TAPI DIPLOMA ENGINEERING COLLEGE, SURAT

COMPUTER ENGENEERING DEPARTMENT

SUBJECT: Computer Networking (4340703)

VIRTUAL LAB

Basics of Network Simulation

INTRODUCTION

Network simulation has become an integral part of most research works in the field of Computer Networks. Whether it is for understanding the behaviour of existing protocols, or to determine the performance of a new protocol, one doesn't often get access to real network devices. This gap has been filled up by network simulation to a large extent.

In this experiment we will get familiar with one of the most popular open source network simulators, Network Simulator version 2 (ns2). The experiment makes one familiar with fundamental concepts of ns2, and provides step-by-step instructions on how to install it.

THEORY

Objectives

After completing this experiment you will be able to:

- Learn the basic concepts about open source network simulator NS-2, and how to download, install and work with NS-2
- Defining the different agents and their applications like TCP, FTP over TCP, UDP, CBR over UDP
- Identifying and solving typical errors encountered during installation of NS-2

Time Required

Around 3.00 hours

Introduction

Network Simulator version 2 (NS-2) is discrete event packet level simulator. The network simulator covers a very large number of application of different kind of protocols of different network types consisting of different network elements and traffic models. NS-2 is a package of tools that simulates behavior of networks such as creating network topologies, log events that happen under any load, analyze the events and understand the network. The aim of this first experiment is to learn how to use NS-2, to get acquainted with the simulated objects and understand the operations of network simulation. We will also look at how to analyze the outcome of a simulation.

Platform required to run network simulator

- Unix and Unix like systems

- Linux
- Free BSD
- SunOS/Solaris
- Windows 95/98/NT/2000/XP (requires Cygwin)

Backend Environment of Network Simulator

Network Simulator is based on two languages: C++ and OTcl. OTcl is the object oriented version of Tool Command Language. While the core of NS-2 is written in C++, one uses OTcl to write simulation scripts. C++ helps in the following way:

- It helps to increase the efficiency of simulation.
- Its is used to provide details of the protocols and their operation.
- It is used to reduce packet and event processing time.

OTcl helps in the following way:

- With the help of OTcl we can describe different network topologies
- It helps us to specify the protocols and their applications
- It allows fast development
- Tcl is compatible with many platforms and it is flexible for integration
- Tcl is very easy to use and it is available in free

And of course, there is a linkage between C++ and OTcl, which allows us to run the simulation scripts.

Basics of Tcl Programming for NS-2

Network simulation with NS-2 would involve the following general steps:

1. Initialization and termination aspects of network simulator object
2. Defining the network topology: nodes, links, queues, mobility of nodes, if any
3. Defining the network traffic: creating agents and their applications
4. Setting trace for Network Animator (NAM) [optional]
5. Tracing

In this section, we provide a brief overview of the most commonly used features of NS-2. This summary has been prepared based on various tutorials on, and the manual for, NS-2. See the References section for some of the different tutorials available.

Initialization

To create a new simulator we write

1 set ns [new Simulator]

From the above command we get that a variable ns is being initialized by using the set command. Here the code [new Simulator] is a instantiation of the class Simulator which uses the reserved word new. So we can call all the methods present inside the class simulator by using the variable 'ns'.

Creating the output files

```
1  # Create the trace files
2    set tracefile [open out.tr w]
```

```
3   $ns trace-all $tracefile
 4
5  # Create the nam files
6    set namfile [open out.nam w]
7   $ns namtrace-all $namfile
```

In the above we create a output trace file 'out.tr' and a NAM visualization file 'out.nam'. But in the Tcl script they are not called by their names declared, while they are called by the pointers initialized for them such as 'tracefile' and 'namfile' respectively.The line which starts with # are commented. The next line opens the file 'out.tr' which is used for writing is declared 'w'. The next line uses a simulator method trace-all by which we will trace all the events in a particular format.

The termination program is done by using a 'finish' procedure

```
1 # Defining the 'finish' procedure'
 2
3  proc finish {} {
4       global ns tracefile namfile
5        $ns flush-trace
6        close $tracefile
7        close $namfile
8         exit 0
 9   }
```

In the above, the keyword proc is used to declare a procedure called 'finish'. The keyword global is used to tell what variables are being used outside the procedure.

flush-trace is a simulator method that dumps the traces on the respective files. The command close is used to close the trace files and the command exec is used to execute the NAM visualization. The command exit closes the application and returns zero as default for clean exit.

In ns we end the program by calling the 'finish' procedure

```
1  # End the program
2  $ns at 125.0 "finish"
```

Thus the entire operation ends at 125 seconds.To begin the simulation we will use the command

```
1 # Start the the simulation process
2  $ns run
```

Defining nodes, links, queues (topology)

Way to create a node:

```
1set n0 [$ns node]
```

In the above we created a node that is pointed by a variable n0. While referring the node in the script we use $n0. Similarly we create another node n2. Now we will set a link between the two nodes.

```
1 $ns duplex-link $n0 $n2 10Mb 10ms DropTail
```

So we are creating a bi-directional link between nodes n0 and n2 with a capacity of 10 Mb/sec and a propagation delay of 10 ms.

In NS an output queue of a node is implemented as a part of a link whose input is that node to handle the overflow at the queue. If the buffer capacity of the output queue is exceeded then the last packet arrived is dropped and here we will use a 'DropTail' option. There are other queue types such as RED (Random Early Discard) mechanism, FQ (Fair Queuing), DRR (Deficit Round Robin), SFQ (Stochastic Fair Queuing) also available.

Now we will define the buffer capacity of the queue related to the above link

```
1  # Set queue size of the link
2  $ns queue-limit $n0 $n2 20
```

So, if we summarize the above three things we get

```
01  # Create nodes
 02
03  set n0 [$ns node]
04  set n1 [$ns node]
05  set n2 [$ns node]
06  set n3 [$ns node]
07  set n4 [$ns node]
08  set n5 [$ns node]
 09
10 # Create links between the nodes
 11
12      $ns duplex-link $n0 $n2 10Mb 10ms
                DropTail
13      $ns duplex-link $n1 $n2 10Mb 10ms
                DropTail
14      $ns simplex-link $n2 $n3 0.3Mb 100ms
                 DropTail
15      $ns simplex-link $n3 $n2 0.3Mb 100ms
                 DropTail
16      $ns duplex-link $n0 $n2 0.5Mb 40ms
                DropTail
17      $ns duplex-link $n0 $n2 0.5Mb 40ms
                DropTail
 18
```

```
# Set queue-size of the link (n2-n3) to
19                    20
20 $ns queue-limit $n2 $n3 20
```

Agents and applications

TCP

TCP is used to provide reliable transport of packets from one host to another host by sending acknowledgements on proper transfer or loss of packets. Thus, TCP requires bi-directional links in order for acknowledgements to return to the source.

Now we will show how to set up tcp connection between two nodes

```
1 # Setting a TCP connection
2
3 set tcp [new Agent/TCP]
4 $ns attach-agent $n0 $tcp
5 set sink [new Agent/TCPSink]
6 $ns attach-agent $n4 $sink
7 $ns connect $tcp $sink
8 $tcp set fid_ 1
9 $tcp set packetSize_ 552
```

The command set tcp [new Agent/TCP] gives a pointer called 'tcp' to the TCP agent object of ns. The command $ns attach-agent $n0 $tcp defines the source node of TCP connection. Next the command set sink [new Agent/TCPSink] defines the destination of TCP by a pointer called 'sink'. The next command $ns attach-agent $n4 $sink defines the destination node as n4. Next, the command $ns connect $tcp $sink makes the TCP connection between the source and the destination i.e n0 and n4. When we have several flows (such as TCP, UDP) in a network, to identify these flows we set their flow ID by using the command $tcp set fid_1. In the last line we set the packet size of TCP as 552 byte. The default packet size of TCP is 1000 B.

FTP over TCP

File Transfer Protocol (FTP) is a standard mechanism provided by the Internet for transferring files from one host to another. FTP differs from other client server applications in that it establishes two connections between the client and the server. One connection is used for data transfer and other one is used for providing control information. FTP uses the services of the TCP. The well Known port 21 is used for control connections and the other port 20 is used for data transfer.

Here we will learn in how to run a FTP connection over a TCP:

```
1 # Initiating FTP over TCP
  2
3 set ftp [new Application/FTP]
4 $ftp attach-agent $tcp
```

In above,the command set ftp [new Application/FTP] gives a pointer called 'ftp' which indicates the FTP application. Next, we attach the ftp application with tcp agent as FTP uses the services of TCP.

UDP

The User datagram Protocol is one of the main protocols of the Internet protocol suite. UDP helps the host to send send messages in the form of datagrams to another host which is present in a Internet protocol network without any kind of requirement for channel transmission setup. UDP provides a unreliable service and the datagrams may arrive out of order, appear duplicated, or go missing without notice. UDP assumes that error checking and correction is either not necessary or performed in the application, avoiding the overhead of such processing at the network interface level. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system.

Now we will learn how to create a UDP connection in network simulator.

```
1 # Setup a UDP connection
2 set udp [new Agent/UDP]
3 $ns attach-agent $n1 $udp
4 $set null [new Agent/Null]
5 $ns attach-agent $n5 $null
6 $ns connect $udp $null
7 $udp set fid_ 2
```

The command set udp [new Agent/UDP] gives a pointer called 'udp' which indicates the udp agent which is a object of ns. Then the command $ns attach-agent $n1 $udp defines the source node of UDP connection. Next the command set null [new Agent/Null] defines the destination of udp by a pointer called null. The next command $ns attach-agent $n5 $null defines the destination node as n5. Next, the command $ns connect $udp $null makes the UDP connection between the source and the destination i.e n1 and n5. To identify a particular flow we mark it using the command $udp set fid_2.

Constant Bit Rate (CBR)

Constant Bit Rate (CBR) is a term used in telecommunications, relating to the quality of service.When referring to codecs, constant bit rate encoding means that the rate at which a codec's output data should be consumed is constant. CBR is useful for streaming multimedia content on limited capacity channels since it is the maximum bit rate that matters, not the average, so CBR would be used to take advantage of all of the capacity. CBR would not be the optimal choice for storage as it would not allocate enough data for complex sections (resulting in degraded quality) while wasting data on simple sections.

CBR over UDP Connection

```
1 # Setup CBR over UDP
2
3 set cbr [new Application/Traffic/CBR]
4 $cbr attach-agent $udp
```

```
5 $cbr set packetSize_ 1000
6 $cbr set rate_ 0.01Mb
7 $cbr set random_ false
```

In the above we define a CBR connection over a UDP one. Well we have already defined the UDP source and UDP agent as same as TCP. Instead of defining the rate we define the time interval between the transmission of packets in the command $cbr set rate_ 0.01Mb. Next, with the help of the command $cbr set random_ false we can set random noise in cbr traffic. We can keep the noise by setting it to false or we can set the noise on by the command $cbr set random_ 1. We can set by packet size by using the command $cbr set packetSize_. The packet size is specified in bytes.

Scheduling Events

In ns the tcl script defines how to schedule the events or in other words at what time which event will occur and stop. This can be done using the command $ns at time event. Here in our program we will schedule when the ftp and cbr traffic should start and stop.

```
1 # Scheduling the events
2
3 $ns at 0.1 "$cbr start"
4 $ns at 1.0 "$ftp start"
5 $ns at 124.0 "$ftp stop"
6 $ns at 124.5 "$cbr stop"
```

Network Animator (NAM)

When we will run the above program in ns then we can can visualize the network in the NAM. But instead of giving random positions to the nodes, we can give suitable initial positions to the nodes and can form a suitable topology. So, in our program we can give positions to the nodes in NAM in the following way

```
1 # Give position to the nodes (for NAM)
  2
3 $ns duplex-link-op $n0 $n2 orient-right-down
4 $ns duplex-link-op $n1 $n2 orient-right-up
5 $ns simplex-link-op $n2 $n3 orient-right
6 $ns simplex-link-op $n3 $n2 orient-left
7 $ns duplex-link-op $n3 $n4 orient-right-up
8 $ns duplex-link-op $n3 $n5 orient-right-down
```

We can also define the color of CBR and TCP packets for identification in NAM. For this we use the following command

```
1 # Marking the flows (for NAM)
2  $ns color1 Blue
3  $ns color2 Red
```

To view the network animator we need to type the command: nam

Network Animator could only be run on a desktop. This Virtual Lab does not provide any option to visualize the NAM output (apart from a few screenshots). Henceforth, we would skip creating NAM trace files in our code.

Tracing

Tracing Objects

NS-2 simulation can produce visualization trace as well as ASCII file corresponding to the events that are registered at the network. While tracing ns inserts four objects: EnqT, DeqT, RecvT, and DrpT. EnqT registers information regarding the arrival of packet and is queued at the input queue of the link. When overflow of a packet occurs, then the information of the dropped packet is registered in DrpT. DeqT holds the information about the packet that is dequeued instantly. RecvT hold the information about the packet that has been received instantly.

Event	Time	From node	To node	Pkt type	Pkt size	Flags	Fid	Src addr	Dst addr	Seq num	Pkt id

Structure of Trace Files

The following describe about the structure of the trace files produced for wired networks. The format of the trace files are different for wireless networks, and will be discussed in the Experiment on WiMax.

1. The first field is event.It gives you four possible symbols '+' '-' 'r' 'd'. These four symbols correspond respectively to enqueued, dequeued, received and dropped.
2. The second field gives the time at which the event occurs
3. The third field gives you the input node of the link at which the event occurs
4. The fourth field gives you the the output node at which the event occurs
5. The fifth field shows the information about the packet type.i.e whether the packet is UDP or TCP
6. The sixth field gives the packet size
7. The seventh field give information about some flags
8. The eight field is the flow id(fid) for IPv6 that a user can set for each flow in a tcl script.It is also used for specifying the color of flow in NAM display
9. The ninth field is the source address
10. The tenth field is the destination address
11. The eleventh field is the network layer protocol's packet sequence number
12. The last field shows the unique id of packet

Following are trace of two events:

r 1.84471 2 1 cbr 210 ------- 1 3.0 1.0 195 600

r 1.84566 2 0 ack 40 ------- 2 3.2 0.1 82 602

SIMULATION

Create a network with three nodes namely 0, 1 and 2. Establish a TCP connection between node 0 and node 2 such that node 0 will send TCP packets to node 2 via node 1.

At first the simulator is started and then the trace files, nam files ,finish procedure are defined. Next, three three nodes are created and connected by duplex links while defining bandwidth,delay and queue type.The queue size has been set to define the buffer capacity. A TCP connection is created between node 0 and node 2. Node 0 sends TCP packets to node 2 through node 1 while node 2 in turn send acknowledgements to node 0.The events are scheduled at a particular time.And at last we run to ns to view the simulation and get the required outputs.

PROCEDURE

Steps for conducting the experiment

General Instructions

Follow are the steps to be followed in general to perform the experiments in Advanced Network Technologies Virtual Lab.

1. Read the theory about the experiment
2. View the simulation provided for a chosen, related problem
3. Take the self evaluation to judge your understanding (optional, but recommended)
4. Go to the exercises section, choose a problem, and carefully read the problem description
5. Write a script (or make necessary changes) to simulate the desired scenario in the code editor just below the problem statement
6. Click on the 'Run' button to execute the simulation script
7. Simulation with ns2: If the simulation was successful, and was instructed to create a trace file, contents of the trace file would be displayed in the area below the 'Run' button
8. Simulation with ns3: If the simulation was successful, output of the program would be displayed in the area below the 'Run' button
9. A trace file generated as a result of simulation with ns2 could be used for certain kind of analysis, which would be discussed in a later section

Experiment Specific Instructions

In the theory part we have learned how to work with NS2. In this section we now learn how to make your system ready so that you can work with Network Simulator 2.NS2 is a open source software. It can be downloaded from Internet and installed.

Basic Requirements:

- A computer which is having access to the Internet
- Minimum 512Mb RAM
- Operating system: Linux(Ubuntu 10.04)
- ns-2.34 package
- gcc-4.4.3
- make tools

The following instructions for downloading and installing ns2 are for a system with:

- Operating System: Linux (Ubuntu 10.04)
- ns2: 2.34
- gcc: 4.4.3

Some of the known problems that have been faced during installation as well as their solutions have been discussed in one of the sections below.

The steps for installation should ideally be applicable for other version and/or configuration of Linux also. Any other problem that might arise would require further troubleshooting.

Downloading ns-2.34

To download ns2 go to The Network Simulator: Building Ns. Here you can download the ns all-in-one package or you can download the packages separately. The instructions provided here are largely based on what have been mentioned in The Network Simulator: Building Ns.

Now let's learn how to download the packages separately.

1. First go to The Network Simulator: Building Ns
2. Then download Tcl and Tk from http://www.tcl.tk/software/tcltk/downloadnow84.tml [Note: the versions of Tcl and Tk must be same.]
3. Download OTcl from: http://sourceforge.net/projects/otcl-tclcl/files/OTcl/1.13/
4. Download Tclcl from: http://sourceforge.net/projects/otcl-tclcl/files/TclCL/1.19/
5. Download ns-2 from: http://sourceforge.net/projects/nsnam/files/ns-2/2.34/
6. Download nam from: http://sourceforge.net/projects/nsnam/files/nam-1/1.14/ [Note: download only nam-1.14.tar.gz not ns-allinone package.]
7. Download xgraph from: http://sourceforge.net/projects/nsnam/files/xgraph/xgraph-12.1/

Note: There are some other few things to be downloaded but that depends upon your requirement. For example, if some error occurs for absence of any package, then you need to detect the error and download the required package. A good amount of troubleshooting guidance will be provided in the subsequent sections.

Installation

Following steps illustrate how to install the packages separately

1. All the files will be zip format -- you need to unzip all the files at first. The command to unzip the files:

```
1tar -xzvf  <file_name>
```

for e.g if we want to unzip the Tcl package the type: tar -xzvf tcl8.4.19

To unzip all the files together use the following command:

```
1  for ifile in ` ls *.tar.gz`
2   do
3  tar -xzvf $ifile
4   done
```

2. Now let's begin with installing Tcl. The steps required are:

```
1 cd tcl8.4.19    # Move inside the directory containing (unzipped) source code of
  Tcl
2 ls
3 cd unix
4 ./configure
```

```
5 make
6 sudo make install
```

3. Install Tk:

```
1  cd tk8.4.19
2   ls
3   cd unix
4  ./configure
5   make
6   sudo make install
```

4. Install OTcl:

```
1  cd otcl-1.13
2 ./configure --with-tcl=../tcl8.4.19  # Note: while configuring we need to specify the
                                          tcl
3   make
4   sudo make install
```

5. Install Tclcl-1.19:

```
1  cd tclcl-1.19
2 ./configure --with-tcl=../tcl8.4.19  # Note: while configuring we need to specify the
                                          tcl
3   make
4   sudo make install
```

6. Install ns-2.34:

```
1  cd ns-2.34
2 ./configure --with-tcl=../tcl8.4.19
3   make
4   sudo make install
```

If the above steps complete successfully, open a terminal, and type in ns. If you see a % prompt, then, congrats, your installation was successful!

7. Install NAM:

```
1  cd nam-1.14
2 ./configure --with-tcl=../tcl8.4.19
3   make
4   sudo make install
```

8. Install xgraph:

```
cd xgraph-12.1
./configure
make
sudo make install
```

Probable problems that could appear while installing the packages and their solution

1. While installing Tk, 'make' fails. It generates some error message like:

gcc -c -O2 -pipe -Wall -Wno-implicit-int -fno-strict-aliasing -fPIC -I/home/tamoghna/Downloads/tk8.4.19/unix -I/home/tamoghna/Downloads/tk8.4.19/unix/../generic -I/home/tamoghna/Downloads/tk8.4.19/unix/../bitmaps -I/home/tamoghna/Downloads/tcl8.4.19/generic -DHAVE_LIMITS_H=1 -DHAVE_UNISTD_H=1 -DPEEK_XCLOSEIM=1 -D_LARGEFILE64_SOURCE=1 -DTCL_WIDE_INT_TYPE=long\ long -DHAVE_STRUCT_STAT64=1 -DHAVE_OPEN64=1 -DHAVE_LSEEK64=1 -DHAVE_TYPE_OFF64_T=1 -DHAVE_SYS_TIME_H=1 -DTIME_WITH_SYS_TIME=1 -DSTDC_HEADERS=1 -DHAVE_PW_GECOS=1 -DTCL_NO_DEPRECATED -DUSE_TCL_STUBS /home/tamoghna/Downloads/tk8.4.19/unix/../generic/tk3d.c

In file included from /home/tamoghna/Downloads/tk8.4.19/unix/../generic/tkInt.h:21,

from /home/tamoghna/Downloads/tk8.4.19/unix/../generic/tk3d.h:18,

from /home/tamoghna/Downloads/tk8.4.19/unix/../generic/tk3d.c:16:

/home/tamoghna/Downloads/tk8.4.19/unix/../generic/tk.h:81:23: error: X11/Xlib.h: No such file or directory

In file included from /home/tamoghna/Downloads/tk8.4.19/unix/../generic/tkInt.h:21,

from /home/tamoghna/Downloads/tk8.4.19/unix/../generic/tk3d.h:18,

from /home/tamoghna/Downloads/tk8.4.19/unix/../generic/tk3d.c:16:

/home/tamoghna/Downloads/tk8.4.19/unix/../generic/tk.h:557: error: expected declaration specifiers or '...' before 'Window'

/home/tamoghna/Downloads/tk8.4.19/unix/../generic/tk.h:557: error: 'Window' declared as function returning a function

/home/tamoghna/Downloads/tk8.4.19/unix/../generic/tk.h:557: warning: parameter names (without types) in function declaration

/home/tamoghna/Downloads/tk8.4.19/unix/../generic/tk.h:560: error: expected declaration specifiers or '...' before 'XEvent'

/home/tamoghna/Downloads/tk8.4.19/unix/../generic/tk.h:569: error: expected specifier-qualifier-list before 'Tk_ClassCreateProc'

/home/tamoghna/Downloads/tk8.4.19/unix/../generic/tk.h:663: error: expected specifier-qualifier-list before 'Bool'

/home/tamoghna/Downloads/tk8.4.19/unix/../generic/tk.h:679: error: expected specifier-qualifier-list before ‘Bool’

/home/tamoghna/Downloads/tk8.4.19/unix/../generic/tk.h:756: error: expected specifier-qualifier-list before ‘Display’

...

...

...

make: *** [tk3d.o] Error 1

Solution:

Install the libx11-dev package. Open a terminal, and type in

```
1 sudo apt-get install libx11-dev
```

2. **Tk was installed properly but it failed to run somehow.**

How to identify this:

After installing tk8.4.19 try to run the script tk8.4.19/unix/wish from the terminal. A small window will open, close it. If no error messages appears in the terminal then Tk installation is successful and Tk is working properly. This can also be verified after installing nam and then trying to run nam. The error message would be something like:

nam:

[code omitted because of length]

: no event type or button # or keysym

while executing

"bind Listbox <MouseWheel> {

%W yview scroll [expr {- (%D / 120) * 4}] units

}"

invoked from within

"if {[tk windowingsystem] eq "classic" || [tk windowingsystem] eq "aqua"} {

bind Listbox <MouseWheel> {

%W yview scroll [expr {- (%D)}] units

}

bind Li..."

Solution: If you get error messages then download the patch files tk-8.4.18-tkBind.patch and tk-8.4-lastevent.patch from http://bugs.gentoo.org/show_bug.cgi?id=225999.

then copy those files into Tk directory.Now apply the patches by using the following command:

```
1   patch -p1 < tk-8.4.18-tkBind.patch
2   patch -p0 < tk-8.4-lastevent.patch
```

If fail to apply the patches then open the patch, check the name of the file to be patched, and make the relevant modifications to that file accordingly.

Note: the contents of the original file are shown with a minus(-) sign at the beginning.The modified contents do begin with a plus(+) sign. The contents of the two patch files are shown below for easy reference:

```
01   --- tk8.4.18-orig/generic/tkBind.c  2006-07-21 08:26:54.000000000 +0200
02     +++ tk8.4.18/generic/tkBind.c   2008-07-05 12:17:10.000000000 +0200
03   @@ -586,6 +586,9 @@
04       /* ColormapNotify
       */    COLORMAP,
05         /* ClientMessage */  0,
06         /* MappingNotify */  0,
07   +#ifdef GenericEvent
08     +    /* GenericEvent
           */    0,
09    +#endif
10          /*
   VirtualEvent
     */   VIRTUAL,
11         /* Activate */   ACTIVATE,
12           /* Deactivate */    ACTIVATE,
```

and

view source

print?

```
01   --- generic/tk.h.orig   2008-02-06 16:31:40.000000000 +0100
02     +++ generic/tk.h    2008-07-24 08:21:46.000000000 +0200
```

```
03   @@ -635,17 +635,15 @@
04               *
05      *-----------------------------------------------------------------------
06                                                 */
07    -#define VirtualEvent        (LASTEvent)
08  -#define ActivateNotify      (LASTEvent + 1)
09    -#define DeactivateNotify     (LASTEvent + 2)
10    -#define MouseWheelEvent      (LASTEvent + 3)
11    -#define TK_LASTEVENT        (LASTEvent + 4)
12  +#define VirtualEvent        (MappingNotify + 1)
13    +#define ActivateNotify      (MappingNotify + 2)
14   +#define DeactivateNotify     (MappingNotify + 3)
15    +#define MouseWheelEvent      (MappingNotify + 4)
16     +#define TK_LASTEVENT        (MappingNotify + 5)
17
           #define
18 MouseWheelMask      (1L
           << 28)
19             -
           #define
20 ActivateMask        (1L
            < 29)
21     #define VirtualEventMask     (1L < 30)
22    -#define TK_LASTEVENT        (LASTEvent +
                       4)
 23
 24
 25    /*
```

After this install Tk from beginning again and also verify whether 'wish' runs properly (as indicated above).

3. **Problem while running 'make' for OTcl**

otcl.o: In function `OTclDispatch':

/home/tanay/Desktop/ns2/otcl-1.13/otcl.c:495: undefined reference to `__stack_chk_fail_local'

otcl.o: In function `Otcl_Init':

/home/tanay/Desktop/ns2/otcl-1.13/otcl.c:2284: undefined reference to `__stack_chk_fail_local'

ld: libotcl.so: hidden symbol `__stack_chk_fail_local' isn't defined

ld: final link failed: Nonrepresentable section on output

make: *** [libotcl.so] Error 1

Solution:

1.goto the 'configure' file

2.In line no. 5516

SHLIB_LD="ld -shared"

change the above to

SHLIB_LD="gcc -shared"

Now reapeat the installation process starting from './configure'. For further information you can go to http://nsnam.isi.edu/nsnam/index.php/User_Information

4. Problem wile running ' sudo make install' for ns-2.34

ns: error while loading shared libraries: libotcl.so: cannot open shared object file: No such file or directory

Solution:

We need to set the following environment variables, and store them in the ~/.bashrc file.

```
1   OTCL_LIB=/your/path/ns2_packages/otcl-1.13
2   NS2_LIB=/your/path/ns2_packages/ns-2.34/lib
3   export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB
```

Now open a new terminal, and type ns. This should now work without any error.

5. Problem while running 'make' for nam

rm -f tkcompat.o; gcc -o tkcompat.o -c -Wall -Wno-write-strings -DTCL_TK -DNO_VOID -DNDEBUG -DUSE_SHM

-DHAVE_LIBTCLCL -DHAVE_TCLCL_H -DHAVE_LIBOTCL1_13 -DHAVE_OTCL_H -DHAVE_LIBTK8_4 -DHAVE_TK_H -DHAVE_LIBTCL8_4 -DHAVE_TCLINT_H -DHAVE_TCL_H -I. -I/home/bibudhendu/Desktop/ns2/tclcl-1.19 -I/home/bibudhendu/Desktop/ns2/otcl-1.13 -I/usr/local/include -I/home/bibudhendu/Desktop/ns2/tcl8.4.19/generic -I/home/bibudhendu/Desktop/ns2/tcl8.4.19/generic tkcompat.c

rm -f tkUnixInit.o; gcc -o tkUnixInit.o -c -Wall -Wno-write-strings -DTCL_TK -DNO_VOID -DNDEBUG -DUSE_SHM -DHAVE_LIBTCLCL -DHAVE_TCLCL_H -DHAVE_LIBOTCL1_13 -DHAVE_OTCL_H -

DHAVE_LIBTK8_4 -DHAVE_TK_H -DHAVE_LIBTCL8_4 -DHAVE_TCLINT_H -DHAVE_TCL_H -I. -I/home/bibudhendu/Desktop/ns2/tclcl-1.19 -I/home/bibudhendu/Desktop/ns2/otcl-1.13 -I/usr/local/include -I/home/bibudhendu/Desktop/ns2/tcl8.4.19/generic -I/home/bibudhendu/Desktop/ns2/tcl8.4.19/generic tkUnixInit.c

rm -f xwd.o; gcc -o xwd.o -c -Wall -Wno-write-strings -DTCL_TK -DNO_VOID -DNDEBUG -DUSE_SHM -DHAVE_LIBTCLCL -DHAVE_TCLCL_H -DHAVE_LIBOTCL1_13 -DHAVE_OTCL_H -DHAVE_LIBTK8_4 -DHAVE_TK_H -DHAVE_LIBTCL8_4 -DHAVE_TCLINT_H -DHAVE_TCL_H -I. -I/home/bibudhendu/Desktop/ns2/tclcl-1.19 -I/home/bibudhendu/Desktop/ns2/otcl-1.13 -I/usr/local/include -I/home/bibudhendu/Desktop/ns2/tcl8.4.19/generic -I/home/bibudhendu/Desktop/ns2/tcl8.4.19/generic xwd.c

xwd.c:87:29: error: X11/Xmu/WinUtil.h: No such file or directory

make: *** [xwd.o] Error 1

Solution:

Install the package libxmu-dev. Then run

```
1  ./configure
2  make clean
3  make
4  sudo make install
```

6. **Problem while running 'make' for xgraph**

/usr/include/stdio.h:651: note: expected ‘size_t * __restrict__’ but argument is of type ‘char *’

dialog.c:780: error: too few arguments to function ‘getline’

dialog.c: In function ‘getline’:

dialog.c:899: error: argument ‘lptr’ doesn’t match prototype

/usr/include/stdio.h:651: error: prototype declaration

dialog.c:899: error: number of arguments doesn’t match prototype

/usr/include/stdio.h:651: error: prototype declaration

make: *** [dialog.o] Error 1

Solution:

Download the patch below-

http://archive.ubuntu.com/ubuntu/pool/universe/x/xgraph/xgraph_12.1-12.diff.gz

Note: copy and unzip the above patch file into xgraph-12.1

```
1patch < xgraph_12.1-12.diff
```

After applying the patch if you see in get any problem with the configure.in file like

configure.in:3: version mismatch. This is Automake 1.11,

configure.in:3: but the definition used by this AM_INIT_AUTOMAKE

configure.in:3: comes from Automake 1.11.1. You should recreate

configure.in:3: aclocal.m4 with aclocal and run automake again.

then goto configure.in file and add 'AC_LOCAL' in the first line.

The above is an attempt to list frequent problems faced while trying to install ns2. This list is by no means exhaustive. In the worst case, you might require to do a bit of troubleshooting by yourself, which, in most cases, would be installing certain missing packages.

You can also look at the following places for further help:

1. Installing ns2.31 on Ubuntu7.04
2. Distribution Specific instructions

Trace File Analysis

A simple tool has been provided as part of this lab to analyze the trace files generated after simulation with ns2. A summary of the available options, and usage guide is given below.

Features List

Following is a list of functionalities provided by the Trace Analysis tool:

1. **Trace file formats**: Following trace file formats are being supported:

    1. Wired
    2. Wireless (new format)
    3. Satellite -- currently redirects to wired mode
    4. Mixed -- when both wired and wireless connections are present in the simulation

2. **General Statistics:** To provide some common statistics about the simulation being run. Currently displays only the simulation duration.

    1. Inputs: None

    2. Output: Text

3. **Average Throughput**: Computes total # of bytes received by a node over the entire simulation duration

    1.Inputs: Node #

    2.Output: Number

4**. Bytes Received:** Plots cumulative count of bytes received by a node over the entire simulation duration

    1.Inputs: Node #; for wireless scenario, trace levels (AGT, MAC, RTR)

    2.Output: Graph

5**. End-to-end Delay**: Plots the end-to-end delay delay encountered by packets while moving from a source node to the destination node

1. Inputs: Source node #, destination node #, scaling factor [optional] -- scaling factor helps to amplify the y-axis values

2. Output: Graph

6. Packet Retransmissions: Plots # of retransmission(s) of a given packet occurs between the source and destination nodes

1. Inputs: Source node #, destination node #

2. Output: Graph

7. Hop Count: Plots the # of hops traveled by a packet to reach the destination node from the source node. It counts the destination node as well.

1. Inputs: Source node #, source port #, destination node #, destination port #

2. Output: Graph

Limitations

1. For analyzing the problems in the "Satellite Networks" experiment, please use the Wired mode of analysis
2. Analysis of trace files for mixed mode of simulations (wired & wireless) is not supported currently
3. Outputs produced do not necessarily have accuracy for scientific publications. In particular, the plot of hop counts may vary a bit from the original count (in wireless mode) in cases when a packet has been forwarded to more than one node.
4. The tool currently allows only a single instance of a given type of plot. For example, this doesn't let you plot end-to-end delays between multiple (source, destination) node pairs

REFERENCES

Following books and websites have been consulted for this experiment. You are suggested to go through them for further details.

Bibliography

1. Introduction to Network Simulator NS2, Teerawat Issariyakul, Ekram Hossain, Springer, 1st edition, 2008

Webliography

I. Marc Greis tutorial
II. The NS Manual
III. NS by Example
IV. Network Simulator 2