Lab 2 TCP over Ad Hoc Networks

August 11, 2010

Contents

1 Introduction

The experiment will make you get familiar with NS2 wireless simulation process. Through some simple scenarios, you will get intuition of different routing protocols,e.g. DSDV, AODV, DSR. These knowledge will be the fundamentals of your further research. You are expected to follow the instructions in the first part and do some of new test based on this.

2 Simulating TCP over DSDV

2.1 Simulation scenario

We start by presenting simple script that runs a single TCP connection over a 3-nodes network over an area of a size of 500m over 400m depicted in figure 1. The location process is as follows.

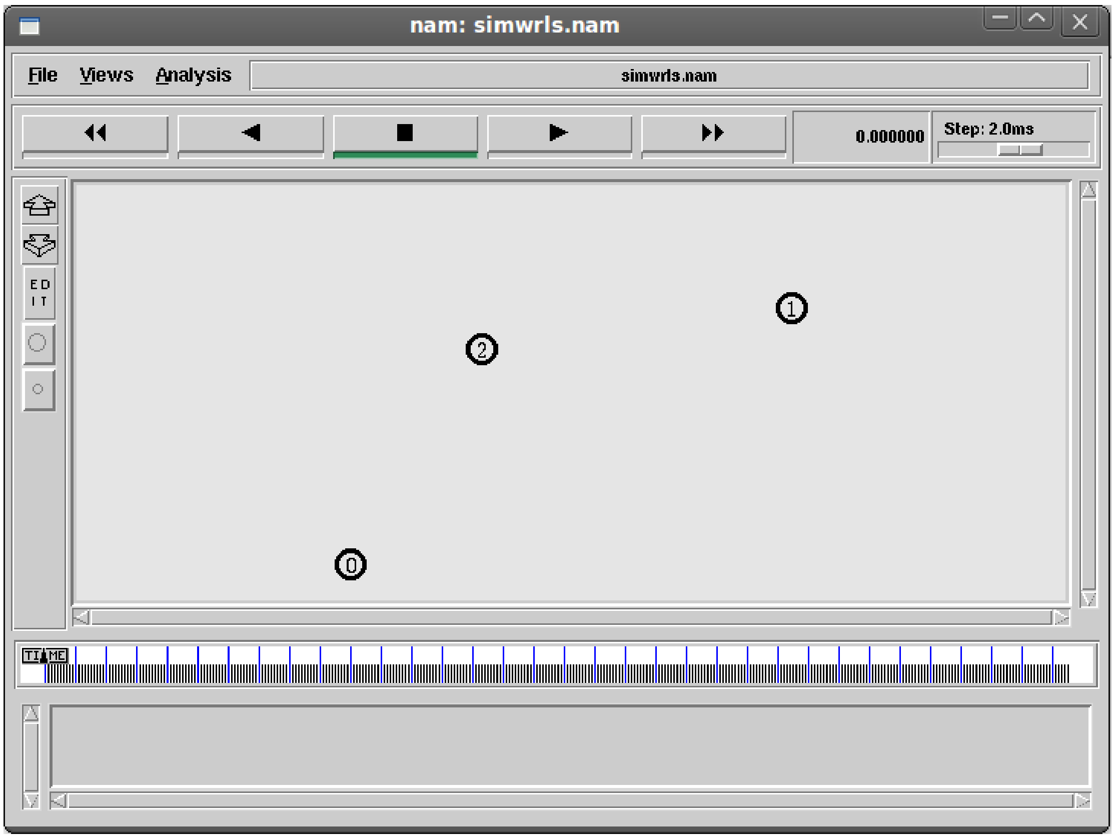

Figure 1: Example of a 3 node ad hoc network

- The initial location of nodes 0,1,and 2 are respectively (5,5), (490,285) and (150,240) (the z coordinate is assumed throughout to be 0).
- At time 10, node 0 starts moving towards point (250,250) at a speed of 3m/sec. At time 15, node 1 starts moving towards point (45, 285) at a speed of 5m/sec. At time 110, node 0 starts moving towards point (480,300) at a speed of 5m/sec.

The simulation lasts 150 sec. At time 10, a TCP connection is initiated between node 0 and node 1.

We shall use below the DSDV ad-hoc routing protocol and the IEEE802.11 MAC protocol.

2.2 Writing the tcl script

We begin by specifying some basic parameters for the simulations, providing information for the different layers. This is done as follows:

```
set val(chan)   Channel/WirelessChannel    ;#channel type
set val(prop)   Propagation/TwoRayGround   ;#radio-propagation model
set val(netif)  Phy/WirelessPhy            ;#networks 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) 50                         ;#mac packet in ifq
set val(nn)     3                          ;#number of mobilenodes
set val(rp)     DSDV                       ;#routing protocol
set val(x)      500                        ;#X dimension of topography
set val(y)      400                        ;#Y dimension of topography
set val(stop)   150                        ;#time of simulation end
```

These parameters are used in the configuring of the nodes, which is done with the help of the following command:

```
$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)       \
                -channelType $val(chan)    \
                -topoInstance $topo        \
                -agentTrace ON             \
                -routerTrace ON            \
                -macTrace OFF              \
                -movementTrace ON
```

```
for {set i 0} {$i < $val(nn) {incr i} {
    set node_($i) [$ns node]
}
```

The four last options in node-config can each be given a value of ON or OFF. The agentTrace will give in our case the trace of TCP, routerTrace provides tracing of packets involved in the routing, macTrace is related to tracing MAC protocol packets, and movementTrace is used to allow tracing the motion of nodes (for nam).

The initial location of node 0 is given as follows:

```
    $node_(0) set X_ 5.0
    $node_(0) set  Y_  5.0
    $node_(0) set  Z_  0.0
```

and similarly we provide the initial location of other nodes.

A linear movement of a node is generated by specifying the time in which it starts, the x and y values of the target point and the speed. For example, node's 1 movement will be written as:

```
$ns at 15.0 "$node_(1)  setdest  45.0  258.0  5"
```

We need to create the initial node position for nam using

```
for {set i 0} {$i < $val(nn)} { incr i } {
      # 30 defines the node size for nam
     $ns initial_node_pos $node_($i) 30
}
```

We tell nodes when the simulation ends with

```
for { set i 0 } { $i < $val(nn) } { incr i} {
     $ns at $val(stop) "$node_($i) reset"
}
```

We then create the TCP connection and ftp application over it.

2.3 Trace format

An example of a line in the output trace is

```
r 40.639943289 _1_ AGT --- 1569 tcp 1032 [a2 1 2 800] ------- [0:0 1:0 32 1]
         [35 0] 2 0
```

- The first field is a letter that can have the values r, s, f, D for "received", "sent", "forwarded" and "dropped", respectively. It can also be M for giving a location or a movement indication, this is described later.
- The second field is the time.
- The third field is the node number.
- The fourth field is MAC to indicate if the packet concerns a MAC layer, it is AGT to indicate a the transport layer (e.g. tcp) packet, or RTR if it concerns the routed packet. It can also be IFQ to indicate events related to the interference priority queue (like drop of packets).
- After the dashes come the global sequence number of the packet (this is not the tcp sequence number).
- At the next field comes more information on the packet type (e.g. tcp, ack or udp).
- Then comes the packet size in bytes.

- The 4 numbers in the first square brackets concern mac layer information. The first hexadecimal number, a2 (which equals 162 in decimal) specifies the expected time in seconds to send this data packet over the wireless channel. The second number, 1, stands for the MAC-id of the sending node, and the third, 2, is that of the receiving node. The fourth number, 800, specifies that the MAC type is ETHERTYPE_IP.
- The next numbers in the second square brackets concern the IP source and destination addresses, then the ttl (Time to Live) of the packet (in our case 32),
- The third brackets concern the tcp information: its sequence number and the acknowledgement number.

There are other formats related to other routing mechanisms and/or packet types. A movement command has the form:

```
M 10.00000 0 (5.00, 5.00, 0.00), (250.00, 250.00), 3.00
```

Where the first number is the time, the second is the node number, then comes the origin and destination locations, and finally is given the speed.

2.4 Procedure

Input the following command at the $ prompt to run the script:

```
$ ns example3.tcl
```

Then open the trace file simple.tr. Like the format of the trace file.

Run following comand to plot figure.

```
$ xgraphy win.tr
```

2.5 Analysis of simulation results

At the beginning the nodes are too far away and a connection cannot be set. The first TCP signaling packet is transmitted at time 10 sec but the connection cannot be opened. Meanwhile nodes 0 and nodes 1 start moving towards node2. After 6 second (timeout) a second reatempt occurs but still the connection cannot be established and the timeout valure is doubled to 12sec. At time 28 another transmission attempt occurs. The timeout value is doubled again to 24 sec and again to 48 sec. Thus only at time 100 sec the connection established. The mobiles get further apart till the direct link breaks. The routing protocol is too slow to react and to create an alternative route. The window evolution is given in figure 2.

Next we slightly change the parameters of the simulation. The only change is in fact that the ftp transfer will start now at time 12 instead of at time 10. This will cause both nodes 0 as well as node 1 to be within the radio of node 2 when the timeout at around 53 sec expires so that when tcp connection is reattempted at that time a two hop path is established between node 0 a direct connection is established. The window size evolution is given in figure 3. At the moment of the path change there is a single TCP packet loss that cause the window to decrease.

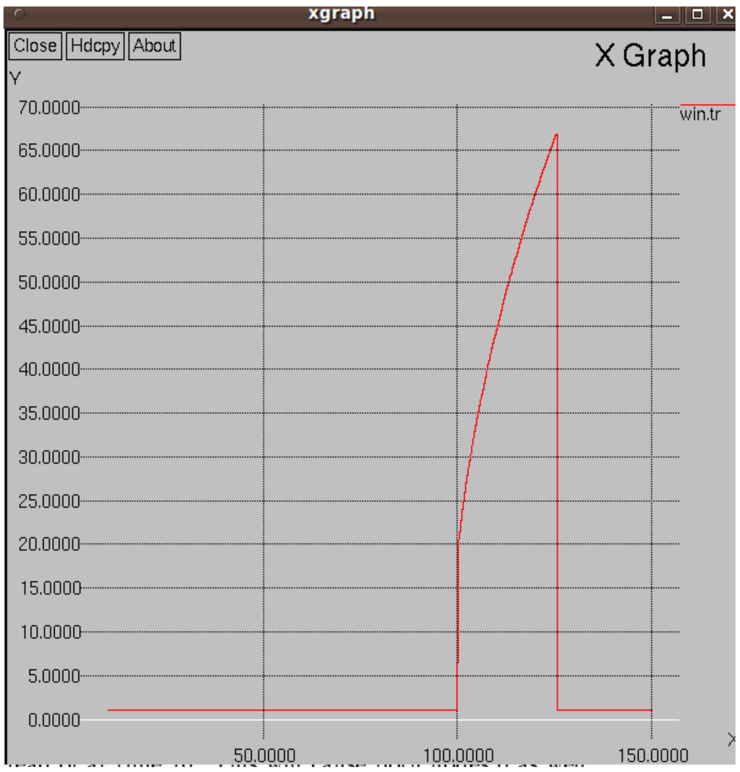

Figure 2: TCP Windows Size in a three nodes scenario with DSDV routing protocol

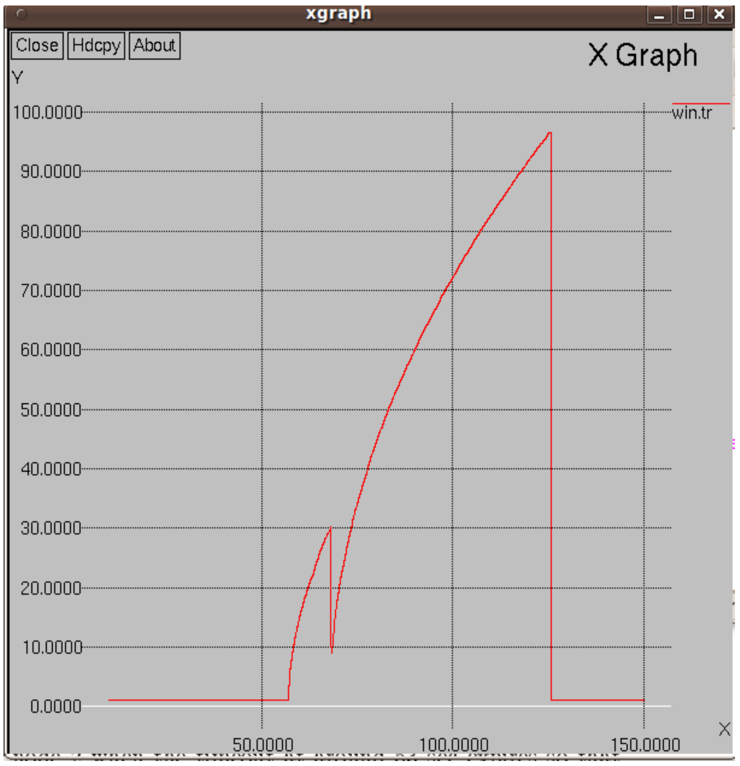

Figure 3: TCP windows size in a three nodes scenario with DSDV routing protocol

At time 125.5 nodes 0 and 1 are too far apart for the connection to be maintained and the conncetion breaks.

We note that in the DSDV, the connection was ended much earlier. The total number of TCP packets transfered using DSDV is 2079.We can obtain this information by typing

```
grep "^r" simple.tr | grep "tcp" | grep "_1_ AGT" > tcp.tr
```

and then counting the number of lines. Or we can be more precise and look at the sequence number of the last received tcp packet.

3 Comparision with other ad-hoc routing

3.1 TCP over AODV and TORA

The simulations with the same parameters as before is repeated with AODV. Change the corresponding script as follow:

```
set val(rp)    AODV   ; # routing protocol
```

The window size is giving in figure 4. The connection transfered altogether 3924 TCP data packets. It had throughout a long singe phase in which the same two hop path was used, in which node 2 relayed the packets.

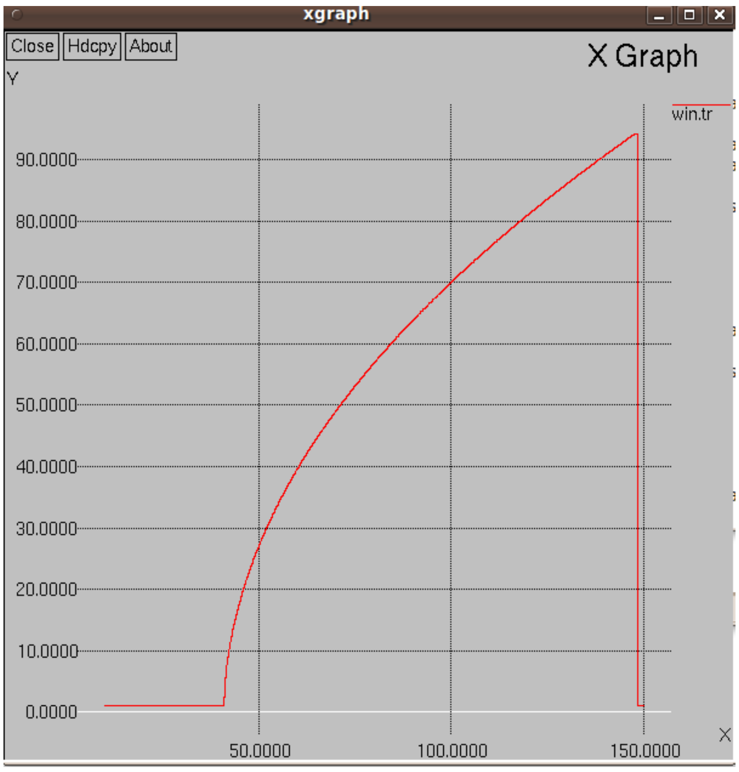

Figure 4: Window size evolution of the TCP connection for AODV

Due to the fact that changes in paths were avoid, there were no losses so the window remained high.

There are some minor errors in TORA implmentation in NS2. If you want to run this script wrls-dsdv.tcl using TORA, you have to modify some codes in ”tora.h” and ”tora.cc”. Before you do any modification, remember to backup the files. The most important thing is know what you are doing.

3.2 TCP over DSR

We first change the routing protocol to DSR by changing in wrls-dsdv.tcl the corresponding line to

```
set val(ifq)  CMUPriQueue   ; # Queue special design for DSR
set val(rp)   DSR  ; # routing protocol
```

Change the Queue/DropTail/PriQueue to CMUPriQueue is mandatory to avoid runtime errors.

3.3 Comments

In the examples that we considered, losses occurred either when the geographical range was too far for reception or when there was a route change, and there were no losses due to buffer overflow. This is due to the fact that we used the default value of the maximum window size of TCP of 20. If we use a maximum window size of 2000. There will be losses due to overflow.

3.4 Questions

Please answer the following questions in your project report.

1. How many packets have been transmitted by TCP using AODV as underlying protocol?

2. Try to plot the windows size evolution in DSR(TORA) vs DSDV, save and submit your figure. Compare their windows size evolution performance. Give your judgement.

3. In the protocols we discussed here, which one have transmitted most packets? How to calculate the total packets transmitted or received by a node?

4. Among the DSDV,DSR and AODV, which protocol most suits highly mobility systems? Provide the reasons to justify your answer.

References

[1] Marc Greis, *Tutorial for the Network Simulator ”NS”*, `http://www.isi.edu/nsnam/ns/tutorial/`

[2] *NS Manual*, `http://www.isi.edu/nsnam/ns/`

[3] Zhike Heng, *NS2 Tutorial*, `http://140.116.72.80/~smallko/ns2/book.htm`

A The Source Code

```
#specify basic parameters for the simulations
set val(chan)   Channel/WirelessChannel        ;#channel type
set val(prop)   Propagation/TwoRayGround       ;#radio−propagation model
set val(netif)  Phy/WirelessPhy                ;#networks 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) 50                             ;#mac packet in ifq
set val(nn)     3                              ;#number of mobilenodes
set val(rp)     DSDV                           ;#routing protocol
set val(x)      500                            ;#X dimension of topography
set val(y)      400                            ;#Y dimension of topography
set val(stop)   150                            ;#time of simulation end

set ns [new Simulator]

#open a standard trace file for analyzing.
set tracefd          [open simple.tr w]
set namtrace         [open simwrls.nam w]
set windowVsTime2   [open win.tr w]

$ns trace−all $tracefd
$ns namtrace−all−wireless $namtrace $val(x) $val(y)

#Set a topograhy object for ensuring the nodes move inside the Topological
        boundary
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

#Create a god object.
create−god $val(nn)

#nodes configuring
$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) \
              −channelType $val(chan) \
              −topoInstance $topo \
              −agentTrace ON \
              −routerTrace ON \
              −macTrace OFF\
              −movementTrace ON

for {set i 0} { $i < $val(nn) } { incr i} {
    set node_($i) [$ns node]
}

#Provide the initial locations of the nodes
$node_(0) set X_ 5.0
```

```
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0
$node_(1) set X_ 490.0
$node_(1) set Y_ 285.0
$node_(1) set Z_ 0.0
$node_(2) set X_ 150.0
$node_(2) set Y_ 240.0
$node_(2) set Z_ 0.0
#nodes’ movement
$ns at 10.0 ”$node_(0)␣setdest␣250.0␣250.0␣3.0”
$ns at 15.0 ”$node_(1)␣setdest␣45.0␣285.0␣5.0”
$ns at 110.0 ”$node_(0)␣setdest␣480.0␣300.0␣5.0”

#create the TCP connection and ftp application between node 0 and node 1, which
        will be initiated at time 10
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach−agent $node_(0) $tcp
$ns attach−agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach−agent $tcp
$ns at 1.0 ”$ftp␣start”

#print window size
proc plotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file ”$now␣$cwnd”
$ns at [expr $now+$time] ”plotWindow␣$tcpSource␣$file” }
$ns at 10.1 ”plotWindow␣$tcp␣$windowVsTime2”

#initial node position for nam using
for {set i 0} { $i < $val(nn) } { incr i } {
    #30 defines the node size for nam
    $ns initial_node_pos $node_($i) 30
}

#end simulation condition
for {set i 0} { $i < $val(nn) } { incr i} {
    $ns at $val(stop) ”$node_($i)␣reset”;
}
#tell the simulator to call the procedures
$ns at $val(stop) ”$ns␣nam−end−wireless␣$val(stop)”
$ns at $val(stop) ”finish”
$ns at 150.1 ”puts␣\”end␣simulation\”␣;␣$ns␣halt”
proc finish {} {
    global ns tracefd namtrace
    $ns flush−trace
    close $tracefd
    close $namtrace
}

#run the simulation
$ns run
```