

EEC 687 Mobile Computing (Spring, 2008)

Ns-2 Laboratory #8

Chansu Yu

Cleveland State University

MANET Simulation (in-class lab #1)

- ❑ This assignment aims at making the students familiar with routing protocols used in ad hoc networks and see the performance of DSDV protocol.
- ❑ DSDV is table-driven routing scheme for ad hoc mobile wireless networks based on the Bellman-Ford algorithm. It uses sequence numbers to mark each node to improve upon the loop problem. Routing information is distributed between nodes via sending "full dumps" and incremental updates.
- ❑ Performance Metrics used
 - Packet delivery fraction : packets delivered / packets generated
 - Routing load : routing packets / packets delivered.

ns2 Instructions

- ❑ Get, read, and run the tcl script (dsdv-test.tcl)
 - The topology - 3 nodes over 500x400 m² area (where are they?)
 - The mobility model – pre-defined (what is it?)

- ❑ Determine the performance metrics using the trace file dsdv-test.tr
 - To get the number of packets sent
`grep "^s.*-NI AGT.*-It tcp.*" dsdv-test.tr | wc -l`
 - To get the actual number of packets received
`grep "^r.*-NI AGT.*-It tcp.*" dsdv-test.tr | wc -l`
 - To get the total number of routing packets sent
`grep "^\\(s\\|f\\).*-NI RTR.*-It message.*" dsdv-test.tr | wc -l`

 - Calculate Packet Delivery Fraction (received packets/sent packets)
 - Calculate Routing Load using (routing packets/received packets)
 - How to get average packet delay?

3

c.yu91@csuohio.edu

Reactive MANET Routing Protocols - DSR and AODV (in-class lab #2)

- ❑ Compare the performance of DSR and AODV protocols.

- ❑ Performance Metrics used
 - Packet delivery fraction : packets delivered / packets generated
 - Routing load : routing packets / packets delivered.

4

c.yu91@csuohio.edu

NS2 Instructions

- ❑ Get the tcl script compare.tcl
- ❑ This script takes 4 command line arguments - scenario file, traffic file, output trace file and routing protocol (1 = DSR and 2 = AODV). Script usage is :
\$ ns compare.tcl -scen {scen} -tfc {tfc} -tr {tr} -rpr {rpr}
- ❑ Use setdest to create five mobility files
\$ setdest -v 1 -n 25 -p 0 -M 20 -t 100 -x 500 -y 500 > scen-25-0
\$ setdest -v 1 -n 25 -p 10 -M 20 -t 100 -x 500 -y 500 > scen-25-10
\$ setdest -v 1 -n 25 -p 20 -M 20 -t 100 -x 500 -y 500 > scen-25-20
\$ setdest -v 1 -n 25 -p 40 -M 20 -t 100 -x 500 -y 500 > scen-25-40
\$ setdest -v 1 -n 25 -p 100 -M 20 -t 100 -x 500 -y 500 > scen-25-100
- ❑ Use cbrgen.tcl to create four traffic files
ns cbrgen.tcl -type cbr -nn 25 -seed 1 -mc 5 -rate 8.0 > cbr-25-5
ns cbrgen.tcl -type cbr -nn 25 -seed 1 -mc 10 -rate 8.0 > cbr-25-10
ns cbrgen.tcl -type cbr -nn 25 -seed 1 -mc 15 -rate 8.0 > cbr-25-15
ns cbrgen.tcl -type cbr -nn 25 -seed 1 -mc 20 -rate 8.0 > cbr-25-20

5

c.yu91@csuohio.edu

NS2 Instructions

- ❑ With five mobility and four traffic files, we have 20 simulation scenarios.
- ❑ For each of DSR and AODV, we will use a script file (run-dsr.sh and run-aodv.sh) to run them all. You can run these scripts as follows.
\$ chmod +x run-aodv.sh
\$ chmod +x run-dsr.sh
\$./run-aodv.sh
\$./run-dsr.sh
- ❑ The above scripts will create 3 files each. The files have a suffix recv, sent, route_pkts, to mean received, sent and routing packets. Each line in the file is "Pause Time", "CBR Load", "Extracted Number".
- ❑ Handin
 - Plot each performance metric for both DSR and AODV versus pause-times, for each CBR Load.
 - State if any peculiar behavior is observed. Give a brief report as to the interpretation of the graphs.

6

c.yu91@csuohio.edu

Reactive MANET Routing Protocols - DSR and AODV (Lab Report)

- ❑ Try the whole experiment with seed = 2 in the run-aodv.sh and run-dsr.sh, in the "ns cbrgen.tcl..." command. And make the plots again. Does the behavior change markedly?
- ❑ Change one of the parameters in DSR to see it's effect on network performance.
 - Make sure to compile and re-make the ns executable when you modify a source file.
- ❑ Bonus: Can you measure RREQ, RREP and RERRs?

c.yu91@csuohio.edu

dsragent.cc

```
/****** selectors *****/
bool dsragent_snoop_forwarded_errors = true; // give errors we forward to our cache?
bool dsragent_snoop_source_routes = true; // should we snoop on any source routes we see?
bool dsragent_reply_only_to_first_rreq = false; // should we only respond to the first route request we receive from // a host?

bool dsragent_propagate_last_error = true; // should we take the data from the last route error msg sent to us // and propagate it around on the next propagating route request we do? // this is aka grat route error propagation

bool dsragent_send_grat_replies = true; // should we send gratuitous replies to effect route shortening?
bool dsragent_salvage_with_cache = true; // should we consult our cache for a route if we get a xmitfailure // and salvage the packet using the route if possible

bool dsragent_use_tap = true; // should we listen to a promiscuous tap?
bool dsragent_reply_from_cache_on_propagating = true; // should we consult the route cache before propagating rt req's // and // answer if possible?
bool dsragent_ring_zero_search = true; // should we send a non-propagating route request as the first action // in each route discovery action? // NOTE: to completely turn off replying from cache, you should // set both dsragent_ring_zero_search and // dsragent_reply_from_cache_on_propagating to false

bool dsragent_dont_salvage_bad_replies = true; // if we have an xmit failure on a packet, and the packet contains a // route reply, should we scan the reply to see if contains the dead link? // if it does, we won't salvage the packet unless there's something aside // from a reply in it (in which case we salvage, but cut out the rt reply)

bool dsragent_require_bi_routes = true; // do we need to have bidirectional source routes? // [XXX this flag doesn't control all the behaviors and code that assume // bidirectional links -dam 5/14/98]
```

c.yu91@csuohio.edu

How To Obtain #RREQ/RREP

- DSR Trace %d [%d %d] [%d %d %d %d->%d] [%d %d %d %d->%d]
- Number Of Nodes Traversed
- Routing Request Flag, Route Request Sequence Number
- Routing Reply Flag, Route Request Sequence Number, Reply Length
- Source Of Source Routing -> Destination Of Source Routing
- Error Report Flag (?), Number Of Errors, Report To Whom
- Link Error From -> Link Error To

http://nslam.isi.edu/nslam/index.php/NS-2_Trace_Formats

9

c.yu91@csuohio.edu

DSR traces using DSR code

- Trace file analysis
 - Using grep and awk to get #RREQ

```
$ cat file.tr | grep DSR | awk '$19=="[1]" {print $2}' | wc -l
```
 - Using grep and awk to get #RREP:

```
$ cat file.tr | grep DSR | awk '$21=="[1]" {print $2}' | wc -l
```
 - Using grep and awk to get #RERR:

```
$ cat file.tr | grep DSR | awk '$25=="[1]" {print $2}' | wc -l
```

** RERR can be piggybacked on RREQ, which is called gratuitous RERR and can be enabled by "dsragent_propagate_last_error"*

c.yu91@csuohio.edu