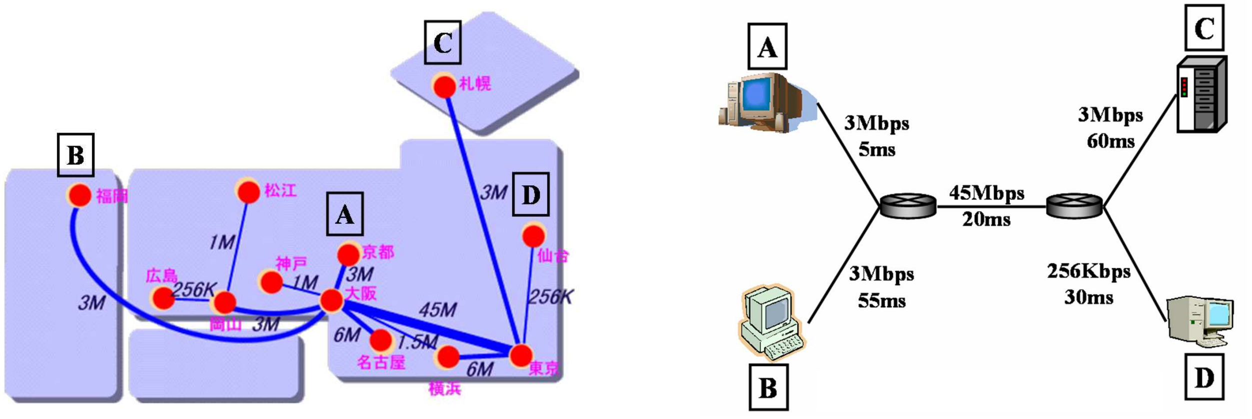
NS2 (Network Simulator version 2) Manual

Information Sciences Information and Communications Technology
School of Engineering Internet Engineering

1 What is Network Simulator?

Setting up a network to do some real experiments is the best way for studying about communication in internet. However, setting a network is not easy and costly. For this reason, a virtual network provided by network simulator is used for experiment in only one computer. Specially, NS2 which is free and easy to use is the popular all over the world.

For example, to observe the communication between A,B,C,D in a network as shown in Fig. 1(a), we can set up a network topology as shown in Fig.1(b) for simulator to do experiment.



(a) The reality network

(b) The network for simulation

Figure 1: Difference between real and simulation network

2 NS2 Manual

NS2 use Tcl language for creating simulation scenario file (for example, `sample.tcl`). Network topology, transmission time, using protocol etc... are defined in scenario file. If we execute this scenario file, the simulation result will be output to `out.tr` and `out.nam` file.

out.tr all the information about communication is written in this file. We can find out the way a packet was forwarded. This file is called as trace file.

out.nam contains the data for animation of the experiment result. This file can be execute by Nam, an animation software. The state of forwarding packet in Nam is shown in Fig.2.

If simulation use TCP, we can also observe the state of TCP congestion control by a trace file.

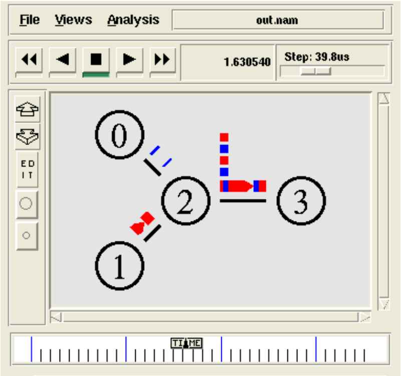


Figure 2: Animation soft Nam

out.tcp Record the change of TCP parameters by time. Using Gnuplot software to plot a graph, it is easy to observe the appearance of congestion control. This file is called as TCP trace file.

3 Scenario File

3.1 Indicate a Network in NS2

To write a scenario file, we need to understand about the network inside NS2.

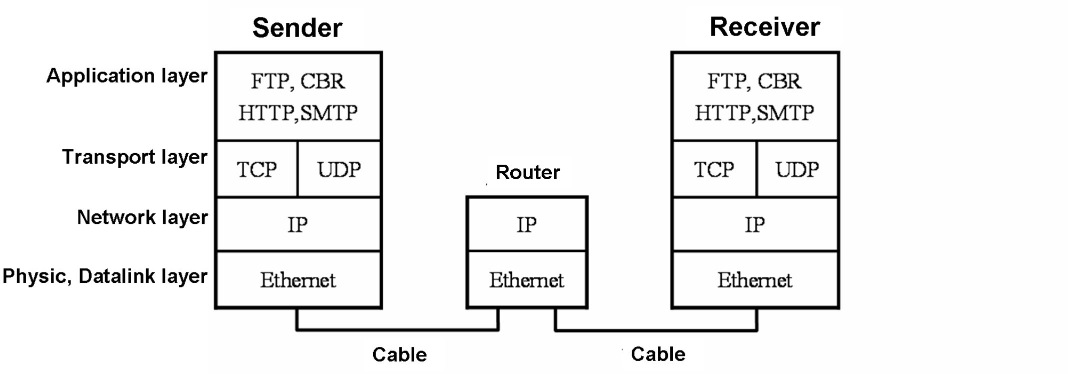


Figure 3: The realistic network

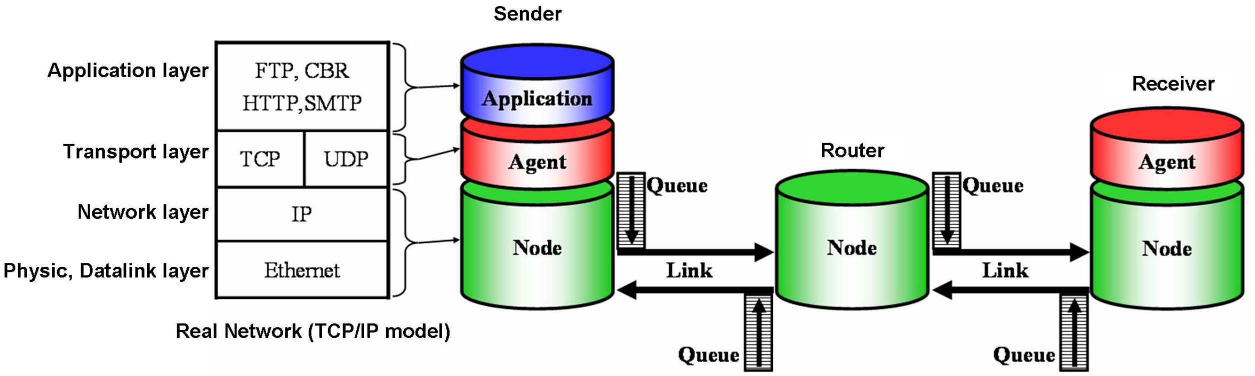


Figure 4: The network inside NS2

The node (terminal, router) in real network have 4 layer using TCP/IP model as shown in Fig. 3. Actually, forwarding router have only 2 bottom layer.

In the otherwise, 4 layers of TCP/IP model in NS2 are shown in Fig. 4. Two bottom layers are ready automatically when a node setting up. TCP or UDP in transport layer are shown as Agent. FTP, CBR are in application layers. Actually, the sender will set the application for creating data while the receive will not because it is no need to simulate the receiving data in application layer.

Realistic network use cable for link between 2 node. One cable is available for biconnection. In NS2, a link is use for one connection. Two lines are needed for biconnection. Each link has a queue which similar to buffer in realistic network. Packet, which is sent from node should be queuing in queue. When queue is empty, it will be send to other node via link.

The following is steps to creating a scenario file.

Step0 Declare Simulator and setting output file

Step1 Setting Node and Link

Step2 Setting Agent

Step3 Setting Application

Step4 Setting Simulation time and schedules
Step5 Declare finish.

3.2 Write Scenario File

In this chapter, we introduce the scenario of sample file `sample.tcl`.

3.2.1 Step0: Declare Simulator and setting output file

Declare simulator and setting output file is written as below.

```
--------------------------------------------------
set ns [new Simulator]
set file [open out.tr w]
$ns trace-all $file
set namfile [open out.nam w]
$ns namtrace-all $namfile
set tcpfile [open out.tcp w]
Agent/TCP set trace_all_oneline_ true
--------------------------------------------------
```

3.2.2 Step1: Setting Node and Link

Setting a node is shown as below.

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

then the node which have name as `n0` is ready to use. (in Tcl file it will refer to $n0). Node is numbered from 0. Link is declared as below.

```
--------------------------------------------------
$ns duplex-link $n0 $n2 3Mb 5ms DropTail
$ns duplex-link-op $n0 $n2 orient right-down
--------------------------------------------------
```

In line 1, the link between two nodes `n0` and `n2` has bandwidth `3Mbps` and delay `5ms`. A biconnection link between `n0` and `n2` is declared. `DropTail` is a waiting queue type, if the queue overflow then the new entered packet will be dropped, similar to buffer of general realistic network. Line 2 setting position of node and link for Nam. It does not affect to the result of simulation. The length of queue is set as below

```
--------------------------------------------------
$ns queue-limit $n2 $n3 20
$ns duplex-link-op $n2 $n3 queuePos 0.5
--------------------------------------------------
```

In line 1, the length of Queue on the link from `n2` to `n3` is `20`[packets]. In line 2, the position of Queue is set for executing simulation result on Nam, `0.5` is the angle between link and queue, it equal to (0.5π).

3.2.3 Step2: Setting Agent

UDP Agent

Using UDP for simulation, the sender set the Agent as UDP Agent while the receiver set to Null Agent. Null Agents do nothing except receiving the packet. Setting of Agent as flowing.

```
--------------------------------------------------
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 0
$ns color 0 blue
--------------------------------------------------
```

In first 4 lines, `udp, null` Agent is set for `n0, n3`. Line 5 declared the transmission between `udp` and `null`. Line 6 set the number for data flow of `udp`. This number will be recorded to all packet which are sent from `udp`. By using this number, we can easily observer the flow that packet is belong to by looking up trace file. Similarly, in line 7, we mark the color to discrete packet for showing result on Nam.

TCP Agent

Using TCP for simulation, the sender set the Agent as TCP Agent while the receiver set to TCPSink Agent. When receiving a packet, TCPSink Agent will reply an acknowledgment packet (ACK). Setting Agent for TCP is similar to UDP.

```
--------------------------------------------------
set tcp [new Agent/TCP]
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$ns color 1 red
--------------------------------------------------
```

Confirm that `fid_` and `color` is differ from `udp`.

Trace file of TCP is written as below.

```
--------------------------------------------------
$tcp attach-trace $tcpfile
$tcp trace cwnd_
--------------------------------------------------
```

3.2.4 Step3: Setting Application

In general, UDP Agent use CBR Application while TCP Agent use FTP Application.

```
--------------------------------------------------
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
set ftp [new Application/FTP]
$ftp attach-agent $tcp
--------------------------------------------------
```

3.2.5 Step4: Setting time schedule for simulation

Time schedule of a simulation is setting as below:

```
--------------------------------------------------
$ns at 1.0 "$cbr start"
$ns at 1.5 "$ftp start"
$ns at 3.0 "$ftp stop"
$ns at 3.5 "$cbr stop"
$ns at 4.0 "finish"
proc finish {} {
global ns file namfile tcpfile
$ns flush-trace
close $file
close $namfile
close $tcpfile
exit 0
}
--------------------------------------------------
```

`cbr` transmit data from `1.0`[sec] to `3.5`[sec] and `ftp` transmit data from `1.5`[sec] to `3.0`[sec]. The `finish` function is use for output data file which always at the end of simulation.

3.2.6 Step5: Declare finish

After finish setting, declare finish is written at the end of file.

```
--------------------------------------------------
$ns run
--------------------------------------------------
```

4 Execute Simulation and start Nam

Execute by blow command line, simulation will be started.

```
--------------------------------------------------
ns sample.tcl
--------------------------------------------------
```

After simulation finish running, Nam will be started and show the animation of simulation.

```
--------------------------------------------------
nam out.nam
--------------------------------------------------
```

5 View trace file (out.tr)

Below 3 lines is apart of a trace file.

```
--------------------------------------------------
+ 1.825127 2 3 tcp 1040 ---A--- 1 1.0 3.1 30 303
- 1.825367 2 3 cbr 210 ------- 0 0.0 3.0 203 274
r 1.82558 3 2 ack 40 ------- 1 3.1 1.0 29 302
--------------------------------------------------
```

Each line shown the information of a packet. We will explain the meaning of each column.

Column 1 The name of event

`+` : Packet enter to a Queue

`-` : Packet exit a Queue

`r` : Packet is received by a node

`d` : Packet is drop

Column 2 Time for event in second[sec]

Column 3, 4 Position of event (Link). Number in column 3, 4 is the number of node the event is occurred when data send from node number in column 3 to node number in column 4.

Column 5 The type of Packet

`cbr` : Packet created by CBR Application, sent by UDP Agent

`tcp` : Packet created by FTP Application, sent by TCP Agent

`ack` : Packet created by TCPSink Agent (ACK)

Column 6 Packet size [bytes].

Column 7 Flag: using for first retransmit packet in retransmission control `A` is Flag value. Actually, the flag isn't shown for all retransmission packet.

Column 8 Flow number which packet belong to(refer to 3.2.3).

Column 9 Packet sender. For example, if the value is `1.0`, it means that node which have ID equal 1 send data from port 0.

Column 10 Packet receiver. For example, if the value is `3.1`, it means that node which has ID equal 3 receive data by port 1

What is a port?

In realistic network, a computer have to receive data from many service (Web, Mail...). In NS2, there are many Agent will be set. For this reason, if we only record the ID of sender and receiver, we will not know the packet belong to which Agent. Using different port for each Agent is a way to solve this problem. Actually, in relistic network, individual port is used for each service.

Column 11 The sequence number of transmit packet

Colum 12 The ID number of that Packet. Each packet have a individual ID number.

In the flowing 2 lines.

```
--------------------------------------------------
+ 1.70342 2 3 tcp 1040 ------- 1 1.0 3.1 30 231
d 1.70342 2 3 tcp 1040 ------- 1 1.0 3.1 30 231
--------------------------------------------------
```

At time `1.70342`[sec], packet enter the queue and then be drop at the same time. It means that, packet try to enter to the queue and be drop because the queue is overflow.

6 Trace File (out.tcp)

The below is a part of TCP trace file (actually, it is one line).

```
----------------------------------------------------------------------------------------
time: 1.57005  saddr: 1  sport: 0  daddr: 3  dport: 1  maxseq: 2  hiack: 1  seqno: 3
cwnd: 3.000  ssthresh: 20  dupacks: 0  rtt: 0.040  srtt: 0.030  rttvar: 0.015  bkoff: 1
----------------------------------------------------------------------------------------
```

Each value is the value of TCP parameter when packet is send at sender. The meaning of each parameter is explained below.

`time`	Packet send time
`saddr`	ID number of sender
`sport`	Port number of sender
`daddr`	ID number of receiver
`dport`	Port number of receiver
`maxseq`	Maximum sequence number after sending
`hiack`	Maximum sequence number after receiving
`seqno`	Sending sequence number
`cwnd`	Congestion Window Size
`ssthresh`	Slow Start Threshold value
`dupacks`	Number of duplicate ACK
`rtt`	Round trip time delay
`srtt`	Smoothing of `rtt`
`rttvar`	Average deviation of `srtt`
`bkoff`	times of exponential back-off

To understand about Congestion Control of TCP, parameter `seqno, cwnd, ssthresh` is very important. Pay attention to these parameter. In this lecture, we will not concern about `srtt, rttvar, bkoff`.

7 appendix : full text of sample.tcl

```
##### Declare Simulator
set ns [new Simulator]

##### Setting output file
set file [open out.tr w]
$ns trace-all $file
set namfile [open out.nam w]
$ns namtrace-all $namfile
set tcpfile [open out.tcp w]
Agent/TCP set trace_all_oneline_ true

##### Setting Node
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

##### Setting Link
$ns duplex-link $n0 $n2 3Mb 5ms DropTail
$ns duplex-link $n1 $n2 3Mb 5ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
$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
$ns duplex-link-op $n2 $n3 queuePos 0.5

##### Setting Queue Length
$ns queue-limit $n2 $n3 20

##### Setting UDP Agent
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 0
$ns color 0 blue

##### Setting CBR Application
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp

##### Setting TCP Agent
set tcp [new Agent/TCP]
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
```

```
$tcp set fid_ 1
$ns color 1 red
### Setting output file of TCP Agent
$tcp attach-trace $tcpfile
$tcp trace cwnd_

##### Setting FTP Application
set ftp [new Application/FTP]
$ftp attach-agent $tcp

##### Setting time schedule of simulation
$ns at 1.0 "$cbr start"
$ns at 1.5 "$ftp start"
$ns at 3.0 "$ftp stop"
$ns at 3.5 "$cbr stop"
$ns at 4.0 "finish"
proc finish {} {
    global ns file namfile tcpfile
    $ns flush-trace
    close $file
    close $namfile
    close $tcpfile
    exit 0
}

##### Finish setting and start simulation
$ns run
```