S-38.180 – Quality of Service in Internet

Introduction to the exercises

Timo Viipuri

22.9.2004

Exercise Subjects

1) General matters in doing the exercises
    - Work environment
    - Making the exercises and returning the reports
2) Introduction to NS-2 Network Simulator
    - Basic understanding on how to work with it

Work Environment

- Exercises held in Maari-c
  - http://www.hut.fi/cc/computers/Maari-C.html
- NS-2 is used in most of the exercises
  - You can use it in any of Computing Centre's Linux-computers
    - http://www.hut.fi/atk/luokat/ ("unix")
    - A modified version of NS-2 is installed there
    - ➔ the exercise simulations won't work anywhere else
  - Can be used locally or with SSH

Exercises

- Exercise schedule and info at course home page:
  - http://www.netlab.hut.fi/opetus/s38180/2004/schedule.shtml
- Each exercise session (2 hrs) consists of:
  - (Review of the previous exercise)
  - Introduction to the new exercise
  - Begin work on the simulations with course staff present

Exercise Reports

- Two hard deadlines:
  - Exercises 1-4: **October 29th, 4 pm**
  - Exercises 5-6: **November 3rd, 4 pm**
- It is advised to return reports before the next exercise
  - Return format is either **PDF** or **paper**
  - Late returns are automatically discarded!
- Total exercise points are scaled to 1-6
  - Used in the exam grading to replace the points from the lowest scoring answer

S-38.180 – Quality of Service in Internet

Exercise 1: NS-2 Network Simulator

Timo Viipuri

22.9.2004

Exercise Objectives

- To familiarize yourself with the work environment
- To learn to work with NS-2 at the level that you can:
  1. Write simple simulation scripts
  2. Read and understand more complex simulation scripts

Tasks of the Day

1. A few words about the background and structure of NS-2
   - to give you some idea of what you are working with
2. Line-by-line study of a simple simulation scenario
   - to explain the minimum requirements needed to create a simulation
3. Begin making your own simulation

NS-2 Forewords

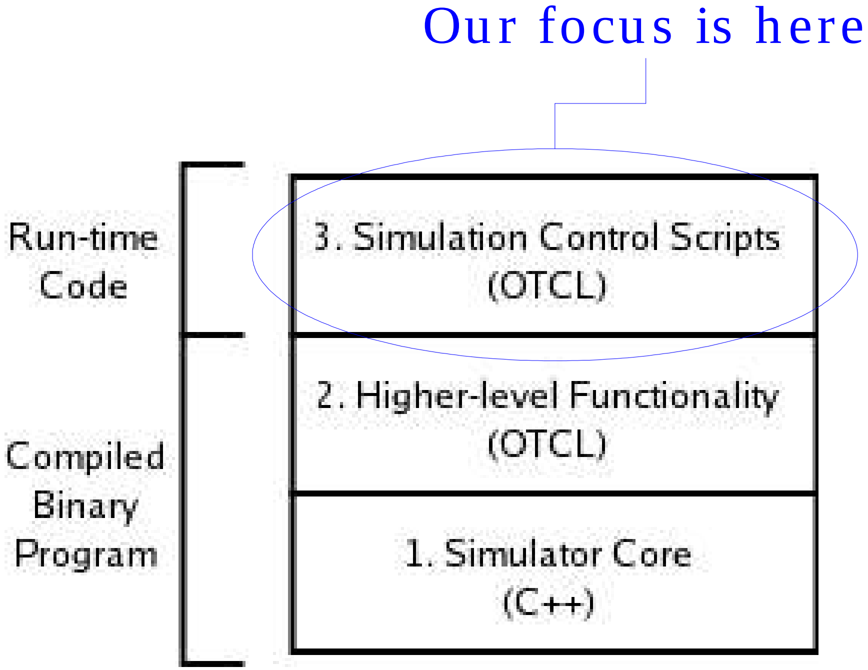
- Open source software
  - Possible to tailor the code to exactly fit the needs
  - Thousands of developers => rapid increase in functionality
  - No one is liable for the code => use at your own risk
- Nowadays it is argueably the most popular network simulator in the world
  - Used extensively by both businesses and universities

NS-2 Software Structure

- NS-2 uses two programming languages to combine efficiency and ease of extentability
  - C++
  - OTCL (Object Tool Command Language)
- NS-2 software is written in both C++ and OTCL
  - Generally doesn't need to be modified
- Simulation scripts are written in OTCL
  - Used to set up and control the simulation

NS-2 Software Structure 2

- Simulator software is separated to 3 layers:
  1. Basic functionality: C++
  2. Experimental protocols and complex applications: OTCL
  3. Simulation control scripts: OTCL



Simulation Scripts

- Used to set up a simulation scenario:
  - Network topology
  - Traffic agents
  - Simulation events, e.g. when to start sending data
  - Gathering results: monitoring and tracing
- Written in OTCL
  - No need to compile; scripts are interpreted at run-time
- For help in writing simulation scripts, refer to NS-2 manual
  - http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf

Simulation Example (1)

- Topology
  - A network of two nodes connected with a duplex link
    - Bandwidth: 5 Mbps
    - Packet delay: 10 ms
- Traffic agents
  - 1 TCP-connection
  - 1 UDP-connection with a CBR-traffic generator
- Simulation events
  - TCP starts sending 15 kB of data at 0.5 s
  - UDP starts sending at a rate of 800 kbps at 0.2 s and stops at 0.8 s
- Gathering data
  - Monitor traffic flows

Example: Topology (2)

- Create nodes n0 and n1

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

Create a node and assign it to variable n0

Assign a variable n0

```
set n1 [$ns node]
```

- Create a duplex-link between the nodes

```
$ns duplex-link $n0 $n1 5Mb 10ms DropTail
```

Call procedure 'duplex-link' of object $ns

Bandwidth 5Mbps, delay 10ms

Buffer management method: DropTail

Set link between nodes n1 and n2

Example: UDP-agents (3)

- Create UDP- and null-agents

```
set udp0 [new Agent/UDP]
set null0 [new Agent/Null]
```

A null-agent acts as an UDP-sink

- Attach them to nodes n0 and n1

```
$ns attach-agent $n0 $udp0
$ns attach-agent $n1 $null0
```

Parameters: $node $agent

- Connect the agents

```
$ns connect $udp0 $null0
```

Parameters: $agent $agent

(NS-2 manual: “30: UDP Agents”)

Example: CBR-traffic (4)

- Create a CBR traffic source

```
set cbr0 [new Application/Traffic/CBR]
```

Application type

- Set traffic parameters

```
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
```

$$\Rightarrow \text{Send Rate} = \frac{8 * 500\ \text{b}}{0.005\ \text{s}} = 800\ \text{kbps}$$

Time interval
between packets

- Attach the traffic generator to an agent

```
$cbr0 attach-agent $udp0
```

Example: TCP-agents (5)

- Create a TCP-connection pair

```
set src [new Agent/TCP/FullTcp]
set sink [new Agent/TCP/FullTcp]
```

FullTcp includes a three-way handshake and a connection tear-down

- Attach agents to nodes

```
$ns attach-agent $n0 $src
$ns attach-agent $n1 $sink
```

- Connect the agents

```
$ns connect $src $sink
```

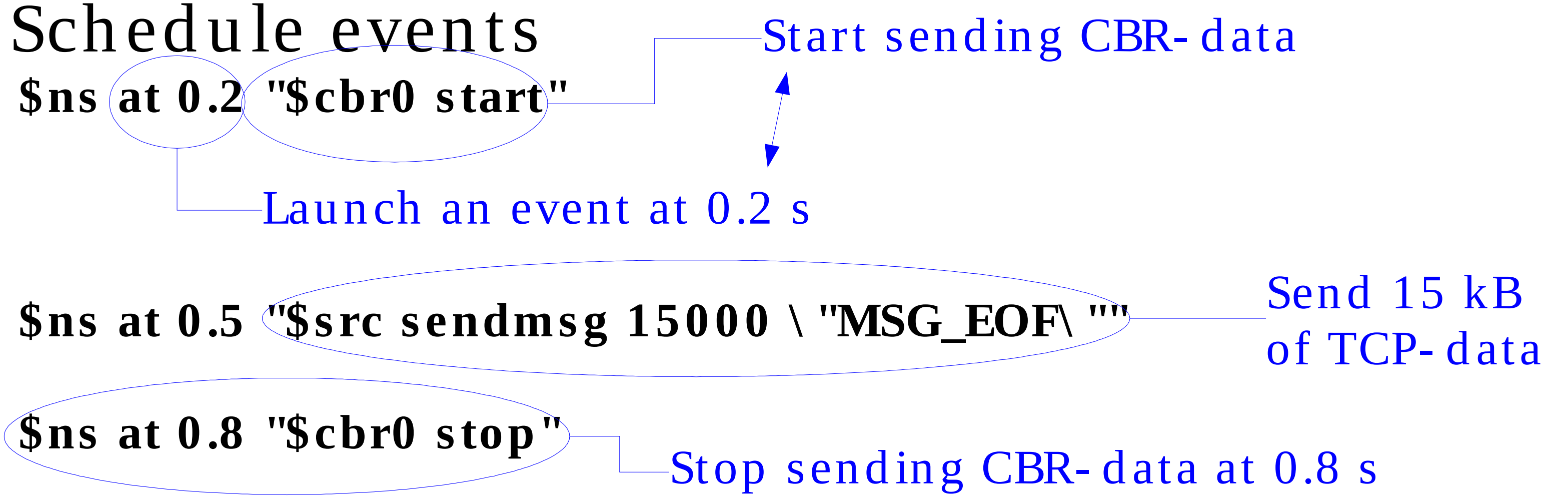
- Assign the *sink*-agent to listening mode (*src* initiates the connection)

```
$sink listen
```

(NS-2 manual: “31.3 Two-Way TCP Agents (FullTcp)”)

Example: Events (6)

- Schedule events



```
$ns at 0.2 "$cbr0 start"
```

Start sending CBR-data

Launch an event at 0.2 s

```
$ns at 0.5 "$src sendmsg 15000 \ "MSG_EOF\ ""
```

Send 15 kB of TCP-data

```
$ns at 0.8 "$cbr0 stop"
```

Stop sending CBR-data at 0.8 s

- Call the finish procedure after 1.0 s of simulation time

```
$ns at 1.0 "finish"
```

- Start the simulation in the end of the script

```
$ns run
```

Example: Monitoring (7)

- Create a flow monitor

```
set flow_mon [$ns makeflowmon Fid]
```

Use flow ID's to identify different flows

- Attach the flow monitor to the link

```
$ns attach-fmon [$ns link $n1 $n0] $flow_mon 0
```

Attach the monitor between nodes $n1 and $n0

- Assign an output file

```
$flow_mon attach [open output_file.fmon w]
```

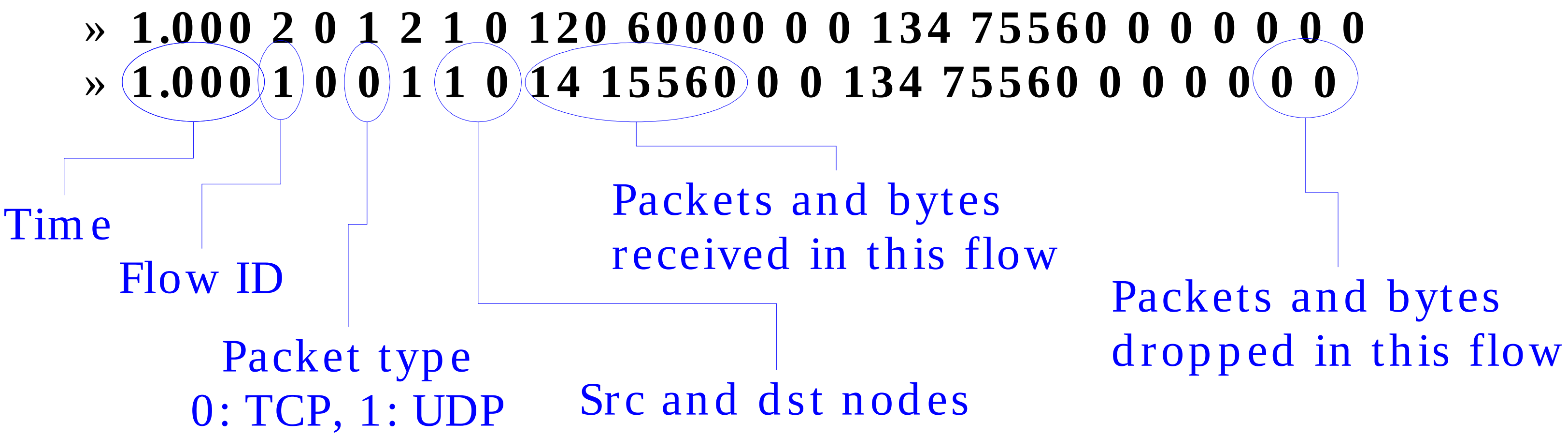
- Print the statistics at given time

```
$ns at 1.0 “$flow_mon dump”
```

  - Hint! You can put the quoted command in the finish-procedure

Example: Results (8)

- Sample of the flow monitor output (with 2 flows):



```
» 1.000 2 0 1 2 1 0 120 60000 0 0 134 75560 0 0 0 0 0 0
» 1.000 1 0 0 1 1 0 14 15560 0 0 134 75560 0 0 0 0 0 0
```

Time

Flow ID

Packet type
0: TCP, 1: UDP

Src and dst nodes

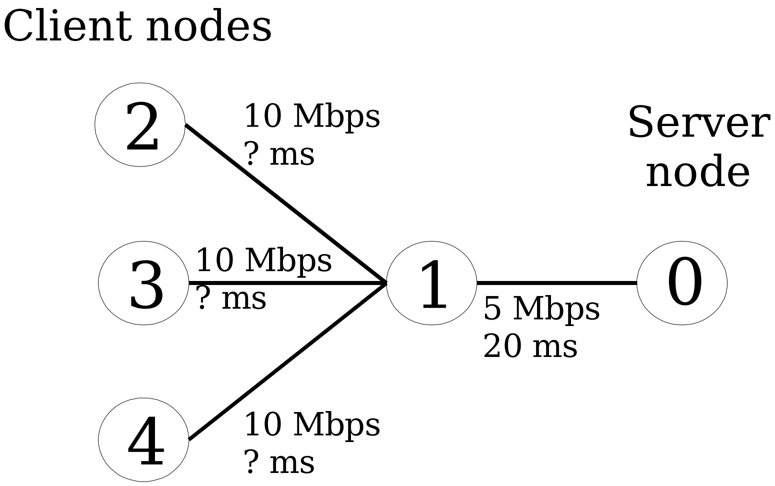
Packets and bytes
received in this flow

Packets and bytes
dropped in this flow

(NS-2 Manual: “23.7.2 Flow monitor trace format”)

Simulation: Link Delay

- Topology
  - 1 FTP server node
    - 3 server agents in node 0
  - 3 FTP client nodes
    - 3 client agents in nodes 2-4
- Task: Study the effect of link delay to the throughput of a TCP-connection



Random Numbers

- NS-2 produces only pseudo-random numbers
  - they aren't random but only appear to be
- A seed value is needed for the generation of pseudo-random numbers
  - If the seed value is the same the number sequence generated will be the same
  - Modified with: *"$defaultRNG seed 1"*,
  - using seed 0 will cause a random seed to be generated on each new simulation
- e.g. RED uses random numbers to calculate the drop probability
- NS-2 manual: “22.1 Random number generation”

NS-2 Material

- Development pages:
  - http://www.isi.edu/nsnam/ns
  - Especially useful topics:
    - "Mark Greis's NS-2 tutorial"
    - "Ns manual"
  - Visit them!
- TCL tutorials
  - http://users.belgacom.net/bruno.champagne/tcl.html
  - http://hegel.ittc.ukans.edu/topics/tcltk/tutorial-noplugin
- OTCL tutorial
  - http://www.openmash.org/developers/docs/otcl-doc/doc/tutorial.html