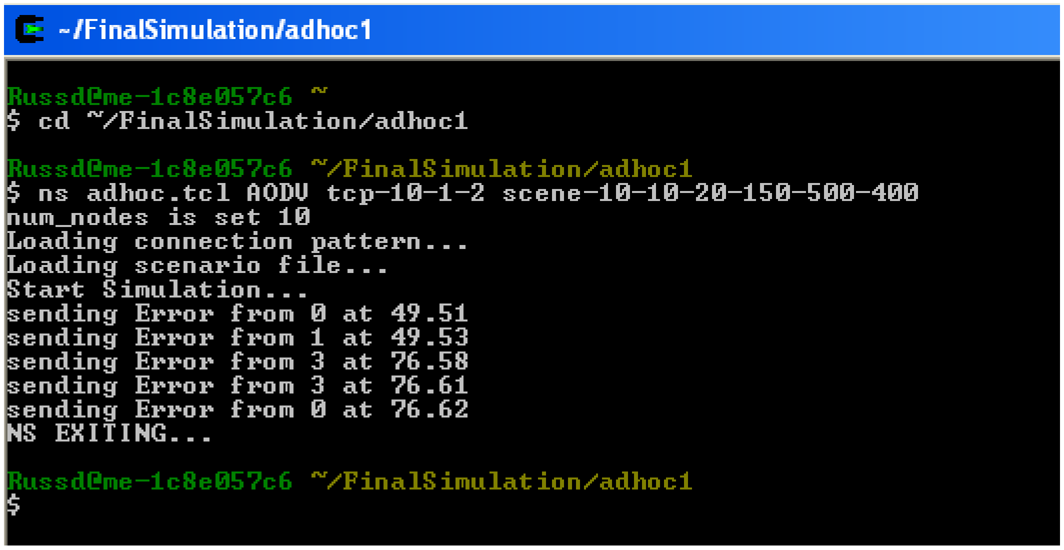
```
if {$argc !=3} {
        puts "Usage: ns adhoc.tcl  Routing_Protocol Traffic_Pattern Scene_Pattern "
        puts "Example:ns adhoc.tcl DSDV cbr-50-10-8 scene-50-0-20"
        exit
}

set par1 [lindex $argv 0]
set par2 [lindex $argv 1]
set par3 [lindex $argv 2]

set val(chan)           Channel/WirelessChannel    ;
# channel type
set val(prop)           Propagation/TwoRayGround   ;
# radio-propagationmodel
set val(netif)          Phy/WirelessPhy            ;
# network interface type
set val(mac)            Mac/802_11                 ;
# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;
# interface queue type
set val(ll)             LL                         ;
# link layer type
set val(ant)            Antenna/OmniAntenna        ;
# antenna model
set val(ifqlen)         10                         ;
```

```
# max packet in ifq
set val(rp)             $par1                    ;
# routing protocol
set val(x)              500
set val(y)              400
set val(seed)           0.0
set val(tr)             temp.tr
set val(nn)             10
set val(cp)             $par2
set val(sc)             $par3
set val(stop)           150.0

set ns_             [new Simulator]

set tracefd     [open $val(tr) w]
$ns_ trace-all $tracefd
#$ns_ use-newtrace



set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

set chan_1_ [new $val(chan)]

        $ns_ node-config -adhocRouting $val(rp) \
                         -llType $val(ll) \
                         -macType $val(mac) \
                         -ifqType $val(ifq) \
                         -ifqLen $val(ifqlen) \
                         -antType $val(ant) \
                         -propType $val(prop) \
                         -phyType $val(netif) \
                         -channel $chan_1_ \
                         -topoInstance $topo \
                         -agentTrace ON \
                         -routerTrace ON \
                         -macTrace OFF

        for {set i 0} {$i < $val(nn) } {incr i} {
                set node_($i) [$ns_ node]
                $node_($i) random-motion 0            ;
                             # disable random motion
        }
  puts "Loading connection pattern..."
source $val(cp)

puts "Loading scenario file..."
source $val(sc)

for {set i 0} {$i < $val(nn) } {incr i} {
   $ns_ initial_node_pos $node_($i) 30
}

for {set i 0} {$i < $val(nn) } {incr i} {
   $ns_ at $val(stop).000000001 "$node_($i) reset";
}
$ns_ at $val(stop).000000001 "puts \"NS EXITING...\"; $ns_ halt"
puts "Start Simulation..."
$ns_ run
```

Running Simulation:

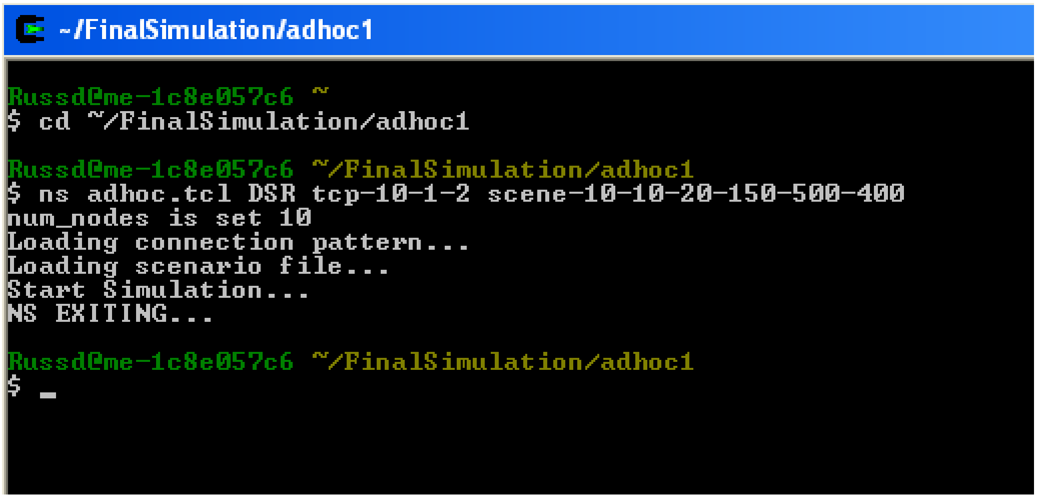
After creating the traffic and scenario pattern files, they are saved in the same folder as the adhoc.tcl script. The commands for the first scenario are shown below:

1. AODV



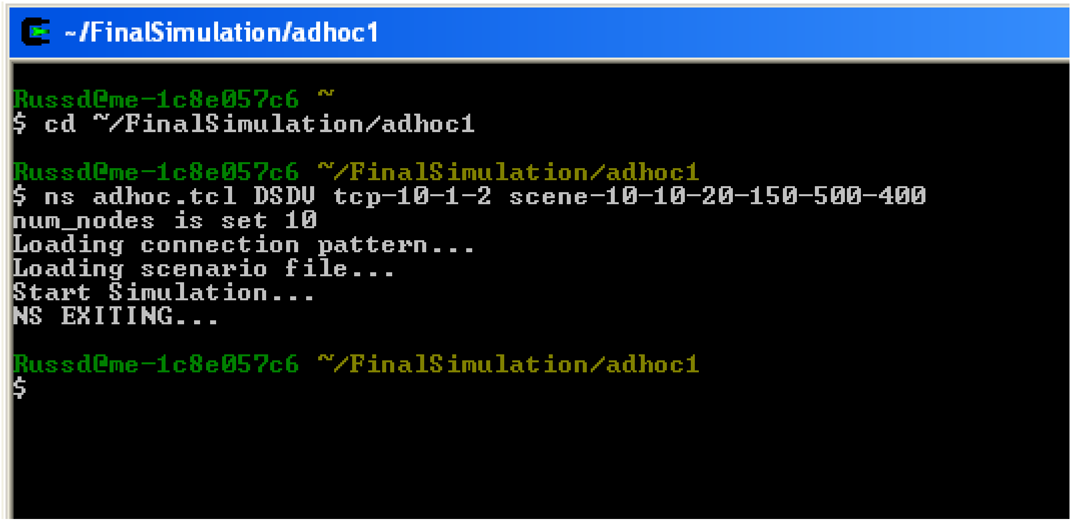
```
Russd@me-1c8e057c6 ~
$ cd ~/FinalSimulation/adhoc1

Russd@me-1c8e057c6 ~/FinalSimulation/adhoc1
$ ns adhoc.tcl AODV tcp-10-1-2 scene-10-10-20-150-500-400
num_nodes is set 10
Loading connection pattern...
Loading scenario file...
Start Simulation...
sending Error from 0 at 49.51
sending Error from 1 at 49.53
sending Error from 3 at 76.58
sending Error from 3 at 76.61
sending Error from 0 at 76.62
NS EXITING...

Russd@me-1c8e057c6 ~/FinalSimulation/adhoc1
$
```

2. DSR



```
Russd@me-1c8e057c6 ~
$ cd ~/FinalSimulation/adhoc1

Russd@me-1c8e057c6 ~/FinalSimulation/adhoc1
$ ns adhoc.tcl DSR tcp-10-1-2 scene-10-10-20-150-500-400
num_nodes is set 10
Loading connection pattern...
Loading scenario file...
Start Simulation...
NS EXITING...

Russd@me-1c8e057c6 ~/FinalSimulation/adhoc1
$ _
```

3. DSDV



```
Russd@me-1c8e057c6 ~
$ cd ~/FinalSimulation/adhoc1

Russd@me-1c8e057c6 ~/FinalSimulation/adhoc1
$ ns adhoc.tcl DSDV tcp-10-1-2 scene-10-10-20-150-500-400
num_nodes is set 10
Loading connection pattern...
Loading scenario file...
Start Simulation...
NS EXITING...

Russd@me-1c8e057c6 ~/FinalSimulation/adhoc1
$
```

SIMULATION PERFORMANCE METRICS

1. Throughput: The Total bytes received by the destination node per sec. (Data packets and Overhead).

2. Goodput:

- **Goodput (In terms of Number of packets):**

The **ratio** of the total number of data packets that are sent from the source **to** the total number of packets that are transmitted within the network to reach the destination.

- **Goodput (In terms of Packet Size in Bytes):**

The **ratio** of the total bytes of data that are sent from the source **to** the total bytes that are transmitted within the network to reach the destination.

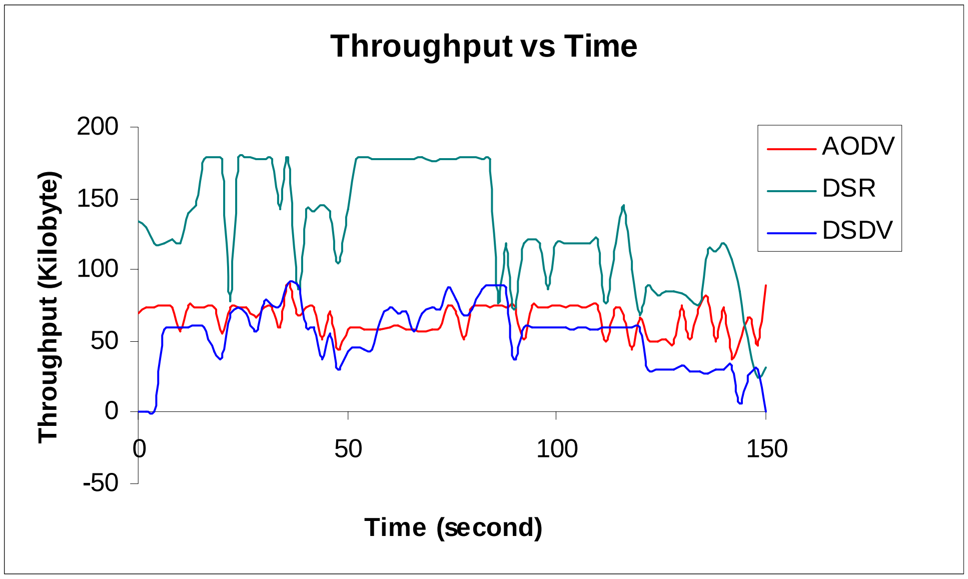
3. Routing Load:

- **Routing Load (In terms of Number of packets):**

The **ratio** of the total number of routing packets that are send within the network **to** the total number packets that are transmitted within the network to reach the destination.

- **Routing Load (In terms of Packet Size in Bytes):**

The **ratio** of the total bytes of routing packets that are send within the network **to** the total number bytes that are transmitted within the network to reach the destination.

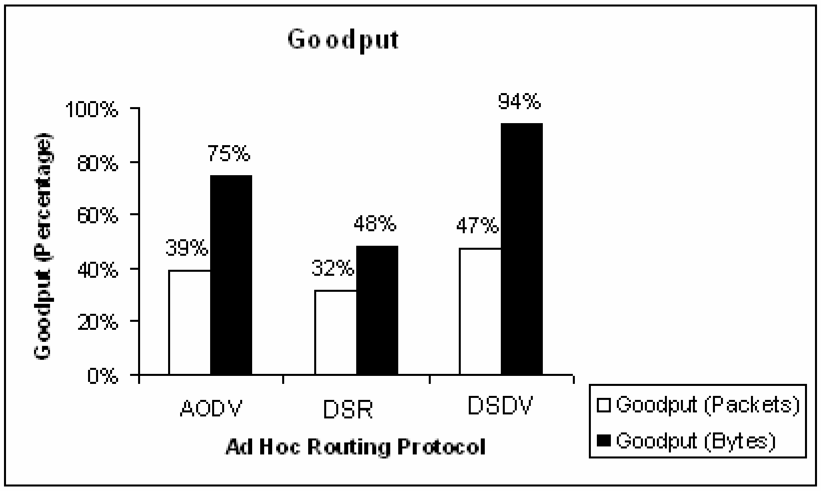


AODV: Data transfer starts quickly and the data rate is more stable.

DSR: Starts out quickly, however, date rate fluctuates over time.

DSDV: Takes time to start off but the data rate has more stable.

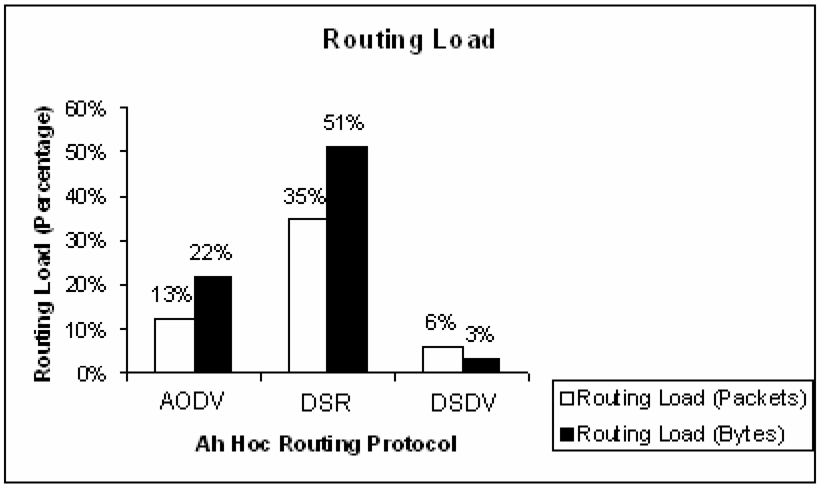
The total number of TCP packets transferred using DSR is much larger than in DSDV. For example, in DSR, 11274 TCP (data) packets have been received during the simulation of scenario Adhoc-1, whereas in DSDV with the same parameters it is 10906. For AODV, the connection transferred altogether 10387 data packets which is less than that in DSR.



From the graph we see that per 100 packets transmitted in the network, 39 packets would be data packets for AODV, 32 for DSR and 47 for DSDV.

And in term of bytes, in an average per 100 bytes of packets transmitted in the network, 75 bytes would be data packets for AODV, 48 bytes for DSR and 94 bytes for DSDV.

So it seems that though DSDV takes more time to converge, it is actually sending more data packets in number as well as bytes than AODV and DSR.



From the above two graphs we observe that though DSR has a better throughput, it actually contains more overhead as routing packets. However DSDV has a relatively lower routing load than AODV and DSR.

CONCLUSION

To sum up our research, we can say that using the simulation software we were able to achieve the capability for comparison between the three protocols. We created three different scenarios for all the protocols under random motion.

We found that DSDV has the best goodput and lesser routing load. But DSDV also takes the most amount of time to converge. This leads to being a disadvantage in a continuously changing topology.

AODV was found to have a moderate goodput and routing load but had the ability to adapt to the changing network easily. Thus it has a stable throughput. This leads to AODV becoming useful in conditions where a fast topology change exists.

DSR was found to be the least favorable as even after having the greatest throughput it had high routing load. It shows a lot of fluctuations in the throughput graph which is not preferred in a wireless network.

FUTURE WORK

In our thesis we were able to simulate for 3 protocols only, in the future other protocols can be taken into account. Protocols such as TORA and OLSR can be analyzed and compared with the rest of the protocols giving us a clear view of the merits and demerits.

Till now we have only been able to analyze the separate protocols of ad hoc network theoretically. In the future a practical implementation would likely be in order for which the theoretical analysis was necessary. Now we want to focus on the implementation aspect of Ad Hoc networks.

REFERENCES

1. Ad-hoc On-Demand Distance Vector routing (AODV), C. E. Perkins and E. M. Royer in Proc. 2nd IEEE Wksp. Mobile Computer Systems and Applications, pp. 90-- 100, 1999.
2. Wireless Ad Hoc Sensor Networks, Advanced Network Technology Division, NIST.
3. Proactive Routing, University of Luxembourg, SECAN-Lab
4. List of ad-hoc routing protocols, Wikipedia
5. The Destination Sequenced Distance Vector (DSDV) protocol, Dr. R. B. Patel
6. Highly Dynamic Destination-Sequenced Distance-Vector Routing, University of Luxembourg, SECAN-Lab
7. PERFORMANCE ANALYSIS OF ADhOC NETWORK ROUTING PROTOCOLS, P. Chenna Reddy and Dr. P. Chandrasekhar Reddy, Computer Science and Engineering, JNTU College of Engg, Anantapur.
8. Mobile ad-hoc networks, Pro-active routing protocols (DSDV, OLSR), Janico Greifenberg
9. The Destination Sequenced Distance Vector (DSDV) protocol, Rabindra
10. A Quick Guide to AODV Routing, Luke Klein-Berndt, Wireless Communications Technologies Group, National Institute of Standards and Technology
11. Dynamic Source Routing, University of Luxembourg, SECAN-Lab