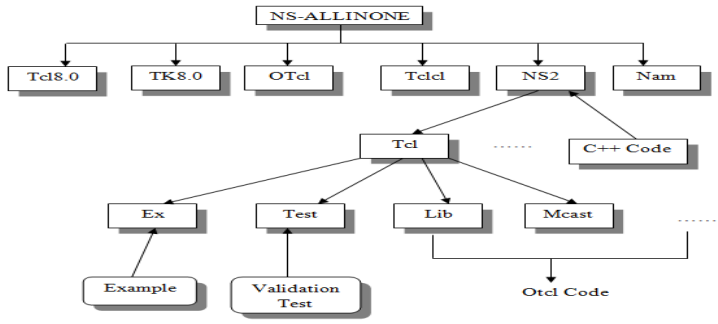
2.6.2 Komponen Pembangunan NS-2

Instaler NS versi NS-allinone berisi keseluruhan komponen wajib dan optional yang dibutuhkan oleh simulasi. Masing-masing komponen tersebut terdapat pada folder NS-allinone yang diinstal. Gambar 2.6 adalah gambar komponen-komponen pembangun NS-2.



Gambar 2.6 Komponen Pembangun NS-2

Keterangan:

1. TCL: Tool Command Language
2. Otcl: Object TCL
3. TK: Tool Kit
4. Tclcl: Tcl/C++ Interface
5. NS2: NS versi 2
6. Nam: Network Animator

2.6.3 Cara Membuat dan Menjalankan Skrip NS-2

Membuat skrip simulasi NS sangat mudah. Skrip simulasi bisa dibuat dengan menggunakan program teks *editor* yang ada pada linux, dan disimpan dalam sebuah folder dengan ekstensi .tcl.

Contoh:

Simulasi1.tcl

Berikut ini akan dijelaskan dasar-dasar bahasa Tcl yang berguna dalam membangun simulasi :

***Syntax* Dasar**

Syntax dasar perintah tcl yaitu :

```
Command Arg1 arg2 arg3 …
```

Command tersebut bisa berupa nama dari *built in command*, atau sebuah prosedur Tcl. Contoh :

```
Expr 2*3
Puts *ini adalah contoh command*
```

Variable* dan *Array

Untuk membuat *variable*, digunakan perintah `set`. Adapun contoh penggunaannya adalah sebagai berikut :

```
Set x *ini adalah contoh variable*
Set y 20
```

Pemanggilan *variabel* dilakukan dengan menggunakan tanda `$` seperti contoh di bawah ini :

```
Puts “ $x, semuanya berjumlah $y ”
```

Sedangkan pembuatan *array* ditandai dengan menggunakan tanda kurung setelah nama *array* tersebut yang dapat dituliskan sebagai berikut:

```
Set opts (bottleneckinkrate) 1.2 Mb
Set opts (ENC) *on*
Set n(0) [$ns node]
Set n(1) [$ns node]
```

Untuk menjalankan simulasi yang telah dibuat adalah dengan cara sebagai berikut :

[root @ accessnet your_folder]# ns Simulasi1.tcl

2.6.4 Tahap-tahap Membangun Simulasi NS-2

Pembangunan simulasi NS-2 dilakukan secara bertahap. Berikut ini merupakan contoh tahap-tahap dasar pembuatan simulasi NS-2 :

Langkah 1: Mendefinisikan *Variable* Global

Menurut Wirawan & Indarto (2004), dalam membangun simulasi JSN apabila membutuhkan *variable-variable* global yang akan digunakan oleh keseluruhan program, maka *variable-variable* harus didefinisikan terlebih dahulu dengan menggunakan perintah `set<spasi>nama`*`variabel`*`(identitas`*`variabel`*`) <spasi>`*`value`*. Contoh pendefinisian tersebut adalah sebagai berikut :

```
set val(chan) Channel/WirelessChannel;
set val(prop) Propagation/TwoRayGround;
set val(netif) Phy/WirelessPhy;
set val(mac) Mac/802_11;
set val(ifq) Queue/DropTail/PriQueue;
set val(ll) LL;
set val(ant) Antenna/OmniAntenna;
set val(ifqlen) 50;
set val(nn) 5;
set val(rp) AODV;
set val(seed) 0;
set val(x) 300;
set val(y) 300;
set val(stop) 200;
set val(mobility) Static;
```

Variable-variable yang didefinisikan di atas memiliki fungsi yang berbeda-beda sesuai dengan kebutuhan pembuatan simulasi. Keterangan dari *variable-variable* di atas adalah sebagai berikut :

1. Chan

   Merupakan tipe *channel* yang digunakan dalam simulasi, seperti *channel wireless*.

2. Prop

   Merupakan model propagasi. Model propagasi bisa bernilai *OneWayGround* atau *TwoWayGround*.

3. Netif

   Merupakan tipe jaringan *wireless* yang digunakan.

4. Mac

   Merupakan tipe MAC yang digunakan sesuai dengan *channel* yang digunakan.

5. Ifq

   Merupakan tipe antarmuka antriannya, yang menunjukkan perlakuan *node* terhadap paket apabila memori yang digunakan telah penuh.

6. Ll

   Merupakan T\tipe *link layer*.

7. Ant

   Merupakan model antena *node* yang digunakan.

8. Ifqlen

   Merupakan ukuran maksimum antrian paket.

9. Nn

   Merupakan jumlah *node* yang digunakan.

10. Rp

    Merupakan protokol *routing* yang digunakan.

11. Seed

    Merupakan nilai s*eed* yang digunakan sebagai nilai awal dari penggunaan nilai *random*.

12. X

    Merupakan nilai topografi x.

13. Y

    Merupakan Nilai topografi y.

14. Stop

    Merupakan nilai waktu dimana simulasi akan dihentikan.

15. Mobility

    Merupakan mobilitas dari *node* apakah pergerakan *node* bersifat statis atau dinamis.

Langkah 2: Inisialisasi Simulasi

Menurut Wirawan & Indarto (2004), untuk memulai pembuatan simulasi sederhana, dapat menggunakan salah satu teks editor yang ada pada linux yang digunakan. Kemudian *file* tersebut disimpan dalam sebuah folder.

Simulasi NS dimulai dengan menuliskan skrip Tcl seperti di bawah ini :

```
#memanggil simulator object
Set ns [new Simulator]
#open file handle untuk simulator nam  trace data
Set nf [open out.nam w]
$ns namtrace-all $nf
#prosedur finish berguna untuk menyelesaikan simulasi
Proc finish {} {
#menutup file dan memulai nam (network animator)
Global ns nf
$ns flush-trace
Close $ns
Exec nam out.nam &
Exit 0
}
#mengeksekusi prosedur finish pada saat detik ke 5.0
$ns at 5.0 “finish”
#menjalankan simulasi
$ns run
```

Dimana baris yang diawali dengan tanda # dianggap sebagai komentar yang digunakan untuk menjelaskan masing-masing perintah.

Langkah 3 : Pembuatan Topologi

Topologi dibangun oleh *node* dan *link* yang dijelaskan sebagai berikut :

Node

Merupakan sebuah objek *node* pada NS-2 didefiniskan dengan command `$ns node`. Sebagai contoh pembuatan node pada NS-2 adalah sebagai berikut :

```
Set node [$ns node]
```

Link

Ada dua jenis link yang bisa digunakan pada NS-2, yaitu *simplex link* dan *duplex link.* Berikut ini adalah perintah pembuatan link beserta parameternya :

1. Untuk simplex link :

   ```
   $ns simplex-link <node1><node2><bw><delay><qtype>
   ```

   Link satu arah dari <node1> ke <node2>.

2. Untuk duplex link :

   ```
   $ns simplex-link <node1><node2><bw><delay><qtype>
   ```

   Link dua arah dari <node1> ke <node2> dan sebaliknya.

Langkah 4 : Membuat aliran data

Proses pengiriman data pada NS-2 dilakukan dengan membuat *transport agent* dan aplikasi pembawanya. *Transport agent* dibuat berpasangan, satu berfungsi sebagai sumber data dan pasangannya sebagai tujuannya.

Pada simulasi ini, kita menggunakan paket TCP dengan menggunakan aplikasi FTP. Pengiriman data tersebut diawali dengan membuat *agent* pengirim data.

```
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(1) $tcp
$ns attach-agent $node_(4) $sink
$ns connect $tcp $sink
```

```
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"
```

#membuat objek simulator yang berupa TCP Agent

```
set tcp [new Agent/TCP]
```

#attach-agent berfungsi untuk mengambil object agent yang sudah didefinisikan. *Attach-agent* tersebut dibagi menjadi 2 bagian, yaitu bagian pengirim dan bagian penerima.

#node_(1) sebagai *node* sumber

```
$ns attach-agent $node_(1) $tcp
```

#node_(4) sebagai *node* tujuan

```
$ns attach-agent $node_(4) $sink
```

Untuk dapat mengalirkan data di antara 2 *node* tersebut, maka kedua *node* tersebut harus dihubungkan dengan perintah sebagai berikut :

```
$ns connect $tcp $sink
```

Kemudian dibuat aplikasi yang berjalan di atas *transport agent* tersebut. Pada contoh ini, kita gunakan generator trafik dengan fungsi FTP dengan perintah sebagai berikut :

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

Setelah itu dilakukan pengaturan waktu dimana data TCP akan dialirkan, dengan perintah sebagai berikut :

```
$ns at 10.0 "$ftp start"
```

Artinya bahwa data akan dikirim setelah detik ke 10.0.

Menurut Wirawan & Indarto (2004), ada 2 jenis TCP *Agent* yang didukung oleh NS-2, yaitu *One Way* TCP *Agent* dan *Two Way* TCP *Agent*. Dimana *One Way* TCP

Agent tidak menggunakan komunikasi 2 arah antara *node* sumber dan *node* tujuannya, sedangkan *Two Way* TCP *Agent* mengadakan kesepakatan dua arah dalam proses pengiriman datanya.

2.7 *Practical Extraction and Report language* (Perl)

Menurut Jusak (2011), Perl merupakan bahasa pemrograman yang banyak digunakan untuk pemrosesan data *file* ASCII pada system UNIX. Bahasa pemrograman ini dibuat oleh Larry Wall dengan tujuan untuk memudahkan tugas-tugas administrasi sistem UNIX. Saat ini, Perl telah berevolusi menjadi bahasa pemrograman dan merupakan salah satu sarana yang dapat digunakan untuk pemrograman web. Di bawah ini merupakan contoh sederhana prosedur queue.pl untuk mem*filter* data :

```
$infile=$ARGV[0];
open (DATA,"<$infile") || die "Can't open $infile $!";
while (<DATA>)
{
    @x = split(' ');
    if ($x[6] eq 'tcp')
{
print STDOUT "$x[0]           $x[1]                $x[2]
}
}
close DATA;
exit(0);
```

Prosedur queue.pl di atas dijalankan dengan perintah sebagai berikut :

```
Perl queue.pl simple.tr > queue
```

Yang berarti bahwa prosedur queue.pl digunakan untuk memfilter data dari file simple.tr yang kemudian hasilnya akan disimpan pada *file* baru bernama queue.