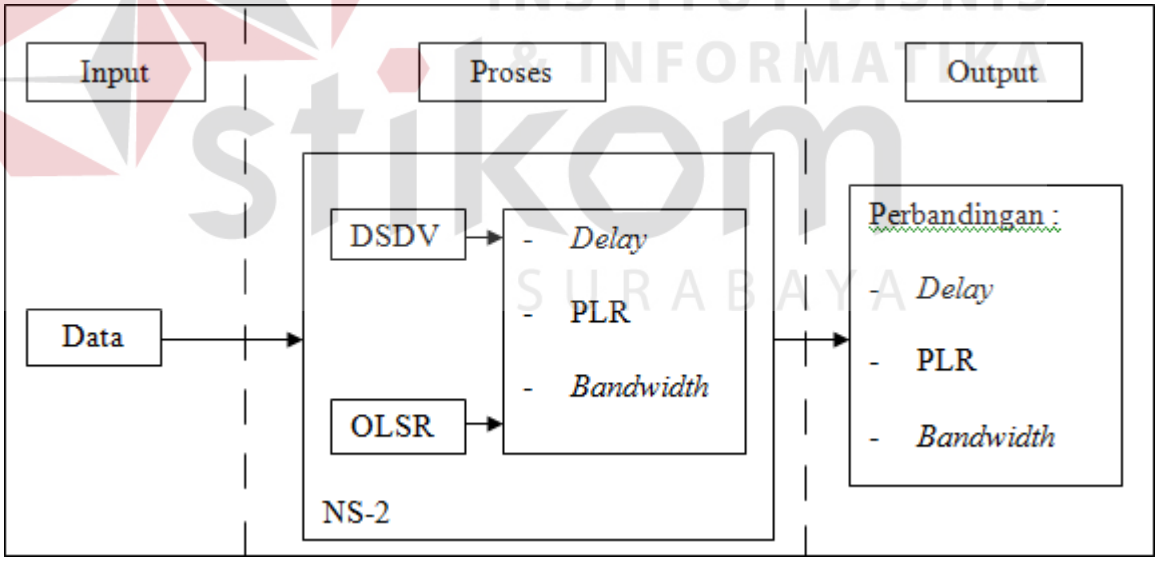
BAB III

METODE PENELITIAN DAN PERANCANGAN SISTEM

3.1 Metode Penelitian

Metode penelitian yang digunakan dalam TA ini dilakukan dengan cara mencari informasi yang berkaitan dengan data-data yang dibutuhkan untuk menyelesaikan TA ini. Data-data tersebut meliputi karakteristik protokol yang digunakan yaitu protokol DSDV dan OLSR sebagaimana yang diketahui bahwa kedua protokol tersebut merupakan protokol Proactive *routing*, data yang digunakan sebagai aliran trafik yaitu TCP, serta prameter-parameter *QoS* yang digunakan dalam perhitungan dan analisis. Penelitian yang dilakukan dapat dijelaskan dengan lebih baik melalui diagram blok pada Gambar 3.1.



Gambar 3.1 Diagram Blok

Dari diagram blok pada Gambar 3.1, dapat diketahui bahwa terdapat 3 bagian penting yaitu bagian *input*, proses, dan *output*.

3.1.1 Bagian *Input*

Bagian input terdiri dari data yang digunakan untuk simulasi yaitu berupa data trafik yang dibangkitkan dari simulator NS-2. Data TCP tersebut dialirkan dari node sumber menuju node tujuan menggunakan protokol DSDV dan OLSR.

3.1.2 Bagian Proses

Pada bagian ini akan dibangun simulasi jaringan menggunakan NS-2. Proses ini meliputi konfigurasi jaringan, desain topologi, mengatur skenario simulasi dan parameter eksternal, dan menjalankan simulasi. Simulasi yang dilakukan menggunakan topologi dengan posisi *node* yang bersifat acak (*random*), dimana untuk menghasilkan posisi yang acak tersebut menggunakan parameter eksternal berupa nilai *seed* yang ditentukan untuk masing-masing percobaan.

Setelah semua tahap proses dilakukan dengan beberapa percobaan, akan dilakukan pengambilan data-data yang berguna untuk melakukan penghitungan *delay*, PLR, dan *utilisasi bandwidth*. Data-data tersebut dapat diperoleh dari *file* hasil simulasi yaitu *file* "simple.tr" yang berisi segala informasi aktivitas yang terjadi mulai dari awal hingga akhir. Data-data yang diambil adalah data-data yang merupakan aktivitas-aktivitas yang terkait dengan data TCP saja, karena ukuran paket TCP ± sama dengan ukuran paket data yang dikirim dan diterima pada penerapan JSN yaitu maksimal ±1040 *byte*. Hasil dari pengolahan data tersebut berupa analisis seperti yang disebutkan sebagai berikut :

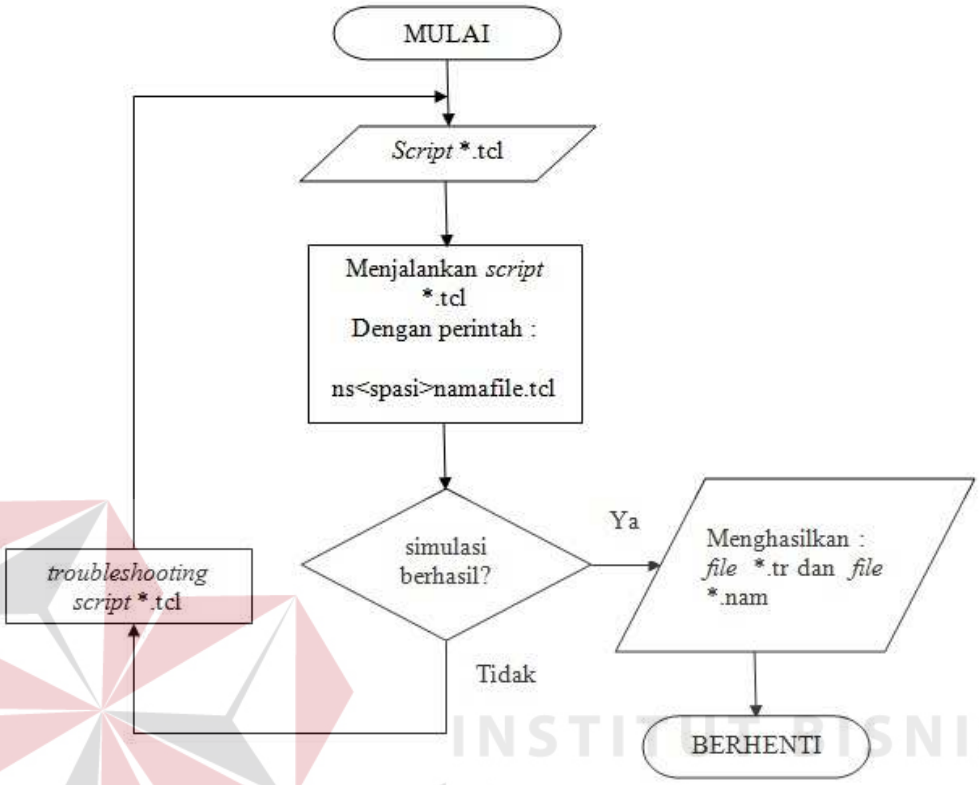
1. Analisis perbandingan *delay* : yaitu rata-rata waktu penundaan yang terjadi terhitung dari saat paket dikirim dari *transmitter* hingga paket diterima oleh *receiver*.

2. Analisis perbandingan PLR : yaitu rata-rata besarnya paket data yang hilang dan gagal diterima oleh *receiver*. Apabila nilai PLR dari suatu protokol itu semakin kecil, maka semakin baik kualitas protokol dalam pengiriman data.
3. Analisis perbandingan *utilisasi bandwidth* : yaitu rata-rata *bandwidth* yang dihabiskan untuk melakukan pengiriman dan penerimaan data. Semakin kecil rata-rata *bandwidth* yang digunakan oleh sistem dengan protokol tertentu, maka protokol tersebut semakin efektif bila digunakan.

3.1.3 Bagian *Output*

Bagian *output* meliputi hasil perbandingan protokol DSDV dan OLSR yang terdiri dari beberapa parameter *QoS* yaitu *delay*, PLR, dan *utilisasi bandwidth*. Sehingga dapat dilihat protokol yang paling sesuai untuk digunakan pada pengimplementasian JSN *Ad Hoc* berdasarkan unjuk kerja dari masing-masing protokol tersebut.

3.2 Arsitektur Sistem Jaringan



Gambar 3.2 Diagram Alur Proses Simulasi

Dari Gambar 3.2 dapat diketahui bahwa proses simulasi dimulai dengan adanya *script* Tcl. *Script* tersebut dibangun sesuai dengan parameter-parameter yang akan dijelaskan pada subbab berikutnya. Kemudian *script* tersebut akan dijalankan dengan perintah :

```
root@fauzan:/home/fauzan/TA#ns<spasi>nama file.tcl
```

Perintah di atas dituliskan pada jendela terminal ubuntu dengan *format* yang telah dijelaskan pada bab sebelumnya. Jika simulasi tersebut berhasil, maka akan dihasilkan *file* hasil simulasi, yaitu :

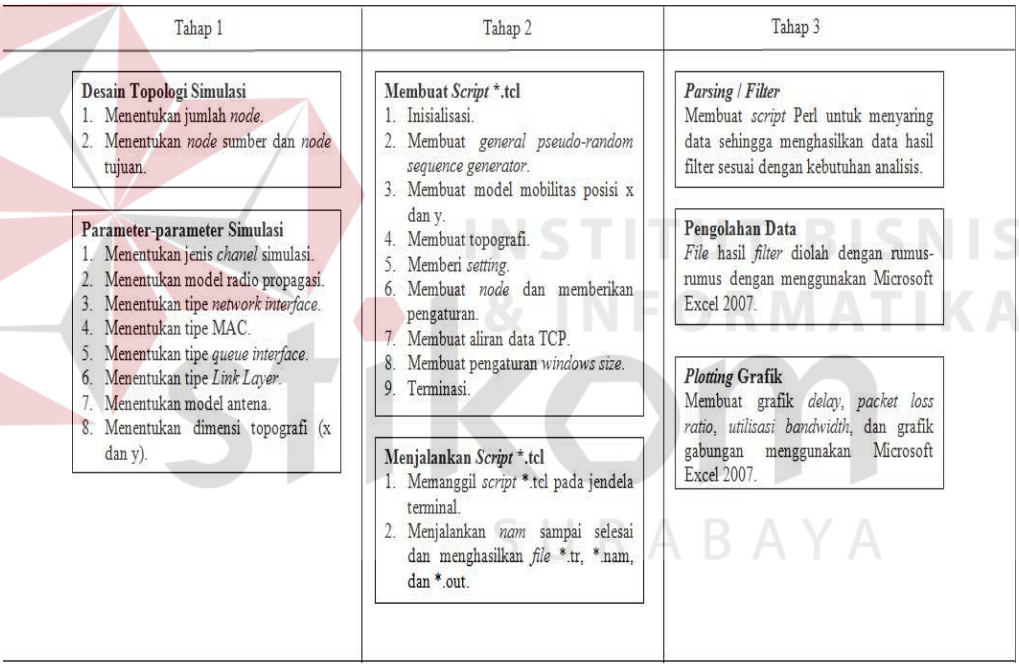
1. *File*.tr : untuk melihat *trace* yang terjadi selama proses komunikasi.

2. *File*.nam : digunakan untuk melihat *display* dari hasil program. Dapat dieksekusi dengan cara “nam<spasi>namafile.nam

Jika simulasi tidak berhasil, maka harus dilakukan *troubleshooting* terhadap *script* Tcl yang telah dibuat dan memperbaikinya. Setelah itu, penjalanan simulasi bisa dilakukan kembali.

3.3 Tahap Perancangan Sistem

Tahap-tahap yang dilakukan dari awal perancangan sistem, pembuatan sistem, serta analisisnya dapat dilihat pada Gambar 3.3.

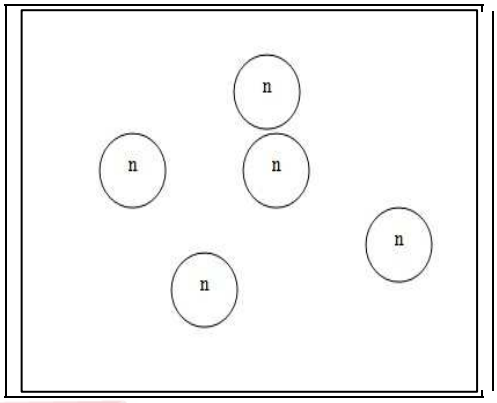


Gambar 3.3 Tahap Perancangan Sistem

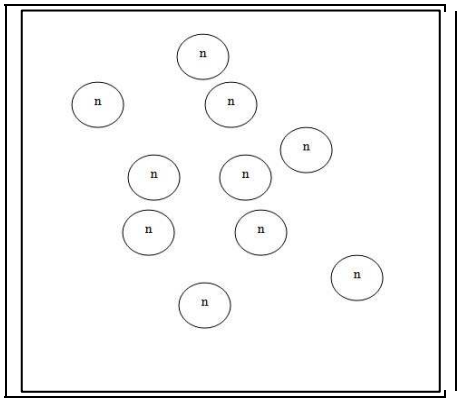
Tahap-tahap perancangan sistem pada gambar di atas akan dijelaskan pada sub-bab berikutnya.

3.3.1 Desain Topologi Simulasi

Desain topologi simulasi yang digunakan dibagi menjadi 2 yaitu topologi dengan 5 buah *node* dan 10 buah *node*. Desain topologi tersebut dapat diilustrasikan seperti Gambar 3.4 dan Gambar 3.5.



Gambar 3.4 Topologi 5 buah *node*



Gambar 3.5 Topologi 10 buah *node*

Desain topologi sistem seperti yang diilustrasikan pada Gambar 3.4 dan Gambar 3.5, masing-masing posisi *node* nya ditentukan secara *random*. Jadi, untuk masing-masing percobaan akan menghasilkan posisi-posisi *node* yang berbeda. Akan tetapi, *node* sumber dan *node* tujuannya sudah ditentukan. Penentuan *node* sumber dan *node* tujuannya dapat dilihat pada Tabel 3.1.

Tabel 3.1 Penentuan *Node* Sumber dan *Node* Tujuan

	DSDV	OLSR
5 node	*node* sumber : *node* 1 *node* tujuan : *node* 4	*node* sumber : *node* 1 *node* tujuan : *node* 4
10 node	*node* sumber : *node* 1 *node* tujuan : *node* 2	*node* sumber : *node* 1 *node* tujuan : *node* 2

Penentuan *node* mana yang menjadi sumber dan *node* mana yang menjadi tujuan bisa dilihat dari *syntax* berikut :

```
$ns attach-agent $node_(1) $tcp  ……(1)
$ns attach-agent $node_(2) $sink ……(2)
```

Syintax (1) menunjukkan nomor *node* yang menjadi sumber, sedangkan *syntax* (2) menunjukkan nomor *node* yang menjadi tujuan. *Syntax* tersebut dapat digunakan jika menggunakan tipe data TCP yang akan dialirkan.

3.3.2 Parameter-parameter Simulasi

Pembuatan sistem tidak terlepas dari parameter-parameter yang digunakan untuk menghasilkan sistem sesuai dengan kebutuhan. Parameter-parameter yang digunakan pada pembangunan sistem pada penelitian ini dapat dilihat pada Tabel 3.2.

Tabel 3.2 Parameter Simulasi

No.	Parameter	Nilai
1.	*Channel*	*WirelessChannel*
2.	Model propagasi	*Free Space*
3.	Tipe antarmuka antrian	*Drop Tail*
4.	Model antena	*Omni antenna*
5.	Tipe protokol *routing*	DSDV dan OLSR
6.	*Network Interface*	*WirelessPhy*
7.	Tipe MAC	802.11
8.	Tipe *Link Layer*	LL
9.	Dimensi topografi	300 x 300 m
10.	Waktu simulasi	± 200 detik

Dari Tabel 3.2 dapat terlihat parameter-parameter yang menunjang pembangunan sistem. Parameter-parameter tersebut dapat dijelaskan sebagai berikut :

1. *Channel = WirelessChannel*

Channel yang diaktifkan adalah *channel Wireless*, karena yang dibuat adalah simulasi jaringan yang menggunakan *wireless* (jaringan tanpa kabel).

2. Model propagasi = *Free Space*

   Model propagasinya bersifat *free space*.

3. Tipe antarmuka antrian = *Drop Tail*

   Tipe antarmuka antrian ini menggunakan *Drop Tail* dimana data yang diterima pada saat kapasitas.

4. Model Antena = Omni *Antenna*

   Model antena yang digunakan adalah antena omni untuk masing-masing *node* yang digunakan dalam simulasi.

5. Tipe protokol *routing* = DSDV dan OLSR

   Protokol *routing* yang digunakan adalah DSDV dan OLSR. Kedua protokol *routing* tersebut merupakan protokol yang dapat digunakan dalam tipe jaringan *WirelessPhy*. Kedua protokol tersebut merupakan protokol yang proaktif, sehingga selalu ada komunikasi antar-*node* sehingga rute yang terbentuk akan selalu terjaga.

6. *Network Interface* = *WirelessPhy*

   Jaringan yang digunakan adalah jaringan *wireless* dan jaringan tersebut memiliki beberapa tipe. Yang digunakan dalam penelitian ini adalah dengan tipe standar *wirelessphy* (802.11 phy) dengan data *rate* sebesar 1.2 Mbps dan *operating frequency* sebesar 2.4 GHz. (Gong, Shan. 2006).

7. Tipe MAC = 802.11

Tipe MAC yang digunakan adalah 802.11 karena menggunakan standar *wireless* 802.11.

8. Tipe *Link Layer* = LL

Tipe *link layernya* LL.

9. Dimensi topografi = 300 x 300 m

Dimensi topografi yang digunakan adalah panjang = 300 m dan lebar = 300 m. Dimensi ini lebih besar 3 kali dari dimensi topografi yang digunakan oleh Nofianti, Dwi, dkk (2011), agar posisi *node* yang tersebar lebih terlihat komunikasi.

10. Waktu simulasi = 200 detik

Waktu simulasi yang digunakan adalah selama 200 detik.

Selain parameter-parameter yang sudah disebutkan di atas, untuk pengaturan posisi *node* secara *random* menggunakan parameter *seed* yang sama, agar hasil yang diperoleh dapat dibandingkan. Parameter *seed* yang diterapkan pada sistem dapat dilihat pada Tabel 3.3.

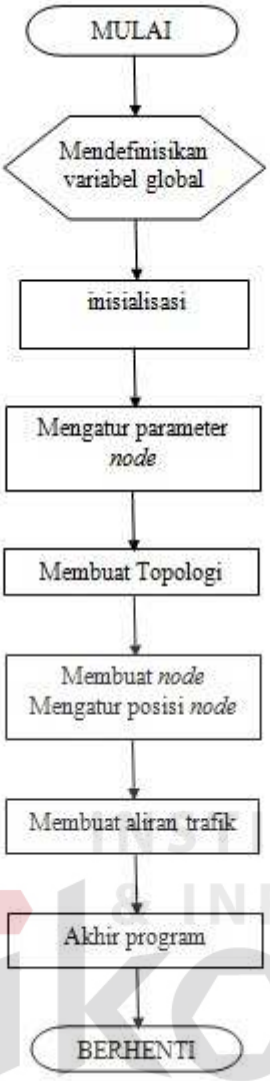
Tabel 3.3 Nilai Parameter *Seed*

Percobaan Ke-	DSDV		OLSR	
	5 node	10 node	5 node	10 node
1.	0			
2.	0.001			
3.	0.002			
4.	0.003			
5.	0.004			
6.	0.005			
7.	0.006			
8.	0.01			
9.	0.02			
10.	0.03			
11.	0.04			
12.	0.05			
13.	0.06			

Percobaan Ke-	DSDV		OLSR	
	5 node	10 node	5 node	10 node
14.	0.07			
15.	0.08			
16.	0.09			
17.	0.1			
18.	0.2			
19.	0.3			
20.	0.4			
21.	0.5			
22.	0.6			
23.	0.7			
24.	0.8			
25.	0.9			
26.	1			
27.	2			
28.	3			
29.	4			
30.	5			

Nilai-nilai *seed* pada Tabel 3.3 berpengaruh pada pembuatan *generator* nilai *random* yang akan digunakan pada simulasi. Nilai-nilai tersebut ditentukan berdasarkan *range* maksimal nilai *seed* yang berpengaruh pada hasil simulasi sehubungan dengan nilai *seed* tersebut akan menghasilkan nilai posisi yang masih bisa terjangkau oleh *node-node* lain ataupun tidak. Nilai *seed* = 5 adalah nilai maksimal yang dapat dijangkau dengan ukuran topografi yang ditentukan.

3.3.3 Membuat *Script* *.tcl



Gambar 3.6 Diagram Alur Pembuatan *Script* *.tcl

Sebelum melakukan proses simulasi, terlebih dahulu dilakukan pembuatan *script* *.tcl sesuai dengan parameter-parameter dan bentuk topologi jaringan yang sudah ditetapkan sebelumnya. Dari Gambar 3.6 dapat dijabarkan bahwa tahap pembuatan *script* *.tcl adalah sebagai berikut :

1. Mendefinisikan Variabel Global

Dalam membangun simulasi JSN *Ad Hoc* terlebih dahulu harus mendefinisikan variabel-variabel global yang dibutuhkan. Berikut adalah cara pendefinisian variabel dengan nilai yang sudah ditentukan :

```
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) DSDV;
set val(seed) 0;
set val(x) 300;
set val(y) 300;
set val(stop) 200;
set val(mobility) Static;
```

Script di atas bertujuan untuk mendeklarasikan variabel-variabel yang digunakan. Variabel-variabel yang akan digunakan pada Tabel 3.4.

Tabel 3.4 Variabel-variabel Global

No.	Nama Variabel	*Value*	Keterangan
1.	Chan	WirelessChannel	Tipe *channel* yang digunakan adalah *channel wireless*
2.	Prop	TwoRayGround	Model propagasinya TwoRayGround
3.	Netif	WirelessPhy	Tipe jaringan *wireless* yang digunakan adalah standar *wirelessphy*
4.	Mac	802_11	Tipe MAC yang digunakan adalah *wireless* 802.11
5.	Ifq	DropTail/PriQueue	Tipe antarmuka antriannya bersifat *droptail* yang menggunakan *priqueue*
6.	Ll	LL	Tipe *link layer* nya LL
7.	Ant	OmniAntenna	Antena yang digunakan pada *node* adalah antena Omni
8.	Ifqlen	50	Ukuran maksimum antrian paket adalah 50
9.	Nn	5 / 10	Jumlah *node* yang digunakan adalah 5 dan 10 (secara terpisah)
10.	Rp	DSDV / OLSR	Protokol *routing* yang digunakan adalah DSDV dan OLSR (secara terpisah)
11.	Seed	0 – 5	*Seed* yang digunakan bervariasi sesuai dengan *range* yang ditunjukkan pada

No.	Nama Variabel	*Value*	Keterangan
			Tabel 3.3 pada subbab 3.3.3
12.	X	300	Nilai topografi x adalah 300 meter
13.	Y	300	Nilai topografi y adalah 300 meter
14.	Stop	200	Lama waktu simulasi adalah ± 200 detik
15.	Mobility	Static	Mobilitas dari *node* bersifat statis

2. Inisialisasi

Setelah mendefinisikan variabel-variabel yang dibutuhkan, selanjutnya dilakukan inisialisasi dari simulasi tersebut. *Script* di bawah ini harus selalu dituliskan ketika akan membangun simulasi. *Script* tersebut adalah sebagai berikut:

```
set ns [new Simulator]
set tracefd [open simple.tr w]
set namtrace [open simwrls.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

Dengan *script* di atas maka akan dihasilkan 2 jenis *file* yaitu “simple.tr” yang digunakan untuk menyimpan hasil simulasi dan “simwrls.nam” untuk menyimpan data visualisasi dari simulasi. Perintah “open” pada *script* di atas digunakan untuk membuat *file* dan perintah “w” yang berada di belakang untuk menuliskan hasil pada *file* yang telah dibuat.

3. Mengatur Parameter *Node*

Karena topologi yang dibuat adalah topologi yang posisi *node-node*nya ditentukan secara *random*, maka harus dibuat sebuah parameter yang digunakan untuk menjalankan fungsi *random*. *Script*nya adalah sebagai berikut :

```
# General pseudo-random sequence generator
set genSeed [new RNG]
$genSeed seed $val(seed)
set randomSeed [new RandomVariable/Uniform]
$randomSeed use-rng $genSeed
$randomSeed set min_ 1.0
$randomSeed set max_ 200.0

# Mobility model: x node position [m]
set genNodeX [new RNG]
$genNodeX seed [expr [$randomSeed value]]
set randomNodeX [new RandomVariable/Uniform]
$randomNodeX use-rng $genNodeX
$randomNodeX set min_ 1.0
$randomNodeX set max_ [expr $val(x) - 1.0]

# Mobility model: y node position [m]
set posNodeY [new RNG]
$posNodeY seed [expr [$randomSeed value]]
set randomNodeY [new RandomVariable/Uniform]
$randomNodeY use-rng $posNodeY
$randomNodeY set min_ 1.0
$randomNodeY set max_ [expr $val(y) - 1.0]
```

Script di atas digunakan untuk membuat nilai *random*. Sebelum melakukan *random* pada nilai variabel X dan Y, harus dibuat general *random* terlebih dahulu dengan perintah `set genSeed [new RNG]`, yang digunakan untuk membangkitkan *generator random*nya. Kemudian membuat variabel yang akan digunakan seperti `$genSeed seed $val(seed)`, dengan perintah tersebut akan diberi nilai *generator*nya berdasarkan nilai parameter *seed* yang telah disebutkan pada sub bab sebelumnya. Setelah membangkitkan *generator*nya, selanjutnya dibuat variable yang digunakan untuk menampung nilai *random* yaitu dengan perintah `set randomSeed [new RandomVariable/Uniform]`, variabel tersebut akan menampung nilai yang dihasilkan oleh generator *seed* yang diatur sebelumnya dengan perintah `$randomSeed use-rng $genSeed`. Kemudian dilakukan pengaturan nilai minimal dan maksimal nilai *random*, dalam penelitian ini digunakan range 1-200 yang diatur dengan menuliskan perintah `$randomSeed set min_ 1.0` dan `$randomSeed set max_ 200.0`. Dimana batas minimal dan

maksimal yang digunakan berdasarkan kemampuan jangkauan dari masing-masing *node* dikarenakan tipe jaringan *wireless* 802.11 memiliki jangkauan ± 100 meter. (Gong, Shan. 2006).

Setelah itu, pengaturan posisi *node* dapat dilakukan. Untuk pemberian nilai *random* untuk titik X dan titik Y harus dibedakan variabel penampung dan variabel generatornya, agar tidak terjadi pemberian nilai yang sama utnuk masing-masing titik tersebut. Dimana generator yang digunakan untuk titik X adalah genNodeX dan untuk titik Y adalah posNodeY dengan perintah `set genNodeX [new RNG]` dan `set posNodeY [new RNG]`. Sedangkan variabel penampungnya digunakan variable randomNodeX dan randomNodeY. Proses terakhir pemberian nilai *random* untuk masing-masing titik adalah mengatur nilai minimal dan maksimalnya juga. Dari perintah `$randomNodeX set min_ 1.0` dan `$randomNodeX set max_ [expr $val(x) - 1.0]`, nilai minimal yang digunakan adalah 1 dan nilai maksimalnya adalah 300-1. Karena nilai maksimal dari generatornya adalah 200 maka nilai maksimal dari masing-masing mobilitas *node* X dan Y adalah 200.

Setelah melakukan pengaturan-pengaturan di atas, sebelum membuat topologi dan proses-proses yang lain dilakukan pengaktifan konfigurasi *node* dengan variabel-variabel global yang sudah diinisialisasikan sebelumnya beserta nilai-nilainya dengan perintah sebagai berikut :

```
$ns node-config -adhocRouting $val(rp) -llType $val(ll) \
-macType $val(mac) -ifqType $val(ifq) \
-ifqLen $val(ifqlen) -antType $val(ant) \
-propType $val(prop) -phyType $val(netif) \
-topoInstance $topo a-agentTrace ON -routerTrace OFF \
-macTrace ON -movementTrace ON -channel $chan_
```

4. Membuat Topologi

Untuk membangun simulasi, maka dibutuhkan objek topografi, sehingga harus dituliskan *script* dibawah ini :

```
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
set chan_ [new $val(chan)]
```

Perintah di atas digunakan untuk membuat objek topografi dan mendefinisikan ukuran topologi sebesar nilai dari variabel x dan nilai dari variabel y. Nilainya sesuai dengan yang telah didefinisikan pada variabel global, yaitu 300 x 300 meter. Selain itu, juga dibuat GOD (*General Operation Director*) untuk menyimpan informasi keseluruhan *mobile node* dan melakukan perhitungan jumlah hop terpendek untuk menghubungkan satu *node* dengan lainnya.

5. Membuat *Node* dan Mengatur Posisi *Node*

Objek *node* digunakan sebagai ilustrasi sebuah sensor. Pada NS-2, untuk membuat sebuah *node* digunakan perintah :

```
set nama_node [$ns node]
```

Maka perintah yang digunakan adalah seperti di bawah ini:

```
for {set i 0} {$i < $val(nn) } {incr i} {
      set node_($i) [$ns node]
      $node_($i) random-motion 0        ;
}
```

Script di atas menunjukkan bahwa *node* yang dibuat adalah sebanyak nilai “nn” yang sudah dideklarasikan pada saat pertama kali, yaitu sebanyak 5 atau 10 buah. Jika nilai nn=5, maka banyak *node* yang dibuat berjumlah 5, dan pemberian nama *node*-nya karena dimulai dari 0 (seperti yang disebutkan dari perulangan dengan variabel *i*, dimulai dari 0 hingga *i* kurang dari 5), maka nama *node* tersebut adalah node_0, node_1, node_2, node_3, node_4. Demikian pula untuk jumlah nn = 10 dan untuk jumlah nn lain yang dibutuhkan.

Karena posisi *node* yang ditentukan bersifat *random*, maka posisi *node* yang dihasilkan harus diatur letak titik X dan titik Y nya dengan menggunakan general-pseudo-random yang sudah dibuat di *script* sebelumnya. Cara menggunakan perintahnya adalah sebagai berikut :

```
if {$val(mobility) == "Static"} {

for {set i 0} {$i < $val(nn)} {incr i} {
            set X [expr [$randomNodeX value] ]
            $node_($i) set X_ $X
            set Y [expr [$randomNodeY value] ]
            $node_($i) set Y_ $Y
            $node_($i) set Z_ 0.0
      }}
```

Dengan *script* di atas, akan diciptakan sebuah kondisi yaitu jika nilai dari variabel mobility adalah “Static” maka akan terjadi pengulangan dimulai dari 0 – nn untuk membuat posisi X dan Y dari *node-node* yang bersangkutan bernilai *random*.

Perintah “expr” yang tercantum pada *script* digunakan untuk menandai bahwa ada operasi matematika. Seperti yang terdapat pada *script* di atas yaitu [$randomNodeX value] yang merupakan perintah untuk memberikan nilai variabel X dengan hasil dari operasi randomNodeX, begitu juga dengan nilai variabel Y. Sedangkan untuk nilai titik Z digunakan posisi 0, karena koordinat yang digunakan adalah koordinat 2 dimensi.

6. Membuat Aliran Trafik

Untuk mendapatkan *input* data, maka dibuat aliran data. Proses pengaliran data dilakukan dengan membuat transport agent dan aplikasi di atasnya. Dalam penelitian ini, menggunakan agent TCP dan FTP sebagai aplikasinya. *Script* yang digunakan dapat dituliskan sebagai berikut :

```
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"
```

Dari *script* di atas aliran data TCP akan dialirkan dari node_1 yang ditunjukkan dengan perintah **`$ns attach-agent $node_(1) $tcp`** yang menunjukkan bahwa node_1 merupakan *node* sumber menuju node_4 dengan perintah **`$ns attach-agent $node_(4) $sink`** sebagai tujuannya. setelah ditentukan arah alirannya, maka kedua *node* tersebut dihubungkan dengan perintah :

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

Kemudian TCP tersebut dialirkan dengan aplikasi *File Transfer Protokol* (FTP) pada detik ke 10.0.

7. Akhir Program

Selain *script-script* yang dibuat di atas, untuk proses akhir dari program, maka harus dibuat *script* penutup. *Script* akhir bertujuan untuk me-*reset* semua nilai yang sudah digunakan dan untuk mengakhiri simulasi. Berikut adalah perintah untuk me-reset nilai-nilai :

```
for { set i 0} { $i < $val(nn) } { incr i } {
$ns initial_node_pos $node_($i) 30
}

for { set i 0} { $i < $val(nn) } { incr i } {
$ns at 200.0 "$node_($i) reset";
}

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 200.01 "puts \"end simulation\" ; $ns halt"
```

Sedangkan, untuk menghentikan program simulasi digunakan perintah sebagai berikut :

```
proc stop {} {
global ns tracefd namtrace
$ns flush-trace
$ns use-newtrace
close $tracefd
close $namtrace
exec nam simwrls.nam &
exit 0
}
```

Prosedur di atas menggunakan variabel global `ns`, `tracefd`, dan `namtrace`. Perintah “flush-trace” digunakan untuk menyimpan seluruh data hasil simulasi ke dalam *tracefile* dan *namfile*. Sedangkan, perintah “exit” digunakan untuk mengakhiri aplikasi dan memberikan status 0 kepada sistem. Nilai status tersebut adalah *default* untuk membersihkan memori sistem. Prosedur di atas akan dijalankan pada detik 200.01 seperti yang tertera pada perintah :

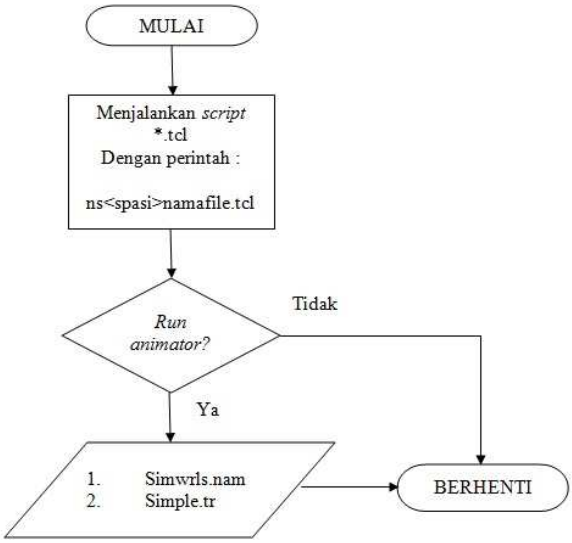
```
$ns at 200.01 "puts \"end simulation\" ; $ns halt"
```

Setelah menuliskan semua objek simulasi, jangan lupa untuk menuliskan perintah :

```
$ns run
```

3.3.4 Melakukan *Trace* Simulasi

Dengan perintah tersebut objek simulasi akan dieksekusi secara berurutan ketika dijalankan, perintah-perintah yang ditulis setelah `$ns run` tidak akan pernah dieksekusi.



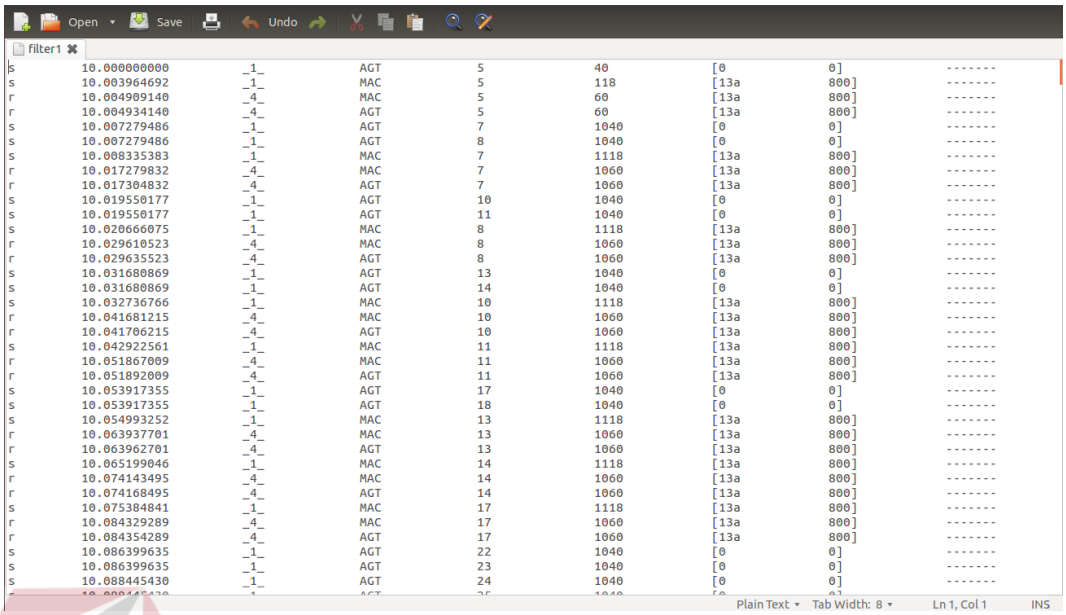
Gambar 3.7 Diagram Alur Trace Simulasi

Gambar 3.7 merupakan langkah-langkah melakukan *trace* simulasi, pertama-tama kita melakukan panggilan/menjalankan *script* *.tcl yang telah dibuat. Kemudian jendela *Network Animator* yang dihasilkan dari pemanggilan *script* tersebut dijalankan hingga selesai. Sehingga dihasilkan 2 *file* yaitu *file* *.nam yang berisi data visualisasi dari simulasi dan *file* *.tr yang berisi data-data aktivitas yang terjadi pada saat simulasi berjalan hingga selesai.

3.3.5 Proses *Parsing* Data

Proses *parsing* data yang dilakukan bertujuan untuk mengambil data-data yang dibutuhkan untuk melakukan penghitungan selanjutnya dari *file* *.tr yang telah dihasilkan. Pada penelitian ini *filter* data dapat dilihat selengkapnya pada pengujian Bab 4.3.3.

Tidak semua informasi yang tersimpan pada *tracefile* digunakan untuk proses selanjutnya. Adapun *tracefile* yang dihasilkan adalah sebagai berikut :



Gambar 3.8 Hasil *Tracefile*

File yang ditunjukkan oleh Gambar 3.8 merekam semua aktivitas yang terjadi mulai dari detik ke-0 (nol) hingga detik ke-200 sesuai dengan *script* Tcl yang telah dibuat. Gambar 3.9 menunjukkan cuplikan baris pertama dari hasil *tracefile* Gambar 3.8.

S	0.031853939	_1_	MAC	---	0	message	90	[0 ffffffff 1 800]	[1:255 -1:255 32 0]
K. 0	K. 1	K. 2	K. 3	K. 4	K. 5	K. 6	K. 7	K. 8	K.9

Gambar 3.9 Cuplikan *Tracefile*

Dimana :

K = Kolom ke-*i* , misal : K. 7 = Kolom 7.

Data-data yang terekam ada beberapa kolom meliputi : (Wirawan, Andi B. dan Indarto, Eka. 2004)

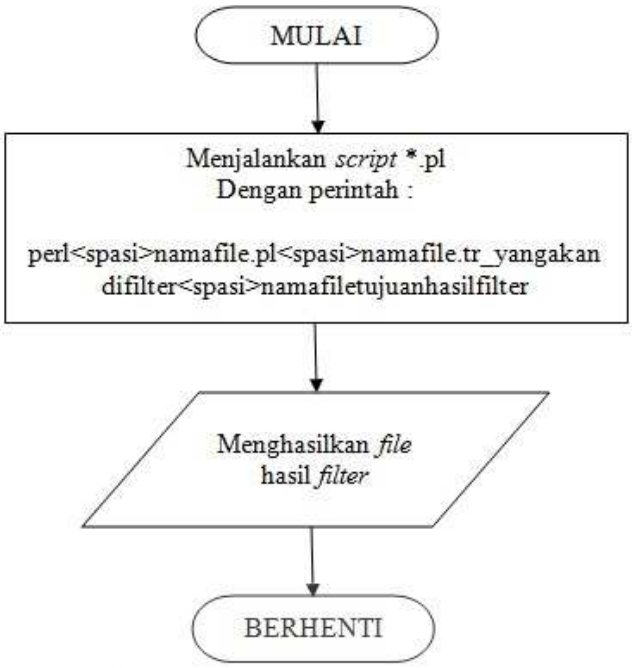
1. Kolom 0 menunjukkan *event* yang sedang terjadi. Pada *file* di atas terdapat 3 *event* yaitu s (*sent*) yang menunjukkan bahwa sedang terjadi pengiriman, *event* r (*receive*) yang menunjukkan bahwa sedang terjadi penerimaan, dan

event D (*Drop*) yang menunjukkan bahwa ada paket yang dibuang (*drop*) biasanya terjadi pada detik-detik terakhir saat simulasi akan berakhir.

2. Kolom 1 menunjukkan waktu *event* sedang terjadi.
3. Kolom 2 menunjukkan nomor *node* yang sedang terlibat dengan aktivitas pada saat itu.
4. Kolom 3 menunjukkan *trace level* yaitu MAC dan AGT. MAC merupakan MAC layer dan AGT merupakan paket pada transport layer.
5. Kolom 4 merupakan pemisah.
6. Kolom 5 adalah nomor urut paket dimulai dari urutan 0 (nol).
7. Kolom 6 menunjukkan tipe dari paket (tcp, ack, arp, cts, dsb).
8. Kolom 7 menunjukkan ukuran (*length*) dari paket dalam satuan *byte*.
9. [0 ffffffff 1 800] menunjukkan informasi MAC layer.
10. [1:255 -1:255 32 0] menunjukkan IP sumber dan tujuan berikutnya *time to live* (ttl) dari paket.
11. Apabila terdapat informasi [0 0] merupakan nomor urut dan pemberitahuan nomor (informasi tcp).

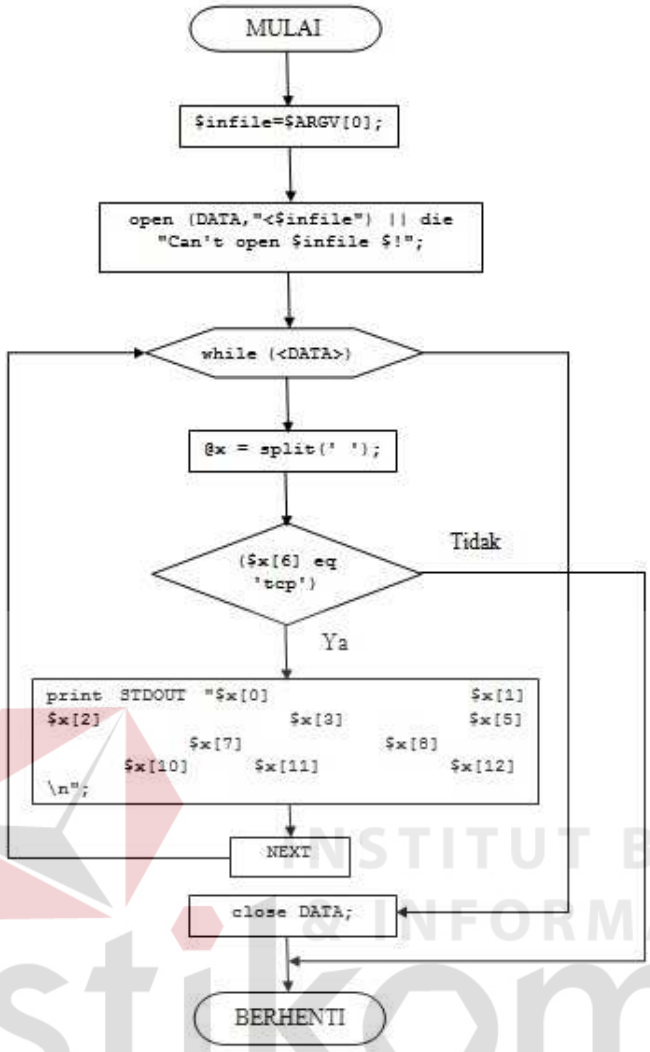
    0 0 merupakan format mekanisme *routing type pack*.

File di atas *difilter* dengan langkah-langkah pada diagram alur pada Gambar 3.9.



Gambar 3.10 Diagram Alur Melakukan *Filter* Data

Gambar 3.10 merupakan langkah-langkah melakukan proses *parsing file* dimulai dengan memanggil *script* *.pl seperti yang terlihat pada diagram alur Gambar 3.10. *File* yang di*filter* pada penelitian ini adalah *file* simple.tr, data-data yang ada di dalam *file* tersebut di*filter* sesuai dengan kebutuhan untuk penghitungan *delay*, PLR, dan *utilisasi bandwidth*. Jika proses *filter* berhasil, maka akan dihasilkan *file* baru yang berisi data-data yang telah ter*filter* dengan nama yang diberikan. Data-data yang dibutuhkan untuk proses penghitungan selanjutnya dapat dilihat pada Gambar 3.11.



Gambar 3.11 *Flowchart Script* Perl

Dari Gambar 3.11 dapat diketahui bahwa data-data yang dibutuhkan adalah data yang menunjukkan aliran data TCP sesuai dengan sistem yang dibuat. Tetapi tidak semua informasi yang digunakan. Dari *flowchart* di atas data-data yang dibutuhkan adalah data pada kolom-kolom berikut :

1. Kolom 0 ($x[0])

   Terdapat 3 jenis *event* yaitu s (sent) yang menunjukkan bahwa sedang terjadi pengiriman, *event* r (receive) yang menunjukkan bahwa sedang terjadi penerimaan, dan *event* D (Drop) yang menunjukkan bahwa ada paket yang dibuang (*drop*) biasanya terjadi pada detik-detik terakhir saat simulasi akan berakhir.

2. Kolom 1 ($x[1])

   Satuan waktu dimana suatu *event* sedang berlangsung.

3. Kolom 2 ($x[2])

   Nomor *node* yang sedang terlibat dengan aktivitas pada saat itu.

4. Kolom 3 ($x[3])

   Trace level yaitu MAC dan AGT. MAC merupakan MAC layer dan AGT merupakan paket pada transport layer.

5. Kolom 5 ($x[5])

   Nomor urut paket dimulai dari urutan 0 (nol).

6. Kolom 7 ($x[7])

   Ukuran (*length*) dari paket baik yang dikirim, diterima, maupun yang *drop* dalam satuan *byte*.

7. Kolom 8 ($x[8])

   Informasi MAC layer.

8. Kolom 11 ($x[11])

   Nomor urut dan pemberitahuan nomor (informasi tcp).

9. Kolom 12 ($x[12])

   Format dari mekanisme. Misalnya 0 0 adalah mekanisme *send* dan 0 1 adalah mekanisme *receive*.

Informasi-informasi yang dijelaskan pada subbab di atas digunakan dalam penghitungan yang akan dijelaskan pada subbab-subbab di bawah ini.

3.3.6 Proses Penghitungan Parameter *Delay*

Dalam penghitungan *delay* yang dibutuhkan adalah selisih/ waktu tunda antara waktu pada saat paket diterima dengan waktu pada saat paket dikirim. Informasi yang digunakan sebagai acuan pertama-tama data diurutkan berdasarkan nomor urut paket yang tertera pada kolom 5, lalu dikelompokkan berdasarkan *trace level* pada kolom 4 dimulai dari AGT terlebih dahulu karena AGT merupakan paket pada *transport layer* dan MAC adalah MAC *layer*nya. Secara otomatis data akan urut dengan pola *event* s, r, s, r, …, s, r. Sehingga dapat dihitung *delay* yang terjadi satu per satu, yang digunakan adalah waktu dengan status *event* s dan r pada kolom 1. Rata-rata *delay* yang terjadi pada masing-masing percobaan diperoleh dari total seluruh *delay* yang terhitung dibagi dengan banyaknya proses komunikasi yang terjadi dimana suatu proses komunikasi yang sukses adalah ketika data yang dikirim berhasil diterima oleh *receiver*.

Sedangkan, paket yang dikirim tetapi di*drop* oleh sistem tidak digunakan dalam penghitungan *delay*. Tetapi akan digunakan pada penghitungan PLR. Informasi-informasi yang lain digunakan untuk memastikan data-data yang sudah diurutkan adalah benar.

3.3.7 Proses Penghitungan Parameter PLR

Dalam penghitungan PLR yang dibutuhkan adalah ukuran paket yang hilang pada saat komunikasi berlangsung dengan menggunakan data yang sudah diurutkan seperti yang sudah dijelaskan pada penghitungan *delay* pada subbab sebelumnya.

Dalam penghitungan PLR, yang digunakan adalah ukuran paket pada kolom 7 yaitu ukuran paket yang dikirim (*sent*) dikurangkan dengan ukuran paket yang berhasil diterima (*received*). Rata-rata PLR yang terjadi pada masing-masing percobaan diperoleh dari total seluruh paket yang hilang pada saat komunikasi terjadi dijumlah dengan ukuran seluruh paket yang di*drop* dibagi dengan total ukuran paket yang dikirim pada masing-masing percobaan.

3.3.8 Proses Penghitungan Parameter *Utilisasi Bandwidth*

Dalam penghitungan *utilisasi bandwidth* yang dibutuhkan adalah total ukuran paket yang dikirim dan diterima pada saat komunikasi berlangsung dengan menggunakan data yang sudah diurutkan seperti yang sudah dijelaskan pada penghitungan *delay* sebelumnya. Dalam penghitungan *utilisasi bandwidth*, yang digunakan adalah ukuran paket pada kolom 7 yaitu ukuran paket yang dikirim dijumlahkan dengan ukuran paket yang berhasil.

Setelah diperoleh total ukuran paket yang berhasil dikirim dan diterima, rata-rata *utilisasi bandwidth* yang terjadi pada masing-masing percobaan dapat diperoleh dari total seluruh paket yang dikirim dan berhasil diterima pada saat komunikasi terjadi dikalikan dengan 8 bit untuk memperoleh penghitungan dalam satuan bit karena ukuran paket yang ditunjukan dari tracefile dalam ukuran *byte* dibagi dengan *bandwidth* sistem yang disediakan oleh sistem. *Bandwidth default*

yang disediakan jika tidak diatur secara manual adalah 1.2 Mbps (1200000 bit per sekon). Sesuai dengan tipe jaringan yang digunakan yaitu standar *Wirelessphy* 802.11. (Gong, Shan. 2006).