# COMMUNICATION NETWORKS I

*LABORATORY WORK*

*2016/2017*

## GENERAL

The purpose of this laboratory work is to get students familiar with network simulator 2 (ns-2) software and observe and study different wireless networking issues.

The laboratory work is designed to be carried out by **groups of two students**. If you do not find a partner, you can do the laboratory work alone.

A written report must in the thesis format be returned by **14th of January 2017** in pdf format through OPTIMA (Mailbox). No Email or paper form reports are accepted. Your written report, as minimum, should contain the answers to ALL the questions and the appropriate figures as a result from the simulations. The reports will be graded as fail or pass.

Successful completion of the work requires that you have an account for the department network.

For further information/assistance please contact:

**Assistant:**

Alireza Shams Shafigh

e-mail: ashamssh@oulu.fi

# PART I

*AD HOC WLAN NETWORKING*

## 1. INTRODUCTION

In this laboratory work you will need to use software such as ns-2, awk and xgraph. All these can be ran from a command line with your stekt account. Usage of these softwares will become familiar while reading this document. It is recommended to perform this laboratory work at the department Unix lab, but if you want to work at home through SSH connection you will need X11 tunneling enabled and XServer client (for xgraph) running in your PC. You can try X-Win32 [6].

### 1.1. ns-2,

ns-2 is an object-oriented simulator developed as part of the VINT project at the University of California in Berkeley. As stated at the ns-2 homepage [1]: *Ns-2 is a discrete event simulator targeted at networking research. Ns-2 provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. Ns-2 is written in* C++ *and an Object oriented version of* Tcl *called* OTcl.

Usually ns-2 is used by creating a tcl (Tool Command Language) script where all the options are defined. C or C++ is only needed when implementing new protocols or altering the existing ones. Programs like awk and xgraph can be used to analyze and present the results.

### 1.2. tcl

Tcl [3] is a dynamic scripting language. It is mainly used for rapid prototyping, scripted applications, GUIs and testing. Tcl is easy to learn but yet powerful tool when used right.

In ns-2, tcl is used for configuration and setup. User creates a fairly simple tcl script which includes all information about the setup. Type of MAC, antenna, queue, routing protocol, topology, etc. is defined. This script is then ran by ns-2.

Example of tcl language with for loops and a procedure:

```
for {set i 1} {$i <= $val(nn) } {incr i}
  { set node_($i) [$ns_ node]

  }

proc coord_proc {a} {
    return [expr 10 * $a ]

  }

for {set i 1} {$i <= $opt(nn)} {incr i}
    { $node_($i) set X_ [coord_proc $i]
    $node_($i) set Y_ [coord_proc $i]

  }
```

## 1.3. AWK and xgraph

AWK [4] is a programming language developed for processing text-based data. In this laboratory work we will use AWK to analyze trace file created by ns-2. With AWK we can filter for example only tcp traffic from node A to node B. This way it is easy to represent for example throughput between certain nodes.

An AWK program has the general form:

```
   BEGIN {<initializations>} <search
   pattern 1> {<program actions>}
 <search pattern 2> {<program actions>}

   ...
   END                {<final actions>}
```

With xgraph [5] it is easy to graph the result created with awk.

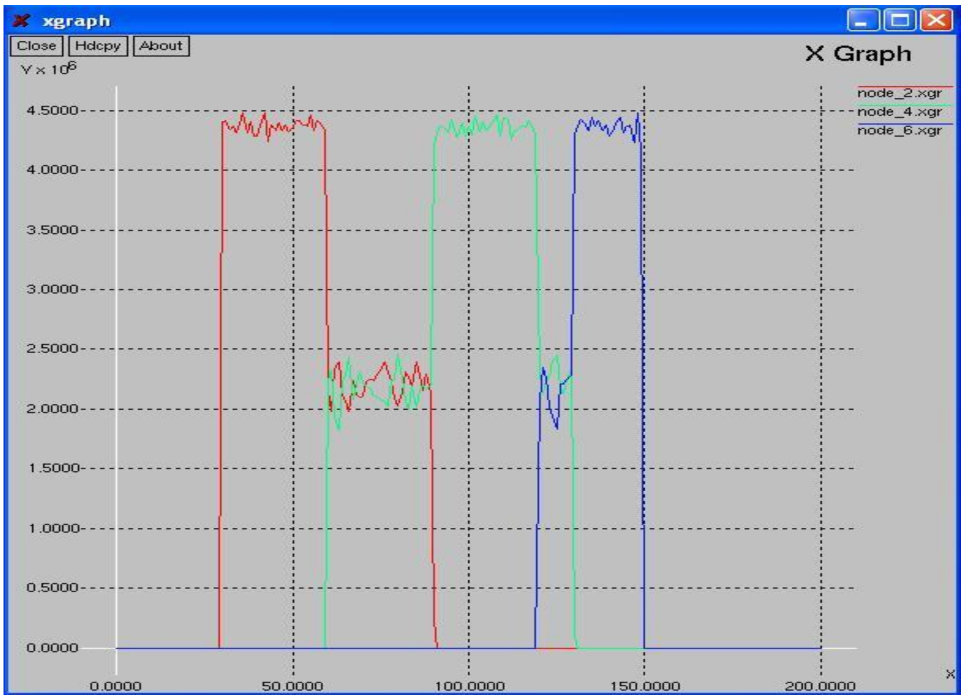


**Figure 1: Example of xgraph.**

## 2. EXERCISE 1 – AD HOC NETWORK

In the first exercise you will get to know basics of ns-2. You will also learn tcl syntax and a way to analyze and present the results. We will approach the problem by first doing the actual simulation and reviewing the results. After that we will go through the simulation script line by line.

### 2.1. Overview

In the exercise we will investigate a simple ad hoc network with six nodes positioned as shown in Figure 2. The default communication range of 802.11 nodes, in ns-2, is 240m. Therefore all six nodes considered here are within the receive range of each other. The purpose is to analyse throughput within the network.

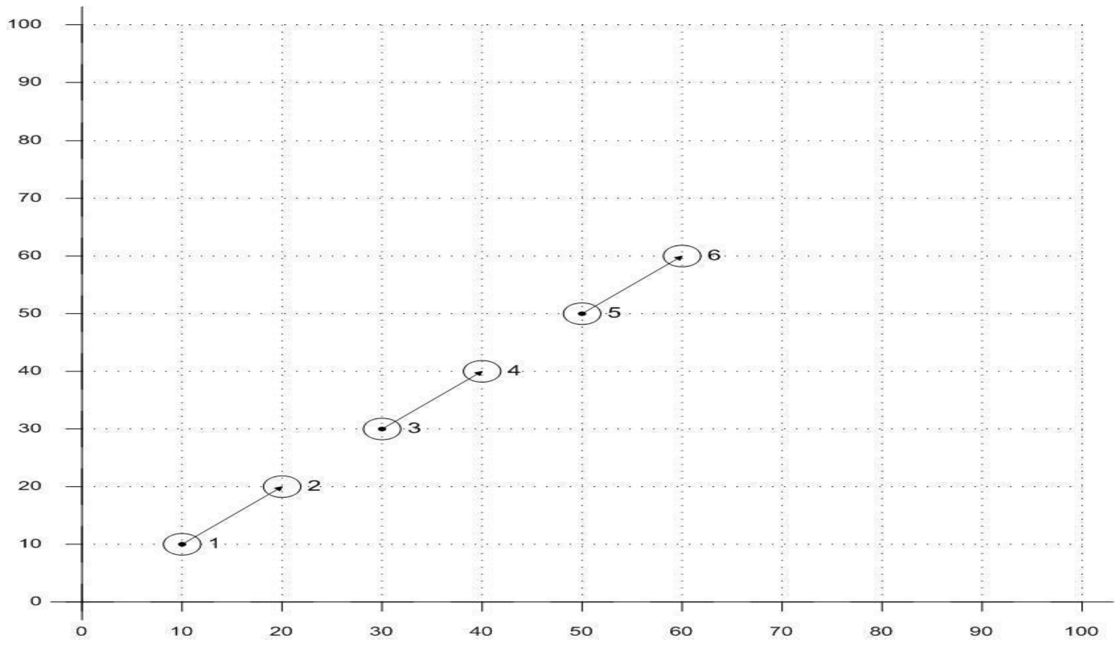


**Figure 2: The network topology for Exercise 1**

### 2.2. Start simulation

At first, you need to download the files (exerc1.tcl, fil2.awk, fil4.awk and fil6.awk) from Optima. In the unix prompt, go to the folder, where you saved the simulation script (exerc1.tcl) for Exercise 1 and give the following command:

```
ns exerc1.tcl
```

This will launch the simulation and shortly you will see an xgraph window popping up with three curves on it (see Figure 3). The graph presents throughputs of the nodes 1, 3 and 5 when they try to send TCP packets to nodes 2, 4 and 6 respectively. They start at $t_1$ = 30s, $t_3$ = 60s and $t_5$ = 120s respectively. They all stop transmitting and also the simulation ends at $t_{\text{end}}$ = 150s.

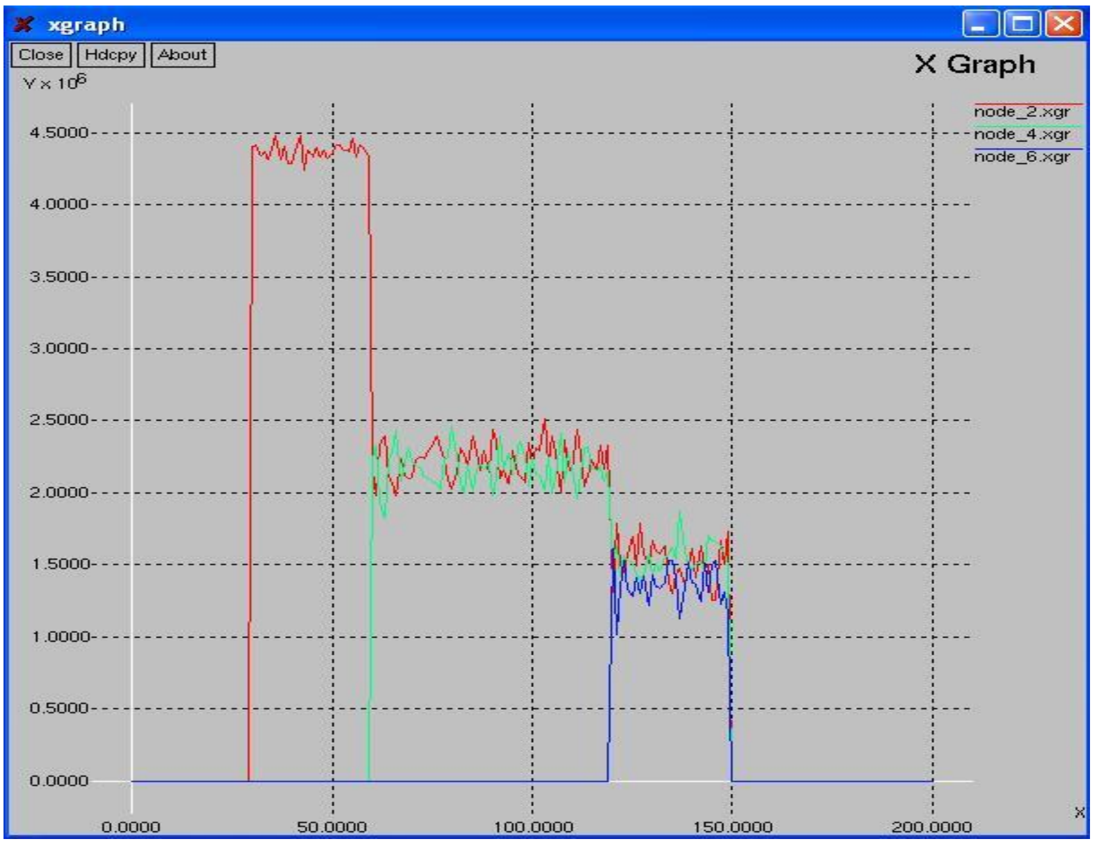


**Figure 3: Throughputs of nodes: node_(1), node_(3) and node_(5)**

### 2.3. Review the code

Next we will go through the exerc1.tcl script and learn how to get the result shown above.

#### *2.3.1. Options*

```
# ====================================================================#

#    Options          #
# ====================================================================#

set val(rp)        DumbAgent                 ;# ad-hoc routing protocol
set val(ll)        LL                        ;# Link layer type
set val(mac)       Mac/802_11                ;# MAC type
set val(ifq)       Queue/DropTail/PriQueue   ;# Interface queue type
set val(ifqlen)    50                        ;# max packets in ifq
set val(ant)       Antenna/OmniAntenna       ;# Antenna type
set val(prop)      Propagation/TwoRayGround  ;#radio-propagation model
set val(netif)     Phy/WirelessPhy           ;#network interface type
set val(chan)      Channel/WirelessChannel   ;# channel type

set val(x)          100                      ;# horizontal dimension
set val(y)          100                      ;# vertical dimension
set val(nn)         6                        ;# number of wireless nodes
set val(stop)       150.0                    ;# end time of simulation
set val(tr)         out.tr                   ;# name of the trace file

# MAC

Mac/802_11 set dataRate_ 11Mb
# RTS/CTS
Mac/802_11 set RTSThreshold_ 3000
```

First we define some options. Since all the nodes in our setup are in the same collision domain, ad-hoc routing protocol is not needed. With the wireless nodes we want to use 802.11 MAC. ifqlen specifies the interface queue size. This has large influence on the packet delays and drop rate. For more information about ns-2 specified options you can review parts of [2, table 5.1].

We use a 100m x 100m topology with six wireless nodes. We set the simulation to end at time 150.0 and define out.tr as name for the trace file, which will contain the resulting information (packet movements, etc.) of the simulation.

We want to simulate 802.11b standard, but ns-2 uses 2Mbps data rate as default. Therefore we set the dataRate_ value to 11Mb. Ns2 has RTS/CTS turned on by default, but we want it to be turned off. We set RTSThreshold to 3000 to achieve that.

### *2.3.2. General setup*

```
set ns_ [new Simulator]
```

First we need to create a new simulator instance. We will call it ns_.

```
Set up trace file
set tracefd [open $val(tr) w]
$ns_ trace-all $tracefd
```

We define ns to use file out.tr ($val(tr)) for writing the trace information. With the second command we tell ns to write all traces in to according trace file.

```
# Create the "general operations director"
create-god $val(nn)
```

god is an object that is used to store connectivity/routing information of the topology. It needs to be created always, even when not using any routing in the simulation.

```
# Create and configure topography (used for wireless scenarios)
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
```

With these two lines we will create a topology of 100m x 100m.

### *2.3.3. Nodes configuration and setup*

```
# ==================================================================#
#    Nodes configuration and setup          #
# ==================================================================#

$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 [new $val(chan)] \

        -topoInstance $topo \
        -agentTrace ON \
        -routerTrace OFF \

        -macTrace OFF \ -
        movementTrace OFF
```

Here we will set the options defined earlier in the options section. It is also stated that we only need agent traces for this simulation. Those are created by traffic sources/destinations. Router, MAC and movement traces are not needed at the moment.

```
# Creating nodes
for {set i 1} {$i <= $val(nn) } {incr i} {
        set node_($i) [$ns_ node $i]
}
```

Nodes are created with a simple for loop. We will name them node_(1), node_(2), ... , node_(6).

```
$node_(1) set X_ 10.0
$node_(1) set Y_ 10.0
$node_(2) set X_ 20.0
$node_(2) set Y_ 20.0
$node_(3) set X_ 30.0
$node_(3) set Y_ 30.0
$node_(4) set X_ 40.0
$node_(4) set Y_ 40.0
$node_(5) set X_ 50.0
$node_(5) set Y_ 50.0

$node_(6) set X_ 60.0
$node_(6) set Y_ 60.0
```

We want to place the nodes to 100m x 100m grid as shown in Figure 1. This could also be done with a for loop and tcl procedure (see section  1.2), but with only six nodes this is a simpler way.

### *2.3.4. Agents*

```
# ==============================================================#
#    Agents            #
# ==============================================================#
```

Agents represent endpoints where network-layer packets are constructed or consumed.

We are using TCP agents with FTP traffic applications.

```
# 1500 - 20 byte IP header - 40 byte TCP header = 1440 bytes
Agent/TCP set packetSize_ 1440 ;# This size EXCLUDES the TCP header
```

We will define the TCP packet size for 1440 bytes to achieve 1500 byte packets.

```
for {set i 1} {$i < $val(nn)} {incr i 2} {
 set agent($i) [new Agent/TCP]
 $ns_ attach-agent $node_($i) $agent($i)

 set sink($i) [new Agent/TCPSink]

 $ns_ attach-agent $node_([expr $i +1])
 $sink($i) $ns_ connect $agent($i) $sink($i)
 set app($i) [new Application/FTP]

 $app($i) attach-agent $agent($i)
}
```

We will use a for loop to attach TCP agents to nodes node_(1), node_(3) and node_(5) for sending and TCPSink agents to nodes node_(2), node_(4) and node_(6) for receiving. Respective sending and sink agents must be connected to each other. FTP applications are created and attached to the sending agents.

```
# Start times for traffic sources
$ns_ at 30.0 "$app(1) start"
$ns_ at 60.0 "$app(3) start"
$ns_ at 120.0 "$app(5) start"

# End times for traffic sources
$ns_ at $val(stop) "$app(1) stop"
$ns_ at $val(stop) "$app(3) stop"
$ns_ at $val(stop) "$app(5) stop"
```

We schedule the created FTP applications to start at different times and end at the same time.

```
# End the simulation

$ns_ at $val(stop).1 "finish"
$ns_ at $val(stop).2 "$ns_ halt"
```

At the end of the simulation (at *t* = val(stop)) we stop the simulator and call the procedure finish, shown below.

### *2.3.5. End*

```
# ==================================================================#
#   End         #
# ==================================================================#

# Calling some external programs
proc finish {} {
for {set i 2} {$i <=6} {incr i 2} {
exec rm -f node_$i.xgr
exec awk -f fil$i.awk out.tr > node_$i.xgr
}
exec xgraph node_2.xgr node_4.xgr node_6.xgr &
puts "Finishing ns..."
exit 0
}
```

With procedure finish we will be able to easily analyse the trace file and output everything on a neat graph. We will go through this in detail in section 3.4.

```
puts "Starting Simulation..."
$ns_ run
puts "Simulation done."
```

After every options is defined, nodes, agents and applications are created and after-simulation procedures called we can finally tell ns to run the simulation.

```
$ns_ flush-trace
close $tracefd
```

At the very end we will flush all trace object write buffers and close the tracefd.

## 2.4. Analyzing the results

We are using external tools awk and xgraph to filter, compute and display the throughput graphs. Everything is based on the trace file out.tr. It is a good idea to open it and figure out what it contains. It might look something like this:

```
s 20.050000000 _0_ AGT --- 10 tcp 1480 ...
r 20.051231990 _1_ AGT --- 9 tcp 1480 ...
```

That is, at second 20.05 the agent in node_(1) (_0_ - note that in the trace file the node numbering starts from 0) sent (s) tcp packet number 10 of size 1480 bytes. At second 20.051231990 agent in node_(2) received ftp packet number 9. For more information see [2, ch. 16.1.6]

```
proc finish {} {
for {set i 2} {$i <=6} {incr i 2} {
exec rm -f node_$i.xgr
exec awk -f fil$i.awk out.tr > node_$i.xgr
}
```

We want to compute the throughput of a three nodes sending FTP traffic. To do so, we need to sum up all the packet sizes (column 8) of the rows starting with r within each second. For this we will call external program awk. After removing unnecessary files from previous simulations, awk filters the trace file out.tr using filter fil2.awk and directs the output to node_2.xgr. The same is done with fil4.awk and fil6.awk. Content of a filter file is studied in  2.4.1.

```
exec xgraph node_2.xgr node_4.xgr node_6.xgr &
puts "Finishing ns..."
exit 0
}
```

Last, the output files (node_2.xgr, node_4.xgr and node_6.xgr) are displayed using xgraph.

### 2.4.1. Filtering

Filter files fil2.awk, fil4.awk and fil6.awk have very similar content. They only differ in which nodes' throughput is computed.

```
BEGIN{
  sim_end = 200;

  i=0;
  while (i<=sim_end) {sec[i]=0; i+=1;};
}
```

First we initialize an array sec[i], which after the next step will contain the throughput at second *i*.

```
{
 if ($1=="r" && $7=="tcp" && $3=="_1_") {

   sec[int($2)]+=$8;
 };
}
```

This will be applied to each line of the trace file. The structure states that if the value in column 1($1) is r (packet received) and the value in column 7($7) is tcp (packet type) and the value in column 3($3) is _1_ (node_(2)), then add packet size ($8) to the throughput in array sec[int($2)].

```
END{
 i=0;

 while (i<=sim_end) {print i " " sec[i]*8; i+=1;};
}
```

After going through every line of the trace file, awk will display (output directed to xgr-file) the array values (in bits/s) as:

| X | Y |
|---|---|
| sec[i] | throughput[i] |

## 2.5. Questions

1. Explain the output graph. Why does the throughput decrease at certain times?

2. Figure 1 (in section 1.3) represents similar setup as in exercise 1, but some changes have been made. Try to explain the reasons of the differences between the two graphs, Figure 1 and Figure 3. It might be helpful to try and change some parameters from the original script (exerc1.tcl).

3. Explain the example tcl-script from 2.2.
4. Explain why we set RTS threshold value to 3000 to disable RTS/CTS.

## 3. EXERCISE 2 – HIDDEN NODE

In second exercise you need to change the script from exercise 1 according to assignment described below.

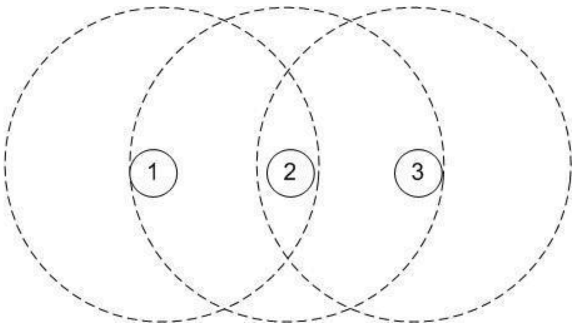


**Figure 4: The hidden node scenario**

1. Your task is to simulate a scenario with three nodes, 1, 2 and 3, where 1 is hidden to 3, and vice versa (Figure 4). Both nodes, 1 and 3 use the basic mode (no RTS/CTS) and they will start transmitting at different times. You should set data rate to 2Mbps anduseUDP/CBRtraffic.Don’tforgettochangethefilteringsettingswhenplotting the results.

   When setting up the hidden node scenario you will have to reduce the carrier sense range using the following:

   *Phy/WirelessPhy set CSThresh_ 3.5e-10*

   Tips for UDP/CBR traffic:

   - UDP agent is called UDP (vs. TCP).

- Sink agent is called Null (vs. TCPSink).
- CBR application is called Traffic/CBR (vs. FTP) for nodes 1 and 3.

- Configure the network so that node 2 receives (sink) the data and nodes 1 and 3 (agent) both transmit to node 2 (connect 1 and 3 to 2).

Here is some help to get you going. You also need to write some additional code. It is easier to implement this simulation without a loop presented in previous example.

```
set agent(1) [new Agent/UDP]

$ns_ attach-agent $node_(1)
$agent(1) set sink(2) [new
Agent/Null]

$ns_ connect $agent(1) $sink(2)

set app(1) [new Application/Traffic/CBR]

$app(1) attach-agent $agent(1)
```

See section 2.1 for a hint of how to achieve hidden node situation.

Both CBR sources should be set to send 1000-Bytes packets each 5ms. You can do it by setting:

```
$app(i) set packetSize_ 1000
$app(i) set interval_ 0.005
```

where i is number of the node. These should be placed in between the “set app”- and “$app” -commands

Note that in result files that you need to filter the node numbering starts from 0. Filter the received CBR traffic by node 2 from nodes 1 and 3. View the out.tr file to get an idea of what lines and columns to filter as an output result.

**Questions:**

1. Show the throughputs of nodes 1 and 3 in a graph. What do you see?
2. Explain how did you implement the hidden node scenario?

Modify the tcl-script so that the nodes will use RTS/CTS method. Then run again the simulation.

3. Show the throughputs of nodes 1 and 3 in a graph. What do you see?
4. How did you change the setup to use RTS/CTS?

5. What do you notice when comparing the two graphs (with and without RTS/CTS)? Explain the reason for such results.

## 4. EXERCISE 3 – ROUTING

In third exercise you will create a setup with five wireless nodes positioned as shown in Figure 5. You should use DSDV as routing protocol ([2, Table 5.1] might help). Create TCP/FTP traffic from node 1 to node 5. Note that you should create and attach TCP and sink agents to all nodes, but only connect nodes 1 and 5.

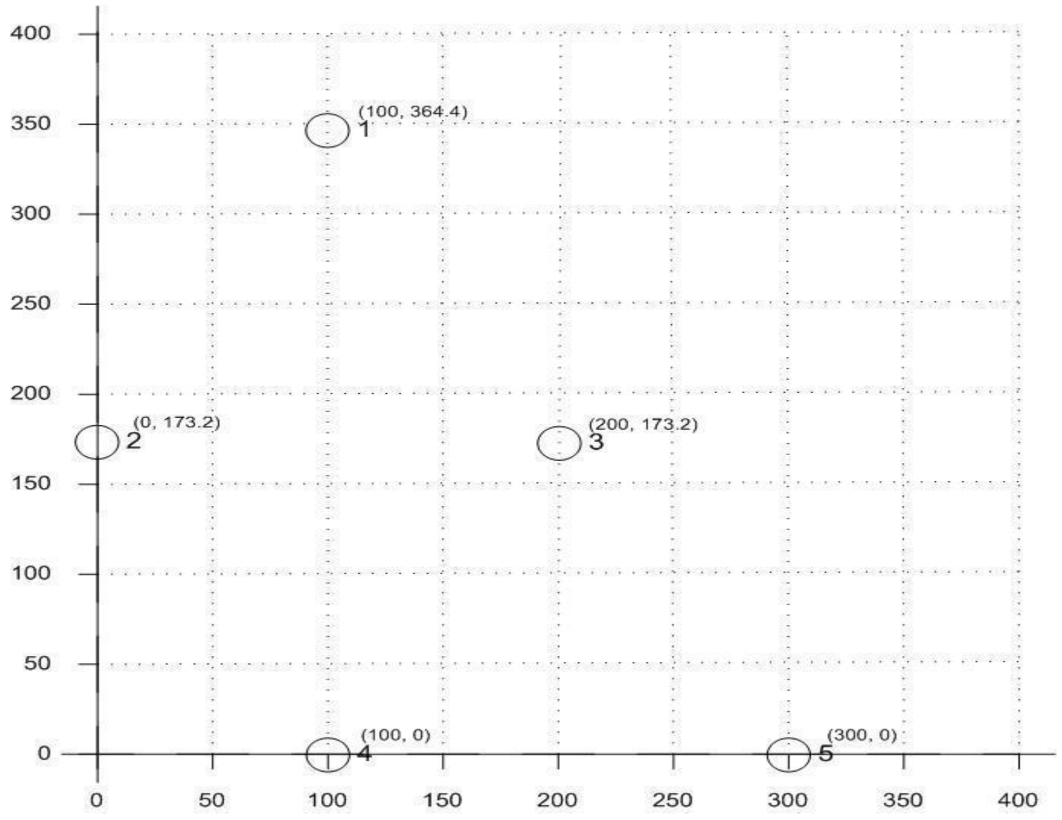


**Figure 5: Scenario for exercise 3**

**Questions:**

1. Explain the simulation script you created for this exercise.

2. Plot the throughput on a graph and explain it. Try analysing the situation in depth. If there are changes in throughput, tell why. What happens at certain times?

# PART II

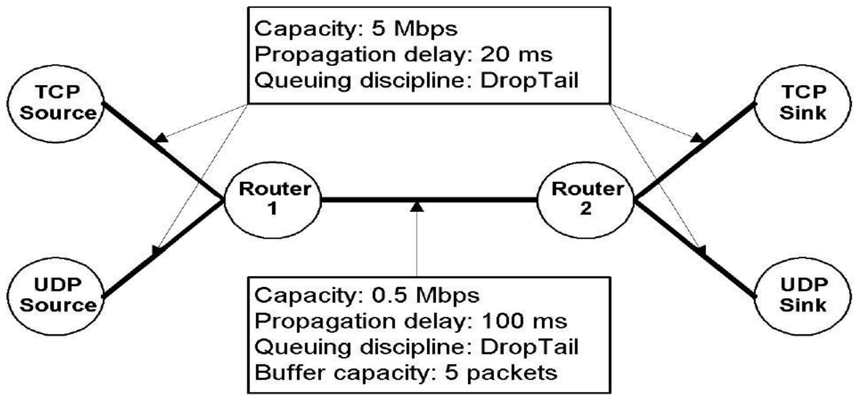
*TCP PROTOCOL*

## 5. EXERCISE 4 – TCP PROTOCOL

In this exercise, you will use the ns simulator for a comparison of congestion control algorithms of the TCP protocol.

### 5.1: Compatibility of TCP and UDP

a) Modify the following script to simulate the following topology:



```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w] $ns
namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {

     global ns nf $ns
     flush-trace

     #Close the trace file
     close $nf

     #Execute nam on the trace file
     exec nam –a out.nam&

     exit 0

}

# Creates four nodes n0, n1, n2, n3
set n0 [$ns node]

set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
```

```
# Create the following topology, where
# n0 and n1 are connected by a duplex-link with
# capacity 5Mbps, propagation delay 20ms, dropping
# discipline “DropTail”.

$ns duplex-link $n0 $n1 5Mb 20ms DropTail

# n1 and n2 are connected by a duplex-link with
# capacity 0.5Mbps, propagation delay 100ms, and
# dropping discipline “DropTail”.

$ns duplex-link $n1 $n2 0.5Mb 100ms DropTail

# n2 and n3 connected by a duplex-link with capacity 5Mbps,
# propagation delay 20ms, dropping discipline “DropTail”.

$ns duplex-link $n2 $n3 5Mb 20ms DropTail

# Create a bottleneck between n1 and n2, with a maximum queue
# size of 5 packets

$ns queue-limit $n1 $n2 5

# Instruct nam how to display the nodes
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n1 $n2 queuePos 0.5

# Establish a TCP connection between n0 and n3
set tcp [new Agent/TCP]

$ns attach-agent $n0 $tcp set
sink [new Agent/TCPSink] $ns
attach-agent $n3 $sink $ns
connect $tcp $sink

# Create an FTP transfer (using the TCP agent)
# between n0 and n3

set ftp [new Application/FTP]
$ftp attach-agent $tcp

# Start the data transfer:
$ns at 0.1 “$ftp start" $ns
at 5.0 “$ftp stop"
# Launch the animation

$ns at 5.1
“finish” $ns run
```

b) Both of the TCP and UDP sources should start transmitting at time 0.1 second, and stop at time 5.0 seconds.

c) Configure the UDP source to be a CBR source, sending packets of size 1000 bytes, at a rate of 0.1Mbps. Configure the UDP sink to be a LossMonitor agent. Keep the same TCP source and sink as in the script hw3-tahoe.tcl.

d) Use Chapter VIII of the ns tutorial ( http://www.isi.edu/nsnam/ns/tutorial/nsscript4.html#second) to write a procedure that records the throughput of the TCP flow and the throughput of the UDP flow, starting a time 0.0 seconds, until time 5.0 seconds, with a sliding window of size 0.5 seconds. Using xgraph, plot the throughput obtained by both flows on the same graph. Submit your script and the

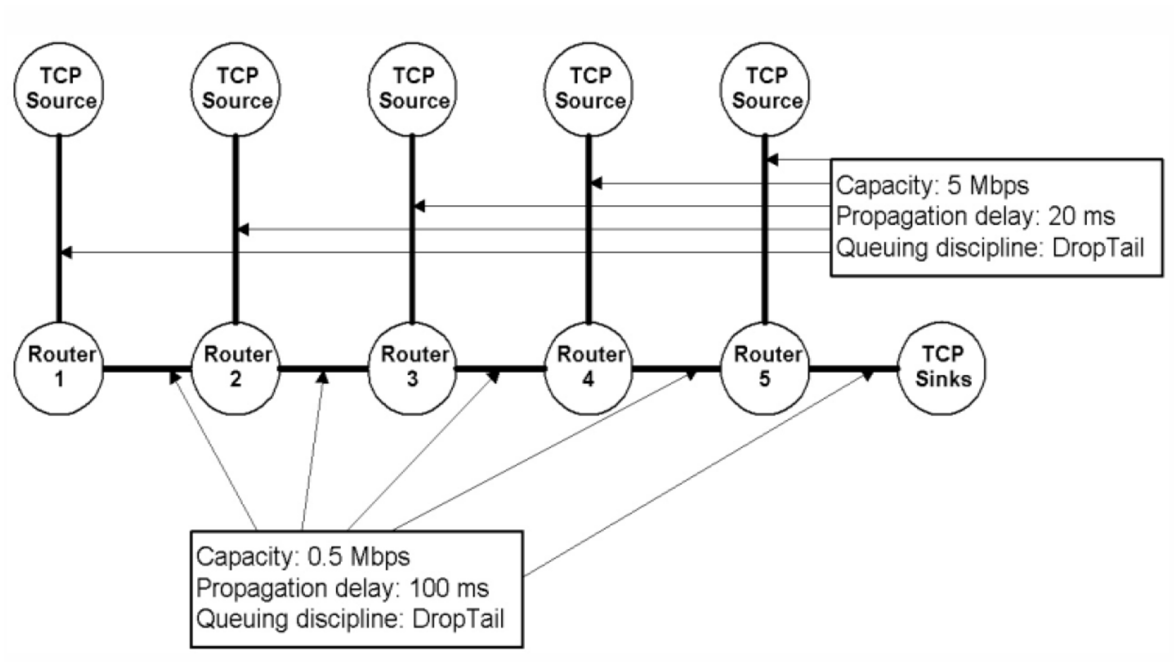
plot. Make sure the axis and the curves are properly labelled.

e). Modify the script and repeat the simulation so that the UDP source sends at a rate of

1. 0.2 Mbps.
2. 0.5Mbps, and
3. 1 Mbps.

For each of these three simulations, plot the throughput obtained by both of the TCP and UDP flows on the same graph. Submit your plots and describe the results. Explain how TCP and UDP interact.

5.2: a) Write a script simulating the following topology:



Remarks: In ns, you may attach multiple sinks to the same node and do not limit the queue size at any of the links.

All sources start transmitting at time t=0.1 s, and stop transmitting at time t=10 s.

b) Modify the procedure for recording the throughput you have used in section 5.1 to monitor the throughput of all five TCP flows at the sinks, averaged over a sliding window of size 0.5 seconds. Plot the results on the same graph. Submit your script and the plots.

c) Do all flows get a fair share of the available bandwidth? Provide a discussion of the fairness.

## 6. REFERENCES

[1] The Network Simulator – ns-2 homepage
http://www.isi.edu/nsnam/ns

[2] The ns manual http://www.isi.edu/nsnam/ns/ns-documentation.html

[3] Tcl/Tk homepage
http://www.tcl.tk/

[4] Awk at wikipedia
http://en.wikipedia.org/wiki/Awk

[5] xgraph
http://www.isi.edu/nsnam/xgraph/

[6] X-Win32 download page
http://www.starnet.com