

National University of Computer and Emerging Sciences



Laboratory Manuals
for
Computer Networks - Lab
(CL -3001)

Course Instructor	Sir Nauman Moazzam Hayat
Lab Instructor(s)	Mr. Usama Khan Mr. Muzammil Muneer
Section	BCS-6B
Semester	Spring 2023

*Department of Computer Science
FAST-NU, Lahore, Pakistan*

Lab Manual 09

Objective:

- Run basic TCL commands
- Introduction to NS-2
- Simulate a basic topology

Basics of the TCL Language

Tcl and Otcl Programming Tutorial

Tcl(Tool Command Language) is a language with a very simple syntax and it allows easy integration with other languages. The characteristics of this language are:

1. It allows a faster development
2. It provides a graphical interface
3. It is compatible with many platforms
4. It is easy to use

Basics of Tcl and Otcl Programming

1. Assigning a value to a variable depends on the “set” command; “set b 0” is equivalent to “b=0”, the set command is irrespective of datatype i.e “int b=0” is equivalent to “set b 0” and “double pi=3.142” is equivalent to “set pi 3.142”. In Tcl variables are not typed. A variable can be an integer, double or string depending on the value you assign to it.
2. If we want to use a value assigned to a variable we should use a \$ sign before it.

C++	Tcl
<pre>int a=20; int b; b=a;</pre>	<pre>set a 20 set b \$a</pre>

3. To display an output the command “puts” is used.

C++	Tcl
<pre>int x=67; cout<<x;</pre>	<pre>set x 67 puts “x \$x”</pre>

4. To take input in a variable gets stdin is used

C++	Tcl
<pre>int x; cin>>x;</pre>	<pre>set x [gets stdin]</pre>

- The sign # starts a commented line that is not part of the program, so the tcl interpreter will not execute this line.
- A mathematical operation is done using the expression command. For example we wish to assign to a variable x the sum of values of a and b, we should write `set x [expr $a + $b]`

C++	Tcl
<code>x=a+b</code>	<code>set x [expr \$a + \$b]</code>

- The structure of an if command is as follows:

```

if {expression} {
    <execute some commands>
} else {
    <execute some commands>
}

```

C++	Tcl
<pre> If(x<2) cout<<"x is lesser"; else cout<<"x is greater"; </pre>	<pre> if {\$x < 2} { puts "x is lesser" } else { puts "x is greater" } </pre>

The if command can be nested within other if and else that can appear in the <execute some commands> part. When testing equality we should use `==`. Inequality is written with `!=`.

- Loops have the following form:

```

for {set i 0} {$i < 5} {incr i} {
    <execute some commands>
}

```

C++	Tcl
<pre> for (int x=0; x<5; x++) { ... } </pre>	<pre> for {set x 0} {\$x < 5} {incr x} { ... } </pre>

Steps for compiling tcl code

- Save your tcl code on Desktop **filename.tcl**
- Open terminal (go to Desktop), write **ns filename.tcl**

In Lab Statement 1:

[5]

Write a TCL Script that will take a number from user (user will give you a number greater than 4):

- a. If the number is even then you will print all even integers (starting from 4) until that number
- b. If the number is odd then you will print all the odd integers (starting from 3) until that number

INTRODUCTION TO NS-2

Network Simulator-2

An NS Simulator starts with the following command:

```
set ns [new Simulator]
```

In order to have output files for visualizations (**nam** files) we need to create files using open command:

```
# opennam file
setnf [open out.nam w]
$ns namtrace-all $nf
```

The termination of the program is done using the finish procedure:

```
# define a finish procedure
proc finish {} {
    global ns nf
        $ns flush-trace
    close $nf
    execnamout.nam
    exit 0
}
```

We should call the procedure finish at the end of the ns program and specify at what time the termination should occur. For example:

```
$ns at 5.0 "finish"
```

This command will be used to call finish at 5.0 sec.

The simulation can begin by using the following command. This command should be the last line of the code.

```
$ns run
```

Demo Codes:

You are provided with two demo codes in which two nodes are linked to each other and data is being sent from one node to other node. One code uses **UDP and CBR for data transfer** and other **code uses TCP and FTP for data transfer**. Go through the slides and codes and implement the In Lab statement given below.

In lab Statement 2:

[15]

Write tcl script to implement the simple network shown in the figure below

1. This network consists of **4 nodes (n0, n1, n2, n3)**
2. The duplex links between n0 and n2, and n1 and n2 **have 2 Mbps of bandwidth and 10 ms of delay.**
3. The duplex link between n2 and n3 has **1.7 Mbps of bandwidth and 20 ms of delay.**
4. Each node uses a **DropTail queue**, of which the maximum size is 10. You will have to orient the nodes as shown in the diagram below.
5. A "tcp" agent is attached to n1, and a connection is established to a tcp "sink" agent attached to n3.
6. A tcp "sink" agent generates and sends ACK packets to the sender (tcp agent) and frees the received packets.
7. A "udp" agent that is attached to n0 is connected to a "null" agent attached to n3. A "null" agent just frees the packets received.
8. A **"ftp"** and a **"cbr" traffic** generator are attached to "tcp" and "udp" agents respectively, and the "cbr" is configured to generate packets having size of **1 Kbytes at the rate of 100 packets per second.**
9. FTP will control the traffic automatically according to the throttle mechanism in TCP.
10. The **traffic flow of UDP must be colored red** and traffic flow of **TCP must be colored blue.**
11. The "cbr" is set to start at 0.1 sec and stop at 4.5 sec,
12. "ftp" is set to start at 0.5 sec and stop at 4.0 sec.

