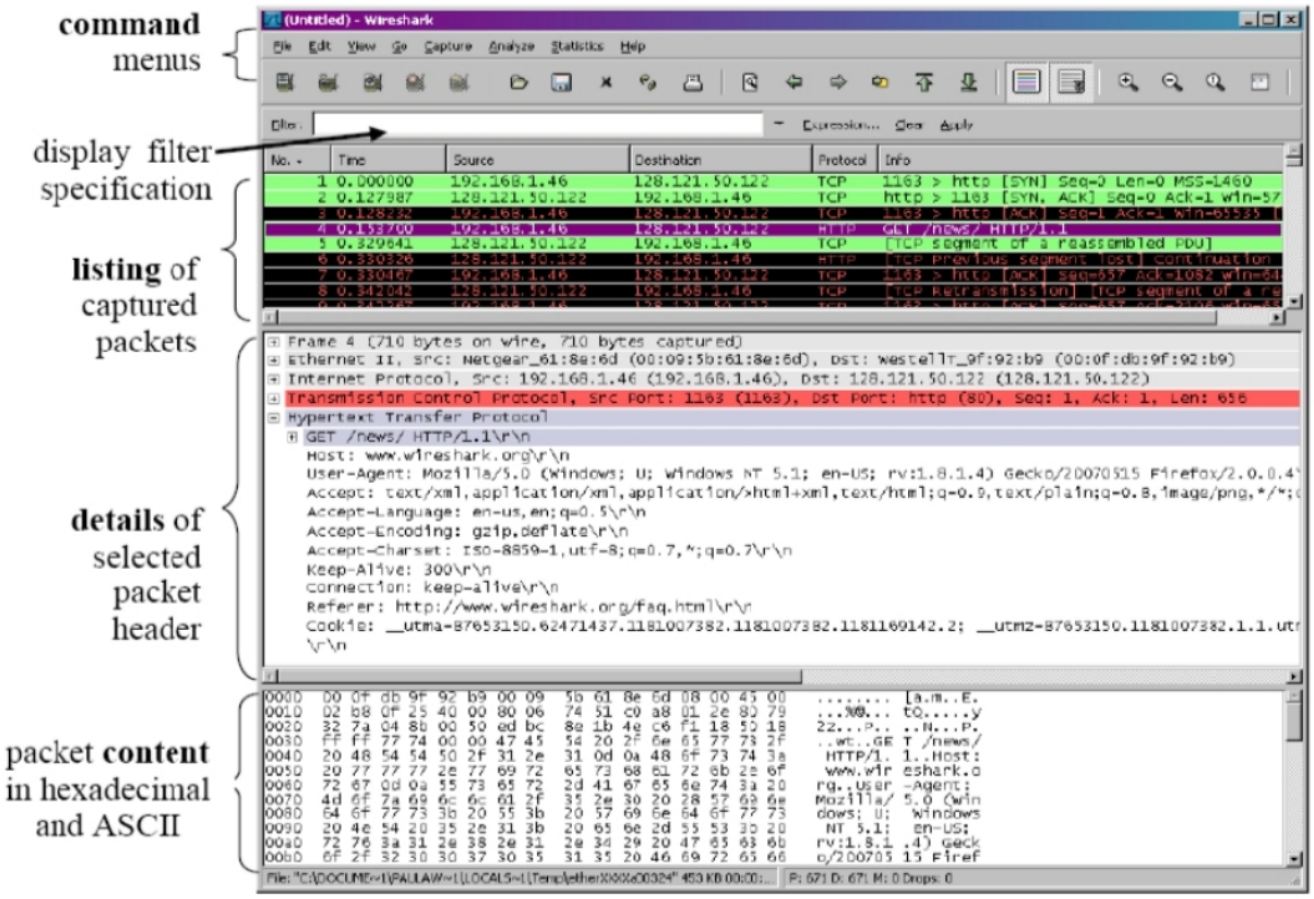


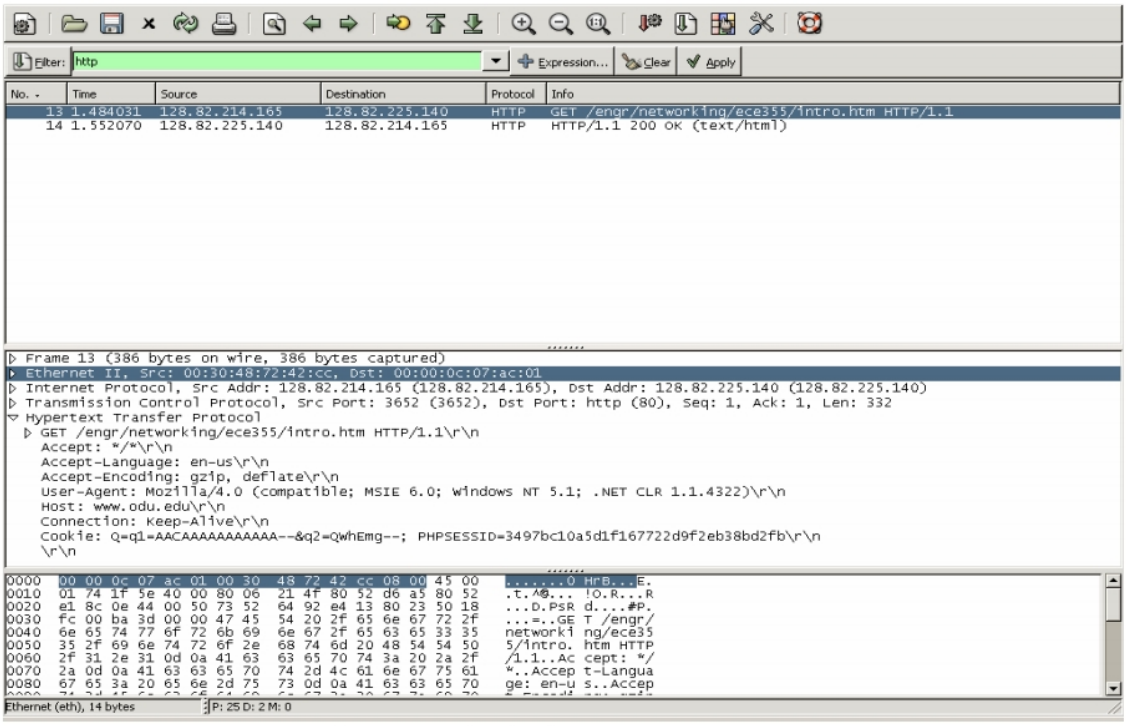
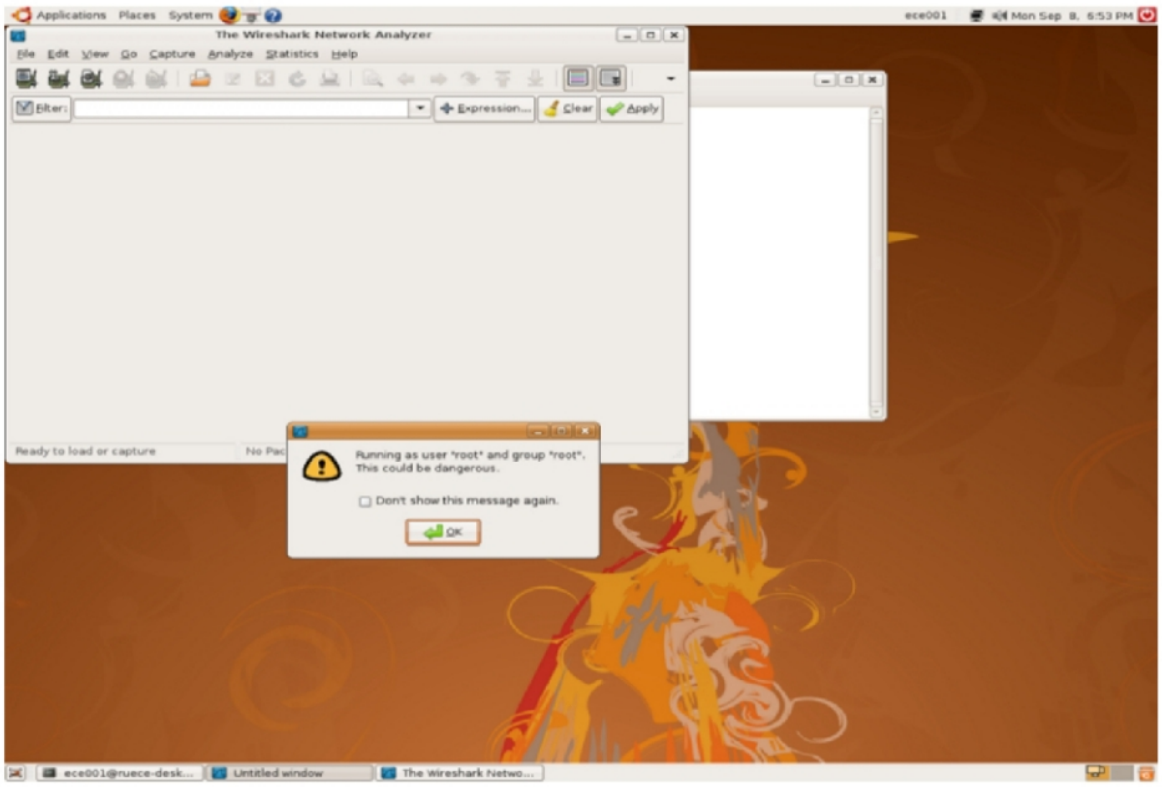
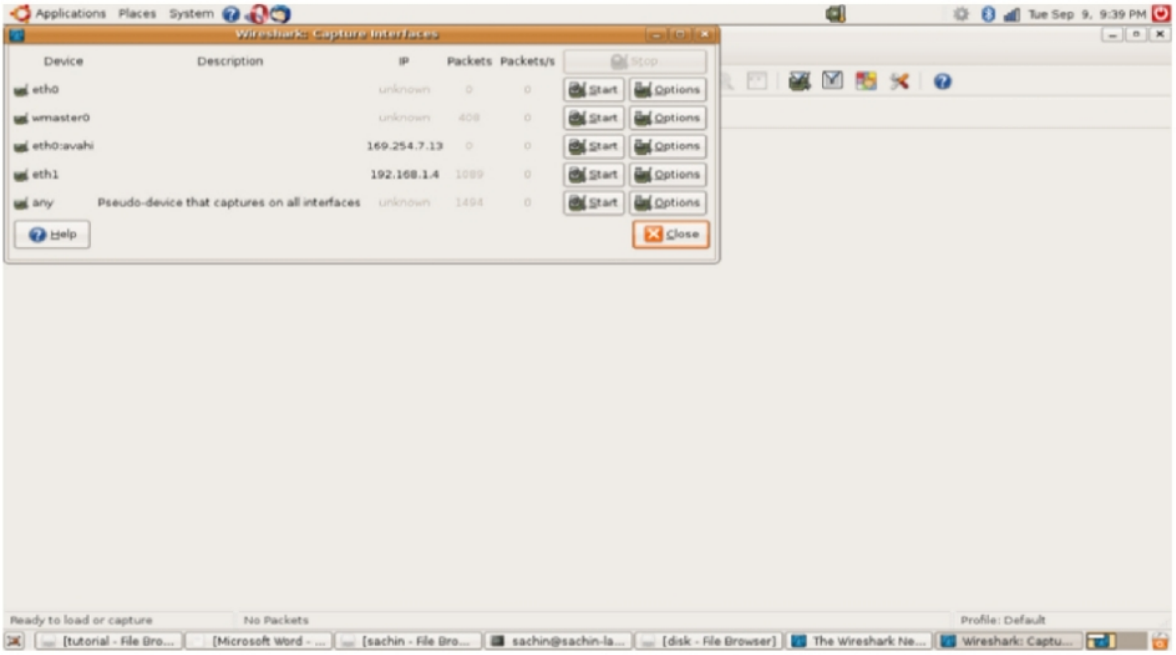


Figure 5: Wireshark display after step 9


Applications Places System
ece001
Mon Sep 8, 6:53 PM
The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply
Ready to load or capture
Running as user "root" and group "root". This could be dangerous.
Don't show this message again.
OK
ece001@ruece-desk...
Untitled window
The Wireshark Netwo...


Applications Places System
Tue Sep 9, 9:39 PM
Wireshark: Capture Interfaces
Device Description IP Packets Packets/s
Stop
eth0 unknown 0 0 Start Options
wmaster0 unknown 408 0 Start Options
eth0:avahi 169.254.7.13 0 0 Start Options
eth1 192.168.1.4 1089 0 Start Options
any Pseudo-device that captures on all interfaces unknown 1494 0 Start Options
Help
Close
Ready to load or capture
No Packets
Profile: Default
[tutorial - File Bro...
[Microsoft Word - ...
[sachin - File Bro...
sachin@sachin-la...
[disk - File Browser]
The Wireshark Ne...
Wireshark: Captu...

This will install nam and ns2 on your Linux. Now lets check if everything is functional by executing a small TCL/Tk simulation script.

```
#Create a simulator object
 set ns [new Simulator]
#Define different colors for data flows
$ns color 1 Blue
 $ns color 2 Red

#Open the nam trace file
set nf [open out.nam w]
 $ns namtrace-all $nf

#Define a 'finish' procedure
 proc finish {} {
  global ns nf
$ns flush-trace
 #Close the trace file
close $nf

#Execute nam on the trace file
exec nam out.nam &amp;amp;
exit 0
 }

#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 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
 $ns duplex-link $n3 $n2 1Mb 10ms SFQ

$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
```

```
#Monitor the queue for the link between node 2 and node 3
 $ns duplex-link-op $n2 $n3 queuePos 0.5

#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$udp0 set class_ 1
$ns attach-agent $n0 $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

#Create a UDP agent and attach it to node n1
set udp1 [new Agent/UDP]
$udp1 set class_ 2
$ns attach-agent $n1 $udp1

# Create a CBR traffic source and attach it to udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
 $cbr1 attach-agent $udp1

#Create a Null agent (a traffic sink) and attach it to node n3
 set null0 [new Agent/Null]
$ns attach-agent $n3 $null0

#Connect the traffic sources with the traffic sink
$ns connect $udp0 $null0  
$ns connect $udp1 $null0

#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"

 #Call the finish procedure after 5 seconds of simulation time
```



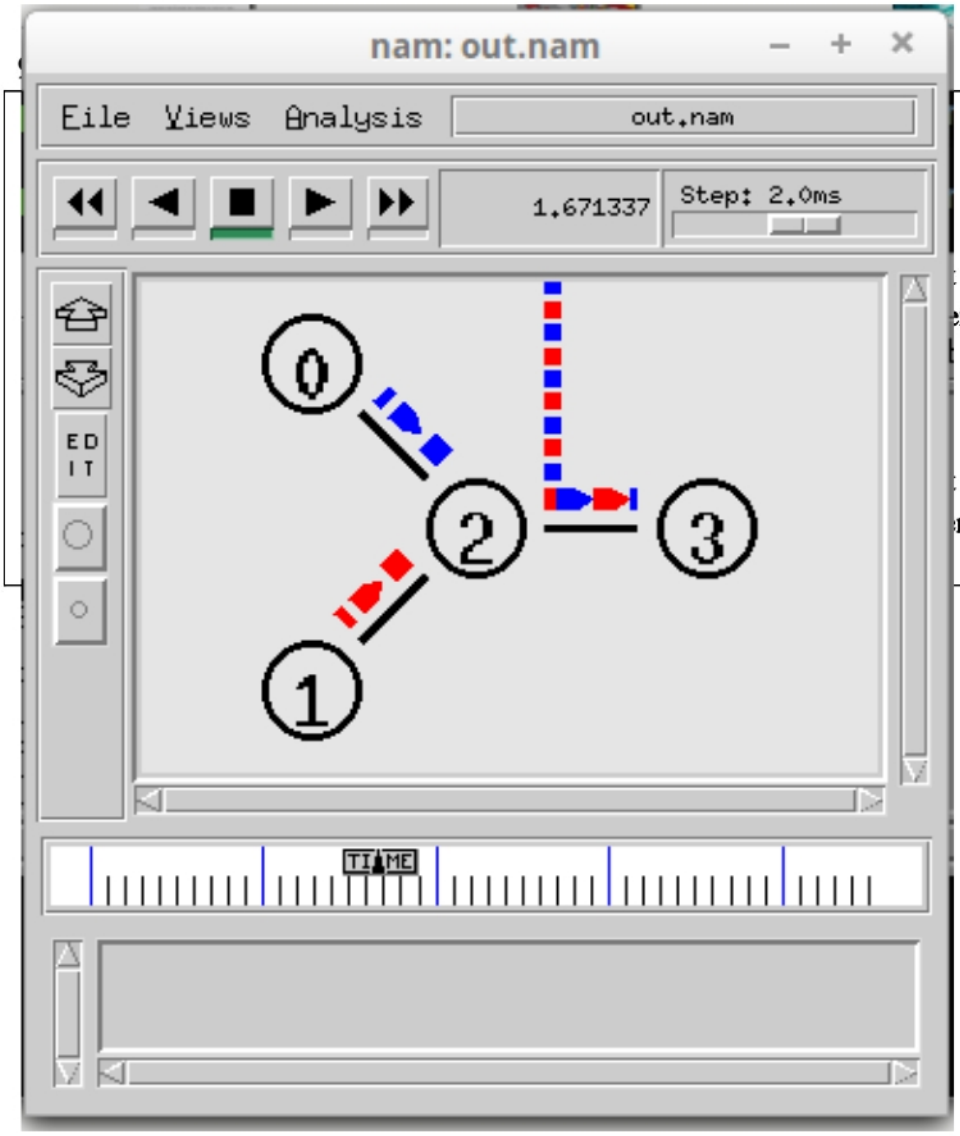
```
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

Save it into example.tcl and run the following command:

ns example.tcl

if everything goes fine, which shows the animator network on nam as shown in figure 12.1


nam: out.nam
File Views Analysis
out.nam
1.671337
Step: 2.0ms
ED
IT
0
1
2
3
TIME