CSE 4606
Computer Networks Lab

List of Experiments	
Experiment No	**Title and short introduction**
Experiment 1	**Introduction to Network Simulation**
Experiment 2 & 3	**Introduction to NS programming and Network Performance Analysis**
Experiment 4 & 5	**Simulating complex topology , link failure and network dynamics**
Experiment 6	**Simulation Local Area Networks**
Experiment 7	**Simulation of Wireless Network**
Experiment 8	**Trace File Analysis and Performance Evaluation of Wireless Networks**
Experiment 9	**Installation and Configuration of Domain Name System (DNS) Server**
Experiment 10	**Installation and Configuration of Mail Server**
Experiment 11	**Installation and Configuration of Server FTP and Proxy Server**
Experiment 12	**Installation and Configuration of SAMBA and DHCP Server**

Course Code : CSE 4605 **Credits : 1.00**
Course Name: Computer Networks Lab **Semester: Summer**

Course Overview

1. Course Goals:

The goals of CSE 4605 are:

- To learn the basics of network simulation (NS) and NS programming to simulate different type of network and analyze their performance
- To learn how to configure and install different servers in Linux environments and then decide where to place those servers in the network.

2. Course Content:

Introduction to computer networks, Uses of computer networks, Network models, Network topology, Layered approach of networking protocols, Design issues of layers, and TCP/IP protocol suite. Data link layer: Design issues; error control, detection and correction; Logical link control sub-layer, Medium access sub-layer; Multiple access protocols, Medium access mechanisms – ALOHA, slotted ALOHA, CSMA, CSMA/CD, CSMA/CA, WDMA; Medium access protocols – IEEE 802.3: Ethernet, IEEE 802.4: Token bus, IEEE 802.5: Token ring, Introduction to Wi-Fi; High speed LANs, FDDI, Fast Ethernet, and Gigabit Ethernet; LAN extension – Bridges, Switches, and VPN, Network layer: IP addressing, IP packet forwarding, Subnetting, CIDR, Internet protocol, ICMP, ARP, RARP, DHCP, and IPv6 overview; Routing protocols. Transport layer: Functionalities; User datagram protocol (UDP) – UDP operations and UDP package modules, Transmission control protocol (TCP) – TCP features, TCP Connection establishment and termination, TCP Flow control and error control, Congestion control. Application layer: DNS, Electronic mail (SMTP, POP, IMAP), FTP, WWW, SAMBA, DHCP, Proxy server and other linux base servers.

3. Course Organization:

Students have to complete their own assignments individually. Every student will be asked to explain their own assignments on each lab.

3.1 Marks Distribution:

- ✓ **Lab assignment**

Coding capability	30%
Problem solving capability	20%
Total Marks on lab assignment	**60%**

- ✓ **Lab Examination** **30%**
- ✓ **Lab Viva** **20%**

3.2 Acceptance and Evaluation:

Students have to finish their respective lab assignment within the lab period. If anyone fails to complete his task during the lab time can get one day more to finish his task and submit it to respective teacher. Late submission of lab assignment will deduce 50% of total marks assigned for lab assignment.

Islamic University of Technology (IUT)

Department of Computer Science and Engineering (CSE)
CSE 4606: Computer Networks Lab

Experiment No. 1

Introduction to Network Simulation

1. Objectives:

To understand the

- Discrete event simulation
- Simulation in NS-2
- Tool Command Language (TCL) basics

After performing this lab you will have a basic idea about discrete event simulation, one of the prominent network simulators known as NS-2 and tcl basics. Although we will not simulate any network in the first lab, students will have an idea of different elements of network simulation.

2. Organization:

- Students have to understand the lab material and submit the report (hard copy) individually.
- Labs will be performed in Lab 5 of CSE Department
- In case you do not understand anything feel free to contact the course teacher at sakhawat@iut-dhaka.edu.

3. Theory:

3.1 Simulation:

Simulation is the imitation of the operation of a real-world process or system over time.

- Simulation requires:
  - a model to be developed
- The model represents the system itself, whereas the simulation represents the operation of the system over time.

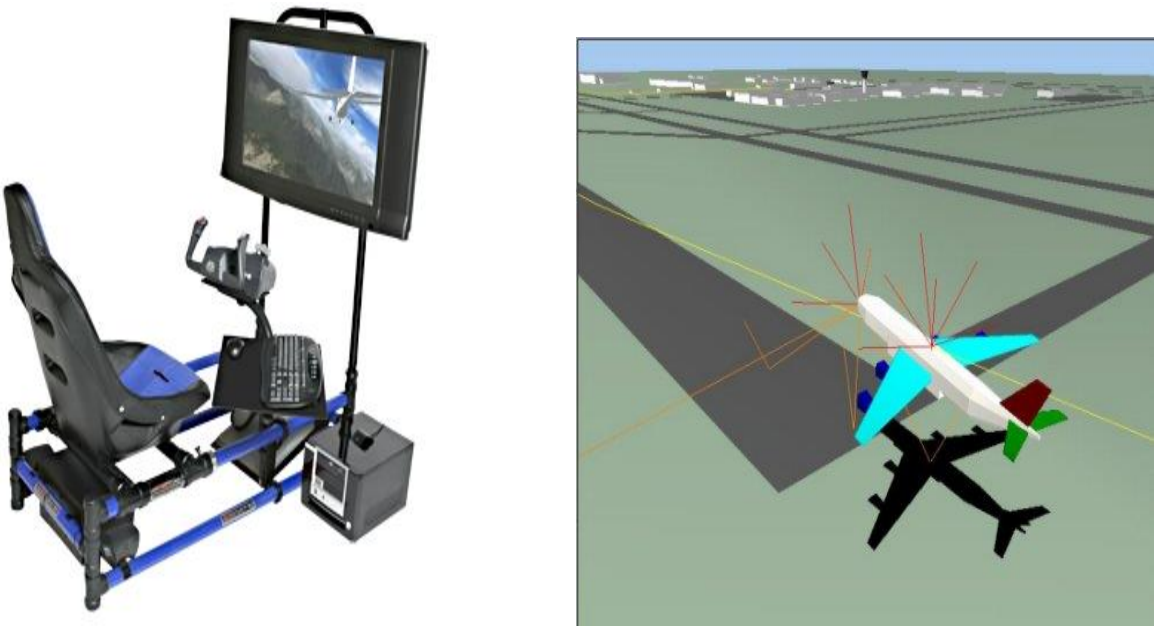


Figure: Flight Simulator

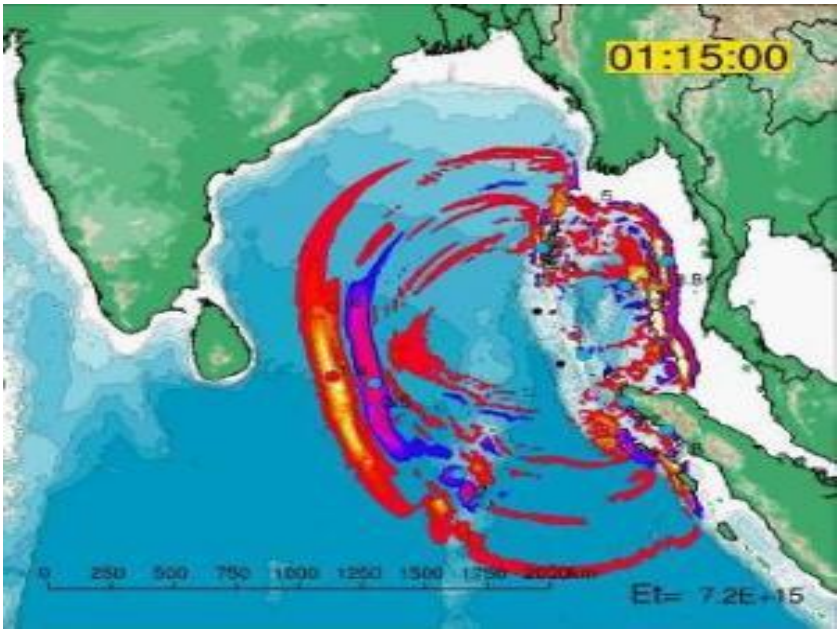


Figure: Using sophisticated computational techniques to simulate the tsunami

3.2 Network Simulation:

Network simulation is a technique where a program models the behavior of a network

- ✓ either by calculating the interaction between the different network entities (hosts, routers, data links, packets, etc.) using mathematical formulas
- ✓ Or actually capturing and playing back observations from a production network.

3.3 Network Simulator:

A **network simulator** is a piece of software or hardware that predicts the behavior of a network, without an actual network being present.

- A network simulator is a software program that imitates the working of a computer network.

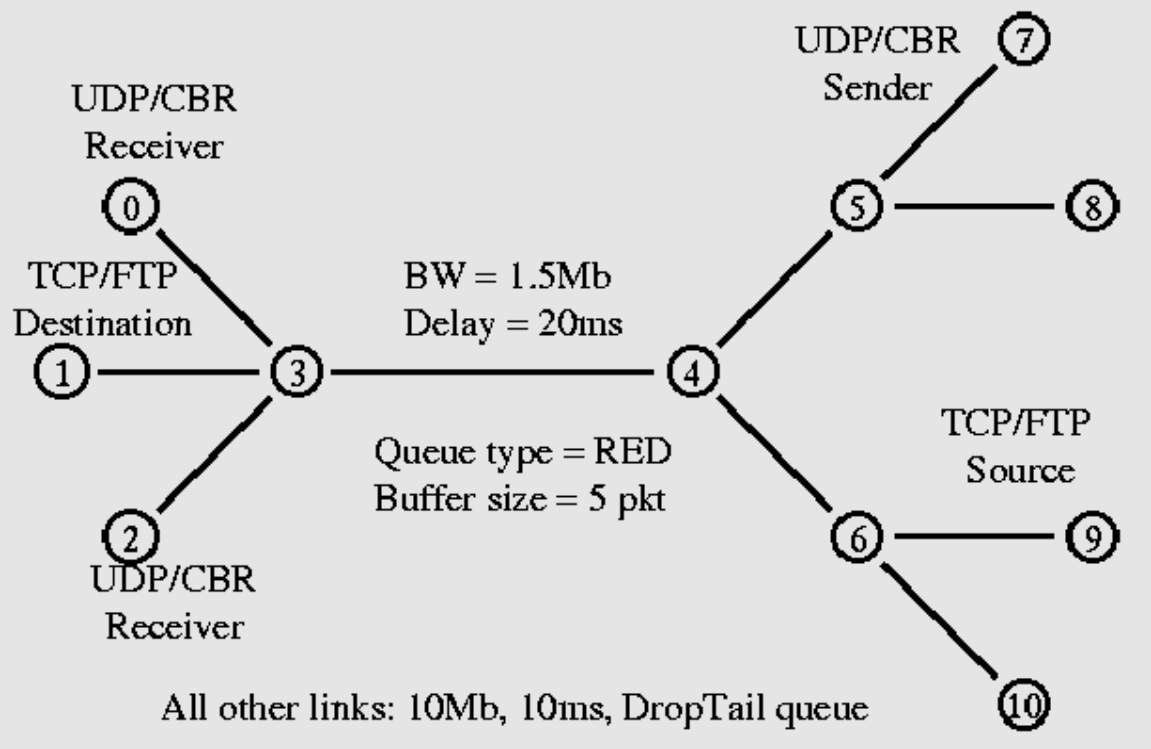
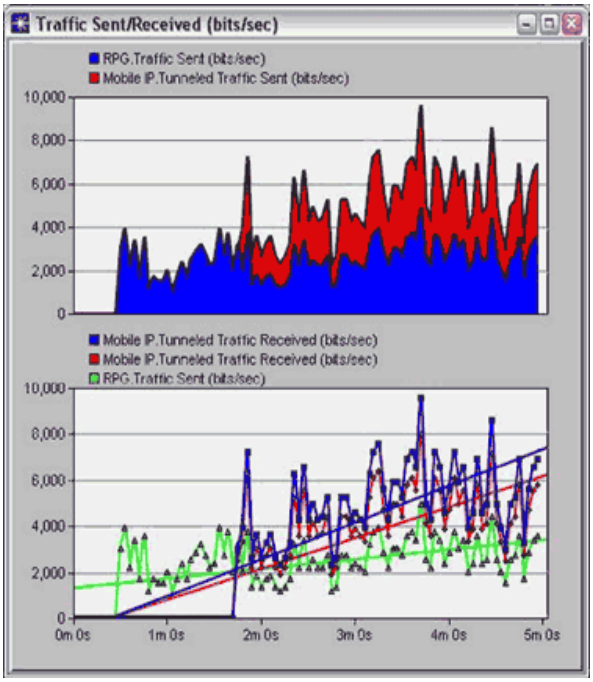
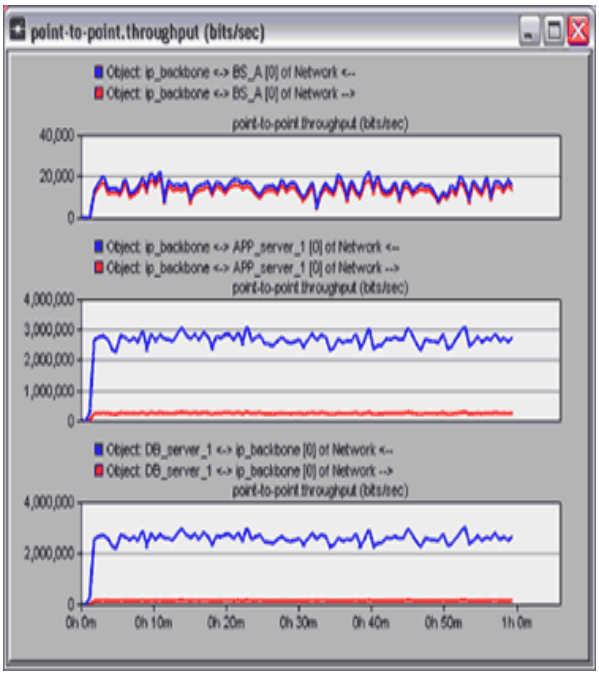


Figure: Example of simulation network with different parameters





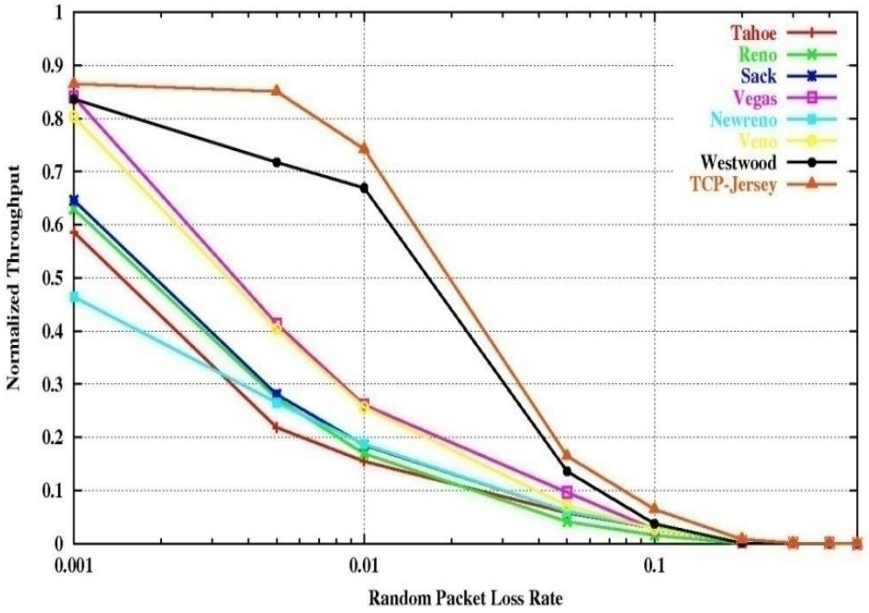


Figure: Network Simulation Result Analysis

3.4 Network Simulator (NS) Introduction:

NS is an object oriented discrete event simulator. NS focuses on modeling network protocols and main features are as follows

- Wired, wireless, satellite
- TCP, UDP, multicast, unicast
- Web, telnet, ftp
- Ad hoc routing, sensor networks
- Infrastructure: stats, tracing, error models, etc
- Traffic Models: CBR, VBR, Web etc
- Error Models: Uniform, bursty etc
- Misc: Radio propagation, Mobility models , Energy Models
- Topology Generation tools
- Visualization tools (NAM), Tracing

Simulator maintains list of events and executes one event after another. Model world as *events*

- Simulator has list of events
- Process: take next one, run it, until done
- Each event happens in an instant of *virtual (simulated) time*, but takes an arbitrary amount of *real* time

• NS uses simple model: single thread of control => no locking or race conditions to worry about (very easy)

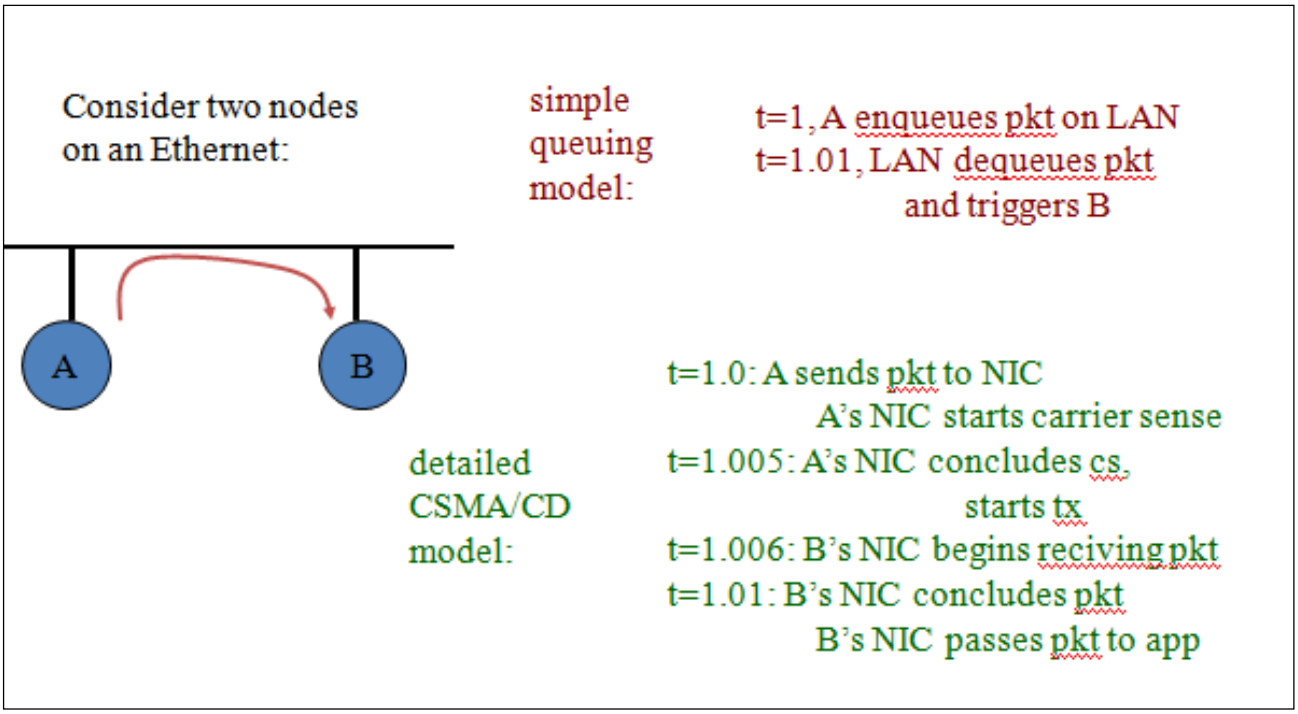


Figure: Discrete Event Examples

3.4.1 NS Goals:

- Support networking research and education
  - Protocol design, traffic studies, etc
  - Protocol comparison
- Provide a *collaborative* environment
  - Freely distributed, *open source*
    - Share code, protocols, models, etc
  - Allow easy *comparison* of similar protocols
  - *Increase confidence* in results
    - More people look at models in more situations
    - Experts develop models
- *Multiple levels of detail* in one simulator

3.4.2 NS Components:

1. NS, the simulator itself
2. Nam, the network animator
   - Visualize *ns* (or other) output
   - Nam editor: GUI interface to generate ns scripts
3. Pre-processing:
   - Traffic and topology generators
4. Post-processing:
   - Simple trace analysis, often in Awk, Perl, or Tcl

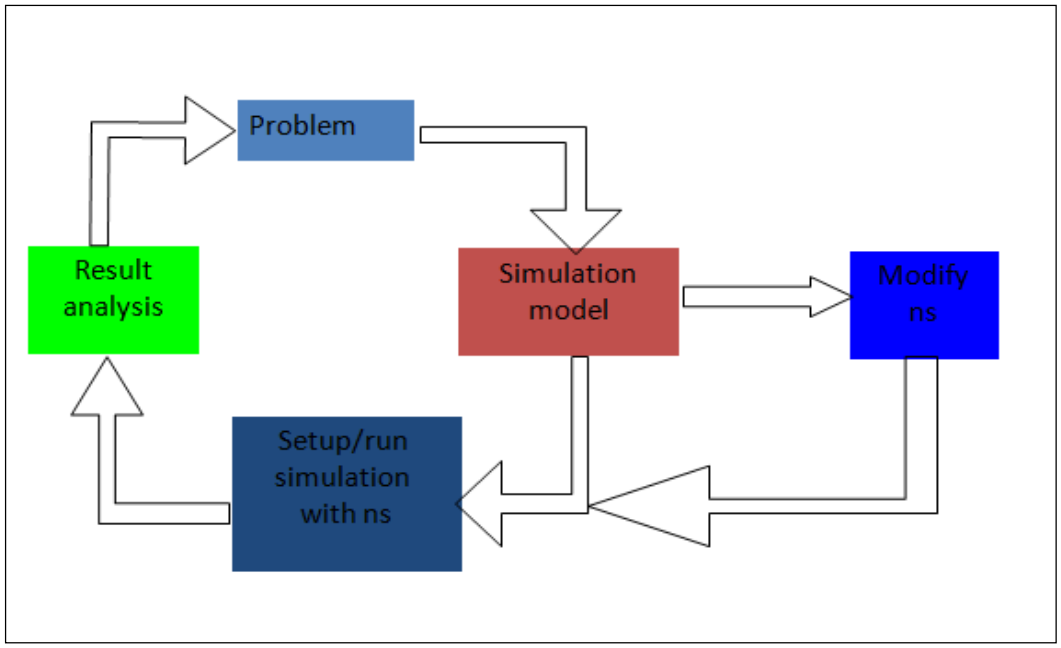


Figure: NS working procedure

OTcl (object variant of Tcl) and C++ share class hierarchy. Back end is C++ event scheduler that implements most of the protocols and those are fast to run and easier to control. Front end is oTCL that creates scenarios and extensions to C++ protocols. oTCL isfast to write and change. TclCL is glue library that makes it easy to share functions, variables, etc

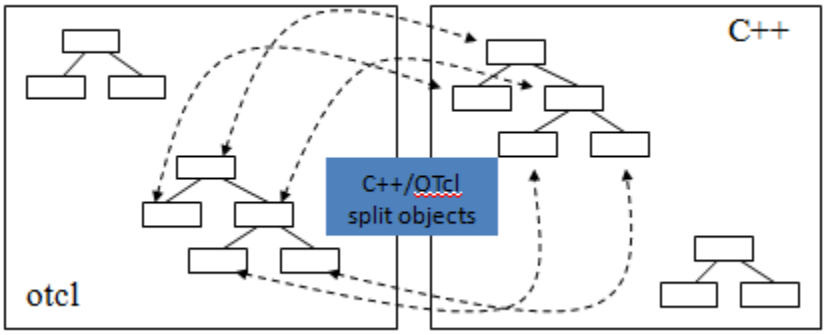


Figure: Otcl and C++: The Duality

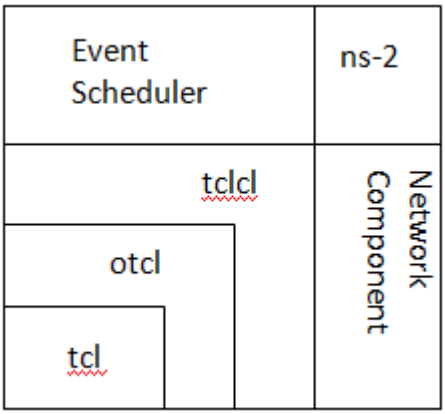
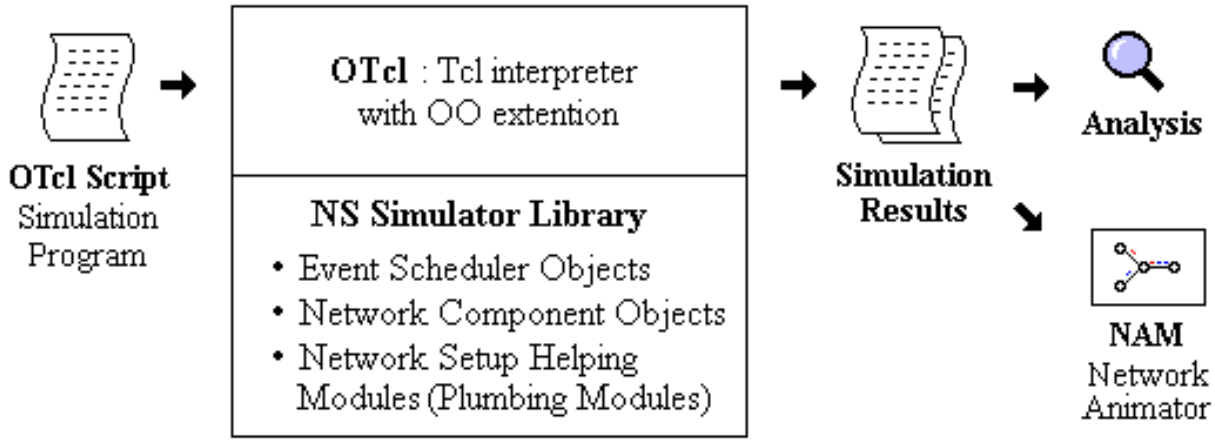


Figure: TCL Interpreter with extents

3.4.3 NS Input and output:



4. Discussion:

So far we have some basic idea of simulation, network simulation, network simulator and various components of NS. We has also discussed about tcl and its interaction with other components of NS. In the next lab we are going to simulate a very simple network. Before coming to the next lab all students are advised to learn about the syntax of tcl. In the later part of the document we provide a brief introduction of tcl syntax.

Tcl Basics

Tcl:

- **Tcl** is a very simple, open-source-licensed scripting language (programming language).
- Tcl provides basic language features such as variables, procedures, and control
- It runs on almost any modern OS, such as Unix

Scripting language:

- A programming language to write **scripts ,**programs
- ✓ written for a software environment
- ✓ that automate the execution of tasks which could alternatively be executed one by one by a human operator.
- **Example: Scripting languages for**
  - ✓ **Job control languages and shells**
  - ✓ **web browsers**
  - ✓ **Text processing**

Basic Tcl Language Features:

; or New line	**statement separator**
\	statement continuation if last character in line
#	comments out rest of line
var	simple variable
var(index)	associative array variable
$var	variable substitution
[expr 1+2]	command substitution
"hello $a"	quoting with substitution
{hello $a}	quoting with no substitution (deferred substitution)

System Interaction open:

- **open** item **[access [perms]]**
- Open item (a file, serial port)

- If a new file is created, its permission is set (default 0666).

r	**Read only. File must exist.**
r+	Read and write. File must exist.
w	Write only. Truncate if exists.
w+	Read and write. Truncate if exists.
a	Write only. File must exist. Access position at end.
a+	Read and write. Access position at end.

Intoducing a tcl script:

```
# code starts

set name "Andreas"
set number 3.4

puts $name
puts $number

unset name

puts $name

# code ends
```

Basic Tcl syntax:

variables:

```
set x 10
puts “x is $x”
```

functions and expressions:

```
set y [pow x 2]
set y [expr x*x]
```

control flow:

```
if {$x > 0} { return $x } else {
        return [expr -$x] }
while { $x > 0 } {
        puts $x
        incr x –1
```

}

Control Statements:

- Break
- Continue
- exit [return Code]
  - Terminate the process, returning return Code (an integer which defaults to 0) to the system as the exit status.
- for start test next body
  - Looping command where start, next, and body are Tcl command strings
  - test is an expression string to be passed to expr command.
  - For {set i 0} {$i > 0} {set i [expr $i+1] }{
    puts "$i \n"

}

procedures:

```
proc pow {x n} {
    if {$n == 1} { return $x }
        set part [pow x [expr $n-1]]
    return [expr $x*$part]
    }
```

Creating Procedure:

```
proc test {} {
    set a 43
    set b 27
    set c [expr $a + $b]
    set d [expr [expr $a - $b] * $c]
    for {set k 0} {$k < 10} {incr k} {
        if {$k < 5} {
            puts "k < 5, pow= [expr pow($d, $k)]"
        } else {
            puts "k >= 5, mod= [expr $d % $k]"
        }
    }
}
test
```

Islamic University of Technology (IUT)
Department of Computer Science and Engineering (CSE)
CSE 4606: Computer Networks Lab

Experiment No. 2 and 3

Introduction to NS programming and Network Performance Analysis

1. Objectives:

To understand -

- How to perform NS programming using OTCL
- Creating a very simple network topology for simulation
- Understanding how to perform Tracing and
- Understanding the structure of a trace file

2. Organization:

- Students have to understand the lab material and submit the report (hard copy) individually.
- Labs will be performed in Lab 5 of CSE Department
- In case you do not understand anything feel free to contact the course teacher at sakhawat@iut-dhaka.edu.

3. Theory:

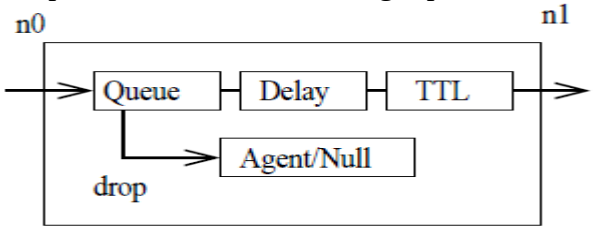
3.1 Steps of a NS simulation

- Define the scenario to simulate:
    1. Create the simulator object
    2. {Turn on tracing}
    3. Setup the network nodes {and links }
    4. Setup the routing mechanism
    5. Create transport connections
    6. Setup user applications
    7. Schedule data transmission
    8. Stop the simulation
- Execute the OTcl script in a Linux shell: > ns example.tcl
- Extract the results from the trace files: awk, xgraph, nam, matlab,perl script etc . .

3.2 Creating a Tcl scenario:

- NS simulation starts with the command:
  set ns [new Simulator]

- ns is now an instance of the class Simulator, so we can use its methods and its attributes.
- To define trace files with the data that needs to be collected from the simulation, we have to create these files using the command open:
  ```
  # open the trace file
  set traceFile [open out.tr w]
  $ns trace-all $traceFile
  # open the Nam trace file
  set namFile [open out.nam w]
  $ns namtrace-all $namFile
  ```
- To schedule an event:
  ```
  $ns at time "event"
  ```
- To terminate a Tcl scenario, we need to call the exit command:
  ```
  $ns at 12.0 "exit 0"
  ```
- Usually, we declare a finish procedure to dump the traces and to close the files:
  ```
  proc finish {} {
  global ns traceFile namFile
  $ns flush-trace
  close $traceFile
  close $namFile
  exec nam out.nam &
  exit 0 }
  ```
- Similarly, we need to declare explicitly to NS when to call the finish function in order to stop the simulation:
  ```
  $ns at 12.0 "finish"
  ```
- The simulation can then begin with the last command in the Tcl script:
  ```
  $ns run
  ```
- Creation of nodes and links in NS:
  - To create a node: `set n0 [$ns node]`
  - To create a given number nodeNb of nodes:
    ```
    for {set i 1} {$i <= $nodeNb} {incr i} {
    set n($i) [$ns node] }
    ```
- To create a link between two nodes, we need to specify the parameters of the link:
  ```
  $ns simplex-link/duplex-link $n0 $n1 "bandwidth"Mb "delay"ms "queue_type"
  ```
- A duplex link in NS is represented as two parallel simplex-links.
- A simplex-link has the following representation in NS:



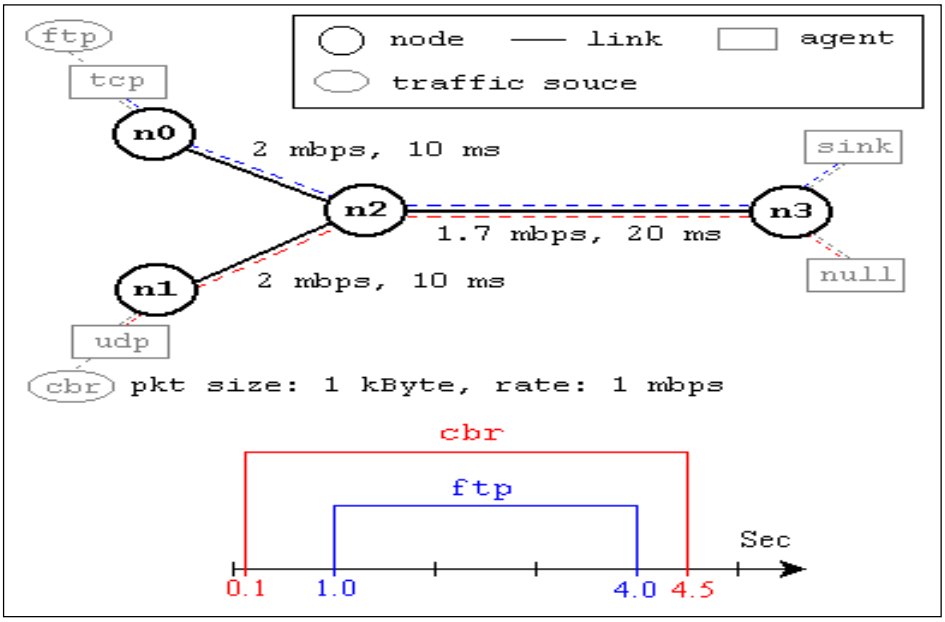
  - Output queue of a node is represented as input queue of the corresponding link.
  - There are different queue types: Drop tail, Stochastic Fair queueing (SFQ), . .
  - Packet overflow is implemented by sending dropped packets to the Null agent.
  - TTL object computes the Time To Live for each received packet.
  - To set the queue size of the ingress nodes of a link to some limit:
    ```
    $ns queue-limit $n0 $n1 20
    ```

- By default, the queue limit is set to 50. All the default values can be found and modified at ***. . . /ns-**/tcl/lib/ns-default.tcl***
- Next step is to create transport agents, to attach them to corresponding nodes, and to associate them to each other. We need also to configure their different parameters.
- For a TCP agent:
  set tcp [new Agent/TCP]
  $ns attach-agent $n(0) $tcp
  set sink [new Agent/TCPSink]
  $ns attach-agent $n(1) $sink
  $ns connect $tcp $sink
- For a UDP agent:
  set udp [new Agent/UDP]
  $ns attach-agent $n(0) $udp
  set null [new Agent/Null]
  $ns attach-agent $n(1) $null
  $ns connect $udp $null
- Next, we need to create the applications, to attach them to corresponding transport agents, and to configure their parameters:
- To setup a FTP application e.g., we need a TCP transport agent:
  set ftp [new Application/FTP]
  $ftp attach-agent $tcp
- To setup a CBR application e.g., we need a UDP transport agent:
  set cbr [new Application/Traffic/CBR]
  $cbr attach-agent $udp
- Before calling the finish procedure, we need to schedule the start time and the stop time for each data sources:
  $ns at 10.0 "$cbr start"
  $ns at 20.0 "$cbr stop"

3.3 Summary: Generic Script Structure

```
Set ns [new Simulator]
# [Turn on tracing]
# Create topology
# Setup packet loss, link dynamics
# Create:
#   - protocol agents
#   - application and/or setup traffic sources
# Post-processing procedures
# Start simulation
```

3.4 Simple Simulation Example 1:



TCL Script Step 1

```
#Create a simulator object
set ns [new Simulator]
#Open the All trace file
Set f [open out.tr w]
$ns trace-all $f
```

TCL Script Step 2

```
#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail
```

TCL Script Step 3

```
#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10
```

TCL Script Step 4

```
#Setup a TCP connection
set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
```

```
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
```

TCL Script Step 5

```
#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
```

TCL Script Step 6

```
#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"
```

TCL Script Step 7

```
#Define a 'finish' procedure
proc finish {} {
        global ns f
        $ns flush-trace
        #Close the trace file
        close $f
                exit 0
}
```

TCL Script Step 8

```
# Trace Congestion Window and RTT
set file [open cwnd_rtt.tr w]
$tcp attach $file
$tcp trace cwnd_
$tcp trace rtt_
```

#Run the simulation
$ns run

3.5 Lab Task:

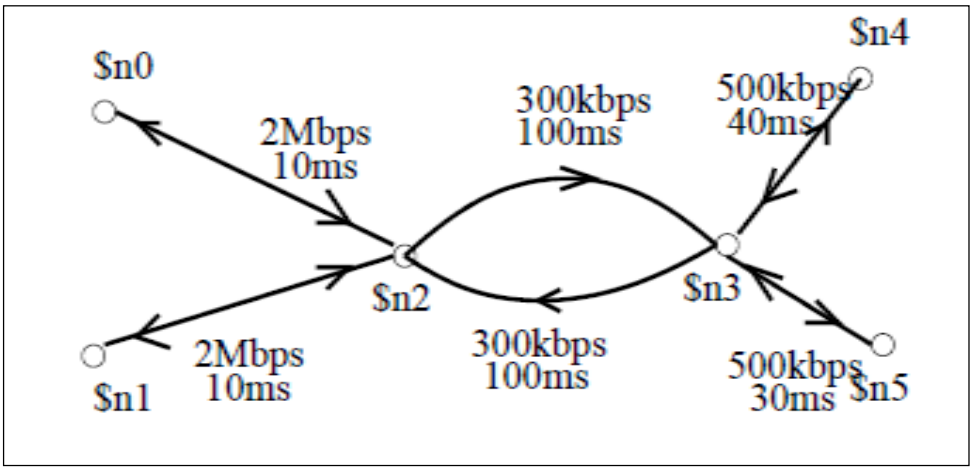
3.5.1 Task1: Simulate the given network in example 1 and understand every sintgle steps in the given solution.

3.5.2 Task2:

Observe the trace file and try to understand what actually NS is tracing. For better understanding the section 3.6 of the document gives a brief overview on the trace files and trace format.

3.5.3 Task3:

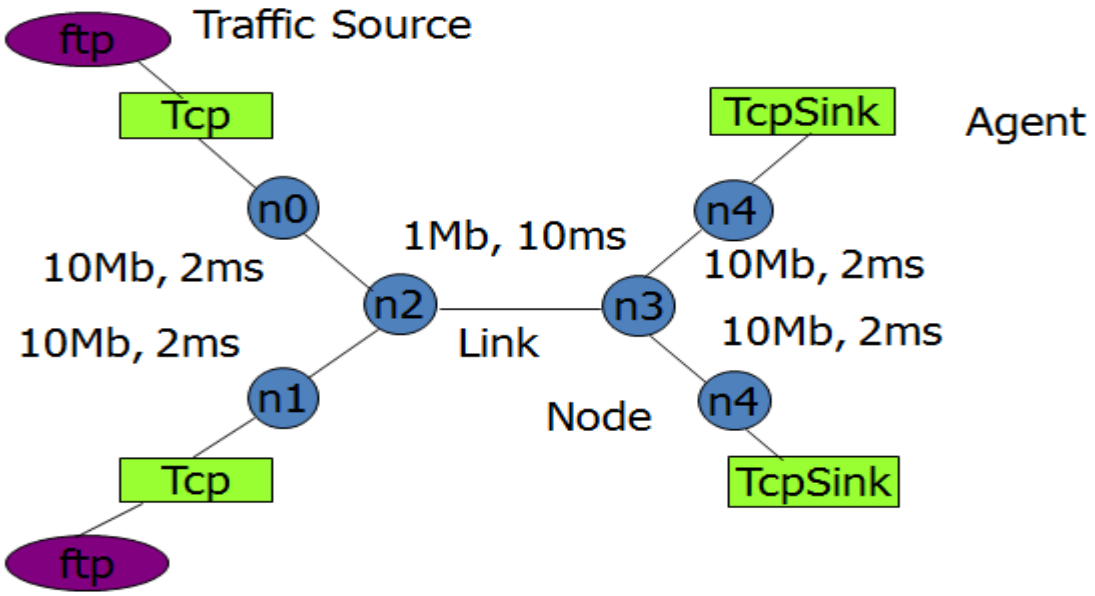
Write the Tcl script for the following network shown below. There is an FTP application running at node n0 and sending traffic to node n4. Packets have size of 512 bytes. There is a CBR application running over a UDP protocol at node n1 and sending traffic to node n5 at the rate of 0.01 Mb. Packets have size of 1000 bytes.



3.5.4 Task 4: Simulate the following network in NS2 and perform the following task.

a. Generate two UDP traffic from node 0 to node 4, and three TCP traffic from node 3 to node 1.

b. Calculate the network throughput and Plot the Time vs Total network throughput in a graph



3.6 The trace file:

• Turn on tracing on specific links

$ns_ trace-queue $n0 $n1

• Each line in the trace file is in the format:

< event > < time > < from > < to > < pkt − type > < pkt − size > < flags >
< fid > < src > < dst > < seq > < uid >

• Trace example:

+ 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
- 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
r 1.00234 0 2 cbr 210 ------- 0 0.0 3.1 0 0

4. Discussion:

Students are expected to finish the first two lab tasks (3.5.1 and 3.5.2) in lab2 and rest of the two tasks should be completed in lab 3. Before coming to lab3 all the students are advised to take a look to the text processing tools like Awk, Perl, grep utility to process/analyze the trace file to calculate the network throughput and fairness of the network.

Islamic University of Technology (IUT)
Department of Computer Science and Engineering (CSE)
CSE 4606: Computer Networks Lab

Experiment No. 4 and 5

Simulating complex topology , link failure and network dynamics:

1. Objectives:

To understand -

- Exposed to more complex oTcl constructs (e.g., *for* loops).
- Exposed to types of events different from "start" and "stop".
- Observe a DV routing protocol in action.

2. Organization:

- Students have to understand the lab material and submit the report (hard copy) individually.
- Labs will be performed in Lab 5 of CSE Department
- In case you do not understand anything feel free to contact the course teacher at sakhawat@iut-dhaka.edu.

3. Theory:

3.1 Inserting Errors:

- Creating Error Module
  - set loss_module [new ErrorModel]
  - $loss_module set rate_ 0.01
  - $loss_module unit pkt
  - $loss_module ranvar [new RandomVariable/Uniform]
  - $loss_module drop-target [new Agent/Null]
- Inserting Error Module
  - $ns lossmodel $loss_module $n0 $n1

3.2 Setup Routing:

- Unicast
  $ns rtproto <type>
  <type>: Static, Session, DV, cost, multi-path
- Multicast
  $ns multicast (right after [new Simulator])
  $ns mrtproto <type>
  <type>: CtrMcast, DM, ST, BST
- Other types of routing supported: source routing, hierarchical routing

3.3 Network Dynamics:

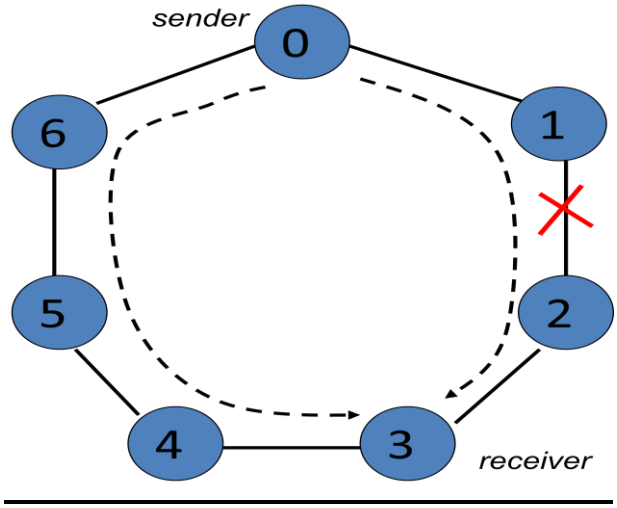
- Link failures
  - Hooks in routing module to reflect routing changes
- Four models
  $ns rtmodel Trace <config_file> $n0 $n1
  $ns rtmodel Exponential {<params>} $n0 $n1
  $ns rtmodel Deterministic {<params>} $n0 $n1
  $ns rtmodel-at <time> up|down $n0 $n1

Parameter list
[<start>] <up_interval> <down_interval> [<finish>]

3.4 Lab Task:

3.4.1 Task1:

Simulate a network according to the following topology where all the seven nodes form ring. Create a UDP agent and attach it to node n(0) and create a CBR traffic source and attach it to udp0 Create a Null agent (a traffic sink) and attach it to node n(3). Connect the traffic source with the traffic sink. After the simulation start, down the link between node 1 and 2 and again up the link and observe the result in the trace file. Submit a report of your observation.



3.4.2Task2:

Calculate the network throughput and Plot the Time vs Total network throughput in a graph before and after the link failure.

3.5 Skeletal script of the task:

```
#Create a simulator object
set ns [new Simulator]
#Tell the simulator to use dynamic routing
$ns rtproto DV
#Define a 'finish' procedure
proc finish {} {
      global ns
      $ns flush-trace
      exit 0
}
#Create seven nodes(use for loops to create multiple nodes)
#Create links between the nodes so that it will form a ring
topology
#Create a UDP agent and attach it to node n(0)
…….
# Create a CBR traffic source and attach it to udp0
…….
#Create a Null agent (a traffic sink) and attach it to node
n(3)
…….
#Connect the traffic source with the traffic sink
…….
#Schedule events for the CBR agent and the network dynamics
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

3.6 Analyzing the trace file:

The following snapshot is from the trace file of the given simulation.

```
r 0.984 0 1 cbr 500 ------- 1 0.0 3.0 94 158
r 0.987 2 3 cbr 500 ------- 1 0.0 3.0 89 153
r 0.988 1 2 cbr 500 ------- 1 0.0 3.0 92 156
r 0.989 0 1 cbr 500 ------- 1 0.0 3.0 95 159
r 0.992 2 3 cbr 500 ------- 1 0.0 3.0 90 154
r 0.993 1 2 cbr 500 ------- 1 0.0 3.0 93 157
r 0.994 0 1 cbr 500 ------- 1 0.0 3.0 96 160
r 0.997 2 3 cbr 500 ------- 1 0.0 3.0 91 155
r 0.998 1 2 cbr 500 ------- 1 0.0 3.0 94 158
r 0.999 0 1 cbr 500 ------- 1 0.0 3.0 97 161
r 1.002 2 3 cbr 500 ------- 1 0.0 3.0 92 156
r 1.004 0 1 cbr 500 ------- 1 0.0 3.0 98 162
r 1.007 2 3 cbr 500 ------- 1 0.0 3.0 93 157
r 1.009 0 1 cbr 500 ------- 1 0.0 3.0 99 163
r 1.010056 1 0 rtProtoDV 7 ------- 0 1.1 0.2 -1 164
r 1.012 2 3 cbr 500 ------- 1 0.0 3.0 94 158
r 1.012056 2 3 rtProtoDV 7 ------- 0 2.1 3.2 -1 165
r 1.014 0 1 cbr 500 ------- 1 0.0 3.0 100 166
r 1.019 0 1 cbr 500 ------- 1 0.0 3.0 101 167
r 1.020112 0 6 rtProtoDV 7 ------- 0 0.2 6.1 -1 170
r 1.022112 3 2 rtProtoDV 7 ------- 0 3.2 2.1 -1 171
r 1.022112 3 4 rtProtoDV 7 ------- 0 3.2 4.1 -1 172
```

To analyze the trace file you re suggest to use grep utility or perl script. Here we provide you with some basics of grep utility

3.6.1 Basic usage of Grep:

- Command-line text-search program in Linux
- Some useful usage:
  - Grep ‘word’ filename # find lines with ‘word’
  - Grep –v ‘word’ filename # find lines without ‘word’
  - Grep ‘^word’ filename # find lines beginning with ‘word’
  - Grep ‘word’ filename > file2 # output lines with ‘word’ to file2
  - ls -l | grep rwxrwxrwx # list files that have ‘rwxrwxrwx’ feature
  - grep -v '^[0-9]‘ filename # find lines beginning with any of the numbers from 0-9
  - Grep –c ‘word’ filename # find lines with ‘word’ and print out the number of these lines
  - Grep –i ‘word’ filename # find lines with ‘word’ regardless of case

3.6.2 One example perl script:

```
#!/usr/bin/perl

$src = "lab2.tr";

#initialization
$total_sent=0;
$total_recv=0;
$total_failed=0;
$T_delay=0;
$num_pkt=0;
$max_node = 6;

for ($i = 0; $i < $max_node; $i++) {
           $sent[$i] = 0;
           $recv[$i] = 0;
}

#collecting trace content into a variable
open(IN,$src);

@line=<IN>;
#print @line;

#handaling each event seperately

foreach $i (@line){
     #print $i;
     @input=split(' ',$i);
     #print $input[0]."\n";

     #calculating sending event
       if($input[0] eq '-' && $input[4] eq 'tcp') {

                    $pck_id=$input[11];

                @temp_src = split('\.',$input[8]);
                     @temp_dst = split('\.',$input[9]);

                $src_ip[$pck_id] = $temp_src[0];
                   $src_port[$pck_id] = $temp_src[1];

                $dst_ip[$pck_id] = $temp_dst[0];
                $dst_port[$pck_id] = $temp_dst[1];

                if($input[2] eq $src_ip[$pck_id]){
                     $sent[$temp_src[0]]++;
                     $start[$pck_id] = $input[1];
                             $status[$pck_id] = 0;

                                $total_sent++;
                        print "Pckt:",$pck_id,"  Transmitted
at:",$start[$pck_id],"  src-ip:port-> ",
```

```
$src_ip[$pck_id],":",$src_port[$pck_id],"  dst-ip:port->",
$dst_ip[$pck_id],":", $dst_port[$pck_id],"\n";
                }
        }

      #calculating receiving event
        if($input[0] eq 'r' && $input[4] eq 'tcp') {

                        $pck_id=$input[11];

                @temp_src = split('\.',$input[8]);
                        @temp_dst = split('\.',$input[9]);

                $src_ip[$pck_id] = $temp_src[0];
                   $src_port[$pck_id] = $temp_src[1];

                $dst_ip[$pck_id] = $temp_dst[0];
                $dst_port[$pck_id] = $temp_dst[1];

                if($input[3] eq $dst_ip[$pck_id]){
                        $recv[$temp_src[0]]++;
                        $finish[$pck_id] = $input[1];
                                $status[$pck_id] = 1;
                        $delay[$pck_id]=$finish[$pck_id]-
$start[$pck_id];

                                        $total_recv++;
                        print "Pckt:",$pck_id,"  Received
at:",$finish[$pck_id],"  src-ip:port-> ",
$src_ip[$pck_id],":",$src_port[$pck_id],"  dst-ip:port->",
$dst_ip[$pck_id],":", $dst_port[$pck_id],"\n";
                }
        }

}

#performance summary


$max = 0;
$min = 1000;

print "\n\nPer Source Survay:\n\n";
for ($i = 0; $i <=$max_node; $i++) {
           if ($sent[$i] > 0 ) {
               print " Source Node:",$i," ->
Throughput(recved/sent): ",$recv[$i],"/", $sent[$i], "=",
$recv[$i]/$sent[$i],"\n";
           }
}
print "\n\nAt a glance :\n";
print "Total_sent: ",$total_sent, "\nTotal_recv:", $total_recv,
"\nTotal_recv/Total_sent:", $total_recv/$total_sent,"\n";

close(IN);
```

4. Discussion:

Students are expected to finish the first task (3.4.1) in lab4 and rest of the task should be completed in lab 5. Before coming to lab5 all the students are advised to take a look to the text processing tools like Awk, Perl, grep utility to process/analyze the trace file to calculate the network throughput and fairness of the network.

CSE 4606
Computer Networks Lab

Experiment No: 6
Name of the Experiment:
Local Area Networks

1. Experiment Objectives:

- Learn how create LAN and simulate an example scenario.

2. Simulating LANs with ns:

The characteristics of the local area networks (LAN) are inherently different from those of point-to-point links. A network consisting of multiple point-to-point links cannot capture the sharing and contention properties of a LAN.

2.1 TCL configuration:

ns simulator simulates three the levels related to a local area network: link layer protocols (such as the ARQ (Automatic Repeat reQuest protocol), the MAC protocol (e.g. Ethernet or token ring) and the physical channel.

The basic way to define a LAN through which a group of nodes is connected is by the command

```
set lan [$ns newLan <arguments>]
```

There are seven arguments:

1. A group of nodes, e.g. `"$n3 $n4 $n5"`,
2. the delay,
3. the bandwidth
4. a link layer type (e.g. "LL"),
5. the interference queue type, e.g. "QueueDrop Tail",
6. the MAC type (e.g. `"Mac/Csma/Cd"` or `"Mac/802_3"`)
7. the Channel type (e.g. "Channel")

As an example, consider the following figure, where n3, n4 and n5 are put on a common LAN. This means that TCP packets destined to node n4 arrive also to node n5, TCP

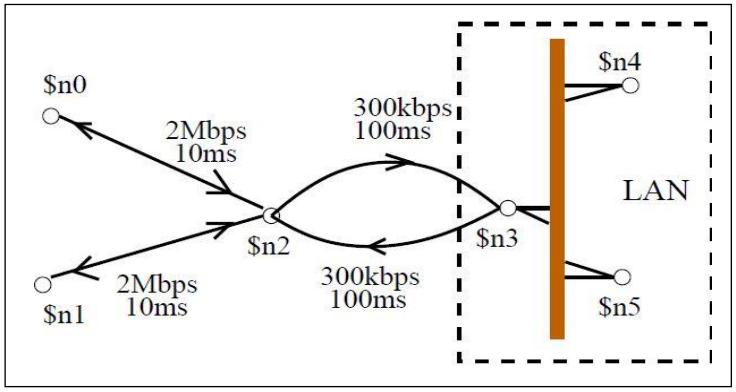


Figure: A LAN Example

acknowledge sent by de n5, and UDP packets destined to node n5 also arrive at node n4.

The script file is then same as the example form the second lab. But here, we replace the commands

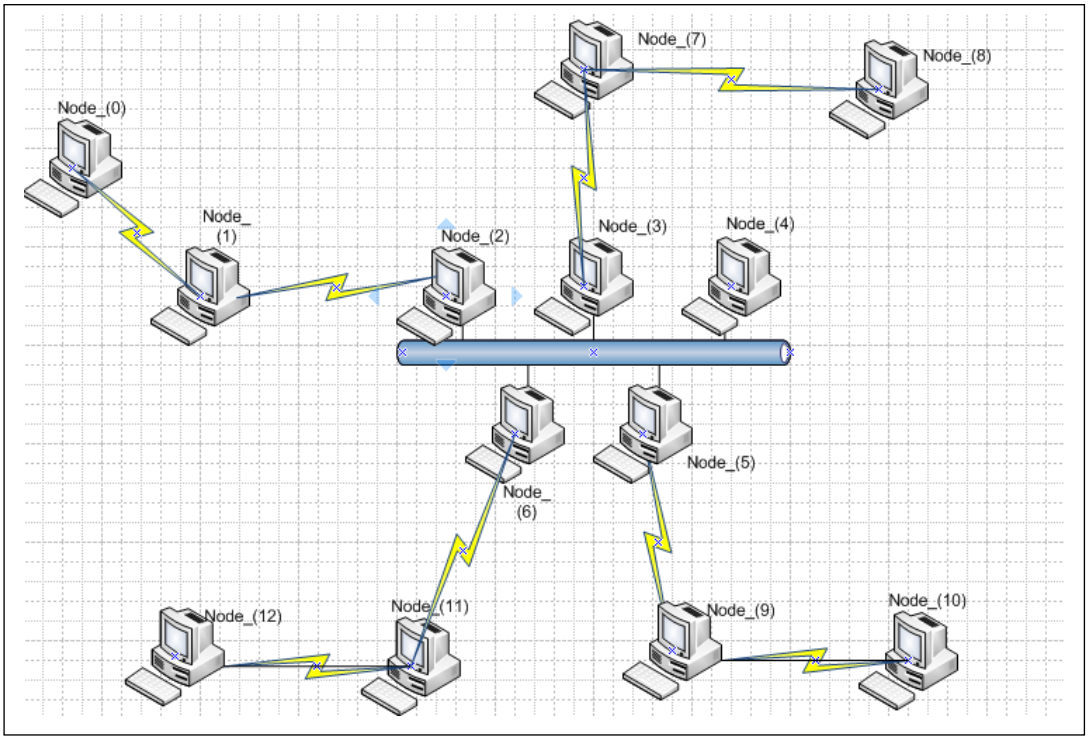
```
$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail
```

by the command

```
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]
```

3. Assignment:

Write a Tcl script for the following figure according to these instructions and perform some simulations with it:



- All access links are full-duplex and have 30 Mbit/s of bandwidth and 2 ms of delay.
- The lan has 10 Mbit/s of bandwidth and 6 ms of delay.
- The queuing discipline of all links is DropTail (FIFO). Limit the maximum queue size at all link to 67 packets

Now do the followings:

- Generate two UDP traffic from node 8 to node 10 , three UDP traffic from node 12 to node 8 and four UDP traffic from node 0 to node 11
- Plot the Time vs Total network throughput in a graph
- Draw the Time vs Average delay graph
- Plot Time vs Per-node throughput
- Plot Time vs Per-node dalay
- Plot Time vs total network reliability
- Calculate the fairness index of overall network.

4. Discussion:

For more on LAN read the following additional documentations:

- NS Simulator for Beginners-Chapter 8
- The ns manual-Chapter 14

CSE 4606
Computer Networks Lab

Experiment No: 7
Name of the Experiment:
Wireless Network

1. Experiment Objectives:

- Learn how to simulate a wireless network.
- Learn how to configure it and run a simple wireless scenario.

2. Simulating Wireless Networks:

The wireless model essentially consists of the MobileNode at the core,with additional supporting features that allows simulations of multi-hop ad-hoc networks, wireless LANs etc. A mobile node consists of network components:

Link Layer (LL)
Interface Queue (IfQ)
the MAC layer
the PHY layer: the wireless channel that the node transmit and receive signals from

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 500m depicted in following figure.

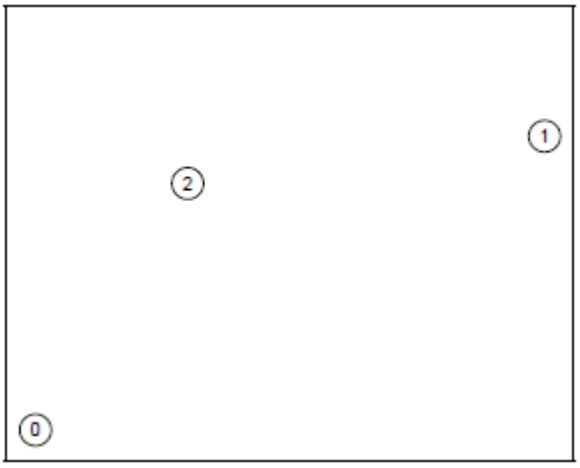


Figure: Example of a 3 nodes ad-hoc network

At the beginning of a wireless simulation, we need to define the type for each of these network components.
Additionally, we need to define other parameters like:
the type of antenna, the radio-propagation model, the type of ad-hoc routing protocol used by mobilenodes, MAC protocol etc.

2.2 Writing the TCL script:

At the beginning of the script we need to specify some basic parameters for the simulation, providing information for different layers. This is done as follows:

- Define options:

```
# Define options #
set val(chan) Channel/WirelessChannel ;        # channel type
set val(prop) Propagation/TwoRayGround ;       # radio-propagation model
set val(ant) Antenna/OmniAntenna ;             # Antenna type
set val(ll) LL ;                               # Link layer type
set val(ifq) Queue/DropTail/PriQueue ;         # Interface queue type
set val(ifqlen) 50 ;                           # max packet in ifq
set val(netif) Phy/WirelessPhy ;               # network interface type
set val(mac) Mac/802_11 ;                      # MAC type
set val(rp) DSDV ;                             # ad-hoc routing protocol
set val(nn) 2 ;                                # number of mobilenodes
```

These parameters are used in configuring 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
```

2.2.1 Defining NS simulator and trace file:

To create a NS simulator object write down the following command:

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

For defining a trace file write the following code:

```
set tracefd [open simple.tr w]
$ns_ trace-all $tracefd
```

2.2.2 Create Topology Object:

The topology is a Topography object that you must create. For defining the topology write down the following:

```
set topo  [new Topography]      ;# Create a Topography object
$topo load_flatgrid  500  500   ;# Make a 500x500 grid topology
```

2.2.3 GOD (General Operations Director) object:

As wireless nodes can MOVE, their distances from one another change over time. To keep track of their positions in the topology grid, we need to create a General Operations Director (GOD) object. Command for it is:

create-god $val(nn)

God is the object that is used to store global information about the state of the environment, network or nodes that an omniscent observer would have, but that should not be made known to any participant in the simulation."

Currently, the God object stores:

- ✓ the total number of mobilenodes
- ✓ a table of shortest number of hops required to reach from one node to another.

2.3 Creating nodes, setting position and node movement:

Next actually create the mobilenodes as follows:

```
for { set j 0 } { $j < $opt(nn)} {incr j} {
set node_($j) [ $ns_ node ]
$node_($i) random-motion 0 ;# disable random motion
}
```

The start-position and future destinations for a mobilenode may be set by using the following APIs:

```
$node set X_ <x1>
$node set Y_ <y1>
$node set Z_ <z1>
$ns at $time $node setdest <x2> <y2> <speed>
```

At $time sec, the node would start moving from its initial position of (x1,y1) towards a destination (x2,y2) at the defined speed.

2.4 Rests:

Rest of the parts of tcl script is same as the wired connection. That means creating traffic flows between nodes, telling the nodes when the simulation ends etc.

3. Assignment:

Create a simple wireless network with following configuration:

- No. of nodes: 5
- MAC type: 802.11
- Routing Protocol: DSR
- Length and width of the topography :700x700
- Create TCP connection between nodes.
- At time 50s Node_(1) starts to move towards node_(0) at speed 15 m/s.
- At time 100s Node_(1) starts to move away from node_(0) 15 m/s.
- Run simulation and stop simulation at 150s.

4. Discussion:

Readers who want to learn more are encouraged to read the following additional documentations:

- The ns manual
- Official ns-2 website (http://www.isi.edu/nsnam/ns/)
- Wireless tutorial by Marc Greis

CSE 4606
Computer Networks Lab

Experiment No: 8
Name of the Experiment:
Trace File Analysis and Performance Evaluation of Wireless Networks

1. Experiment Objectives:

- Learn how create LAN and simulate an example scenario.

2. Wireless Trace File Analysis :

A wireless trace file is not similar to the wired one. It has following fields:

ACTION: [s|r|D]: s -- sent, r -- received, D -- dropped
WHEN : the time when the action happened
WHERE: the node where the action happened
LAYER: AGT – transport l/application ayer,
RTR -- routing,
LL -- link layer (ARP is done here)
IFQ -- outgoing packet queue (between link and mac layer)
MAC -- mac,
PHY -- physical
flags:
SEQNO: the sequence number of the packet
TYPE: the packet type
cbr -- CBR data stream packet
DSR -- DSR routing packet (control packet generated by routing)
RTS -- RTS packet generated by MAC 802.11
ARP -- link layer ARP packet
SIZE: the size of packet at current layer, when packet goes down, size increases, goes up size decreases
[a b c d]: a -- the packet duration in mac layer header
b -- the mac address of destination
c -- the mac address of source
d -- the mac type of the packet body
flags:
[.....]: [source node ip : port_number
destination node ip (-1 means broadcast) : port_number
ip header ttl
ip of next hop (0 means node 0 or broadcast)
]

2.1 Example of Trace Interpretation:

s 76.000000000 _98_ AGT --- 1812 cbr 32 [0 0 0 0] ------- [98:0 0:0 32 0]

Application 0 (port number) on node 98 sent a CBR packet whose ID is 1812 and size is 32 bytes, at time 76.0 second, to application 0 on node 0 with TTL is 32 hops. The next hop is not decided yet.

r 0.010176954 _9_ RTR --- 1 gpsr 29 [0 ffffffff 8 800] ------- [8:255 -1:255 32 0]

The routing agent on node 9 received a GPSR broadcast (mac address 0xff, and ip address is -1, either of them means broadcast) routing packet whose ID is 1 and size is 29 bytes, at time 0.010176954 second, from node 8 (both mac and ip addresses are 8), port 255 (routing agent).

```
Trace beginning:

s 0.029290548 _1_ RTR  --- 0 message 32 [0 0 0 0] ------- [1:255 -1:255 32 0]
s 1.119926192 _0_ RTR  --- 1 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]
M 10.00000 0 (5.00, 2.00, 0.00), (20.00, 18.00), 1.00
s 10.000000000 _0_ AGT  --- 2 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 10.000000000 _0_ RTR  --- 2 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 12.941172739 _1_ RTR  --- 3 message 32 [0 0 0 0] ------- [1:255 -1:255 32 0]
s 13.000000000 _0_ AGT  --- 4 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 13.000000000 _0_ RTR  --- 4 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 13.242656084 _0_ RTR  --- 5 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]
s 19.000000000 _0_ AGT  --- 6 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 19.000000000 _0_ RTR  --- 6 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 24.799296167 _1_ RTR  --- 7 message 32 [0 0 0 0] ------- [1:255 -1:255 32 0]
s 27.719583723 _0_ RTR  --- 8 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]
```

Figure: A sample trace file

2.2 Another Event-> Movement (M):

A movement command has the form –

M 10.00000 0 (5.00, 2.00, 0.00), (20.00, 18.00), 1.00

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

3. TCP Performance Comparison with other ad-hoc routing:

In this lab you will compare the TCP performance over different ad-hoc routing protocols, such as-

- TCP over DSR
- TCP over AODV
- TCP over TORA

4. Assignment:

Write a Tcl script with following configurations and perform some simulations with it:

- No. of nodes: 8
- MAC type: 802.11
- Routing Protocol: DSR/AODV/TORA
- Length and width of the topography :1000x1000
- Create TCP connection between nodes.
- Insert node movement s at random time.
- Run simulation and stop simulation at 150s.

Now do the followings:

- Write a perl script to calculate the Total network throughput, Average delay, Per-node throughput, Per-node delay and total network reliability.
- Use it for all three routing protocols.
- Compare the Time vs Total network throughput for different types of routing protocol using a graph.
- Compare Time vs Average delay graph for different types of routing protocol using a graph.
- Compare Time vs Per-node throughput for different types of routing protocol using a graph.
- Compare Time vs Per-node delay for different types of routing protocol using a graph.
- Compare Time vs total network reliability for different types of routing protocol using a graph.
- Calculate the fairness index of overall network for different types of routing protocol using a graph.

5. Discussion:

For more details on different routing protocols go through following reference:

- NS Simulator for Beginners-Chapter 9

Islamic University of Technology (IUT)
Department of Computer Science and Engineering (CSE)
CSE 4606: Computer Networks Lab

Experiment No. 09

Configuration of Domain Name System (DNS) Server

1. Objectives:

To understand

- What is DNS server?
- How does DNS server work?
- How to configure DNS server in linux environment?

2. Organization:

- Students have to understand the lab material and submit the report (hard copy) individually.
- Labs will be performed in Lab 5 of CSE Department
- In case you do not understand anything feel free to contact the course teacher at sakhawat@iut-dhaka.edu.

3. Problem Discussion:

3.1 Introduction:

Domain Name Server (DNS) is an internet service which basically provides the translation of the domain names into IP addresses. Because it's very hard to remember the IP addresses. As we know that internet is huge and it depends upon the IP addresses. So to make things easier we use DNS servers for translating from domain name to IP address. DNS is a database, no other database is that much big and there are millions of queries daily. So to make life easier we use DNS and also to organize all the addresses.

The Domain Name System (DNS) is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates domain names meaningful for users to the numerical IP addresses needed for the purpose of locating computer services and devices worldwide. An often-used analogy to explain the Domain Name System is that it serves as the phone book for the Internet by translating human-friendly computer hostnames into IP addresses. For example, the domain name www.example.com is translated into the addresses 192.0.43.10 by the DNS Server. Unlike a phone book, the DNS can be quickly updated, allowing a service's location on the network to change without affecting the end users, who continue to use the same host name. Users take advantage of this when they recite

meaningful Uniform Resource Locators (URLs) and e-mail addresses without having to know how the computer actually locates the services.

3.2 Berkeley Internet Name Domain (BIND):

BIND is an open implementation of DNS protocol and is available free for use. The major components of the Domain Name System, including,

- Domain Name System server
- Domain Name System resolver library
- Tools for managing and verifying the proper operation of the DNS server

The basic configuration of the BIND depends upon different file which are,

- named.conf
- named.rfc1912.zones
- morenet.fwd(Forward Zone)
- morenet.rev **(**Reverse Zone**)**

named.conf file is basically primary file, in this file we basically put the IP of the DNS server on which it is listening the requests from the network and the address is 172.16.80.6/26. It also contain, the range which is allowed to query the DNS server in our case it is 192.16.80.0/16 and it includes zone file that has definitions of all zones including forward zone and reverse zone. morenet.fwd contains the forward zone definition and morenet.rev contains the definition of reverse zone.

3.3 Used ports and protocols:

- 53/UDP
- 53/TCP

3.4 Purpose:

- Translating domain name to correspondent ip address.
- Vice versa
- Providing email-routing facilities for mail servers

3.5 Types:

- Primary DNS server
- Secondary DNS server

3.6 Package:

- BIND (Berkeley Internet Name Domain)
- BIND attached a server called “named” whose purpose is to resolve ip address from the hostname.

3.6 Static DNS Server:

- Used for organizations to host their own website in a static ip address.
- Caching servers are used only for reference but for regular organizational use, static servers are beneficial for finding source information.
-

3.7 Configuration environment

Server name:	DNS
Environment:	Red Hat Enterprise Linux 9.0

Server IP address: 192.168.10.6
Package name: bind
Configuration file: /etc/named.conf

3.8 Step by step configuration Process

Step-1: Check if bind package is installed or not. If not installed then install it at first.

```
rpm -qa | grep bind
rpm -ivh bind-9.2.*
```

Step-2: Edit the named.conf file.

```
vi /etc/named.conf
```

Add the following lines:

```
options {
directory "/var/named/";
};
```

Edit the following portion by providing domain name and IP

```
zone "iutcse.com" IN {
        type master;
        file "iutcse.fr";
        allow-update { none; };
};

zone "10.168.192.in-addr.arpa" IN {
        type master;
        file "iutcse.rv";
        allow-update { none; };
};
```

Step-3: Copy the named.conf file to /var/named/chroot/etc/ directory.

```
mkdir –p /var/named/chroot/etc/
cp /etc/named.conf/var/named/chroot/etc/ named.conf
```

Step-4: Copy localhost.zone and named.local file.

```
cd /var/named/
mkdir –p chroot/var/named/
cplocalhost.zone /var/named/chroot/var/named/iutcse.fz
cpnamed.local/var/named/chroot/var/named/iutcse.rv
```

Step-5: Edit the iutcse.fr file.

```
vi iutcse.fr
```

```
$TTL 86400
@       IN SOA      ns1.iutcse.com.     root@iutcse.com. (
                42      ; serial (d. adams)
                3H      ; refresh
                15M     ; retry
                1W      ; expiry
1D )        ; minimum
IN NS       ns1.iutcse.com.
IN NS       iutcse.com.
        IN A    192.168.10.6
```

Step-6: Edit the iutcse.rv file.

```
vicse.rz

$TTL 86400
@       IN  SOA     ns1.iutcse.com.     root@iutcse.com. (
1997022700 ; Serial
28800 ; Refresh
14400 ; Retry
3600000 ; Expire
86400 ) ; Minimum

        IN  NS      ns1.iutcse.com.
IN  NS      iutcse.com.
6       IN  PTR     ns1.cse.com.
```

Step-7:(server+client): Edit the following,

```
vi /etc/resolv.conf
searchiutcse.com
nameserver 192.168.10.6
```

Step-8: Enter the command,

```
vi /etc/hosts
```

Add the following line

```
192.168.10.6        ns1.iutcse.com      ns1
192.168.10.6        iutcse.com          ns1
```

Step-9:

```
chgrp named /var/named/chroot/var/named/iutcse.*
chmod +x /var/named/chroot/var/named/iutcse.*
chown named /var/named/chroot/etc/named.conf
```

Step-10:

```
named-checkconf –t /var/named/chroot
named-checkzone cse.com /var/named/chroot/var/named/iutcse.fr
named-checkzone cse.com /var/named/chroot/var/named/iutcse.rv
```

Step-11:
service network restart
service named restart
chkconfig named on
digns1.iutcse.com
dig -x 192.168.3.100

4. Discussion:

Students are suggested to study the following references to obtain a better understanding:

- Documentation for Linux commands
- E-books on Network administration
- Web links to different Linux resources

Islamic University of Technology (IUT)
Department of Computer Science and Engineering (CSE)
CSE 4606: Computer Networks Lab

Experiment No. 10

Configuration of Mail Server

1. Objectives:

To understand

- What is a mail server?
- How does mail server work?
- How to configure mail server in linux environment?

2. Organization:

- Students have to understand the lab material and submit the report (hard copy) individually.
- Labs will be performed in Lab 5 of CSE Department
- In case you do not understand anything feel free to contact the course teacher at sakhawat@iut-dhaka.edu.

3. Problem Discussion:

3.1 Introduction:

A mail server is the computerized equivalent of your friendly neighborhood mailman. Every e-mail that is sent passes through a series of mail servers along its way to its intended recipient. Although it may seem like a message is sent instantly, zipping from one PC to another in the blink of an eye. The reality is that a complex series of transfers takes place. Without this series of mail servers, you would only be able to send emails to people whose email address domains matched your own - i.e., you could only send messages from one example.com account to another example.com account.

3.2 How Sendmail Works:

Sendmail can handle both incoming and outgoing mail for your domain.

- Client connects through MTA with the mail server.
- Local MTA requests for query to the desired MX, which is stored in DNS.
- Local MTA opens the port TCP/IP 25 for that MX.
- Both mail servers exchanged mail using SMTP.

3.2.1 Incoming Mail:

Usually each user in your home has a regular Linux account on your mail server. Mail sent to each of these users (username@alinsoft.com) eventually arrives at your mail

server and sendmail then processes it and deposits it in the mailbox file of the user's Linux account.

Mail isn't actually sent directly to the user's PC. Users retrieve their mail from the mail server using client software, such as Microsoft's Outlook or Outlook Express that supports either the POP or IMAP mail retrieval protocols.
Linux users logged into the mail server can read their mail directly using a text-based client, such as mail, or a GUI client, such as Evolution. Linux workstation users can use the same programs to access their mail remotely.

3.2.2 Outgoing Mail:

The process is different when sending mail via the mail server. PC and Linux workstation users configure their e-mail software to make the mail server their outbound SMTP mail server.

If the mail is destined for a local user in the alinsoft.com domain, then sendmail places the message in that person's mailbox so that they can retrieve it using one of the methods above.

If the mail is being sent to another domain, sendmail first uses DNS to get the MX record for the other domain. It then attempts to relay the mail to the appropriate destination mail server using the Simple Mail Transport Protocol (SMTP). One of the main advantages of mail relaying is that when a PC user A sends mail to user B on the Internet, the PC of user A can delegate the SMTP processing to the mail server.

3.3 /etc/mail/local-host-names File:

When sendmail receives mail, it needs a way of determining whether it is responsible for the mail it receives. It uses the /etc/mail/local-host-names file to do this. This file has a list of hostnames and domains for which sendmail accepts responsibility. For example, if this mail server was to accept mail for the domains alinsoft.com and another-site then the file would look like this:

alinsoft.com
another-site.com

3.4 /etc/mail/access File:

You can make sure that only trusted PCs on your network have the ability to relay mail via your mail server by using the /etc/mail/access file. That is to say, the mail server will relay mail only for those PCs on your network that have their e-mail clients configured to use the mail server as their outgoing SMTP mail server.

The /etc/mail/access file has two columns. The first lists IP addresses and domains from which the mail is coming or going. The second lists the type of action to be taken when mail from these sources or destinations is received. Keywords include RELAY, REJECT, OK (not ACCEPT), and DISCARD.

3.5 Protocol Used:

 # MUA(Mail User Agent)/LDA(local Delivery Agent): The user application that a user sends/reieves mail. eg.,pine, elm, outlook

 # MTA(Mail Transport Agent): User agents give the mail to the transfer agent, who may pass it onto another agent, or possibly many other transfer agents. eg., sendmail, postfix, qmail, exim, Microsoft Exchange Server, WinGate

 # MDA(Mail Delivery Agent): The server agent that accepts email from MTA, and places into users mailbox. eg., procmail, maildrop

SMTP(Simple Mail Transfer Protocol): MUAs and MTAs are use this portocol for sending emails.

 # POP3(Post Office Protocol 3): Used for retrieving email and, in certain cases, leaving a copy of it on the server.

IMAP(Internet Message Access Protocol): Used for coordinating the status of emails (read, deleted, moved) across multiple email clients. With IMAP, a copy of every message is saved on the server, so that this synchronisation task can be completed. MUAs can use POP3/IMAP protocol to send and receive emails on the server.

3.6 Sendmial vs Qmail:

Why Sendmail:

- Very popular and widely used MTA
- Has given rise to a vast pool of experienced sendmail administrators and consultants.
- Configuration process is more sophisticated than qmail.
- Provides lots of services than qmail and other MTAs.

Why not qmail?

- Not widely used and not matured than sendmail.
- Provides less services than sendmail
- Most of all, configuration steps are not easy to handle
- Not widely documented as sendmail.

3.7 Configuration environment

Server name:	Mail
Environment:	Red Hat Enterprise Linux 9.0
Server IP address:	192.168.10.3
DNS machine IP:	192.168.10.3
Package name:	sendmail
Configuration file:	/etc/mail/sendmail.mc

3.8 Step by step configuration Process

Configuration Steps:

 # dnl(delete new line)

 # Dovecot configuration:-
step:-

1. vim /etc/dovecot.conf
protocols=imap imaps pop3 pop3s

2. service dovecot restart

3. chkconfig dovecot on

#sendmail configuration:-
steps:-

1. rpm -qa|grep sendmail
rpm -qa|grep m4

If send mail is not installed, then
rpm -ivh sendmail
rpm -ivh sendmail-cf
rpm -ivh m4

2. vi /etc/mail/sendmail.mc

At line 116
dnl DAEMON_OPTIONS(`Port=smtp,Addr=127.0.0.1, Name=MTA')dnl

At line 179
LOCAL_DOMAIN(‘alinsoft.com’)

3. vi /etc/mail/local-host-names

192.168.3.102
mail.alinsoft.com
alinsoft.com
ns1.alinsoft.com

4. vi /etc/mail/access

192.168.3.102	RELAY
@alinsoft.com	RELAY
mail.alinsoft.com	RELAY
spam.com	REJECT

7. makemap hash /etc/mail/access.db < /etc/mail/access

8. m4 /etc/mail/sendmail.mc > /etc/mail/sendmail.cf

9. Edit your .fz file

```
        IN      MX    10          mail. alinsoft.com.
        IN      NS                ns1. alinsoft.com.
        IN      NS                alinsoft.com.
        IN      A                 192.168.3.102
mail    IN              CNAME     ns1. alinsoft.com.
```

10. Edit your .rz file

```
                IN      NS              ns1. alinsoft.com.
        IN      NS                alinsoft.com.
        IN      MX    10          mail. alinsoft.com.
        IN      NS                192.168.3.102
102     IN      PTR               mail. alinsoft.com.
```

11. service network restart
12. service named restart
13. service sendmail restart
14. chkconfig sendmail on
15. Send a mail
You can use outlook

4. Discussion:

Students are suggested to study the following references to obtain a better understanding:

- Documentation for Linux commands
- E-books on Network administration
- Web links to different Linux resources

Experiment No. 11

Configuration of File Transfer Protocol (FTP) server and Proxy Server

1. Objectives:

To understand

- What are FTP and Proxy l servers?
- How do FTP and Proxy server work?
- How to configure FTP and Proxy servers in linux environment?

2. Organization:

- Students have to understand the lab material and submit the report (hard copy) individually.
- Labs will be performed in Lab 5 of CSE Department
- In case you do not understand anything feel free to contact the course teacher at sakhawat@iut-dhaka.edu.

3. Discussion on FTP server:

3.1 Introduction:

File Transfer Protocol (FTP) is a standard network protocol used to transfer files from one host or to another host over a TCP-based network, such as the Internet. FTP is built on a client-server architecture and uses separate control and data connections between the client and the server

3.2 Why FTP?

- FTP stands for File Transfer Protocol so FTP mainly transfer file from one location to another location in the network.
- Used as a file server in a network to store files as a repository.
- To exchange files from one machine to another.

Basically we download and upload files from web but to have a complete file server, FTP is needed. FTP provides the following facilities:

- File transfer in a network
- Permits anonymous users to upload files
- Provides a particular FTP service for a number of users.
- Restricts a number of users to use FTP service.
- Confine all local users to work on their home directory (/var/ftp).

3.3 Configuration environment

Server name:	FTP
Environment:	Red Hat Enterprise Linux 9.0

Server IP address: 192.168.10.1
Package name: vsftpd
Configuration file: /etc/vsftpd/vsftpd.conf

3.4 Step by step configuration Process of FTP server

Step-1:

Enter the command to check if vsftpd package is installed or not,

```
rpm –qa|grep vsftpd
```

If not installed then enter the following command,

```
yum install vsftpd* -y
```

Step-2:

Start the service,

```
Service vsftpd start
Chkconfig vsftpd on
```

From any client machine, enter the command

```
ftp 192.168.10.1
```

Step-3:

If we want to give the facility of uploading files to anonymous users then enter the following commands,

```
cd /var/ftp
mkdir incoming
chown root.ftp incoming
chmod 730 incoming
ls –lh incoming
```

Now open the configuration file,

```
vi /etc/vsftpd/vsftpd.conf
```

Add the following lines at the end of the file

```
anon_upload=YES
chown_upload=YES
chown_username=daemon
anon_umask=077
```

Save it and restart the service,

```
Service vsftpd restart
```

Step-4:

If we want to block some particular users form using ftp service then enter the following commands,

vi /etc/vsftpd/ftpusers

Add the usernames (for example mustakim, shafi, ismail, wadud)like the following,

tanveer
mustakim
shafi
ismail
wadud

Now restart the service,

servicevsftpd restart

Step-5:

If we want to give permissions to use ftp to some particular users, do the following,

vi /etc/vsftpd.user_list

Add the usernames like before,

atiq
sami
tausif
imtiaz

Now save the file and open the main configuration file to add some lines,

vi /etc/vsftpd/vsftpd.conf

Add the following line just beneath “userlist_enable=YES” line,

userlist_deny=NO

Save it and restart the service,

servicevsftpd restart

Step-6:

If we want to confine all the local users within their local home directory, then do the following,

vi /etc/vsftpd/vsftpd.conf

Add the following line,

chroot_local_user=YES

Now reload the service,

Service vsftpd reload

We can also confine some particular users in their home directory. To do so, open the main configuration file and do the following,

vi /etc/vsftpd/vsftpd.conf

chroot_list_enable=YES (uncomment it erasing # at the beginning)
chroot_list_file=/etc/vsftpd.chroot_list

Save it and add the users to the /etc/vsftpd.chroot_list like before

tanveer
mustakim
ismail
wadud
shafi

Save it and reload the service,

Service vsftpd reload

4. Discussion on Proxy server:

4.1 Introduction:

In computer networks, a proxy server is a server (a computer system or an application) that acts as an intermediary for requests from clients seeking resources from other servers. A client connects to the proxy server, requesting some service, such as a file, connection, web page, or other resource available from a different server and the proxy server evaluates the request as a way to simplify and control its complexity. Today, most proxies are web proxies, facilitating access to content on the World Wide Web.

4.2 Any other purpose?

* Filtering
* Monitoring
* Anonymous browsing
* Access to country specific contents
* Translation

4.3 Why Squid?

* Freeware
* Can increase performance upto 4 times
* Relays partial requests

4.4 Configuration environment

Server name:	Proxy Server
Environment:	Red Hat Enterprise Linux 9.0
Server IP address:	192.168.128.149
Package name:	Squid
Configuration file:	/etc/squid/squid.conf

4.5 Step by step configuration Process of Proxy server

Step-1:
Check if bind package is installed or not. If not installed then install it at first.

```
rpm -qa squid
```

If Squid is already installed, you will get a response similar to:

```
squid-2.5.STABLE6-3.4E.12
```

If Squid isn't installed, then you can use Yum to install it. Thanks to Yum the installation is quite easy.Just type at a command line:

```
yum install squid
```

Step-2:
Backup configuration file

```
cp /etc/squid/squid.conf /etc/squid/squid.conf.bck.02052007
```

Step-3:
Open the configuration file

```
vi /etc/squid/squid.conf
```

few small changes should be made. You will need to either find and uncomment entries, or modify existing uncommented lines in the squid configuration file. Use your favorite text editor or a text find to quickly locate these lines:

```
visible_hostname machine-name
http_port 3128
cache_dirufs /var/spool/squid 1000 16 256
cache_access_log /var/log/squid/access.log
```

In the acl section near the bottom add:

```
acl intranet 192.168.0.0/24
http_access allow intranet
```

Let me explain what each of these six lines means:

visible_hostname – Create this entry and set this to the hostname of the machine. To find the hostname, use the command hostname. Not entering a value may cause squid to fail as it may not be able to automatically determine the fully qualified hostname of your machine.
http_port 3128 – Uncomment this line but there is no need to edit it unless you want to change the default port for http connections.
cache_dirufs /var/spool/squid 1000 15 256 – Uncomment this line. You may want to append a zero to the value 100 which will make the cache size 1000MB instead of 100MB. The last two values stand for the default folder depth the cache will create on the top and subdirectories respectively. They do not need modification.
cache_access_log – Uncomment this line. This is where all requests to the proxy server will get logged.

acl intranet 192.168.0.0/24 – This entry needs to be added. It should correspond to whatever your local network range is. For example, if your Fedora server is 192.168.2.5 then the entry should be acl intranet 192.168.2.0/24
http_access allow intranet – This allows the acl named intranet to use the proxy server. Make sure to put allow directives above the last 'http_access deny all' entry, as it will overide any allow directives below it.

Step-4:
Turning on squid.Enable the proper run levels:

```
chkconfig squid on
```

Start the service:

```
service squid start
```

Verify that squid isrunning:

```
service squid status
```

Note, if you have problems starting squid, open a separate shell and run:

```
tail -f /var/log/messages
```

Then start the squid service in your original window:

```
service squid start
```

Step-5:
Configuring the clients
If you are using Firefox or Mozilla you will need to add the proxy server as follows:

```
Go to Preferences>Network>Settings
```

Add the name of your new proxy server and port 3128 to the http proxy field (under manual configuration).
Open a shell to your proxy server so you can observe the log file being written to. Use tail, as before:

```
tail -f /var/log/squid/access.log
```

Now surf the web through your proxy server. You should see entries flying by in real time as you surf different http addresses.

5. Discussion:

Students are suggested to study the following references to obtain a better understanding:

- Documentation for Linux commands
- E-books on Network administration
- Web links to different Linux resources

CSE 4606: Computer Networks Lab

Experiment No. 12

Configuration of SAMBA and DHCP Server

1. Objectives:

To understand

- What are SAMBA and DHCP servers?
- How do SAMBA and DHCP server work?
- How to configure SAMBA and DHCP servers in linux environment?

2. Organization:

- Students have to understand the lab material and submit the report (hard copy) individually.
- Labs will be performed in Lab 5 of CSE Department
- In case you do not understand anything feel free to contact the course teacher at sakhawat@iut-dhaka.edu.

3. Discussion on SAMBA server:

3.1 Introduction:

Samba sets up network shares for chosen UNIX directories (including all contained subdirectories). These appear to Microsoft Windows users as normal Windows folders accessible via the network. For example: home directories would have read/write access for all known users, allowing each to access their own files. However, they would still not have access to the files of others unless that permission would normally exist.

3.2 Purpose of SAMBA server?

* Samba is an open source software package that mimics a Windows server.
* Its purpose is to offer an alternative to expensive, unstable Windows servers.
* Samba provides file and print services for various Microsoft Windows clients

3.3Why use SAMBA:

* It is free! NO LICENSING COSTS
* It is reliable.
* It is available for many platforms. IRIX, Solaris, Linux, HP-UX, SCO UnixWare,etc..
* It is secure: There are no security holes known to exist in the current release of Samba

* It is relatively easy to administer yet very customizable. Samba can be set up to function as anything from a simple print server to a complex Windows Domain Controller.

3.4 Configuration environment

Server name:	Mail
Environment:	Red Hat Enterprise Linux 9.0
Server IP address:	192.168.128.151
Package name:	SAMBA
Configuration file:	/etc/samba/smb.conf

3.5 Step by step configuration Process of FTP server

Step-1:
Check whether samba is installed or not.

```
rpm -qa samba
```

If samba is not currently installed, browse to where the file is located and perform the following operation:

```
rpm -i samba.rpm
```

Step-2:
Open the configuration file

```
vi /etc/samba/smb.conf
```

Make the following changes

```
[global]
workgroup = HOME
server string = testing samba
hosts allow = 192.168. 127.
log file = /var/log/samba/%m.log
security = user
encrypt passwords = yes
smbpasswd file = /etc/samba/smbpasswd
socket options = TCP_NODELAY SO_RCVBUF=8192
SO_SNDBUF=8192

[Downloads]
comment = Downloads
path = /home/windisk/Downloads
browseable = yes
writable = yes
public = yes
read only = no
```

[homes] comment = My Home Directory
browseable = yes
writable = yes
public = yes
read only = no

[printers]
path = /var/spool/samba
public = yes
guest ok = yes
printable = yes
browseable = yes
writable = yes
read only = no

Step-3:

Create samba user(first create the password file from linux passwords)

```
cat /etc/passwd | mksmbpasswd.sh > /etc/samba/smbpasswd
```

Change the permissions so that only root has read and write permissions.

```
chmod 600 /etc/samba/smbpasswd
```

However, this file only copies over Linux users to samba users. It doesn't copy over their passwords, as well. Therefore, use the following command to set each samba user's password:

```
smbpasswd username
```

Step-4:

Now that everything has been configured, the final step is to start the samba service. Samba runs in the background as a linux daemon. Therefore, it can be controlled by typing:

```
servicesmb start
servicesmb stop
servicesmb restart
```

Step-5:

Connect To a Samba Resource. For testing purposes, we can connect to

```
localhost.smbclient //localhost/Downloads
```

On Windows machine if you open My Network Places, and browse to workgroup, linux machine should be listed. When trying to access it, prompt for password and username will apear. This can be any of the samba users you previously created. Once logged in, you should see a listing of all available resources available.

4. Discussion on Dynamic Host Congiuration Protocol (DHCP):

4.1 Introduction:

The Dynamic Host Configuration Protocol (DHCP) automatically assigns IP addresses and other network configuration information such as subnetmask, broadcast address e.t.c. to computers on a network. A client configured for DHCP will send out a broadcast request to the DHCP server requesting an address. The DHCP server will then issue a "lease" and assign it to that client. The time period of a valid lease can be specified on the server.

DHCP reduces the amount of time required to configure clients and allows one to move a computer to various networks and be configured with the appropriate IP address, gateway and subnet mask. DHCP servers may assign a "static" IP address to specified hardware.

A DHCP Server assigns IP addresses to client computers. This is very often used in enterprise networks to reduce configuration efforts. All IP addresses of all computers are stored in a database that resides on a server machine.

4.2 DHCP Assignment:

1. *Lease Request*: Client broadcasts request to DHCP server with a source address of 0.0.0.0 and a destination address of 255.255.255.255. The request includes the MAC address which is used to direct the reply.
2. *IP lease offer*: DHCP server replies with an IP address, subnet mask, network gateway, name of the domain, name servers, duration of the lease and the IP address of the DHCP server.
3. *Lease Selection*: Client receives offer and broadcasts to al DHCP servers that will accept given offer so that other DHCP server need not make an offer.
4. The DHCP server then sends an acknowledgment to the client. The client is configured to use TCP/IP.
5. *Lease Renewal*: When half of the lease time has expired, the client will issue a new request to the DHCP server.

4.4 Configuration environment

Server name:	DHCP
Environment:	Red Hat Enterprise Linux 9.0
Server IP address:	192.168.10.4
Package name:	dhcpd
Configuration file:	/etc/dhcpd.conf

4.5 Step by step configuration Process of Proxy server

Step-1:
Edit the file /etc/dhcpd.conf
vi /etc/dhcpd.conf

Now edit the following lines
subnet 192.168.10.0 network 255.255.255.0
default gateway
options routers 192.168.0.4;
optionsubnetmask 255.255.25.0;
option domain name "iutcse.com";
option domain-name-servers 192.168.0.4;
option time offset 1800 (erase the – sign before it);

Range configuration DHCP
range 192.168.10.10 192.168.10.30

Step-2:
Edit the file /etc/dhcpd.conf for MAC dependent ip fixing:
vi /etc/dhcpd.conf

host hostname{
hardware Ethernet 00:AB:E3:67:53:90;
fixed address 192.168.10.15;
}

Step-3:
Do the following
servicedhcpd start
servicechkconfig on

5. Discussion:

Students are suggested to study the following references to obtain a better understanding:

- Documentation for Linux commands
- E-books on Network administration
- Web links to different Linux resources