Laboratory 1: Running Emulab tutorial on DETER

ESHA DESAI

USC ID: 6993245898

Contents

Assignment#1

Assignment 1 is introduction to NS-Network simulator. It's an open-source software for simulating different scenarios where parameters like number of nodes, link delay, packet traffic etc. can be adjusted.

1.Basic NS tutorial:

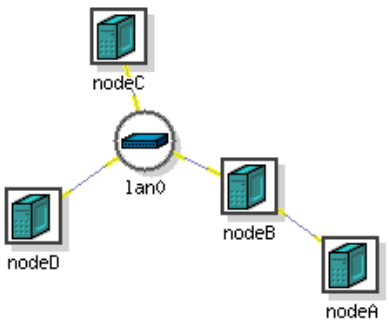
First to begin an experiment, an ns script is uploaded on to the DETER and an experiment is created. A new instance of network simulator is instantiated, and my experiment consisted of a network topology of 4 nodes A, B, C & D which need to be declared first. Then a link between node A and B was created with a bandwidth of 30Mb, delay of 50ms. Parameters like packet size and packet interval and link loss can also be adjusted.

We can change the OS on to the nodes by using the ns script *tb-set-node-os $nodeA RHL-STD.* Here it says that the OS is Red Hat Linux. A LAN between nodes B, C and D was created with a bandwidth of 100Mb. Next static routing is configured and then the network is set to start with the above settings.

After the experiment runs successfully a listing is created. My experiment's listing gave the information that my experiment name was: Experiment: USC558L/labesha.

State of the experiment was active and the experiment was successful!

Experiment **USC558L/labesha**



The "Virtual Lan/Link info" was observed in the listing to be as stated in the ns script i.e. link had 30Mbp and Lan had 100Mb.

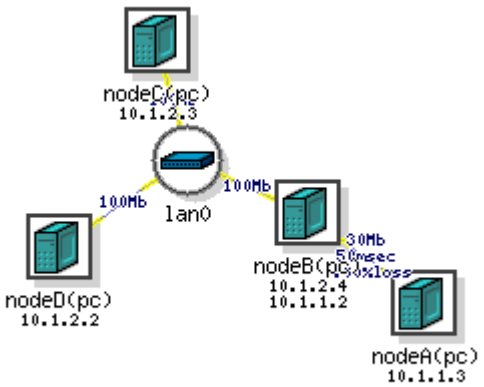
From the "Physical Lan/link mapping", the Ip addresses and MAC addresses of each of the nodes could be noted.

From “Physical delay info”, the delay can be observed to be 50ms as stated in ns script.

The qualified names of the nodes were:

Virtual Node Info:

ID	Type	OS	Qualified Name
nodeA	pc	FBSD-STD	nodeA.labesha.USC558L.isi.deterlab.net
nodeB	pc		nodeB.labesha.USC558L.isi.deterlab.net
nodeC	pc	RHL-STD	nodeC.labesha.USC558L.isi.deterlab.net
nodeD	pc		nodeD.labesha.USC558L.isi.deterlab.net



Node ID	Name	Type	Default OSID
pc083	nodeA	pc3000	FBSD410-STD
pc086	nodeB	pc3000	Ubuntu1004-STD
pc108	tbdelay0	pc3000	FBSD62-STD
pc109	nodeC	pc3000	FBSD410-STD
pc112	nodeD	pc3000	Ubuntu1004-STD

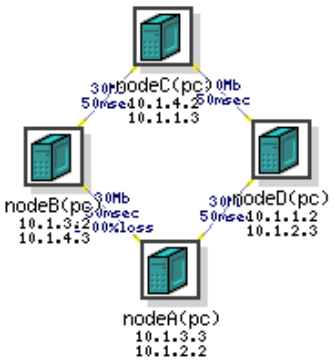
Only 4 nodes were declared but tbdelay0 indicated that a 5th invisible node was created for creating delay.

From the Physical Node Mapping”, it was found that node B was Ubuntu and being comfortable with Ubuntu, I did an ssh to users.deterlab.net and then from there to node B by using the command *ssh nodeB.labesha.USC558L.isi.deterlab.net*.

I did an “ifconfig” on node B and I found that there were four interfaces to node B.

```
users.deterlab.net - PuTTY
  Swap usage:   0%                IP address for eth3: 192.168.1.86
  Processes:    122

  Graph this data and manage this system at https://landscape.canonical.com/

Last login: Fri Aug 26 16:03:41 2011 from users.isi.deterlab.net
sc558ag@nodeb:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:04:23:ae:cc:32
          inet addr:10.1.2.4  Bcast:10.1.2.255  Mask:255.255.255.0
          inet6 addr: fe80::204:23ff:feae:cc32/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:402 (402.0 B)  TX bytes:636 (636.0 B)

eth1      Link encap:Ethernet  HWaddr 00:04:23:ae:cc:33
          inet addr:10.1.1.2  Bcast:10.1.1.255  Mask:255.255.255.0
          inet6 addr: fe80::204:23ff:feae:cc33/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:1086 (1.0 KB)  TX bytes:558 (558.0 B)

eth3      Link encap:Ethernet  HWaddr 00:11:43:d5:fe:33
          inet addr:192.168.1.86  Bcast:192.168.3.255  Mask:255.255.252.0
          inet6 addr: fe80::211:43ff:fed5:fe33/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1573 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1260 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:289229 (289.2 KB)  TX bytes:186265 (186.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:106 errors:0 dropped:0 overruns:0 frame:0
          TX packets:106 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:12563 (12.5 KB)  TX bytes:12563 (12.5 KB)

sc558ag@nodeb:~$
```

Few modifications were made like changing the number of nodes and bandwidth and delay. I implemented the loop example also. And then I implemented a ring topology.



2.Starting application automatically:

I tried to experiment the functionality where we can start our applications automatically since this was just a learning tutorial, my application was a very simple bash script with a for loop.

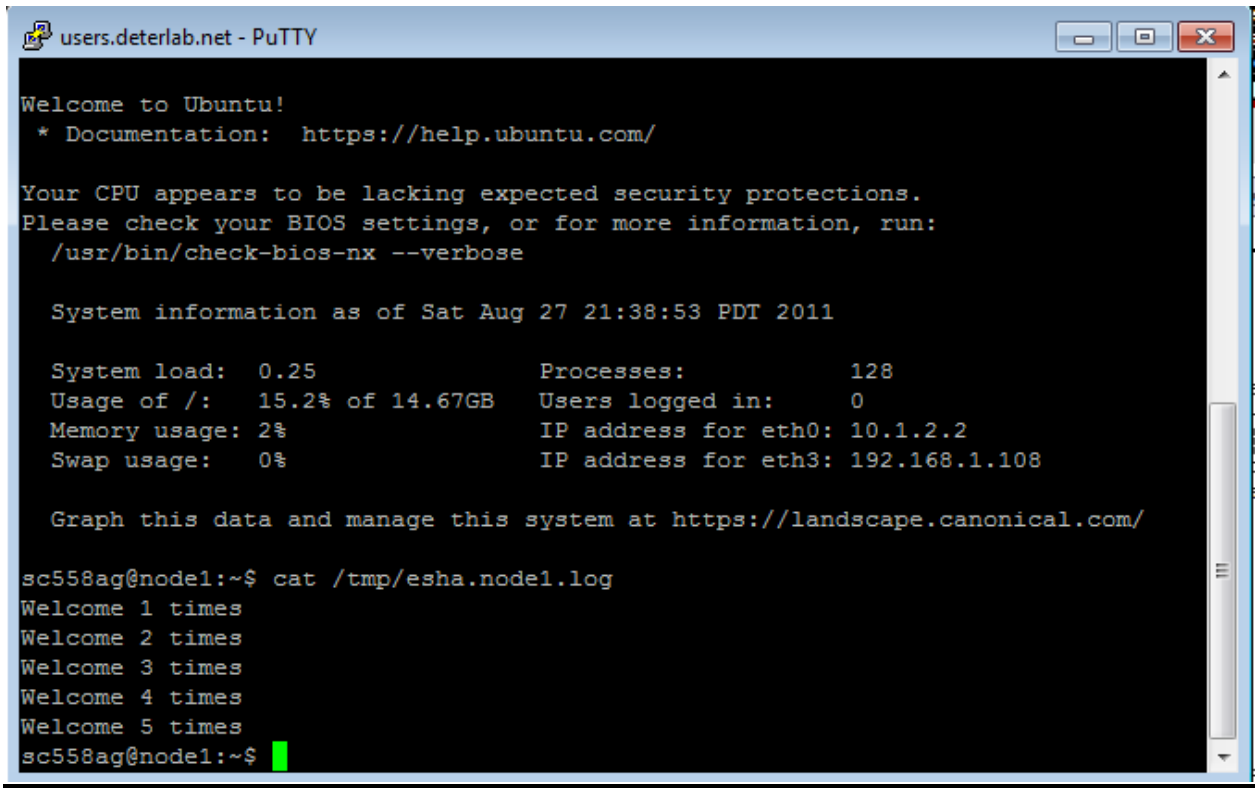
NS Script:

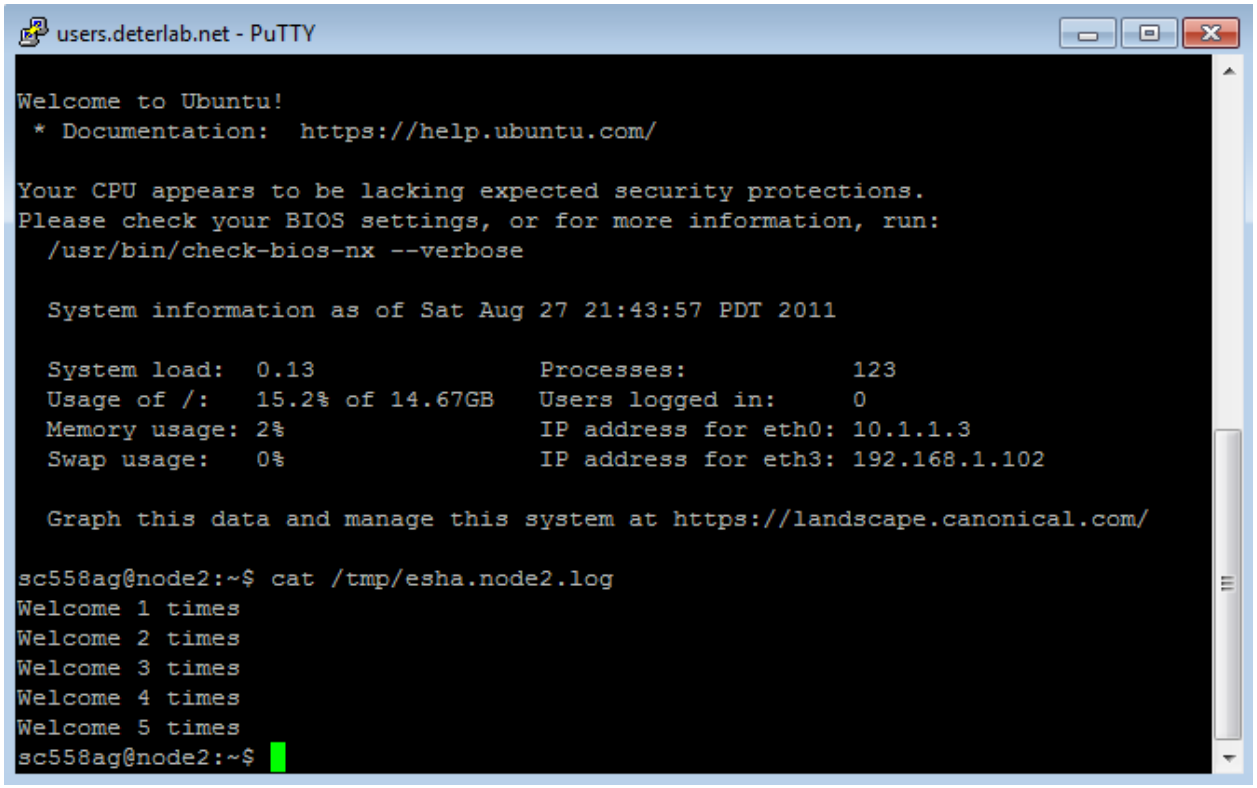
```
source tb_compat.tcl
set ns [new Simulator]
set node1 [$ns node]
set router [$ns node]
set node2 [$ns node]
set linkA [$ns duplex-link $node1 $router 100Mb 0ms DropTail]
set linkB [$ns duplex-link $router $node2 1Mb 10ms DropTail]
$ns rtproto Static
tb-set-node-startcmd $node1 "/proj/USC558L/edesai/esha.node1 >& /tmp/esha.node1.log"
tb-set-node-startcmd $node2 "/proj/USC558L/edesai/esha.node2 >& /tmp/esha.node2.log"
$ns run
```

Bash script:

```
#! /bin/bash
for i in 1 2 3 4 5
do
  echo "Welcome $i times"
done
```

As can be seen, I redirected the application/script output to /tmp/esha.node[1|2].log. Below are the snapshots from node1 and node2 to verify the content of these log files.





Definitely, more complicated functionality can be added to the script as and when needed.

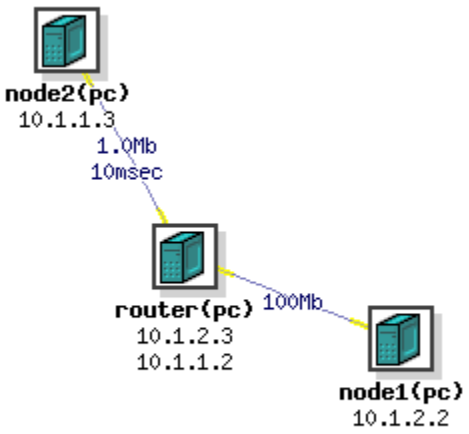
3.Manual Routing:

Till now we were doing static routing, here I tried to do manual routing. As seen in the NS script the node 1 is connected to router and node2 is connected to node1 via the router. Below is the NS script:

NS Script:

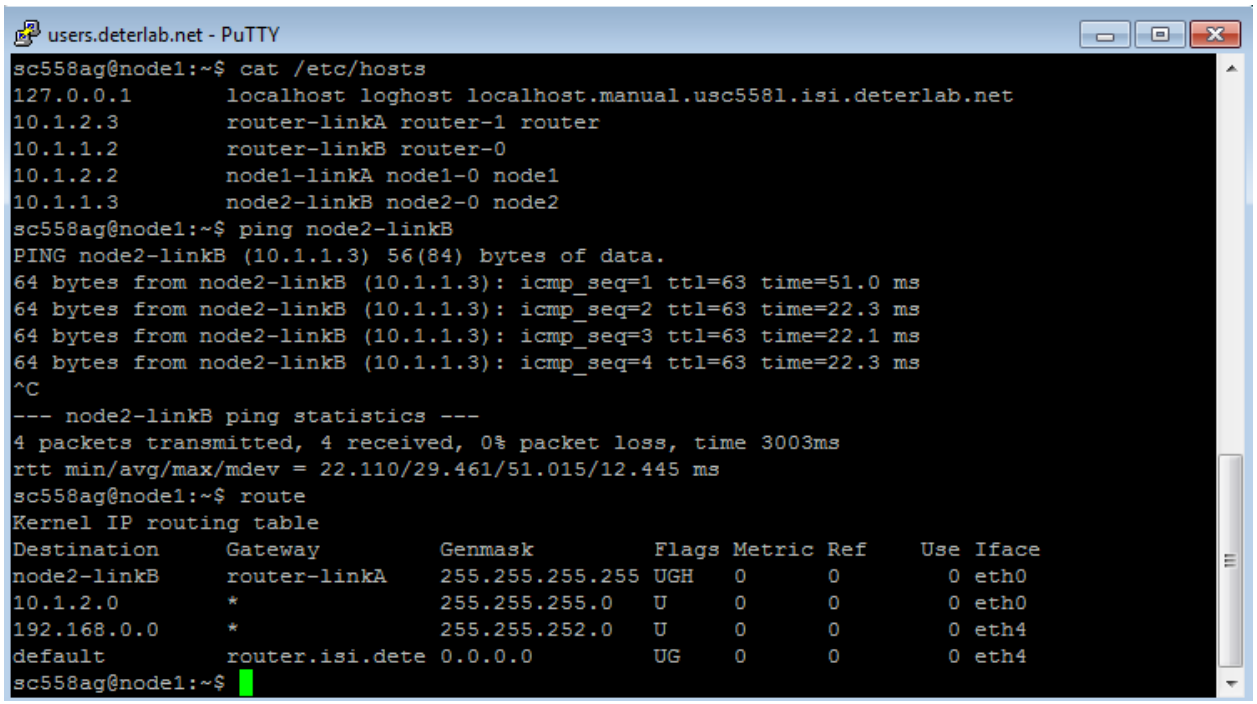
```
source tb_compat.tcl
set ns [new Simulator]
set node1 [$ns node]
set router [$ns node]
set node2 [$ns node]
set linkA [$ns duplex-link $node1 $router 100Mb 0ms DropTail]
set linkB [$ns duplex-link $router $node2 1Mb 10ms DropTail]
$ns rtproto Manual
$node1 add-route $node2 $router
$node2 add-route $node1 $router
$ns run
```

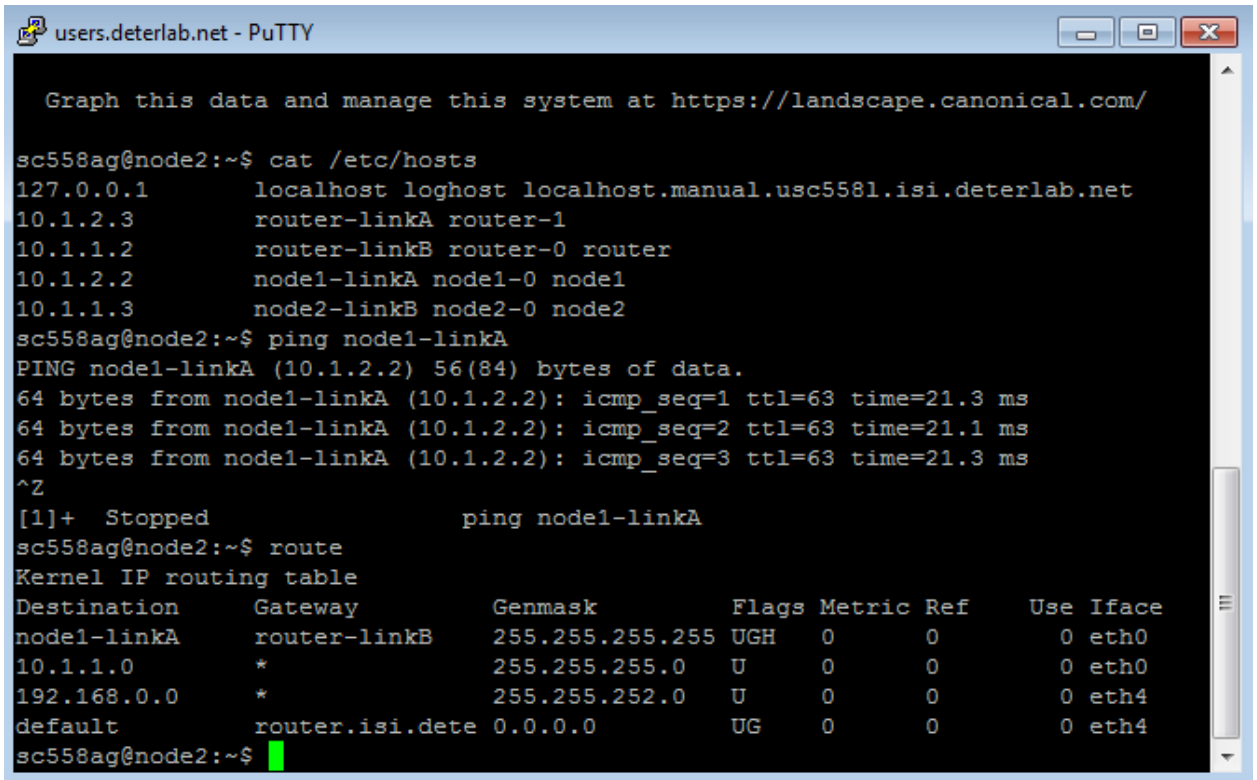
As seen below, I have added a manual route from node1 to node2 via router and vice versa.



Below is the snapshot of the route command at nodes 1 and 2 respectively. It verifies following things:

- Successful ping verifies connectivity
- First route in the routing table shows the manual route we installed through our ns script.


users.deterlab.net - PuTTY
sc558ag@node1:~$ cat /etc/hosts
127.0.0.1 localhost loghost localhost.manual.usc558l.isi.deterlab.net
10.1.2.3 router-linkA router-1 router
10.1.1.2 router-linkB router-0
10.1.2.2 node1-linkA node1-0 node1
10.1.1.3 node2-linkB node2-0 node2
sc558ag@node1:~$ ping node2-linkB
PING node2-linkB (10.1.1.3) 56(84) bytes of data.
64 bytes from node2-linkB (10.1.1.3): icmp_seq=1 ttl=63 time=51.0 ms
64 bytes from node2-linkB (10.1.1.3): icmp_seq=2 ttl=63 time=22.3 ms
64 bytes from node2-linkB (10.1.1.3): icmp_seq=3 ttl=63 time=22.1 ms
64 bytes from node2-linkB (10.1.1.3): icmp_seq=4 ttl=63 time=22.3 ms
^C
--- node2-linkB ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 22.110/29.461/51.015/12.445 ms
sc558ag@node1:~$ route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
node2-linkB router-linkA 255.255.255.255 UGH 0 0 0 eth0
10.1.2.0 * 255.255.255.0 U 0 0 0 eth0
192.168.0.0 * 255.255.252.0 U 0 0 0 eth4
default router.isi.dete 0.0.0.0 UG 0 0 0 eth4
sc558ag@node1:~$


users.deterlab.net - PuTTY
Graph this data and manage this system at https://landscape.canonical.com/
sc558ag@node2:~$ cat /etc/hosts
127.0.0.1 localhost loghost localhost.manual.usc558l.isi.deterlab.net
10.1.2.3 router-linkA router-1
10.1.1.2 router-linkB router-0 router
10.1.2.2 node1-linkA node1-0 node1
10.1.1.3 node2-linkB node2-0 node2
sc558ag@node2:~$ ping node1-linkA
PING node1-linkA (10.1.2.2) 56(84) bytes of data.
64 bytes from node1-linkA (10.1.2.2): icmp_seq=1 ttl=63 time=21.3 ms
64 bytes from node1-linkA (10.1.2.2): icmp_seq=2 ttl=63 time=21.1 ms
64 bytes from node1-linkA (10.1.2.2): icmp_seq=3 ttl=63 time=21.3 ms
^Z
[1]+ Stopped ping node1-linkA
sc558ag@node2:~$ route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
node1-linkA router-linkB 255.255.255.255 UGH 0 0 0 eth0
10.1.1.0 * 255.255.255.0 U 0 0 0 eth0
192.168.0.0 * 255.255.252.0 U 0 0 0 eth4
default router.isi.dete 0.0.0.0 UG 0 0 0 eth4
sc558ag@node2:~$

4.Using emulab Sync:

I setup node1 as master sync server. As can be seen from the startup script, node1 (master) will increment the barrier count by 1 and node2 will sync with the sync server when it boots up.

NS Script:

```
source tb_compat.tcl
set ns [new Simulator]
set node1 [$ns node]
set router [$ns node]
set node2 [$ns node]
set linkA [$ns duplex-link $node1 $router 100Mb 0ms DropTail]
set linkB [$ns duplex-link $router $node2 1Mb 10ms DropTail]
$ns rtproto Static
tb-set-sync-server $node1
tb-set-node-startcmd $node1 "/proj/USC558L/edesai/scripts/esha.sync.node1 master >& /tmp/esha.sync.node1.log"
tb-set-node-startcmd $node2 "/proj/USC558L/edesai/scripts/esha.sync.node2 slave >& /tmp/esha.sync.node2.log"
$ns run
```

Bash Script:

```
#!/bin/sh
if [ "$1" = "master" ]; then
  /usr/testbed/bin/emulab-sync -i 1
else
  /usr/testbed/bin/emulab-sync
Fi
```

I am not sure though if I have achieved what I was exactly trying to achieve.

5.Installing RPM:

We can use Emulab NS extension `tb-set-node-rpms` to install our own RPMs. I copied rpm for apache server in /proj/USC558L/edesai/ and gave the source in my ns script.

NS Script:

```
source tb_compat.tcl
set ns [new Simulator]
set node1 [$ns node]
set router [$ns node]
set node2 [$ns node]
tb-set-node-os $node1 RHL-STD
tb-set-node-os $node2 RHL-STD
set linkA [$ns duplex-link $node1 $router 100Mb 0ms DropTail]
set linkB [$ns duplex-link $router $node2 1Mb 10ms DropTail]
$ns rtproto Static
tb-set-node-rpms $node1 /proj/USC558L/edesai/gcc.rpm
```

tb-set-node-rpms $node2 /proj/USC558L/edesai/gcc.rpm
$ns run

Below is some dump from the output generated after starting experiment which shows the new rpm.

Virtual Node Info:
ID Type OS Qualified Name
--------------- ------------ --------------- --------------------
node1 pc RHL-STD node1.rist.USC558L.isi.deterlab.net
RPMS: /proj/USC558L/edesai/gcc.rpm
node2 pc RHL-STD node2.rist.USC558L.isi.deterlab.net
RPMS: /proj/USC558L/edesai/gcc.rpm
router pc router.rist.USC558L.isi.deterlab.net

Conclusion: Lab 1 as an introduction to ns-2 and ns scripting was useful for me who is new to network simulators. I learnt various features of ns-2 namely changing different parameters of a network like bandwidth, delay, OS etc. Then I learnt that the topology of the network could be changed. Manual routing between the nodes and starting applications automatically and how to install rpms and tar files.