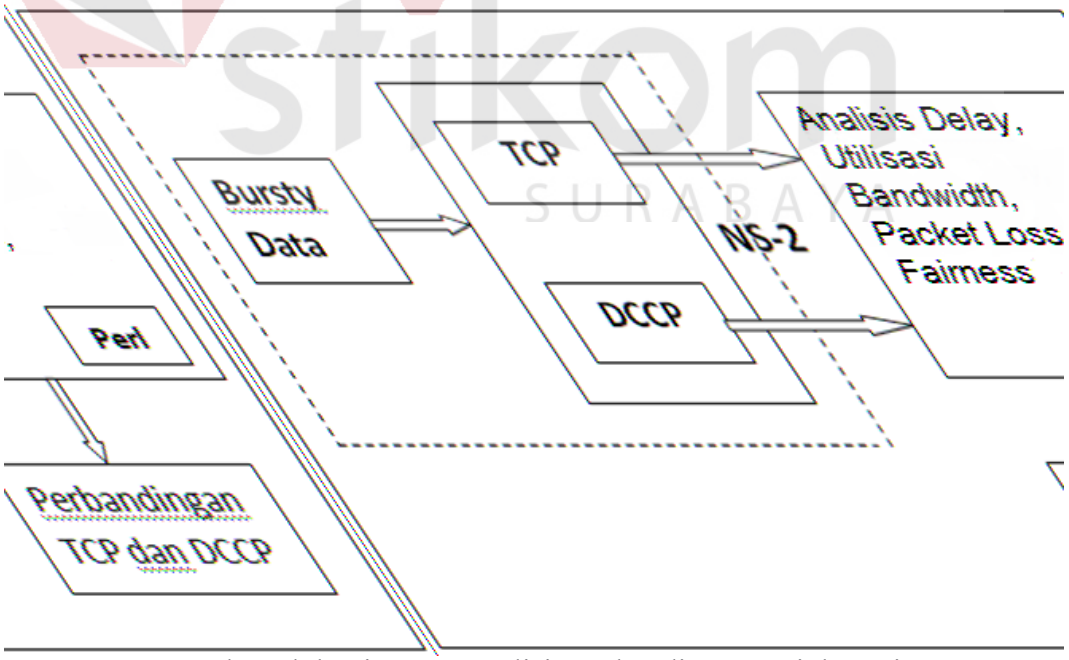
BAB III

MODEL PENELITIAN

3.1 Metode Penelitian

Metode penelitian yang digunakan dalam pengerjaan tugas akhir ini adalah studi kepustakaan, percobaan dan analisis. Dengan ini penulis berusaha untuk mengumpulkan data dan informasi-informasi serta materi yang bersifat teoritis yang sesuai dengan permasalahan. Hal tersebut diperoleh dari buku-buku, materi perkualiahan serta literatur dari internet, jurnal dan percobaan dengan bantuan *Network Simulator 2*.

Analisis perbandingan unjuk kerja protokol TCP dengan DCCP menggunakan data yang bersifat *bursty* ini dapat dijelaskan dengan lebih baik melalui blok diagram seperti yang terlihat pada Gambar 3.1.



Gambar 3.1. Blok Diagram Analisis Perbandingan Unjuk Kerja Protokol TCP dengan DCCP Menggunakan Trafik Data *Bursty*

Pada Gambar 3.1 dapat dikelompokkan menjadi tiga bagian utama, yaitu *input*, proses dan *output* yang berupa hasil analisis perbandingan.

1. *Input*

   Data inputan yang digunakan dalam membandingkan kedua protokol di dapat dengan melakukan pembangkitan paket data pada NS-2 dengan ukuran paket sesuai dengan data *bursty*.

2. Proses

   Data inputan dijalankan di atas protokol TCP dan DCCP menggunakan pemrograman TCL. NS-2 memanggil program TCL sehingga didapatkan hasil *trace file* dan simulasi pada NAM. Hasil *trace file* diolah berdasarkan parameter uji utilisasi *bandwidth, delay, packet loss* dan f*airness* dengan pemrograman *perl.*

3. *Output*

   Bagian *output* menunjukan analisis terhadap data yang dihasilkan berupa analisis perbandingan utilisasi *bandwidth,* analisis perbandingan *delay,* analisis perbandingan *packet loss* dan analisis perbandingan *fairness* dari protokol TCP dengan DCCP. Analisis tersebut disajikan dalam bentuk pembahasan berdasarkan studi literatur dan simulasi yang telah dilakukan pada bagian proses yang nantinya dapat dipaparkan melalui tampilan grafik.

3. 2 Prosedur Penelitian

Prosedur ini menjelaskan tentang langkah-langkah yang dilakukan dalam pengujian seperti diagram alir pada Gambar 3.2.

Gambar 3.2. Prosedur Pelaksanaan Penelitian

3.2.1 Pengumpulan Data dan Parameter Penelitian

Dalam tahap ini dilakukan pengumpulan data yang akan digunakan untuk melakukan pengujian. Protokol yang akan digunakan adalah TCP, DCCP CCID2 dan DCCP CCID3. Waktu percobaan yang akan dipakai selama 30 detik. Penentuan nilai *bandwidth* yang bervariasi untuk memberikan efek dalam membandingkan protokol berdasarkan kondisi-kondisi yang telah ditentukan. Karakteristik antrian yang digunakan yaitu *drop tail* dimana data terakhir yang datang akan dibuang apabila kapasitas dari memori telah penuh (The VINT Project, 2011).

Parameter pembanding yang digunakan untuk analisis masing-masing protokol dalam pengujian ini yaitu utilisasi *bandwidth, packet loss, delay* dan *fairness.*

1. Analisis perbandingan utilisasi *bandwidth*.

   Utilisasi *bandwidth* dianalisis berdasarkan seberapa besar prosentase *bandwidth* suatu *link* yang menghubungkan antara kedua sisi yaitu sisi pengirim dan sisi penerima.

2. Analisis perbandingan *packet loss*

   Packet loss dianalisis berdasarkan berapa banyak paket yang hilang atau gagal mencapai tujuan pada waktu paket sedang berjalan. Paket hilang dengan nilai 0%, termasuk kategori sangat bagus, paket hilang kurang dari 3% termasuk dalam kategori bagus. Kurang dari 15% termasuk dalam kategori sedang dan kategori jelek dengan nilai *packet loss* 25%.

3. Analisis perbandingan *delay.*

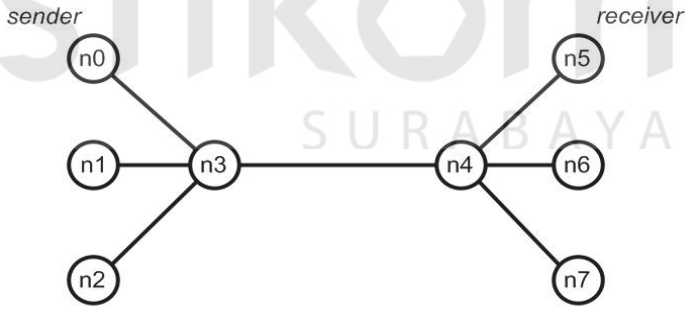
Delay dianalisis berdasarkan berapa waktu tunda dari paket yang diterima sampai ke tujuan dari masing-masing protokol. *Delay* dengan kategori sangat bagus jika < 150 ms, kategori bagus dengan nilai *delay* 150 s/d 300 ms, kategori sedang dengan nilai *delay* 300 s/d 450 ms dan kategori jelek dengan nilai *delay* > 450 ms.

4. Analisis *Fairness*.

Hasil keluaran simulasi dianalisis menggunakan rumus *fairness* dan didapatkan kesimpulan apakah kedua protokol membagi sumber daya dengan adil.

3.2.2 Desain dan Pembuatan Topologi

Topologi pengujian menggunakan delapan *node* seperti pada Gambar 3.3.



Gambar 3.3. Topologi *Dumb-bell* Dalam Sistem

Kedelapan node masing-masing yaitu n0, n1, n2, n3, n4, n5, n6 dan n7. Node n0, n1 dan n2 adalah pengirim, sedangkan node n5, n6 dan n7 merupakan penerima sedangkan *single bottleneck link* terdapat pada jalur node n3-n4. Gambar 3.3. Aliran data mengalir dari satu pengirim menuju ke satu penerima. Data dari n0 menuju ke n5 berjalan di atas protokol DCCP-ID2, n1 menuju ke n6 dengan

protokol TCP dan n2 menuju ke n7 dengan protokol DCCP-ID3. Simulasi dilakukan dengan cara bergantian antara TCP dengan DCCP-ID2 dan TCP dengan DCCP-ID3.

3.2.3 Pembuatan Skrip

Pembuatan skrip adalah pembuatan skrip tcl yang disesuaikan dengan data yang sudah ditentukan. Ada empat langkah dalam pembuatan skrip yaitu inisialisasi, pembuatan *node* dan *link*, penggabungan aplikasi pada TCP dan DCCP serta mengatur waktu jalannya skrip.

1. Inisialisasi

Simulasi dengan NS-2 selalu dimulai dengan mendefinisikan sebuah variabel atau *object* sebagai *instance* dari kelas simulator dengan cara sebagai berikut:

```
set ns [new Simulator]
```

Untuk menyimpan data keluaran hasil dari simulasi (*trace files*) dan juga sebuah *file* lagi untuk kebutuhan simulasi (*nam files*) akan dibuat dua buah *file* dengan perintah "open" seperti berikut:

```
set tracefile1 [open out.tr w]
$ns trace-all $tracefile1
set namfile [open out.nam w]
$ns namtrace-all $namfile
```

Skrip di atas akan membuat *trace file* dengan nama *out.tr* yang akan digunakan untuk menyimpan data hasil simulasi dan *file* out.nam untuk menyimpan data hasil visualisasi. Deklarasi „w" pada bagian akhir dari perintah open adalah perintah *write*.

Selanjutnya cara mendeklarasikan prosedur "finish" seperti di bawah ini:

```
proc finish {} {
       global ns tracefile1 namfile
       $ns flush-trace
       close $tracefile1
       close $namfile
       exec nam out.nam &
       exit 0
       }
```

Perhatikan bahwa prosedur tersebut menggunakan variabel global ns, tracefile1 dan namfile. Perintah *flush-trace* digunakan untuk menyimpan semua data hasil simulasi ke dalam tracefile1 dan namfile. Perintah *exit* akan mengakhiri aplikasi dan mengembalikan status dengan angka 0 ke sistem. Perintah *exit* 0 adalah perintah *default* untuk membersihkan memori dari sistem, nilai yang lain dapat digunakan misalnya untuk memberikan status gagal.

Pada bagian akhir dari program, prosedur 'finish' harus dipanggil dengan indikasi waktu (dalam detik) terminasi dari program, misalnya:

```
$ns at 29.5 finish
```

Selanjutnya untuk memulai simulasi atau menjalankan program dapat dilakukan dengan menggunakan perintah:

```
$ns run
```

2. Membuat *node* dan link

Mendefinisikan sebuah *node* pada NS-2 pada dasarnya adalah membuat sebuah variabel, sebagai berikut:

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
```

Selanjutnya untuk menggunakan *node* n0 dilakukan dengan cara memanggil variabel $n0. Demikian pula *node* yang lain dapat dibuat dengan cara yang sama dengan kebutuhan dalam simulasi.

Setelah *node* terbuat, maka langkah selanjutnya adalah membuat *link* yang akan membuat hubungan antar *node*. Sebuah *link* untuk menghubungkan node $n0 dan $n3 dengan *bidirectional link* berkapasitas 100Mb dan dengan waktu tunda akibat propagasi sebesar 5 ms dapat dibuat dengan cara seperti di bawah ini, begitu pula pada *link* antar *node* lainnya.

```
$ns duplex-link $n0 $n3 100Mb 5ms DropTail
$ns duplex-link $n1 $n3 100Mb 5ms DropTail
$ns duplex-link $n2 $n3 100Mb 5ms DropTail
$ns simplex-link $n3 $n4 1Mb 10ms DropTail
$ns simplex-link $n4 $n3 1Mb 10ms DropTail
$ns duplex-link $n4 $n5 100Mb 5ms DropTail
$ns duplex-link $n4 $n6 100Mb 5ms DropTail
$ns duplex-link $n4 $n7 100Mb 5ms DropTail
```

Pada NS-2, antrian keluaran dari sebuah *node* didefinisikan sebagai bagian dari sebuah *link*. Opsi "DropTail" berarti bahwa data terakhir yang datang akan dibuang apabila kapasitas dari memori telah penuh.

3. Menggabungkan Aplikasi

Untuk mendefinisikan jenis protokol yang akan digunakan NS-2 menggunakan perintah

```
set TCP [new Agent/TCP] #menjalankan TCP
set cbr1 [new Application/Traffic/CBR]
set dccp0 [new Agent/DCCP/TCPlike]
#menjalankan dccp- id2
#untuk menjalankan dccp-id3 perintah "TCPlike"
#diganti dengan "TFRC"
Set cbr0 [new Application/Traffic/CBR]
```

Selanjutnya untuk menggabungkan protokol ini pada sebuah *node* dari sumber yaitu n0 ke tujuan yaitu n5 dan dari n1 ke n6, menggunakan perintah di bawah ini:

```
$ns attach-agent $n0 $dccp0
$ns attach-agent $n5 $dccpsink0
$ns attach-agent $n1 $tcp
$ns attach-agent $n6 $null
```

Langkah terakhir adalah menentukan jenis aplikasi dan menggabungkan dengan protokol TCP dan DCCP yang telah didefinisikan sebagai berikut:

```
$cbr0 attach-agent $dccp0
$cbr1 attach-agent $tcp
```

4. Mengatur jadwal eksekusi pada skrip

Karena NS merupakan simulator kejadian diskrit, maka skrip yang telah dibuat dengan Tcl perlu mendefinisikan waktu eksekusi dari setiap kejadian. Setiap kejadian pada skrip Tcl dapat didefinisikan dengan perintah:

```
$ns at “waktu kejadian”
```

Sehingga untuk menentukan kapan aplikasi cbr pada masing-masing protokol saat mulai mengirimkan data dan kapan selesai mengirimkan data digunakan perintah berikut:

```
$ns at 0.5 "$cbr1 start"
$ns at 5.0 "$cbr0 start"
$ns at 29.5 "$cbr1 stop"
$ns at 29.5 "$cbr0 stop"
```

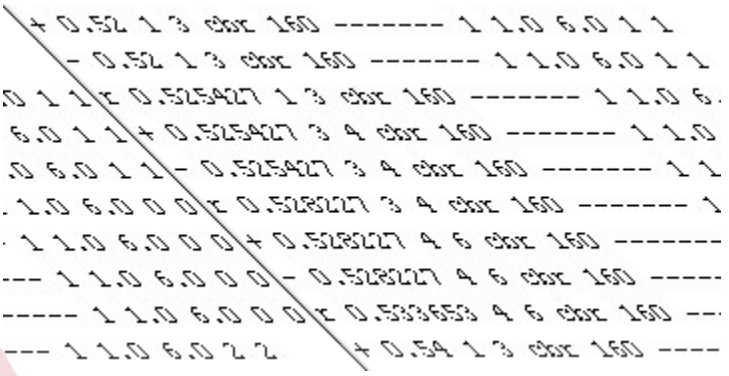
3.2.4 Menjalankan Skrip di NS-2

Setelah pembuatan skrip .tcl selesai, skrip dijalankan diatas aplikasi NS-2 dengan cara mengetik *ns nama_file.tcl* pada terminal. Apabila hasil yang ditampilkan berupa gambar visualisasi NAM maka skrip yang dibuat sudah benar dan sesuai dengan konfigurasi.

Hasil *output* skrip .tcl adalah *file out.tr* dan out.nam. *file out.tr* adalah tempat untuk menyimpan data hasil simulasi sedangkan *file out.nam* adalah tempat untuk menyimpan hasil visualisasi.

3.2.5 Pengolahan Data

Setelah didapatkan hasil dari *file out.tr* seperti Gambar 3.4, langkah selanjutnya adalah melakukan pemrosesan *file* dengan menggunakan bahasa pemrogaman Perl.



Gambar 3.4. *Trace File* dari Program NS-2

Skrip perl yang dibuat terdiri dari lima bagian. Bagian pertama dan kedua skrip sama untuk kelima parameter karena pengecekan dan pembacaan perkolom sama. Berikut lima bagian skrip yang digunakan.

1. Pengecekan *file input*:

```
Open (DATA, "<$infile"),
|| die "cannot open $infile $!";
```

2. Pembacaan perkolom dari *file input* menggunakan perintah:

```
while(<(DATA)>){
@x = split(„ „);
#digunakan pemisah dengan menggunakan spasi
```

3. Proses seleksi untuk perhitungan nilai dan parameter:

```
if ($x[8] eq '1.0' && $x[9] eq '6.0')
      {
       if ($x[0] eq 'r')
            {
             $sum=$sum+$x[5];
            }
       }
if ($x[8] eq '0.0' && $x[9] eq '5.0')
      {
       if ($x[0] eq 'r')
            {
             $sum1=$sum1+$x[5];
            }
```

```
}
```

Proses seleksi di atas merupakan proses seleksi untuk perhitungan nilai dan parameter *throughput* pada TCP dengan DCCP CCID2 (TCP jalan dahulu). Ketiga parameter lainnya akan dijelaskan pada sub-bab 3.3 beserta dengan *flowchart* parameter uji lainnya.

4. Perhitungan nilai dari parameter unjuk kerja.

```
{
$time=$x[1]-0.5;
$time1=$x[1]-5.0;
$throughput=$sum/$time;
$throughput1=$sum1/$time1;
}
```

Perhitungan pada masing-masing skrip disesuaikan dengan kebutuhan parameter yang dihitung. Contoh di atas adalah skrip perhitungan nilai *throughput*. Ketiga parameter lainnya akan dijelaskan pada sub-bab 3.3 beserta dengan *flowchart* parameter uji lainnya.

5. Menampilkan *output* dari hasil perhitungan dengan perintah:

```
Print STDOUT "Throughput TCP $throughput
Throughput CCID2 $throughput1\n"
```

Skrip di atas merupakan skrip untuk menampilkan nilai *throughput* TCP dan DCCP CCID2. Parameter-parameter lainnya akan dijelaskan pada sub-bab 3.3, beserta dengan *flowchart* keseluruhan parameter uji.

Perhitungan utilisasi *bandwidth dan fairness* dilakukan secara khusus. Perhitungan keduanya dilakukan secara manual dengan inputan hasil dari skrip *perl* perhitungan *throughput*.

3.2.6 *Plotting*

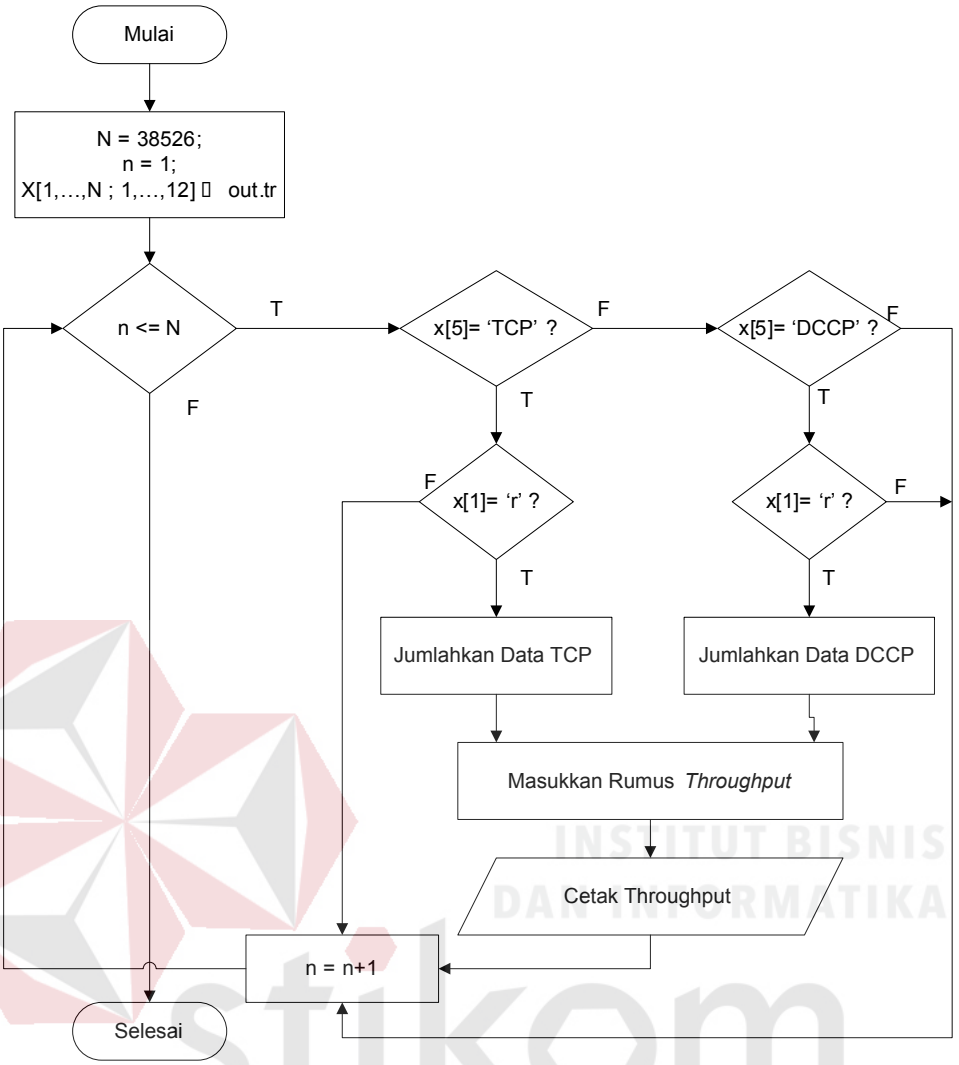
Setelah mendapatkan nilai utilisasi *bandwidth, delay, packet loss* dan *fairness,* selanjutnya adalah menggambarkan ke dalam grafik menggunakan *LibreOffice Calc* untuk memudahkan dalam melakukan perbandingan.

3.3 Perancangan Skrip Perl

Bagian ini adalah perancangan skrip *perl* dari keseluruhan parameter. Bagian pertama dan kedua skrip telah dijelaskan pada sub-bab 3.2.3 dan sub-bab 3.2.5. Berikut ini adalah langkah-langkah selanjutnya untuk masing-masing parameter disertai dengan *flowchart*.

3.3.1 Utilisasi *Bandwidth*

Utilisasi *bandwidth* dihitung dari nilai *throughput*. *Flowchart throughput* ini berisi proses seleksi, perhitungan nilai *throughput* dan menampilkan *output* hasil olah data *trace file.* Gambar 3.5 merupakan *flowchart* dari skrip perl *throughput*. Langkah selanjutnya adalah mengolah data *throughput* dengan rumus utilisasi *bandwidth* menggunakan *calculator*.



Gambar 3.5. *Flowchart Throughput* untuk Parameter Utilisasi *Bandwidth*

Berikut ini adalah penjelasan tentang cara kerja *flowchart throughput* pada Gambar 3.5.

1. Mulai.
2. Inisialisasi variabel, N adalah jumlah baris dalam *file out.tr* , n adalah jumlah baris yang di sedang diakukan perhitungan, X adalah *array* dari *file out.tr* meliputi jumlah baris dan kolom.
3. Pengecekan data. Jika masih ada data pada baris selanjutnya maka menuju nomor 4 jika tidak ada ke nomor 13.

4. Pengecekan kolom sumber yaitu x[8] dan kolom tujuan yaitu x[9]. Jika kedua kondisi sesuai maka kondisi bernilai benar, lanjut ke nomor 5 dan jika kedua kondisi atau salah satu kondisi tidak terpenuhi maka kondisi bernilai salah, kembali ke nomor 6.
5. Pengecekan kolom kejadian yaitu x[0] (kolom ke-0). Jika kolom sama dengan "r" maka lanjut ke nomor 8 dan jika salah, kembali ke nomor 12.
6. Pengecekan kolom sumber yaitu x[8] dan kolom tujuan yaitu x[9]. Jika kedua kondisi sesuai maka kondisi bernilai benar, lanjut ke nomor 7 dan jika kedua kondisi atau salah satu kondisi tidak terpenuhi maka kondisi bernilai salah, lanjut ke nomor 12.
7. Pengecekan kolom kejadian yaitu x[0] (kolom ke-0). Jika kolom sama dengan "r" maka lanjut ke nomor 9 dan jika salah, kembali ke nomor 12.
8. Menjumlahkan data TCP dan menyimpan hasilnya sementara dalam variabel TCP. Kembali ke nomor 3.
9. Menjumlahkan data DCCP dan menyimpan hasilnya sementara dalam variabel DCCP. Kembali ke nomor 3.
10. Perhitungan dengan menggunakan rumus *throughput*. Lanjut ke nomor 11.
11. Cetak hasil perhitungan nomor 10 dan lanjut ke nomor 12.
12. Variabel n ditambah 1, variabel ini digunakan untuk menghitung baris yang ada pada *file out.tr* . Selanjutnya menuju ke nomor 3.
13. Selesai.

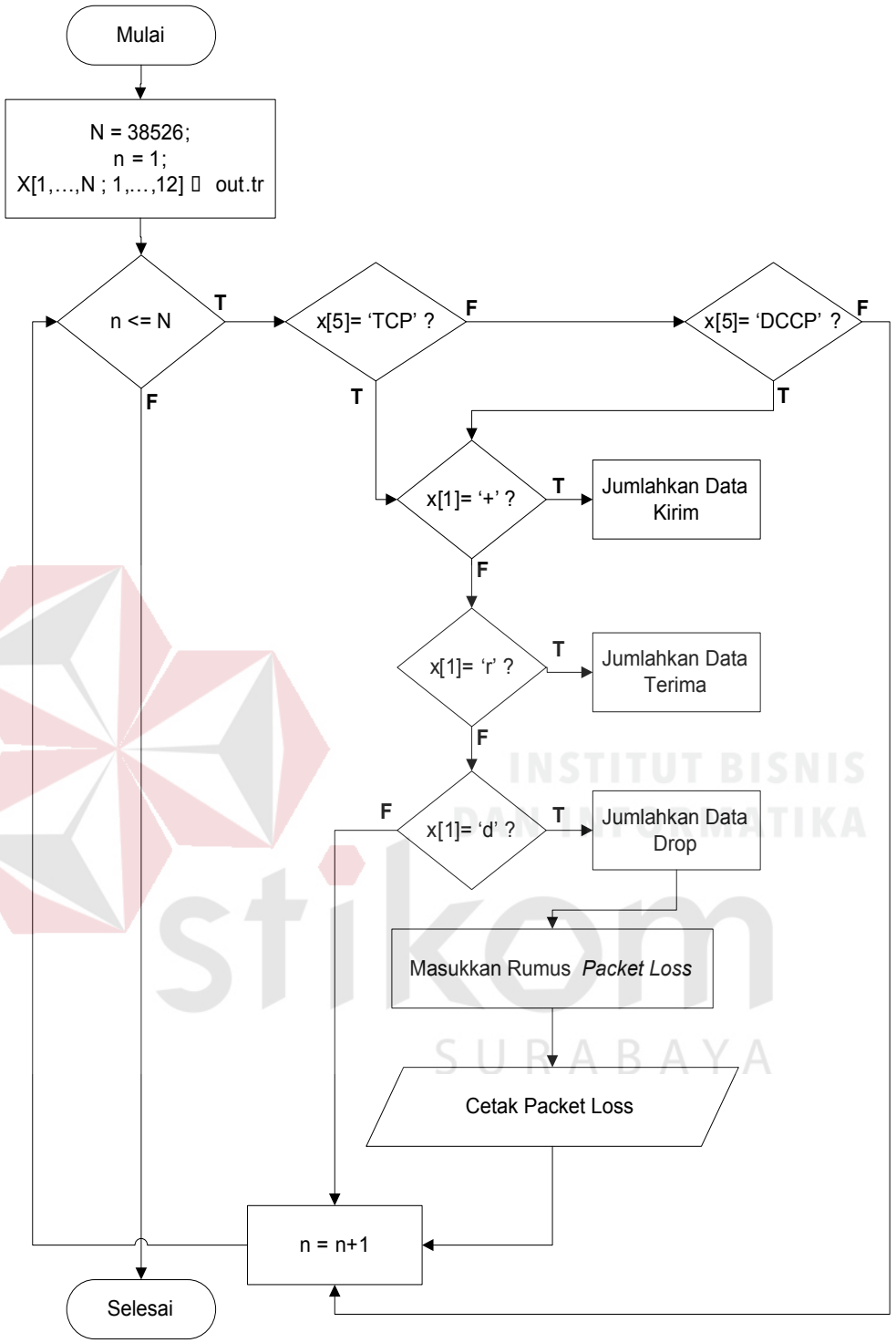
3.3.2 *Packet Loss*

Flowchart packet loss ini berisi proses seleksi, perhitungan nilai dari parameter dan menampilkan *output* hasil olah data *trace file*. Gambar 3.6 merupakan *flowchart* keseluruhan dari skrip perl *packet loss*.

Berikut ini adalah penjelasan tentang cara kerja *flowchart* program p*acket loss* pada Gambar 3.6

1. Mulai.
2. Inisialisasi variabel, N adalah jumlah baris dalam *file out.tr* , n adalah jumlah baris yang di sedang diakukan perhitungan, X adalah *array* dari *file out.tr* meliputi jumlah baris dan kolom.
3. Pengecekan data. Jika masih ada data pada baris selanjutnya maka menuju nomor 4 jika tidak ada ke nomor 15.
4. Pengecekan kolom sumber yaitu x[8] dan kolom tujuan yaitu x[9]. Jika kedua kondisi sesuai maka kondisi bernilai benar (TCP), lanjut ke nomor 5 dan jika kedua kondisi atau salah satu kondisi tidak terpenuhi maka kondisi bernilai salah, lanjut ke nomor 8.
5. Pengecekan kolom kejadian yaitu x[0] (kolom ke-0). Jika kolom sama dengan "+" maka lanjut ke nomor 9 dan jika salah, lanjut ke nomor 6.
6. Pengecekan kolom kejadian yaitu x[0] (kolom ke-0). Jika kolom sama dengan "r" maka lanjut ke nomor 10 dan jika salah, lanjut ke nomor 7.
7. Pengecekan kolom kejadian yaitu x[0] (kolom ke-0). Jika kolom sama dengan "d" maka lanjut ke nomor 11 dan jika salah, lanjut ke nomor 12.
8. Pengecekan kolom sumber yaitu x[8] dan kolom tujuan yaitu x[9]. Jika kedua kondisi sesuai maka kondisi bernilai benar (DCCP), kembali ke nomor

5 dan jika kedua kondisi atau salah satu kondisi tidak terpenuhi maka kondisi bernilai salah, lanjut ke nomor 14.


Mulai
N = 38526;
n = 1;
X[1,…,N ; 1,…,12] out.tr
n <= N
T
F
x[5]= 'TCP' ?
F
T
x[5]= 'DCCP' ?
F
T
x[1]= '+' ?
T
Jumlahkan Data Kirim
F
x[1]= 'r' ?
T
Jumlahkan Data Terima
F
x[1]= 'd' ?
F
T
Jumlahkan Data Drop
Masukkan Rumus Packet Loss
Cetak Packet Loss
n = n+1
Selesai

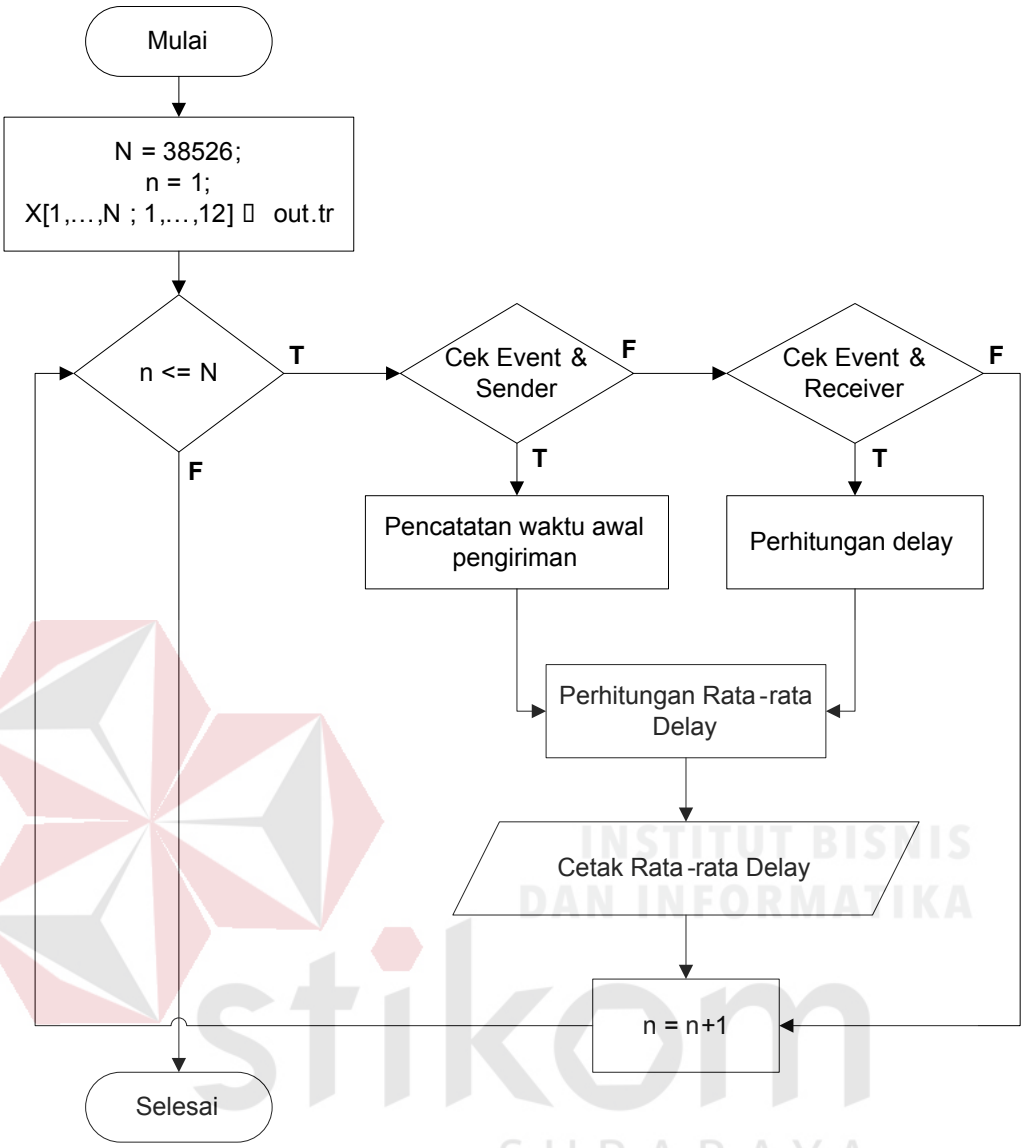
Gambar 3.6. *Flowchart* Program *Packet Loss*

9. Menjumlahkan data kirim, hasilnya disimpan sementara dalam variabel kirim.
10. Menjumlahkan data terima, hasilnya disimpan sementara dalam variabel terima.
11. Menjumlahkan data drop, hasilnya disimpan sementara dalam variabel drop. Lanjut ke nomor 12.
12. Perhitungan dengan menggunakan rumus *packet loss*. Lanjut ke nomor 13.
13. Cetak hasil perhitungan nomor 12 dan lanjut ke nomor 14.
14 Variabel n ditambah 1, variabel ini digunakan untuk menghitung baris yang ada pada *file out.tr.* Selanjutnya menuju ke nomor 3.
15. Selesai.

3.3.3 *Delay*

Berikut ini adalah penjelasan tentang cara kerja *flowchart* program *delay* pada Gambar 3.7.

1. Mulai.
2. Inisialisasi variabel, N adalah jumlah baris dalam *file out.tr* , n adalah jumlah baris yang di sedang diakukan perhitungan, X adalah *array* dari *file out.tr* meliputi jumlah baris dan kolom.
3. Pengecekan data. Jika masih ada data pada baris selanjutnya maka menuju nomor 4 jika tidak ada ke nomor 11.



Gambar 3.7. *Flowchart* Program *Delay*

4. Pengecekan kolom kejadian yaitu x[0] (kolom ke-0) dan kolom node sumber yaitu x[2]. Jika kedua kondisi sesuai maka kondisi bernilai benar, lanjut ke nomor 5 dan jika kedua kondisi atau salah satu kondisi tidak terpenuhi maka kondisi bernilai salah, lanjut ke nomor 6.
5. Melakukan pencatatan pada waktu awal pengiriman data. Melihat pada kolom x[1]. Lanjur ke nomor 9.
6. Pengecekan kolom kejadian yaitu x[0] (kolom ke-0) dan kolom node tujuan yaitu x[3]. Jika kedua kondisi sesuai maka kondisi bernilai benar, lanjut ke

nomor 7 dan jika kedua kondisi atau salah satu kondisi tidak terpenuhi maka kondisi bernilai salah, lanjut ke nomor 10.

7. Perhitungan dengan menggunakan rumus *delay*. Lanjut ke nomor 8.
8. Perhitungan dengan menggunakan rumus rata-rata *delay*. Lanjut ke nomor 9.
9. Cetak hasil perhitungan nomor 8 dan lanjut ke nomor 10.
10. Variabel n ditambah 1, variabel ini digunakan untuk menghitung baris yang ada pada *file out.tr.* Selanjutnya menuju ke nomor 3.
11. Selesai

3.3.4 *Fairness*

Fairness dihitung dari nilai *throughput*. *Flowchart throughput* berisi proses seleksi, perhitungan nilai *throughput* dan menampilkan *output* hasil olah data *trace file. Flowchart throughput* sudah ditampilkan pada Gambar 3.5. Langkah selanjutnya adalah mengolah data *throughput* dengan rumus *fairness* menggunakan *Libre office Calculator*.