Ad Hoc Network Routing Comparison

David Chiasson
dchiasso@purdue.edu

Abstract

A Mobile Ad Hoc Network (MANET) is a wireless network designed to operate without a preexisting infrastructure. Such a network is self supported, with each node acting as a source and drain for information, as well as a router for communication between other nodes. Due to the complexity and instability of an ad hoc network, routing information between nodes becomes a nontrivial problem. The limited battery, processing power, and bandwidth available to wireless nodes make finding an optimal algorithm even more essential to the practicality of such a network. Many different routing protocols have been proposed and implemented in ad hoc networks. Each of these protocols has its own advantages and disadvantages, making the optimal method unapparent. This paper will discuss different methods used by various routing protocols, and the situations in which those protocols excel. The information presented will allow the network designer to more effectively choose the optimal routing protocol based on different network variables.

Keywords: MANET, Ad Hoc, Routing Protocol, Optimal Routing, Protocol Comparison

A Mobile Ad Hoc Network (MANET) is a wireless network designed to operate without a preexisting infrastructure. Such a network is self supported, with each node acting as a source and sink for information, as well as a router for communication between other nodes. There are many different protocols proposed for routing information between nodes. Most of these protocols can be placed into general categories based on the techniques employed. Perhaps the most general category is proactive versus reactive routing philosophies[8].

Proactive Routing Protocols

Proactive routing protocols, or table driven protocols, in their most basic form, store an exhaustive table containing all network connections between every node. This creates very low latency in a network because every node is able to calculate the most efficient route to its destination on demand. However, keeping that table updated poses a difficult problem. Every time a connection between nodes is broken or created, the whole network must be alerted of the change. If connections are changed at a fast enough rate, the network is unable to keep up. This severely limits the size and rate of mobility practical with a proactive routing network. Popular table driven protocols include Destination Sequenced Distance Vector Routing (DSDV)[7] and Optimized Link State Routing Protocol (OLSR)[4]. These table driven protocols can be further categorized based on types of routing tables[10].

Distance Vector Versus Link State Tables

The distance vector tables save routing directions to every possible destination in the network. These tables include both the next hop in the direction of the destination node as well as the distance to that node in the form of hops, time, or link quality[6]. Link state proactive routing protocols also keep exhaustive routing tables saved on each individual node but do not save the routing directions. Instead, these tables save the

connections of each other node in the network, or the topology of the network[1]. This allows each node to calculate for itself the optimal path to the destination node. Link state routing protocols are more reliable in the event of topology change as it is easy for a node to find an alternate path to the destination, but the required algorithm is naturally more complex[10].

Reactive Routing Protocols

Reactive Routing Protocols, also called on-demand or source-initiated routing protocols, do not store connection information on the nodes. Instead, if a node wants to initiate communication with another node, it sends out a mass route request (RREQ) message to the entire network. Once the target node receives this message, it sends a route reply (RREP) message along the reverse path of the RREQ message. This category of routing methods has a higher latency than the proactive methods, but does not require exhaustive routing tables to be stored on every node, nor does it require constant routing updates to flood the network. Ad hoc On-Demand Distance Vector Routing (AODV)[2] is a popular reactive routing protocol.

Packet Addressing Method

Routing protocols can also be categorized based on how packets are addressed. These can be either source routing or hop-by-hop routing. Source routing includes the complete directions to the destination node. Each node along the route simply reads from the packet header where to forward the packet. Hop-by-hop routing on the other hand only includes the destination in the packet header. The source node simple forwards it to the node that is in the direction of the destination node, and every node along the path is responsible for determg the next hop. Hop-by-hop routing is much more robust in the event of topology change, as the route can be altered by any node along the route, but can sometimes lead to loops in network[10].

Flat Versus Hierarchial

Ad hoc networks can also vary in regards to topology of node relationships. The classic MANET has all participant nodes being identical and equal. This is considered a flat routing approach. Other approaches include hierarchical routing and geographic position assisted routing. A hierarchical routing protocol such as the zone routing protocol (ZRP)[9] organizes nodes into smaller groups or zones. These zones will communicate with each other pro actively, but communicate within themselves reactively. Another method used in hierarchical routing is the specialization of nodes. Nodes with greater processing power might be chosen to form a sort of backbone for the network, carrying the majority of the routing load, while less able nodes function predominately as a source or sink. Geographic position assisted routing, such as the location aided routing (LAR)[5] protocol, uses limited geographic information about the nodes to limit the limit the flooding of a network. These two categories of routing protocols have been referred to as hybrid routing protocols since they use a combination of proactive and reactive methods. Such protocols have been shown to greatly enhance the scalability of a network[10].

Protocols Tested

In this study, three different routing protocols of DSDV, AODV, and DSR are explored. These are three of the more standard routing protocols but there are many more in circulation. The exploration of other protocols will be left to further research.

DSDV[7]: The destination sequenced distance vector routing protocol is a proactive distance vector table routing protocol. Routing tables are stored on each node and are frequently broadcasted to neighbors to resolve differences. One of the oldest, this protocol was published in 1994 by Charles E. Perkens at IBM, yet this protocol still receives much attention today as one of the premier proactive algorithms.

AODV[2]: The Ad hoc On-demand Distance Vector routing protocol is a reactive routing protocol that waits until a route is required to discover it. AODV also saves routing tables which are used to return RREP messages and route the data packets. This protocol was developed in 1999 by Charles E. Perkens at Sun Microsystems Laboratories.

DSR[3]: The Dynamic Source Routing protocol is a reactive routing protocol which relies on the two main functions of route discovery and route maintenance to maintain communication between nodes. This protocol is distinctive in it's use of source routing which can be updated on demand through the route maintenance procedure. This procedure was published in 1998 by David B. Johnson at Carnegie Mellon University.

Simulation Set Up

The network model used in this study consists of a set number of wireless nodes contained within a square area, moving towards random destinations at random speeds. Once that destination is reached, a new random destination is set. Fifty random network scenarios are created for every data point in an attempt to average out the negative effects of random generation. The same network scenarios are tested with all three protocols to ensure a fair comparison.

The purpose of this study is not to test each routing protocol in every single possible scenario. Because of the many network variables and imaginable permutations, such an exhaustive study would be impractical. Instead, this study's purpose is to characterize the strong areas of each routing protocol primarily by testing extreme conditions. If a particular protocol preforms significantly better than the other protocols in a certain condition, then more simulations are run to explore that advantage.

As can be seen in appendix one, the testing software takes as variables the values of number of trials, number of nodes, physical size of network boundaries, maximum movement rate, number of connections, and the frequency of communication rate. Adjusting these variables allows one simulate a large number of varying network scenarios. However, they by no means represent the diversity of scenarios found in practice. Further study can be done adding elements of unidirectional links, heterogeneous traffic patterns or movement patterns, sleeping nodes, power restraints and many more.

In this study, the metric of packet delivery ratio is used. This is the number of received packets divided by the number of sent packets. This is the primary and most basic requirement for a network and answers the question, did communication successfully take place? However, it does not address the the quality or efficiency of that communication among other things. In order to gain a more complete view of protocol performance, a designer could consider looking at other network metrics of routing overhead, average delay time, path optimality, security, energy efficiency, route acquisition time, and many others.

Experiment Results

DSDV:
After running approximately 8000 network simulations, not a single scenario has shown DSDV to have a significant throughput advantage over either AODV or DSR. When looking solely at the criteria of packet delivery ratio, DSDV appears to be inferior.

AODV:
The AODV routing protocol was found to perform exceptionally well at high mobility rates. Figure 1 shows results from a scenario of fifty nodes in a rather large square area of 900 by 900 meters. The area is large enough that even with zero movement, the network is still struggling to get all the packets passed. As node mobility is increased, DSDV throughput drops dramatically. DSR also handles reasonably well, but as mobility is increased, AODV has a very significant advantage over the other two routing protocols. AODV also outperformed DSR in larger networks as seen in Figure 2 where the network size is increased with a constant average node density of one node per 8000 meters squared.

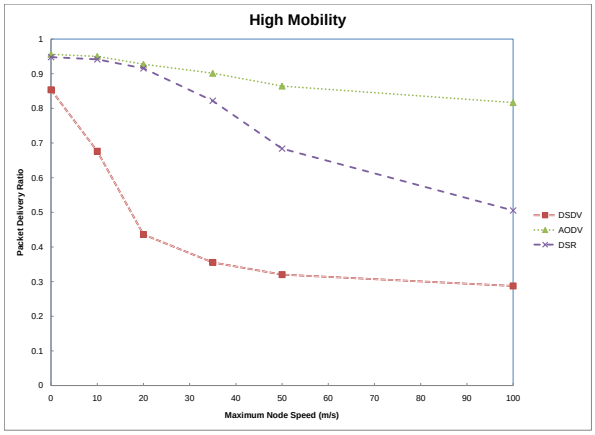

Figure 1. Protocol Performance During Increased Mobility Rate

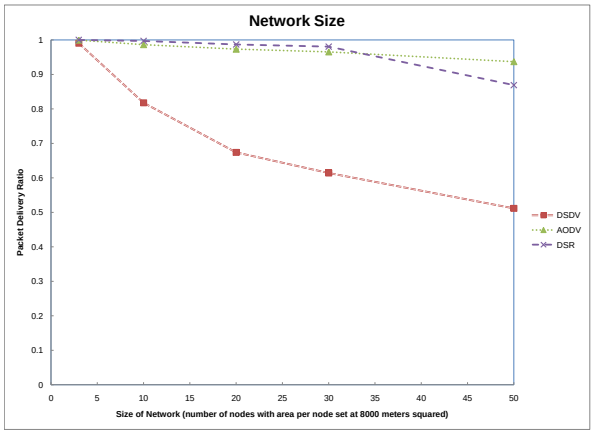

Figure 2. Protocol Performance During Increased Mobility Rate

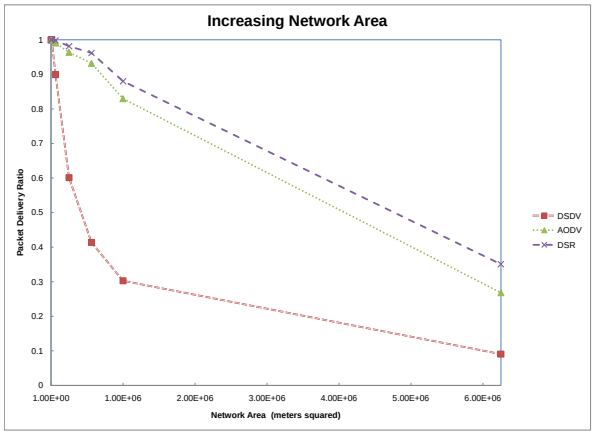

Figure 3. Protocol performance during increased network area with nodes set at 25

DSR:

The DSR routing protocol tended to outperform both DSDV and AODV under moderate testing conditions. Figure 2 shows DSR with slightly higher throughput until the network reaches a certain size. Figure 3, which increases network area with a set 25 nodes, shows that DSR also fares better in scarcely populated networks.

References

[1] E.Baccelli C. Adijh and P. Jacquet. Link state routing in wireless adhoc networks. In *MILCOM '03: Military Communications Conference. IEEE Computer Society*, pages 1274–1279, 2003.

[2] E. Royer C. Perkins. Ad hoc on demand distance vector routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, February 1999.

[3] David A. Maltx David B. Johnson and Josh Broch. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In *MobiCom '98*, 1998.

[4] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, pages 62 – 68, 2001.

[5] Y. B. Ko and N. H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. In *Wirel, Netw,*, volume 6, pages 307–321, 200.

[6] Yi Lu, Weichao Wang, Yuhui Zhong, and Bharat K. Bhargava. Study of distance vector routing protocols for mobile ad hoc networks. In *PerCom'03*, pages 187–194, 2003.

[7] C. Perkens. Highly dynamic destination sequenced distance vector routing (dsdv) for mobile computers. In *ACM SIGCOMM94*, 1994.

[8] Ingo Gruber Rdiger Schollmeier and Michael Finkenzeller. Routing in mobile ad-hoc and peer-to-peer networks a comparison. In *Networking 2002 Workshops, LNCS 2376*, pages 172–186, 2002.

[9] M. Pearlman Z. Haas. The performance of query control schemes for the zone routing protocol. In *IEEE/ACM Transactions on Networking*, volume 9, August 2001.

[10] Stamatis Vassiliadis Zhijiang Chang, Georgi Gaydadjiev. Routing protocols for mobile ad-hoc networks: Current development and evaluation. In *Proceedings of the 16th Annual Workshop on Circuits, Systems and Signal Processing, ProRisc 2005*, number 489-494, November 2005.

Appendices

Bash Shell Scripts

Testing Script

Original code that forms the high level backbone of running the simulation. Takes the parameters of number of trials, number of nodes, physical size of network boundary, maximum movement rate, number of connections, and the frequency of communication rate. Calls the programs to create traffic and movement patterns, calls the network simulator script, then uses two awk programs to record and process results.

```
1  # !/bin/bash
2  # test_script: This script accepts the network variables, creates and runs the
       simulations, then handles result output
3  echo ""
4  echo ""
5  echo "Welcome to David Chiasson's ad hoc routing tester."
6  echo ""
7  echo "DDD    AA   V    V IIIII DDD "
8  echo "D  D  A  A  V    V   I   D  D"
9  echo "D  D  AAAA  V    V   I   D  D"
10 echo "D  D A    A  V  V    I   D  D"
11 echo "DDD  A    A   VV   IIIII DDD "
12 echo ""
13 if [ "$#" != "6" ]; then
14         echo "Usage: test_script.sh trials nodes size movement_rate
               maximum_connections rate"
15         echo "example: ./test_scipt.sh 50 100 500 20 10 2.0"
16         echo "exiting..."
17         exit 1
18 fi
19
20 trials=$1
21 nodes=$2
22 size=$3
23 movement=$4
24 max_connect=$5
25 rate=$6
26
27 echo ======================================
28 echo     Begin Test $nodes-$size-$movement-$max_connect-$rate
29 echo ======================================
30
31 #scenario variable section##############
32
33
34 #end scenario variable section########
35
36 mkdir scenarios/$nodes-$size-$movement-$max_connect-$rate
37 mkdir scenarios/$nodes-$size-$movement-$max_connect-$rate/traffic
38 mkdir scenarios/$nodes-$size-$movement-$max_connect-$rate/movement
39
40 mkdir results/$nodes-$size-$movement-$max_connect-$rate
41
42 ns_address="/home/dchiasso/Downloads/ns-allinone -2.34/ns-2.34/ns"
43 mvmt_address="/home/dchiasso/Downloads/ns-allinone -2.34/ns-2.34/indep-utils/cmu-scen-
       gen/setdest/setdest"
44 trfk_address="/home/dchiasso/Downloads/ns-allinone -2.34/ns-2.34/indep-utils/cmu-scen-
```

```
    gen/cbrgen.tcl"
45
46 scenario_number=0
47
48 #echo "How many  trials would you like to run?";
49 #read inputline
50 #scenarios="$inputline"
51 #while [ -z "${scenarios}" ]; do
52 #       echo "Please enter a number"
53 #       read inputline
54 #       scenarios="$inputline"
55 #done
56 #echo "what you just inputed was:"
57 #echo $scenarios
58 #echo "right?"
59
60 if [ -e results/$nodes-$size-$movement-$max_connect-$rate/DSDV_results.txt ] ; then
61 rm results/$nodes-$size-$movement-$max_connect-$rate/DSDV_results.txt
62 fi
63 if [ -e results/$nodes-$size-$movement-$max_connect-$rate/AODV_results.txt ]; then
64 rm results/$nodes-$size-$movement-$max_connect-$rate/AODV_results.txt
65 fi
66 if [ -e results/$nodes-$size-$movement-$max_connect-$rate/DSR_results.txt ]; then
67 rm results/$nodes-$size-$movement-$max_connect-$rate/DSR_results.txt
68 fi
69
70 while [ $scenario_number -lt $1 ] ; do
71
72         let scenario_number++
73         echo CURRENT TEST:
74         echo nodes=$2 size=$3 movement=$4 connections=$5 rate=$6
75         echo Now creating scenario number $scenario_number out of $1
76
77         trfc_pat=./scenarios/$nodes-$size-$movement-$max_connect-$rate/traffic/
               scenario_$scenario_number
78         move_pat=./scenarios/$nodes-$size-$movement-$max_connect-$rate/movement/
               scenario_$scenario_number
79         rslt_dir=./results/$nodes-$size-$movement-$max_connect-$rate
80
81         echo "creating␣traffic␣pattern..."
82         $ns_address $trfk_address -type cbr -nn $(($nodes-1)) -seed 1 -mc $max_connect
               -rate $rate > $trfc_pat
83         #./traffic_maker.sh $nodes $max_connect $rate > $trfc_pat
84         echo "DONE"
85
86         echo "creating␣movement␣pattern..."
87         $mvmt_address -n $nodes -p 0.0 -M $movement -t 100.0 -x $size -y "0.25␣*␣$size
               " > $move_pat
88         echo "DONE"
89
90         echo Testing DSDV with scenario $scenario_number
91
92         $ns_address adhoc.tcl DSDV $trfc_pat $move_pat $nodes $size > trash.txt
93         awk -f getRatio.awk temp.tr >> $rslt_dir/DSDV_results.txt
94
95         echo Testing AODV with scenario $scenario_number
96
97         $ns_address adhoc.tcl AODV $trfc_pat $move_pat $nodes $size > trash.txt
```

```
        awk -f getRatio.awk temp.tr >> $rslt_dir/AODV_results.txt

        echo Testing DSR with scenario $scenario_number

        $ns_address adhoc.tcl DSR $trfc_pat $move_pat $nodes $size > trash.txt
        awk -f getRatio.awk temp.tr >> $rslt_dir/DSR_results.txt

#       echo Testing TORA with scenario $scenario_number

#       $ns_address adhoc.tcl TORA $trfc_pat $move_pat $nodes $size > trash.txt
#       awk -f getRatio.awk temp.tr >> $rslt_dir/TORA_results.txt


        echo scenario $scenario_number succesfuly completed!

done

echo ======================================
echo Test $nodes-$size-$movement-$max_connect-$rate RESULTS SUMMARY
echo ======================================
echo DSDV
awk -f average_results.awk $rslt_dir/DSDV_results.txt
echo AODV
awk -f average_results.awk $rslt_dir/AODV_results.txt
echo DSR
awk -f average_results.awk $rslt_dir/DSR_results.txt

echo ====================================== >> active_results.txt
echo Test $nodes-$size-$movement-$max_connect-$rate RESULTS SUMMARY  >>
    active_results.txt
echo ====================================== >> active_results.txt
echo DSDV >> active_results.txt
awk -f average_results.awk $rslt_dir/DSDV_results.txt >> active_results.txt
echo AODV >> active_results.txt
awk -f average_results.awk $rslt_dir/AODV_results.txt >> active_results.txt
echo DSR >> active_results.txt
awk -f average_results.awk $rslt_dir/DSR_results.txt >> active_results.txt

echo ====================================== >> $rslt_dir/summary.txt
echo Test $nodes-$size-$movement-$max_connect-$rate RESULTS SUMMARY  >> $rslt_dir/
    summary.txt
echo ====================================== >> $rslt_dir/summary.txt
echo DSDV >> $rslt_dir/summary.txt
awk -f average_results.awk $rslt_dir/DSDV_results.txt >> $rslt_dir/summary.txt
echo AODV >> $rslt_dir/summary.txt
awk -f average_results.awk $rslt_dir/AODV_results.txt >> $rslt_dir/summary.txt
echo DSR >> $rslt_dir/summary.txt
awk -f average_results.awk $rslt_dir/DSR_results.txt >> $rslt_dir/summary.txt
```

Traffic Scenario Maker

This program is original code which takes the parameters of number of nodes, number of connections, and communication rate, then returns a random traffic scenario file.

```
# !/bin/bash
# making traffic patterns
if [ "$#" != "3" ]; then
        echo "Usage: traffic_maker.sh nodes maximum_connections rate"
        echo "example: ./traffic_maker.sh 50 100 2.0"
```

```
        echo "exiting..."
        exit 1
fi

nodes=$1
connect=$2-1
rate=`awk 'BEGIN{printf("%f", 1 / '$3')}'`

current_node=0

for ((i=0;i<=$connect;i+=1))
do

        #time=$[ ( $RANDOM % 100 ) + 1 ]
        #time=`awk 'BEGIN{srand(rand()); printf("%f", rand() * 100)}'`
        rand_node=$[ $RANDOM % $nodes ]

        while [ "$rand_node" == "$i" ]; do
                rand_node=$[ $RANDOM % $nodes ]
        done

        echo "#"
        echo "#connection number "$i
        echo "#"
        echo 'set udp_('$i') [new Agent/UDP]'
        echo '$ns_ attach-agent $node_('$current_node') $udp_('$i')'
        echo 'set null_('$i') [new Agent/Null]'
        echo '$ns_ attach-agent $node_('$rand_node') $null_('$i')'
        echo 'set cbr_('$i') [new Application/Traffic/CBR]'
        echo '$cbr_('$i') set packetSize_ 512'
        echo '$cbr_('$i') set interval_ '$rate
        echo '$cbr_('$i') set random_ 1'
        echo '$cbr_('$i') set maxpkts_ 1'
        echo '$cbr_('$i') attach-agent $udp_('$i')'
        echo '$ns_ connect $udp_('$i') $null_('$i')'
        echo '$ns_ at '$[ ( $RANDOM % 100 ) ]'.'$[ ( $RANDOM % 100 ) ]$[ ( $RANDOM %
            100 ) ]' "$cbr_('$i') start"'

        let current_node++
        if [ $current_node -ge $nodes ]; then
        current_node=0
        fi
done
```

TCL Script

adhoc.tcl

This code is adapted from a script written by Chih-Heng Ke of National Cheng Kung University in Taiwan. This code runs the simulation files through the NS software, and creates the trace files.

```
if {$argc !=5} {
        puts "Usage: ns adhoc.tcl  Routing_Protocol Traffic_Pattern Scene_Pattern
            Node_Count size"
        puts "Example:ns adhoc.tcl DSDV cbr-50-10-8 scene-50-0-20 100 500"
        exit
}

set par1 [lindex $argv 0]
set par2 [lindex $argv 1]
```

```
set par3 [lindex $argv 2]
set par4 [lindex $argv 3]
set par5 [lindex $argv 4]

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

if { $par1=="DSR"} {
  set val(ifq)          CMUPriQueue
} else {
  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)         50                          ;# max packet in ifq
set val(rp)             $par1                       ;# routing protocol
set val(x)                  $par5
set val(y)                $par5
set val(seed)               0.0
set val(tr)               temp.tr                          ;#trace file
set val(nn)             $par4                              ;#number of nodes
set val(cp)               $par2                            ;#traffic_pattern (connection
    pattern)
set val(sc)       $par3                                    ;#scene_pattern (movement)
set val(stop)             100.0                            ;#stop time

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) 20
}

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
```

AWK Scripts

Get Ratio

This awk script was written by Chih-Heng Ke of National Cheng Kung University in Taiwan. It reads the trace files, and returns the packet delivery ratio.

```
BEGIN {
        sendLine = 0;
        recvLine = 0;
        fowardLine = 0;
}

$0 ~/^s.* AGT/ {
        sendLine ++ ;
}

$0 ~/^r.* AGT/ {
        recvLine ++ ;
}


$0 ~/^f.* RTR/ {
        fowardLine ++ ;
} # maybe remote transmission request?

END {
        printf "cbr s:%d r:%d, r/s Ratio:%.4f, f:%d \n", sendLine, recvLine, (recvLine
            /sendLine),fowardLine;
}
```

Average Results

This is an original awk script that averages the results from the fifty trials of each scenario.

```
BEGIN {
        total = 0;
        number = 0;
}

NF > 0  {
```

```
7
8           number++;
9           total += substr($5,7,6);
10  }
11
12  END {
13
14          printf "Collected:%d Average:%.4f \n", number, (total/number);
15  }
```