# PERCOBAAN 2

# PEMROGRAMAN TCL SEDERHANA PADA NS2

## 1.1. Tujuan:

Setelah melaksanakan praktikum ini mahasiswa diharapkan mampu:

- Membuat pemrograman simulasi jaringan sederhana menggunakan Tcl
- Menjalankan program Tcl pada NS2
- Menampilkan hasil simulasi dengan nam, tr dan xgraph

## 1.2. Peralatan:

- OS Ubuntu 10.04
- Simulator NS-2.34

## 1.3. Teori

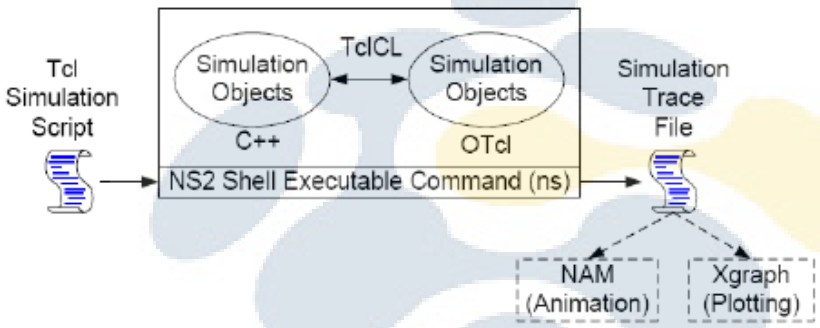
Proyek Network Simulator (NS) pada awalnya merupakan varian dari Simulator Jaringan REAL pada tahun 1989. Selanjutnya pada tahun 1995 disupport oleh *Agency for Advanced Research Projects* USA melalui proyek VINT, yang merupakan joint project dari *National Laboratory Lawrence Berkeley* di Palo Alto Research Center dengan University of California at Berkeley. Versi 2 dari Network Simulator (untuk selanjutnya dinamakan NS-2) ini menggunakan script TcL/Tk untuk mengontrol simulasi dan C++ sebagai modul infrastruktur dari jaringan yang disimulasikan.

NS-2 adalah free software yang dapat didownload dalam bentuk ns-allinone-2.*.tar.gz pada beberapa link yang tersedia. Pada praktikum ini kami menggunakan ns-allinone-2.34.tar.gz yang di-download dari http://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.34/. NS-2 ini dapat dijalankan pada operating system Windows maupun Linux. Untuk Operating System Windows dapat menggunakan WindowsXP atau Windows 7, sedangkan untuk Linux dapat menggunakan Ubuntu 10.04 atau versi di atasnya. Karena NS-2 pada dasarnya dijalankan dengan Linux, maka untuk menjalankan NS-2 pada Windows diperlukan software tambahan yang menjadi interpreter antara Windows dengan Linux, yaitu Cygwin. Untuk mendownload dan menginstall Cygwin dapat mengunjungi: https://cygwin.com/install.html.

Pada praktikum sebelumnya telah dijelaskan cara instalasi NS-2 pada Ubuntu 10.04 hingga NS-2 siap digunakan. Pada praktikum kedua ini mahasiswa akan belajar cara pemrograman Tcl dan mensimulasikan hasil pemrograman pada NS-2.

### 1.3.1. Konsep Dasar NS2

NS-2 dibangun dari 2 bahasa pemrograman: library-library dengan Bahasa pemrograman C++ yang digunakan untuk event scheduler, protocol dan network components, dan Tcl/OTcl yang merupakan Bahasa pemrograman untuk menulis script simulasi. Hubungan antara input simulasi, proses eksekusi dan output simulasi dengan kedua Bahasa pemrograman tersebut di atas ditunjukkan sebagai batang tubuh NS-2, seperti pada gambar 1.1.




Gambar 1.1. Batang tubuh NS-2

Komponen-komponen NS-2 terdiri dari:

1. NS sebagai simulator
2. NAM, sebagai network animator bertugas untuk mem-visualisasikan output dari NS-2. Editor NAM berupa interface GUI yang dipanggil sebagai file ber-ekstensi .nam pada script Tcl.
3. Pre-processing, bertugas membangkitkan trafik dan topologi jaringan
4. Post-processing, berupa analisa hasil simulasi yang ditampilkan pada file .tr dimana sebagian dari hasl simulasi tersebut dapat di-filter menggunakan perintah awk dan dapat dikonversikan dalam bentuk grafik dengan tool XGraph. Penjelasan untuk pem-filter an data simulasi ini secara detail ada di bagian akhir petunjuk praktikum.

### 1.3.2. Langkah-langkah dalam Membangun Simulasi dengan NS-2

1. Buat simulator object dan event scheduler menggunakan tcl programming

2. Buat topologi jaringan
3. Definisikan pola trafik
4. Definisikan trace file
5. Atur jalannya scenario simulasi, traffic flow, trace dan event-event lainnya.
6. Olah data hasil trace file menjadi data yang siap di-plot dan untuk memudahkan analisa

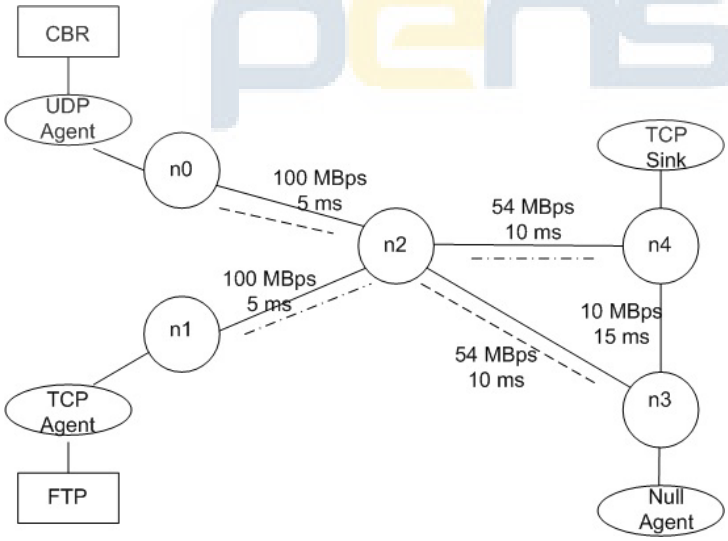
## 1.4. Prosedur Percobaan.

### 1.4.1. Membangun simulasi jaringan Wired dengan NS-2

Skenario simulasi yang akan dibangun ditunjukkan pada gambar 1.2 dengan penjelasan sebagai berikut:

Sebuah jaringan wired terdiri dari 4 node (n0, n1, n2 dan n3). Jalur antara n0 dan n2 adalah duplex dengan bandwidth 100 Mbps, delay 5 ms dan queue tipe DropTail. Hal yang sama berlaku untuk jalur dari n1 ke n2. Di antara n2 dan n4 dibangun jalur duplex dengan bandwidth 54 Mbps delay 10 ms dan DropTail queue, begitu pula dari n2 dan n3. Sementara itu di antara n3 dan n4 ada jalur simplex dengan bandwidth 10 Mbps dan delay 15 ms.

Akan dialirkan paket aplikasi CBR dari node n0 ke n3 yang melalui protokol UDP dengan node n3 adalah Null agent. Sementara paket FTP dialirkan dari node n1 ke n4 melalui protocol TCP, dengan node n4 adalah TCPSink. Simulasi dijalankan selama 6 detik. Pada detik ke 0,5 paket ftp mulai dikirimkan dan berakhir pada detik ke 4,5. Pada detik ke 2 paket cbr dikirimkan dan berakhir pada detik ke 5.




Gambar 1.2. Skenario Simulasi Jaringan Wired

Langkah-langkah Pemrogram Tcl

1. Buka file tcl baru dengan perintah gedit. Misalkan nama file adalah fileku.tcl, maka
   **# gedit fileku.tcl**
2. Pada baris teratas dari file tersebut, definisikan nama simulator. Dalam latihan ini namanya **ns**

```
# Create a Simulator
set ns [new Simulator]
```

3. Buat variable obyek-obyek trace, yaitu **traceku** yang akan dijadikan output file .tr dengan nama *out1.tr*, dan **NAMku** yang akan dijadikan output file .nam dengan nama *out1.nam*. Nama-nama variable obyek dan file output terserah anda.

```
# Create trace object
set traceku [open out1.tr w]
$ns trace-all $traceku

# Create a NAM trace file
set NAMku [open out1.nam w]
$ns namtrace-all $NAMku
```

4.Definisikan prosedur “finish”, yaitu prosedur mengakhiri simulasi

```
# Define a finish  procedure
proc finish {} {
      global ns traceku NAMku
      $ns flush-trace
      close $traceku
      close $NAMku
      puts "running nam..."
      exec nam out1.nam &
      exit 0
}
```

5. Bangkitkan node-node yang diperlukan

```
# Create nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
```

6. Koneksikan masing-masing node dengan duplex link atau simplex link dengan bandwidth dan delay yang telah ditentukan pada scenario.

```
# Connect each node with duplex link
$ns duplex-link $n0 $n2 1Mb 5ms DropTail
$ns duplex-link $n1 $n2 1Mb 5ms DropTail
$ns duplex-link $n2 $n4 5Mb 10ms DropTail
$ns duplex-link $n2 $n3 5Mb 10ms DropTail
$ns simplex-link $n3 $n4 0.5Mb 15ms DropTail
$ns queue-limit $n2 $n3 4
```

7. Gunakan protocol UDP untuk jalur node n0 sampai n3, dimana n0 sebagai asal aliran UDP dan n3 sebagai akhir (null) dari UDP

```
# Create a UDP flow from n0 to n3
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_ 1
```

8. Letakkan aplikasi CBR di atas jalur UDP. Besar paket 500 byte dan interval antar paket 0,005 ms.

```
# Attach CBR source to the UDP flow
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packetSize_ 500
$cbr set interval_ 0.005
```

9. Gunakan protocol TCP untuk jalur node n1 sampai n4, dimana n1 sebagai asal aliran TCP dan n4 sebagai akhir dari TCP (sebagai TCPSink).

```
# Create a TCP flow from n1 to n4
set tcp [new Agent/TCP]
$ns attach-agent $n1 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 2
```

10. Letakkan paket FTP di atas jalur TCP.

```
# Attach FTP source to the TCP flow
set ftp [new Application/FTP]
$ftp attach-agent $tcp
```

11. Buat event scheduler, dimana pada t=0,5 paket ftp diaktifkan dan dihentikan pada t=4,5. Sedangkan pada t=2 paket CBR diaktifkan dan dihentikan pada t=5.

```
# Schedule Events
$ns at  0.5 "$ftp start"
$ns at 2.0 "$cbr start"
$ns at 4.5 "$ftp stop"
$ns at 5.0 "$cbr stop"
```

12. Outputkan pada CLI editor besar paket dan interval antar paket

```
# Put the information to the CLI editor
puts [$cbr set packetSize_]
puts [$cbr set interval_]
```

13. Akhiri proses simulasi

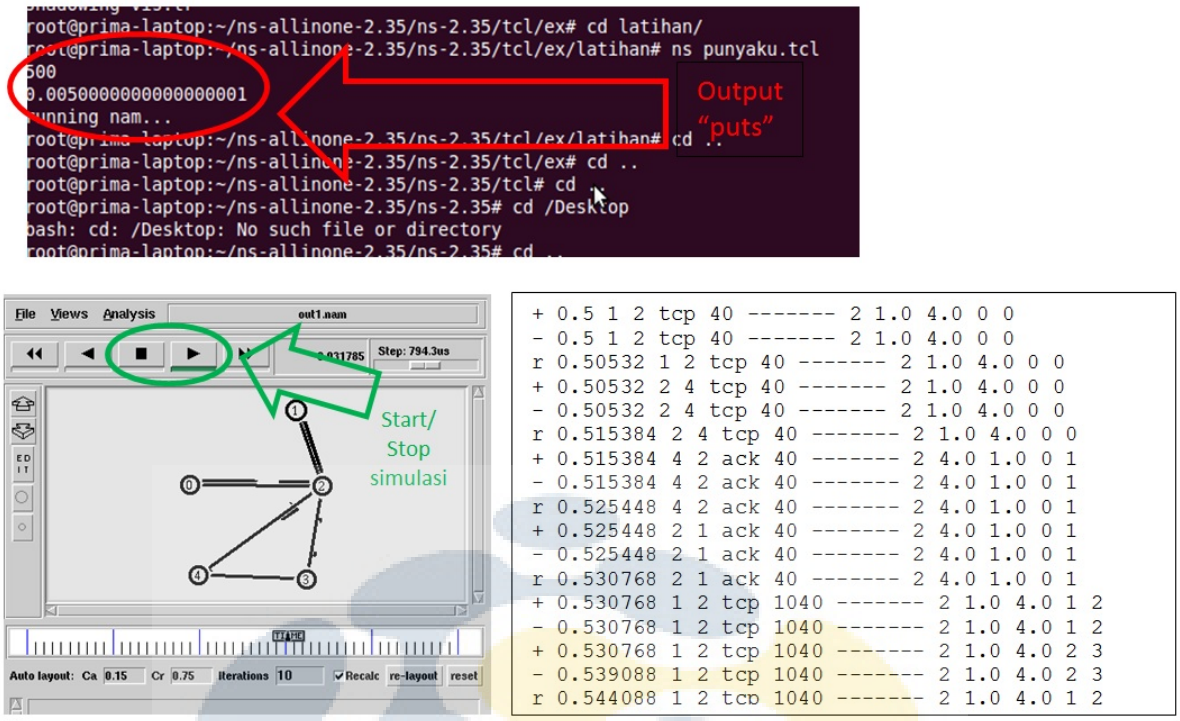
```
$ns at 6.0 "finish"

$ns run
```

14. Setelah seluruh syntax tersebut ditulis, simpan. Pada direktori dimana fileku.tcl berada, compile file tersebut dengan mengetikkan:

```
# ns fileku.tcl
```

15. Jika tidak ada kesalahan dalam penulisan, maka akan ada 3 jenis output yang bisa dilihat saat simulasi dijalankan, yaitu output pada editor CLI yang dikeluarkan dengan perintah “puts”, output pada out1.tr yang berisi data-data hasil simulasi dan output pada out1.nam yang merupakan animasi simulasi seperti ditunjukkan pada gambar 1.3.




Gambar 1.3. Output-output dari simulasi dengan NS-2
Atas: Output "puts" pada editor CLI
Bawah kiri: Output NAM (out1.nam) Bawah kanan: Output trace file (out1.tr)

### 1.4.2. Menganalisa Hasil Simulasi dengan NS-2

Hasil simulasi dapat dianalisa berdasarkan data yang tersimpan di file .tr, dimana struktur data yang tersimpan pada file .tr memiliki aturan seperti pada gambar 1.4.

| Type Identifier | Time | Source Node | Destination Node | Packet Name | Packet Size | Flags | Flow ID | Source Address | Destination Address | Sequence Number | Packet Unique ID |
|---|---|---|---|---|---|---|---|---|---|---|---|

*Enque (+), deque(-), receive(r), drop(d)*

Gambar 1.4. Aturan mengenai isi dari file .tr

#### *1.4.2.1.Parsing data dengan AWK*

Data yang tersimpan pada file .tr bisa difilter sesuai dengan kebutuhan kita. Proses pemfilteran data pada trace file dinamakan *parsing*. Sebagai contoh, dari data trace file yang telah disimpan pada out1.tr hanya akan diambil data waktu dan besar paket data setiap simulasi,

seperti yang ditampilkan pada huruf dengan warna merah pada gambar 1.5. Ketika data-data yang diharapkan tersebut sudah di-parsing, maka data tersebut dapat dianalisa dengan mem-plot ke dalam bentuk grafik.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| r | 4.254.432 | 2 | 1 | ack | 40 | ------- | 2 | 4.0 |
| + | 4.254.432 | 1 | 2 | tcp | 1040 | ------- | 2 | 1.0 |
| r | 42.548 | 2 | 3 | cbr | 500 | ------- | 1 | 0.0 |
| + | 4.255 | 0 | 2 | cbr | 500 | ------- | 1 | 0.0 |
| - | 4.255 | 0 | 2 | cbr | 500 | ------- | 1 | 0.0 |
| - | 4.255.664 | 1 | 2 | tcp | 1040 | ------- | 2 | 1.0 |
| r | 4.255.688 | 2 | 4 | tcp | 1040 | ------- | 2 | 1.0 |
| + | 4.255.688 | 4 | 2 | ack | 40 | ------- | 2 | 4.0 |
| - | 4.255.688 | 4 | 2 | ack | 40 | ------- | 2 | 4.0 |
| r | 4.257.432 | 4 | 2 | ack | 40 | ------- | 2 | 4.0 |

Gambar 1.5 Data-data yang akan diparsing berwarna merah

Salah satu model parsing data yang sering digunakan adalah dengan paket AWK. Pastikan paket AWK sudah diinstall pada Ubuntu anda. AWK dijalankan pada user biasa bukan super user. Untuk menjalankan AWK ini anda harus pindah dari posisi superuser ke user biasa dengan mengetikkan # **exit**, sehingga menuju ke root $.

Pemrograman AWK sangat sederhana, dimana aturan pemrogramannya adalah sebagai berikut:

```
$ awk '{script awk}' file asal >> file tujuan
```

Perhatian:

Untuk menjalankan program AWK yang harus berada di level user biasa, maka file-file yang terlibat (sebagai file input maupun outputnya) harus berada di dalam direktori /home. Oleh karena itu, pindahkan file-file yang akan anda kerjakan menggunakan AWK maupun saat plotting dengan Xgraph pada direktori /home.

Sebelum melakukan parsing, jadikan file yang akan diparsing itu menjadi data input pada level user. Lakukan perintah: **$ cat file_asal**

Kemudian lakukan proses AWK dan Xgraph tersebut pada level user.

Contoh:

Jika ingin mengambil data pada kolom 2 dan kolom 6 saja dari file asal **out1.tr** yang ada pada direktori /home dan menyimpan hasilnya pada file tujuan **outa.txt**, maka cukup menuliskan:

```
$ awk '{print $2,$6}' out1.tr >> outa.txt
```

Sehingga hasil yang didapatkan disimpan di dalam file outa.txt sbb:

| | |
|---|---|
| 4.254.432 | 40 |
| 4.254.432 | 1040 |
| 42.548 | 500 |
| 4.255 | 500 |
| 4.255 | 500 |
| 4.255.664 | 1040 |
| 4.255.688 | 1040 |
| 4.255.688 | 40 |
| 4.255.688 | 40 |
| 4.257.432 | 40 |
| 4.257.432 | 40 |

Pemfilteran dengan kondisi bisa dilakukan dengan AWK, sebagai contoh diinginkan hanya diambil data waktu ($2) dan ukuran paket ($6) pada data dari node 2 ($3==2) ke node 4 ($4==4) dalam kondisi data diterima ($1="r"). Syntax AWK nya adalah sbb:

```
$ awk '{if(($3==2)&&($4==4)&&($1=="r"))print $2,$6}' out1.tr >>outc.txt
```

Untuk menganalisa kualitas jaringan yang telah didisain, ada beberapa parameter yang bisa diukur diantaranya:

1.Jitter

2.Throughput

3.Delay

Cara menghitung ke-3 parameter tersebut dengan rumus sebagai berikut:

Inter arrival Jitter = $t_{terima} - t_{terima-1}$

Throughput=ukuran data yang diterima/waktu pengiriman data

Delay=jeda antara paket dikirim sampai paket diterima

Pada praktikum ini rumus jitter dan throughput dapat ditulis dalam bahasa AWK, dimana hasil proses yang disimpan pada file .awk dapat diimplementasikan pada file hasil parsing sebelumnya. Output dari proses tersebut disimpan pada file baru dimana file tersebut akan diplot menggunakan Xgraph.

Contoh: file awk untuk menghitung Akumulasi throughput

```
BEGIN{
sum=0;
}
{
        if($1>0)
        printf("%d\t%f\n",$1,sum/$1);
        sum=sum+$2
        }
END {
        puts "Done"
}
```

Beri nama **throughput.awk**

Aplikasikan file tersebut kepada file outc.txt dengan perintah di bawah dan disimpan di File1.

```
$ awk –f throughput.awk outc.txt>>File1
```

### *1.4.2.2. Plotting data dengan XGraph*

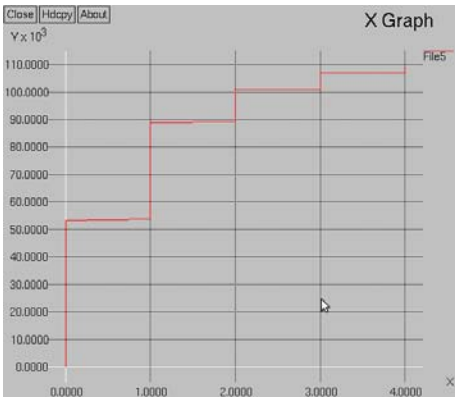
Untuk mem-plot menjadi grafik, file hasil parsing tadi diproses dengan perintah Xgraph. Pastikan paket Xgraph telah diinstall pada Ubuntu kita. Perintahnya adalah sbb:

```
$ xgraph nama_file
```

Nama file hasil dari proses penghitungan troughput di atas adalah File1. Plotlah file tersebut dengan perintah

```
$ xgraph File1
```

Hasil plotting dengan Xgraph ditunjukkan pada gambar 1.6




Gambar 1.6. Hasil plot xgraph untuk File1

### 1.5.Tugas dan Pertanyaan

Konfigurasilah sebuah jaringan dengan 3 node statis yang akan disimulasikan pada NS-2 dengan skenario sebagai berikut:

Node 0 adalah Agen TCP dengan paket FTP dialirkan ke node 2 yang menjadi TCPSink, node 1 juga merupakan agen TCP dengan paket FTP yang mengalir ke node 2 juga. Bandwidth tersedia antar node 0 dan 2 adalah 5 MB dengan delay 10 ms, sedangkan bandwidth dan delay antara node 1 dan node 2 adalah 3 MB dan 20 ms. Baik node 0 maupun node 1 berkomunikasi secara duplex dengan node 2. Jalankan paket-paket FTP pada detik ke 2,5 untuk node 0 dan 4 untuk node 1. Hentikan paket pada detik ke 8. Besar paket yang diijinkan adalah 1000 MB dan interval antar paket sebesar 0,02. Hentikan simulasi pada detik ke 25.

Outputkan hasil animasi pada sebuah file .nam dan hasil simulasi pada file .tr. Selanjutnya ambillah paket yang menuju node 2 dengan besaran > 900 saja, dan tampilkan hasilnya menggunakan grafik.

*Last updated: 26 October 2015 by Prima Kristalina*