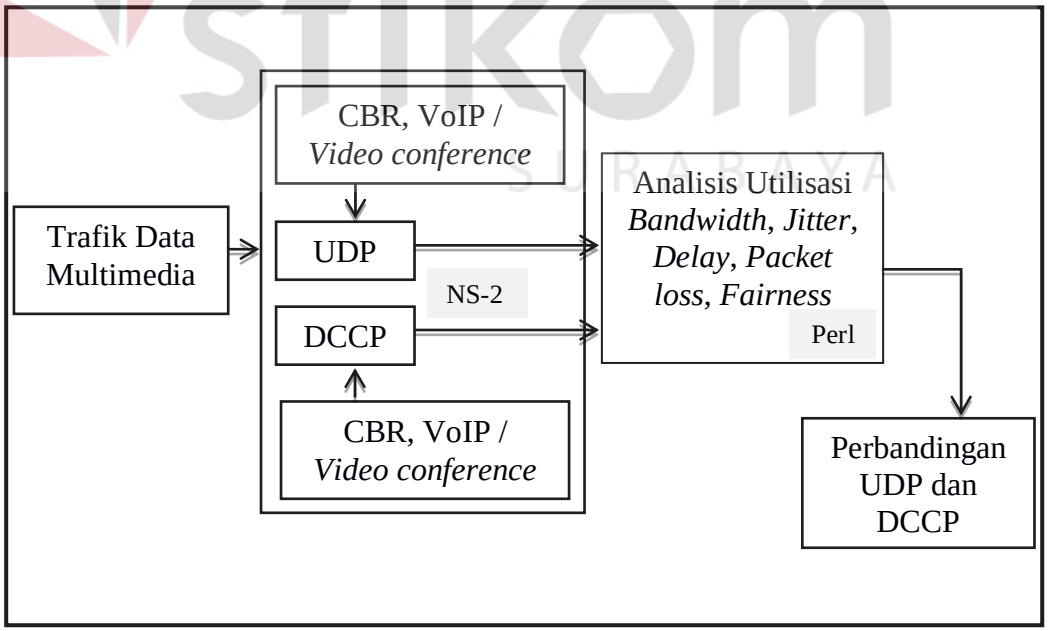
# BAB III

# METODE 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 UDP dengan DCCP menggunakan data multimedia 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 UDP dengan DCCP Menggunakan Trafik Data Multimedia

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

1. Bagian Input Data

   Data inputan yang digunakan dalam membandingkan kedua protokol didapat dengan melakukan pembangkitan paket data pada NS-2 dengan ukuran paket sesuai dengan data multimedia.

2. Proses

   Data inputan dijalankan di atas protokol UDP 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, jitter, packet loss* dan JFI dengan pemrograman *perl.*

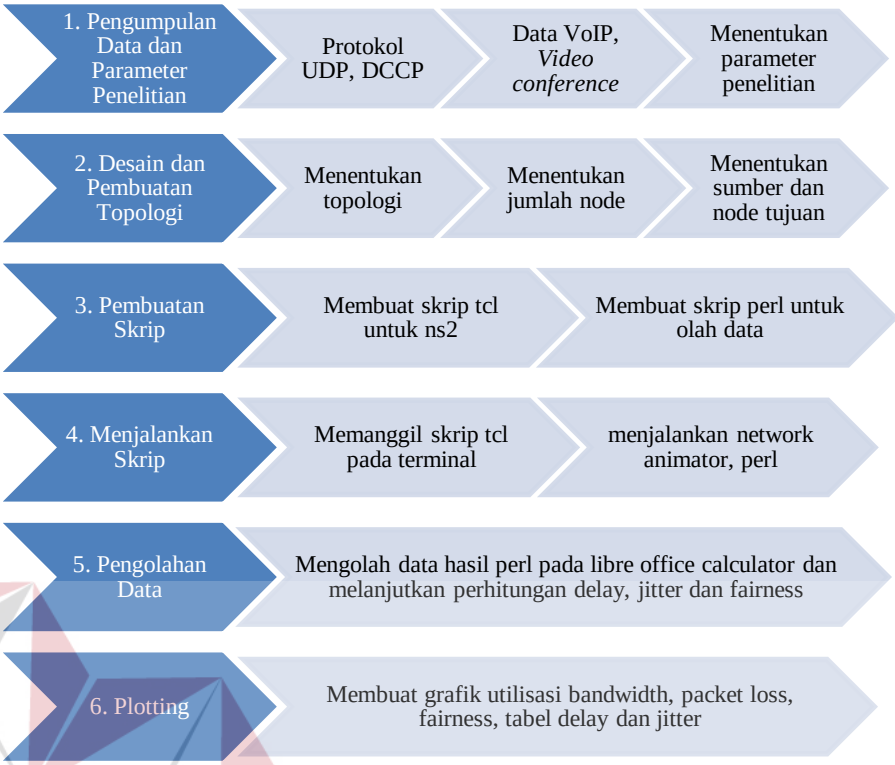
3. Output

   Bagian output menunjukan analisis terhadap data yang dihasilkan berupa analisis perbandingan *utilisasi bandwidth,* analisis perbandingan *delay,* analisis perbandingan *packet loss,* analisis perbandingan *jitter* dan analisis perbandingan *fairness* dari protokol UDP 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. Ada beberapa data yang dibutuhkan meliputi data multimedia yang akan digunakan berupa VOIP dan *Video conference*. Protokol yang akan digunakan adalah UDP, DCCP CCID2 dan DCCP CCID3. Waktu percobaan yang akan dipakai selama 30 detik. Penentuan nilai ukuran paket, *bandwidth* dan *bit rate* 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, latency, packet loss, jitter* 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 kurang dari 3% termasuk dalam kategori bagus. Kurang dari 15% termasuk dalam kategori sedang.

3. Analisis perbandingan *delay*.

   *Delay* dianalisis berdasarkan berapa waktu tunda dari paket yang diterima sampai tujuan dari masing-masing protokol yang dibandingkan dengan data multimedia. Nilai *delay* yang bagus untuk data VoIP dan *Video conference* tidak boleh lebih dari 450 ms.

4. Analisis perbandingan *jitter.*

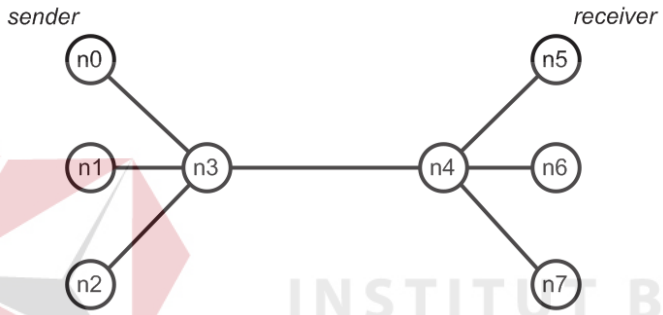
   *Jitter* merupakan variasi dari *delay, jitter* dianalisis berdasarkan keterlambatan transmisi data dari pengirim dan penerima dalam rentang waktu tertentu. Nilai *jitter* bagus jika bernilai kurang dari 76 ms. Sedang jika bernilai kurang dari 125 ms dan jelek jika bernilai 126 ms – 225 ms.

5. Analisis *Fairness.*

Output simulasi dianalisis menggunakan rumus *fairness* dan didapatkan kesimpulan apakah kedua protokol membagi sumberdaya dengan adil.

### 3.2.2 Desain dan Pembuatan Topologi

Topologi pengujian menggunakan delapan *node* seperti pada Gambar 3.3. Pengujian dengan topologi ini menggunakan algoritma data multimedia dan protokol yang berbeda.




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 3.4. Aliran data multimedia mengalir dari satu pengirim menuju ke satu penerima. Data multimedia dari n0 menuju ke n5 berjalan di atas protokol DCCP-ID2, n1 menuju ke n6 dengan protokol UDP dan n2 menuju ke n7 dengan protokol DCCP-ID3. Simulasi dilakukan dengan cara bergantian antara UDP dengan DCCP-ID2 dan UDP 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 UDP 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 3Mb 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 3Mb 5ms DropTail
$ns duplex-link $n1 $n3 3Mb 5ms DropTail
$ns duplex-link $n2 $n3 3Mb 5ms DropTail
$ns simplex-link $n3 $n4 100Kb 10ms DropTail
$ns simplex-link $n4 $n3 100Kb 10ms DropTail
$ns duplex-link $n4 $n5 3Mb 5ms DropTail
$ns duplex-link $n4 $n6 3Mb 5ms DropTail
$ns duplex-link $n4 $n7 3Mb 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 udp [new Agent/UDP] #menjalankan udp
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 $udp
$ns attach-agent $n6 $null
```

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

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

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.

```
+ 0.52 1 3 cbr 160 ------- 1 1.0 6.0 1 1
- 0.52 1 3 cbr 160 ------- 1 1.0 6.0 1 1
r 0.525427 1 3 cbr 160 ------- 1 1.0 6.0 1 1
+ 0.525427 3 4 cbr 160 ------- 1 1.0 6.0 1 1
- 0.525427 3 4 cbr 160 ------- 1 1.0 6.0 1 1
r 0.528227 3 4 cbr 160 ------- 1 1.0 6.0 0 0
+ 0.528227 4 6 cbr 160 ------- 1 1.0 6.0 0 0
- 0.528227 4 6 cbr 160 ------- 1 1.0 6.0 0 0
r 0.533653 4 6 cbr 160 ------- 1 1.0 6.0 0 0
+ 0.54 1 3 cbr 160 ------- 1 1.0 6.0 2 2
```

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 ($x[1]-$clock <= $granularity)
      {
      if ($x[0] eq 'r' && $x[2] eq $fromnode && $x[3]
eq $tonode)
            {
            if ($x[4] eq 'cbr')
                  {
                  $sum1=$sum1+$x[5];
                  }
            else ($x[4] ne 'cbr')
                  {
                  $sum2=$sum2+$x[5];
                  }
            }
      }
```

Proses seleksi di atas merupakan proses seleksi untuk perhitungan nilai dan parameter *throughput.* Keempat parameter lainnya akan dijelaskan pada bagian selanjutnya dalam bab ini beserta dengan *flowchart* kelima parameter.

4. Perhitungan nilai dari parameter unjuk kerja.

```
{
$throughput1=$sum1/$granularity;
$throughput2=$sum2/$granularity;
}
```

Perhitungan pada masing-masing skrip disesuaikan dengan kebutuhan parameter yang dihitung. Contoh di atas adalah skrip perhitungan nilai throughput. Variabel `$granularity` adalah lama waktu pengamatan. Keempat parameter lainnya akan dijelaskan pada bagian selanjutnya dalam bab ini beserta dengan *flowchart* kelima parameter.

5. Menampilkan output dari hasil perhitungan dengan perintah:

```
Print STDOUT "$x[1] $throughput1 $throughput2\n"
```

Skrip di atas merupakan skrip untuk menampilkan nilai throughput UDP dan DCCP. Keempat parameter lainnya akan dijelaskan pada bagian selanjutnya dalam bab ini beserta dengan *flowchart* kelima parameter.

Perhitungan *delay*, *jitter* dan *fairness* dilakukan secara khusus. Skrip Perl *delay* hanya melakukan seleksi pengambilan data yang diperlukan untuk keperluan perhitungan selanjutnya menggunakan LibreOffice Calc sedangkan untuk perhitungan *fairness* dilakukan pada LibreOffice Calc dengan inputan hasil dari skrip Perl perhitungan *throughput*.

### 3.2.6 Plotting

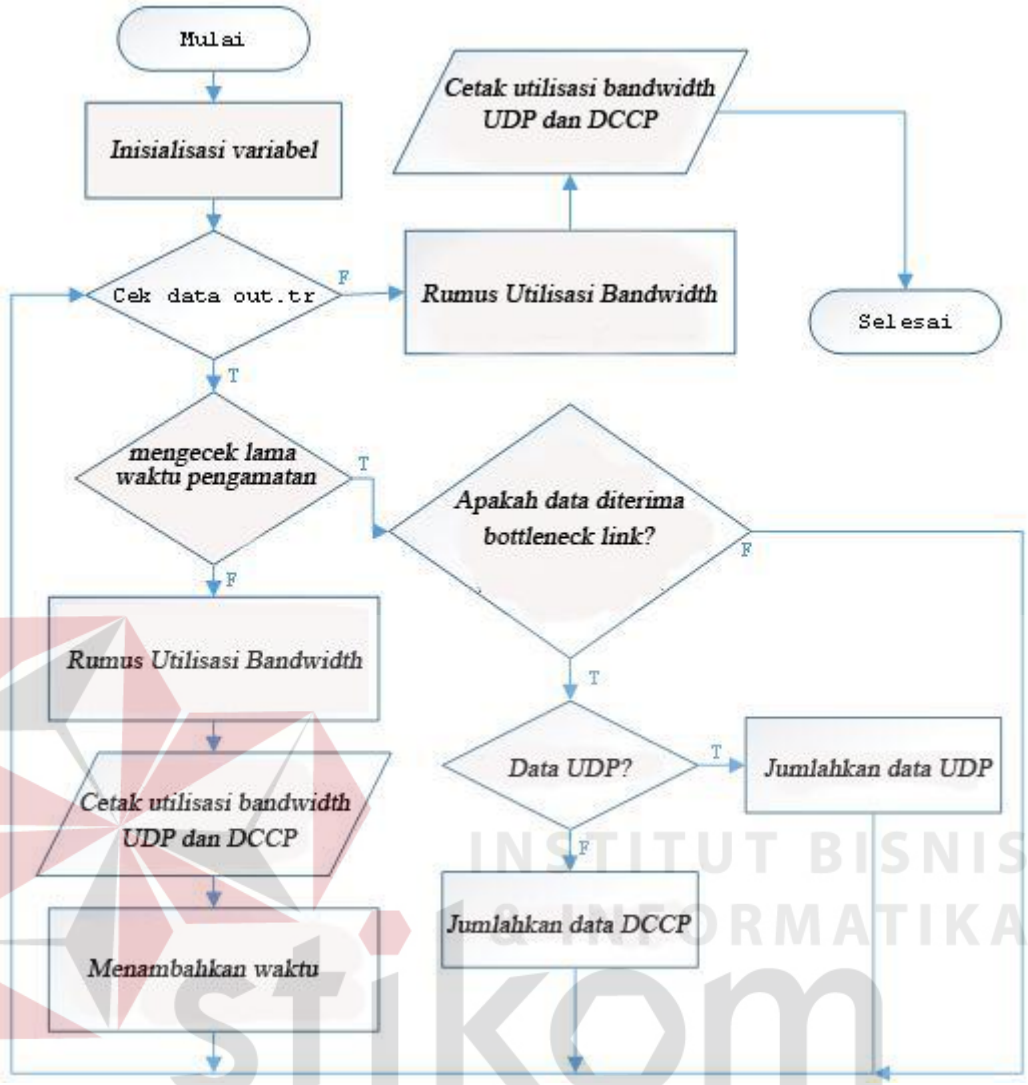
Setelah mendapatkan nilai *utilisasi bandwidth, delay, jitter, packet loss* dan *fairness* selanjutnya adalah menggambarkan ke dalam grafik menggunakan *LibreOffice Calc* untuk memudahkan dalam melakukan perbandingan.

## 3.3 Perancangan Skrip Perl

Bagian ini adalah perancangan skrip perl kelima parameter. Bagian pertama dan kedua skrip telah dijelaskan pada sub-bab sebelumnya. Berikut ini adalah langkah-langkah selanjutnya masing-masing parameter disertai dengan *flowchart*.

### 3.3.1 Utilisasi *Bandwidth*

*Flowchart* utilisasi *bandwidth* ini berisi proses seleksi, perhitungan nilai dari parameter dan menampilkan output hasil olah data *trace file*. Gambar 3.5. merupakan flowchart keseluruhan dari skrip perl utilisasi *bandwidth*.




Gambar 3.5 Flowchart Program Utilisasi *Bandwidth*

Granularity (waktu pengamatan) yang dimasukkan adalah satu sehingga setiap detik terdapat hasil utilisasi *bandwidth* yang ditampilkan. Di dalam rumus perhitungan parameter tertulis angka 12500. Nilai tersebut digunakan untuk menghitung utilisasi *bandwidth* data multimedia VoIP sedangkan untuk *Video conference* nilai dalam skrip diganti dengan nilai 48000 sesuai dengan kapasitas *bandwidth* pada *bottleneck link.*

Berikut ini adalah penjelasan tentang cara kerja *flowchart* utilisasi *bandwidth* pada Gambar 3.5:

1. Mulai.
2. Inisialisasi variabel udp, dccp dan clock menjadi 0.
3. Pengecekan data. Jika masih ada data pada baris selanjutnya maka menuju nomor 4 jika tidak ada ke nomor 12.
4. Pengecekan kolom waktu yaitu x[1] dengan nilai granularitas yang dimasukkan. Jika bernilai benar lanjutkan ke nomor 5 dan jika salah ke nomor 9.
5. Pengecekan kolom kejadian yaitu x[0] (kolom ke-0), kolom sumber yaitu x[2] dan kolom tujuan yaitu x[3]. Jika ketiga kondisi sesuai maka kondisi bernilai benar, lanjut ke nomor 6 dan jika ketiga kondisi atau salah satu kondisi tidak terpenuhi maka kondisi bernilai salah, kembali ke nomor 3.
6. Pengecekan kolom aplikasi yang digunakan yaitu x[4]. Jika kolom sama dengan ‘cbr’ (UDP) maka lanjut ke nomor 7 dan jika salah ke nomor 8.
7. Menjumlahkan data UDP dan menyimpan hasilnya sementara dalam variabel udp. Kembali ke nomor 3.
8. Menjumlahkan data DCCP dan menyimpan hasilnya sementara dalam variabel dccp. Kembali ke nomor 3.
9. Perhitungan dengan menggunakan rumus utilisasi *bandwidth*. Lanjut ke nomor 10.
10. Cetak hasil perhitungan nomor 9 dan lanjut ke nomor 11.
11. Variabel clock ditambah dengan granularitas dan disimpan pada variabel clock. Nilai variabel udp dan dccp dikembalikan menjadi 0. Selanjutnya menuju ke nomor 3.

12. Perhitungan dengan menggunakan rumus utilisasi *bandwidth*. Lanjut ke nomor 13.
13. Cetak hasil perhitungan nomor 12 dan lanjut ke nomor 14.
14. Selesai

### 3.3.2 *Packet loss*

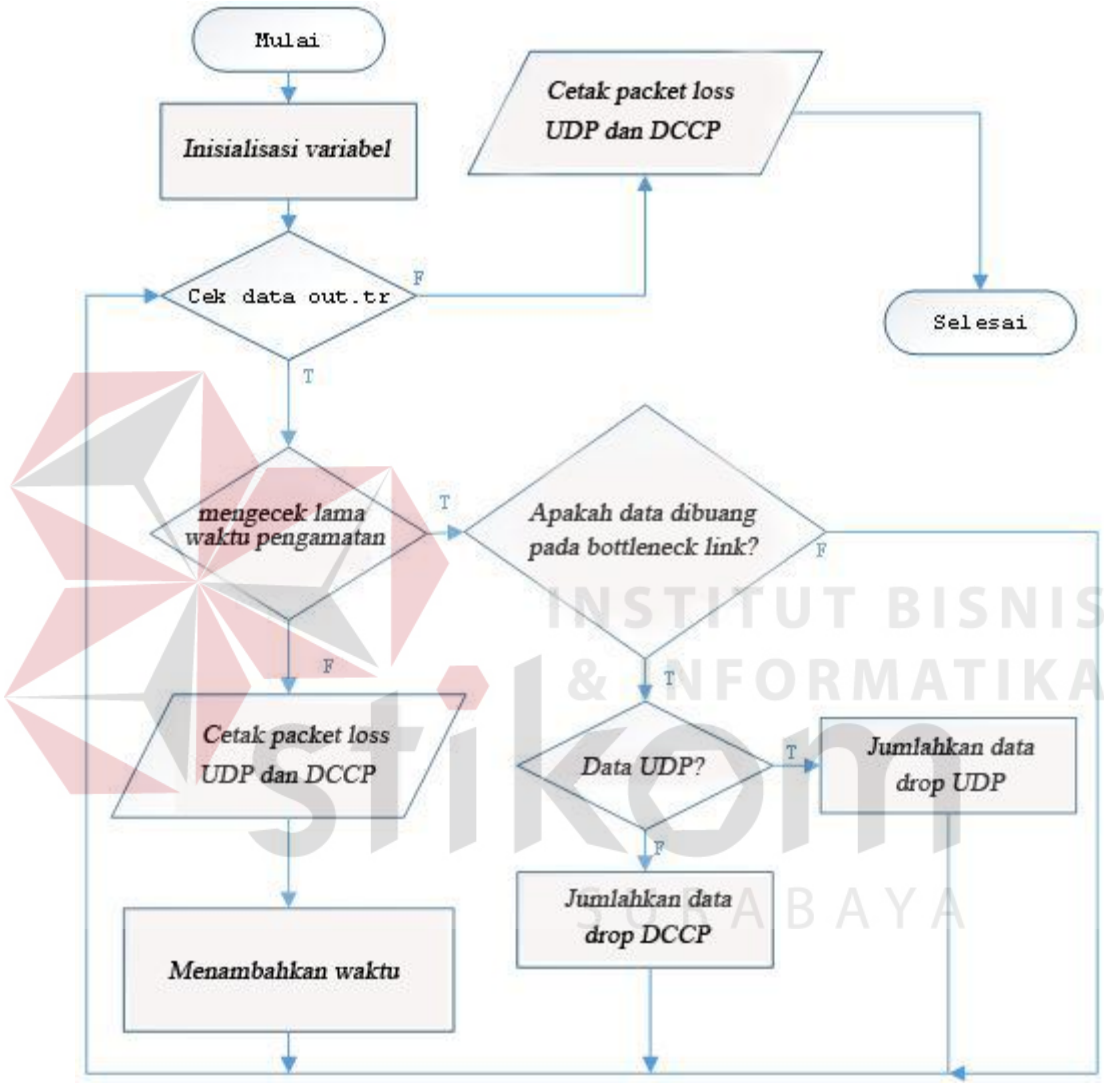
Flowchart *packet loss* ini berisi proses seleksi, perhitungan nilai dari parameter dan menampilkan output hasil olah data *trace file*. Gambar 3.6 merupakan flowchart keseluruhan dari skrip perl *packet loss*.

Sama seperti utilisasi *bandwidth*, granularity yang dimasukkan adalah satu sehingga program akan menampilkan hasil *packet loss* setiap detik. Hasil ini tidak bersifat kumulatif sehingga pembacaan paket hilang setiap waktu lebih mudah.

Berikut ini adalah penjelasan tentang cara kerja *flowchart packet loss* pada Gambar 3.6:

1. Mulai.
2. Inisialisasi variabel udp, dccp dan clock menjadi 0.
3. Pengecekan data. Jika masih ada data pada baris selanjutnya maka menuju nomor 4 jika tidak ada ke nomor 11.
4. Pengecekan kolom waktu yaitu x[1] dengan nilai granularitas yang dimasukkan. Jika bernilai benar lanjutkan ke nomor 5 dan jika salah ke nomor 9.
5. Pengecekan kolom kejadian yaitu x[0] (kolom ke-0), kolom sumber yaitu x[2] dan kolom tujuan yaitu x[3]. Jika ketiga kondisi sesuai maka kondisi

bernilai benar, lanjut ke nomor 6 dan jika ketiga kondisi atau salah satu kondisi tidak terpenuhi maka kondisi bernilai salah, kembali ke nomor 3.



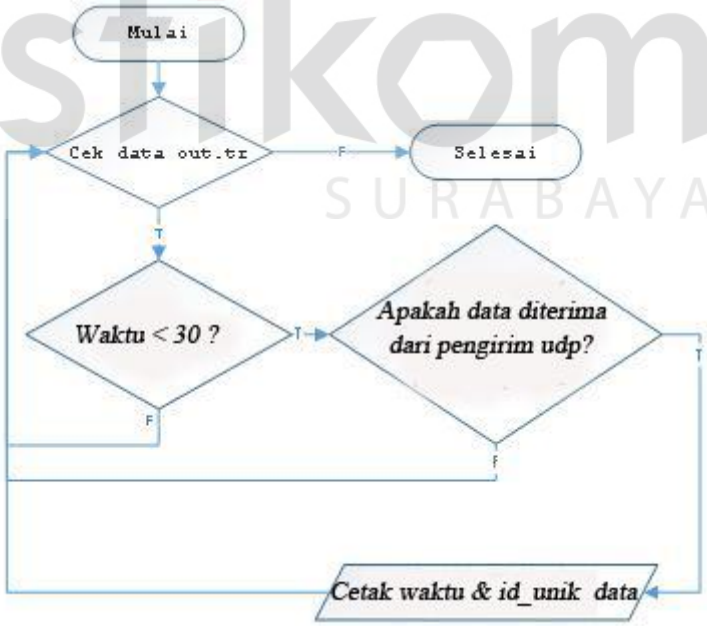

Gambar 3.6. Flowchart Program *Packet Loss*

6. Pengecekan kolom aplikasi yang digunakan yaitu x[4]. Jika kolom sama dengan ‘cbr’ (UDP) maka lanjut ke nomor 7 dan jika salah ke nomor 8.
7. Menjumlahkan data UDP yang hilang dan menyimpan hasilnya sementara dalam variabel udp. Kembali ke nomor 3.

8. Menjumlahkan data DCCP yang hilang dan menyimpan hasilnya sementara dalam variabel dccp. Kembali ke nomor 3.
9. Cetak hasil variabel udp dan dccp dan lanjut ke nomor 10
10. Variabel clock ditambah dengan granularitas dan disimpan pada variabel clock. Nilai variabel udp dan dccp dikembalikan menjadi 0. Selanjutnya menuju ke nomor 3.
11. Cetak hasil variabel udp dan dccp dan lanjut ke nomor 12.
12. Selesai

### 3.3.3 *Delay* dan *Jitter*

*Flowchart delay* ini berisi proses seleksi parameter dan menampilkan output hasil olah data *trace file*. Hasil output program ini dilanjutkan dengan bantuan aplikasi LibreOffice Calc. Gambar 3.7 merupakan *flowchart* seleksi paket kirim dari skrip perl *delay*.



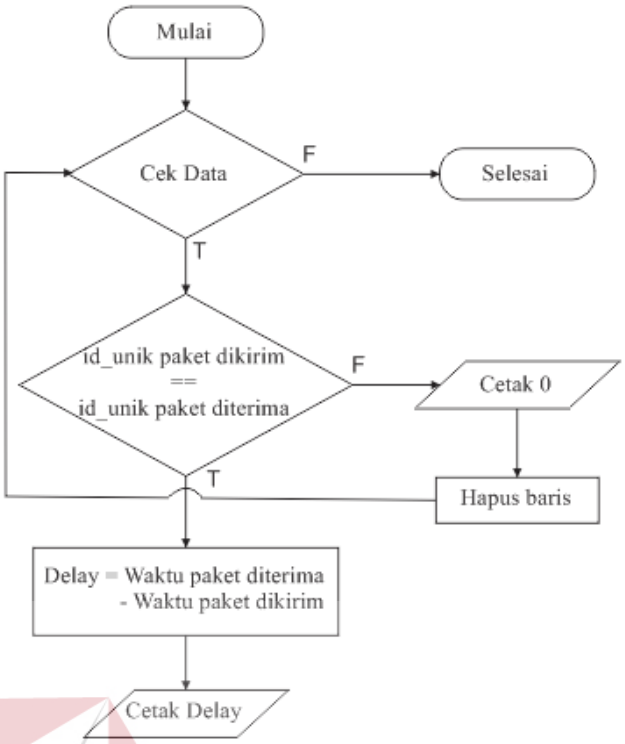

Gambar 3.7 *Flowchart* Seleksi Paket untuk *Delay*

Berikut ini adalah penjelasan tentang cara kerja *flowchart* seleksi paket pada Gambar 3.7:

1. Mulai.
2. Pengecekan data. Jika masih ada data pada baris selanjutnya maka menuju nomor 3 jika tidak ada ke nomor 6.
3. Pengecekan kolom waktu yaitu x[1]. Jika waktu bernilai kurang dari 30 maka bernilai benar lanjutkan ke nomor 4 dan jika salah ke nomor 2.
4. Pengecekan kolom kejadian yaitu x[0] (kolom ke-0), kolom sumber yaitu x[2] dan kolom aplikasi yaitu x[4]. Jika ketiga kondisi sesuai maka kondisi bernilai benar, lanjut ke nomor 5 dan jika ketiga kondisi atau salah satu kondisi tidak terpenuhi maka kondisi bernilai salah, kembali ke nomor 2.
5. Cetak kolom waktu yaitu x[5] dan id unik paket yaitu x[11]. Lanjut ke nomor 2.
6. Selesai.

Seleksi paket terima dilakukan sama seperti proses seleksi paket kirim. Hanya saja pada langkah ke-4 dalam proses seleksi kolom sumber diganti dengan kolom tujuan menjadi `$x[3]=$tonode`.

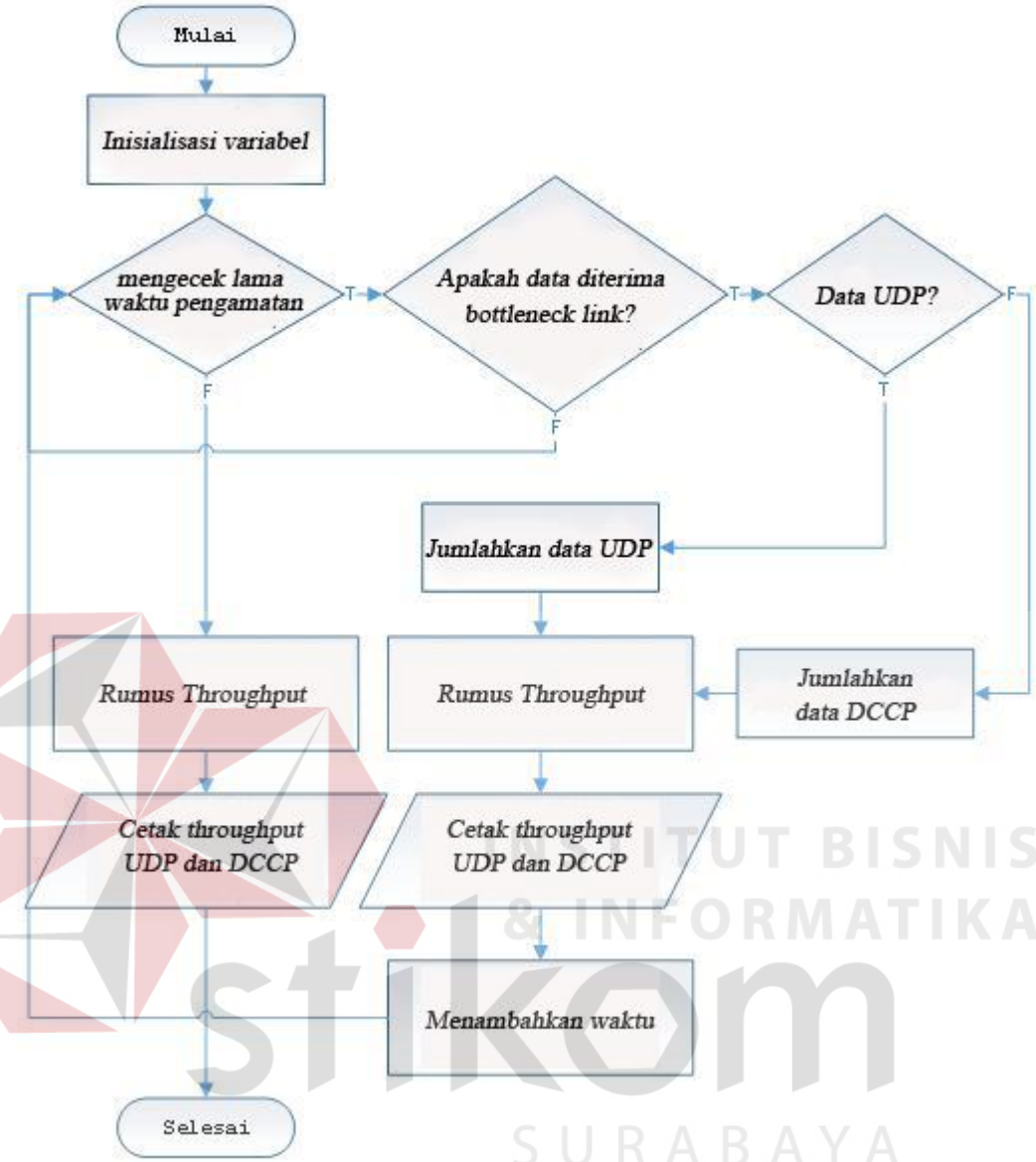
Langkah selanjutnya adalah mengolah paket dikirim dan diterima pada LibreOffice Calc dengan rumus *delay*. Gambar 3.8 merupakan *flowchart* perhitungan *delay* paket data. Kolom B[2] adalah ID unik dari paket data. Setelah data *delay* didapat, dilakukan perhitungan *jitter* dengan cara *delay* paket sekarang dikurangi dengan *delay* paket sebelumnya.




Gambar 3.8 *Flowchart* Perhitungan *Delay*

### 3.3.4 *Fairness*

*Fairness* dihitung dari nilai *throughput*. *Flowchart throughput* ini berisi proses seleksi, perhitungan nilai *throughput* dan menampilkan output hasil olah data *trace file*. Gambar 3.9 merupakan *flowchart* dari skrip perl *throughput*. Granularity yang dimasukkan adalah satu sehingga setiap detik *throughput* ditampilkan. Langkah selanjutnya adalah mengolah data *throughput* dengan rumus *fairness* menggunakan Libre office Calculator.




Gambar 3.9. *Flowchart Throughput* Untuk Parameter *Fairness*

Berikut ini adalah penjelasan tentang cara kerja *flowchart throughput* pada Gambar 3.9.

1. Mulai.
2. Inisialisasi variabel udp, dccp dan clock menjadi 0.
3. Pengecekan data. Jika masih ada data pada baris selanjutnya maka menuju nomor 4 jika tidak ada ke nomor 12.

4. Pengecekan kolom waktu yaitu x[1] dengan nilai granularitas yang dimasukkan. Jika bernilai benar lanjutkan ke nomor 5 dan jika salah ke nomor 9.
5. Pengecekan kolom kejadian yaitu x[0] (kolom ke-0), kolom sumber yaitu x[2] dan kolom tujuan yaitu x[3]. Jika ketiga kondisi sesuai maka kondisi bernilai benar, lanjut ke nomor 6 dan jika ketiga kondisi atau salah satu kondisi tidak terpenuhi maka kondisi bernilai salah, kembali ke nomor 3.
6. Pengecekan kolom aplikasi yang digunakan yaitu x[4]. Jika kolom sama dengan ‘cbr’ (UDP) maka lanjut ke nomor 7 dan jika salah ke nomor 8.
7. Menjumlahkan data UDP dan menyimpan hasilnya sementara dalam variabel udp. Kembali ke nomor 3.
8. Menjumlahkan data DCCP dan menyimpan hasilnya sementara dalam variabel dccp. Kembali ke nomor 3.
9. Perhitungan dengan menggunakan rumus *throughput*. Lanjut ke nomor 10.
10. Cetak hasil perhitungan nomor 9 dan lanjut ke nomor 11.
11. Variabel clock ditambah dengan granularitas dan disimpan pada variabel clock. Nilai variabel udp dan dccp dikembalikan menjadi 0. Selanjutnya menuju ke nomor 3.
12. Perhitungan dengan menggunakan rumus *throughput*. Lanjut ke nomor 13.
13. Cetak hasil perhitungan nomor 12 dan lanjut ke nomor 14.
14. Selesai