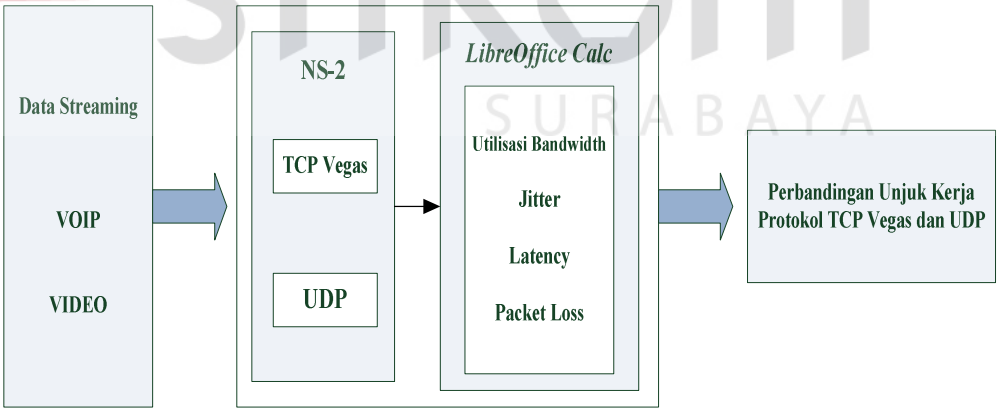
BAB III

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*.

3.1 Desain Penelitian

Analisis Perbandingan Unjuk Kerja Protokol TCP Vegas dan UDP dengan menggunakan data *streaming* ini dapat dijelaskan dengan lebih baik melalui blok diagram seperti yang terlihat pada Gambar 3.1:



Gambar 3.1. Blok Diagram

Pada Gambar 3.1 dapat dikelompokkan menjadi 3 bagian yaitu input data, proses dan output yang berupa hasil analisis perbandingan.

1. Input data

Data inputan yang digunakan adalah data *streaming* berupa VOIP dan Video. Pembanding kedua protokol didapat dengan melakukan generasi pembangkitan paket data pada NS-2 dengan ukuran *packet size, bandwidth* dan *bit rate* yang bervariasi.

2. Proses

Data inputan dijalankan diatas protokol TCP Vegas dan UDP dengan mengunakan pemograman TCL pada NS-2. Setelah didapatkan hasil *trace file*, hasil *trace file* diolah berdasarkan parameter uji *utilisasi bandwidth, jitter, latency dan packet loss* menggunakan *libreoffice calc.*

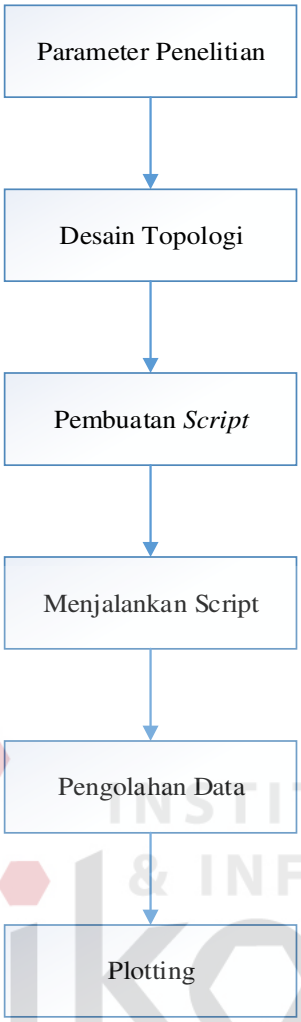
3. Output

Menunjukan analisis terhadap data yang dihasilkan berupa analisis perbandingan *utilisasi bandwidth,* analisis perbandingan *latency,* analisis perbandingan *packet loss*, analisis perbandingan *jitter,* dari protokol TCP Vegas dan UDP.

Analisis diatas disajikan dalam bentuk pembahasan berdasarkan studi literatur dan percobaan yang telah dilakukan pada bagian proses yang nantinya dapat dipaparkan melalui tampilan grafik.

3.2 Prosedur Penelitian

Pada Prosedur penelitian ini menjelaskan tentang langkah-langkah yang dilakukan dalam membuat analisis perbandingan unjuk kerja TCP Vegas dan UDP dengan menggunakan data *streaming,* seperti diagram alir pada Gambar 3.2:



Gambar 3.2. Skema Prosedur Penelitian

3.2.1 Parameter Penelitian

Dalam tahap ini dilakukan pengumpulan data yang akan digunakan untuk melakukan percobaan. Ada beberapa data yang dibutuhkan meliputi data *streaming* yang akan digunakan berupa VOIP dan Video *Streaming*. Protokol yang akan digunakan adalah TCP Vegas dan UDP. Waktu percobaan yang akan dipakai selama 20 detik. Penentuan nilai paket size, *bandwidth*, *delay* dan *bit rate* yang bervariasi. Nilai *Bandwidth*, *delay* dan *bit rate* untuk memberikan efek

dalam membandingkan protokol berdasarkan kondisi-kondisi yang telah ditentukan.

Kateristik antrian yang digunakan yaitu drop tail dimana data terakhir yang datang akan dibuang apabila kapasitas dari memori telah penuh. Karena dianggap memiliki memori (buffer) maka *perl*u mendefinisikan kapasitas antrian sebesar 15.

Parameter pembanding yang digunakan untuk analisis masing-masing protokol yaitu *utilisasi bandw*id*th, latency, packet loss* dan *jitter*.

1. Analisis perbandingan *utilisasi bandwidth*.

Analisis perbandingan *Utilisasi bandw*id*th* pada TCP Vegas dan UDP merupakan hasil akhir yang ingin dicapai dalam penelitian ini. Utilisasi adalah ukuran seberapa sibuk sumber daya yang ada. *Utilisasi bandwidth* dianalisis berdasarkan seberapa besar prosentase *bandwidth* suatu *link* yang menghubungkan antara kedua sisi yaitu sisi pelanggan dan *prov*id*er*. *Utilisasi bandwidth* dapat dihitung dengan persamaan 2.1.

2. Analisis perbandingan *packet loss*.

Analisis perbandingan *packet loss* pada TCP Vegas dan UDP merupakan hasil akhir yang ingin dicapai dalam penelitian ini. *Packet loss* dianalisis berdasarkan berapa banyak paket yang hilang atau gagal mencapai tujuan pada waktu paket sedang berjalan.

Nilai dari *Packet loss* dapat dihitung dengan persamaan 2.2. Menurut versi *TIPHON (Telecommunications and Internet Protocol Harmonization Over Networks) packet loss* dikategorikan seperti pada Tabel 3.1:

Tabel 3.1. Kategori *Packet loss*

Kategori	*Packet loss*
Sangat bagus	0%
Bagus	3%
Sedang	15%
Jelek	25%

3. Analisis perbandingan *latency.*

Analisis perbandingan *lateny* pada TCP Vegas dan UDP merupakan hasil akhir yang ingin dicapai dalam penelitian ini. *Latency* dianalisis berdasarkan berapa waktu tunda dari paket yang diterima sampai tujuan dari masing-masing protokol yang dibandingkan dengan data *streaming.*

Nilai dari *latency* dapat dihitung dengan persamaan 2.3. Menurut versi *TIPHON latency* dikategorikan seperti pada Tabel 3.2:

Tabel 3.2. Kategori *Latency*

Kategori	Latency
Sangat bagus	< 150 ms
Bagus	150 s/d 300 ms
Sedang	300 s/d 450 ms
Jelek	> 450 ms

4. Analisis perbandingan *jitter.*

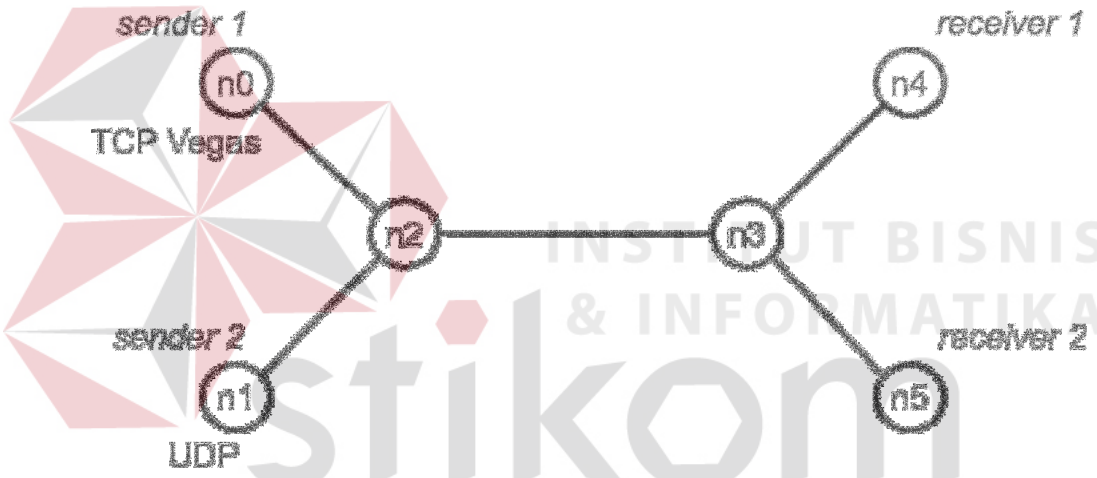
Analisis perbandingan *Jitter* pada TCP Vegas dan UDP merupakan hasil akhir yang ingin dicapai dalam penelitian ini. *Jitter* merupakan variansi dari *delay, jitter* dianalisis berdasarkan keterlambatan transmisi data dari pengirim dan penerima dalam rentang waktu tertentu.

Nilai dari *jitter* dapat dihitung dengan persamaan 2.4. Menurut versi *TIPHON jitter* dikategorikan seperti pada Tabel 3.3:

Tabel 3.3. Kategori *Jitter*

Kategori	Jitter
Sangat bagus	0 ms
Bagus	0 s/d 75 ms
Sedang	76 s/d 125 ms
Jelek	125 s/d 225 ms

3.2.2 Desain Topologi



Gambar 3.3. Topologi

Keterangan topologi pada Gambar 3.3 sebagai berikut:

1. Node yang akan di gunakan yaitu n0,n1,n2,n3,n4,n5.
2. Node n0-n2, n1-n2,n3-n4 dan n3-n5 terjadi hubungan *duplex-link.*
3. Node n2-n3 dan n3-n2 terjadi hubungan *simplex-link.*
4. Node n0 dialiri dengan data TCP Vegas menuju n4
5. Node n1 dialiri data UDP menuju n5

Model topologi yang digunakan pada Gambar 3.3 adalah *dumb-bell topology*. Penggunaan model ini dikarenakan *dumb-bell topology* mempunyai *single bottleneck link* dengan jumlah dua *sender* dan dua *receiver*. Tetapi dengan algoritma *bandwidth*, *bit rate* dan *packet size* yang berbeda. Topologi yang digunakan dalam percobaan menggunakan bersifat *unicast* dimana proses pengiriman dilakukan satu sumber ke satu tujuan. Pada gambar diatas Node n0 dan n1 adalah *sender*, sedangkan node n4 dan n5 merupakan *receiver*. Untuk *bottleneck link* terdapat pada n2 dan n3.

3.2.3 Pembuatan *Script*

Pembuatan *script* adalah pembuatan *script* .tcl yang disesuaikan dengan data yang sudah ditentukan. Ada 3 langkah dalam pembuatan *script* yaitu inisialisasi, pembuatan node dan link, penggabungan protokol TCP vegas dengan UDP, dan mengatur jalannya *script*.

1. Inisialisasi

Simulasi dengan NS 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_voip.tr w]
$ns trace-all $tracefile1
```

```
#Buka NAM trace files
set namfile [open out_voip.nam w]
$ns namtrace-all $namfile
```

Skrip di atas akan membuat *file* out_voip.tr yang akan digunakan untuk menyimpan data hasil simulasi dan *file* out_voip.nam untuk menyimpan data hasil visualisasi. Deklarasi 'w' pada bagian akhir dari perintah open adalah perintah tulis.

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.

2. Membuat node dan link

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

```
set n0 [$ns node]
```

Selanjunya untuk menggunakan node n0 dilakukan dengan cara memanggil *variabel* $n0. Demikian pula node-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 $n2 dengan *bidirectional link* berkapasitas 2Mb dan dengan waktu tunda akibat propagasi sebesar 10ms dapat dibuat dengan cara seperti di bawah ini, begitu pula pada *link* antar node lainnya.

```
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
```

Apabila diinginkan sebuah link satu arah maka dapat digunakan perintah "*simplex-link*" untuk menggantikan "*duplex-link*"

Pada NS, antrian keluaran dari sebuah node didefinisikan sebagai bagian dari sebuah *link*. Karena itu kita *perl*u mendefinisikan karakteristik dari *queue* ini apabila terjadi lebihan data yang datang pada sebuah node. Opsi "DropTail" berarti bahwa data terakhir yang datang akan dibuang apabila kapasitas dari memori telah penuh. Karena *link* dianggap memiliki memori (buffer), maka kita perlu mendefinisikan kapasitas antrian dari *link* sebagai berikut:

```
$ns queue-limit $n2 $n3 15
```

3. Menggabungkan Aplikasi

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

```
set ftp [new Application/FTP]
set tcp [new Agent/TCP/Vegas]
```

```
set cbr [new Application/Traffic/CBR]
Set udp [new Agent/UDP]
```

Selanjutnya untuk menggabungkan protokol ini pada sebuah node dari sumber yaitu n0 ke tujuan yaitu n4 dan dari n1 ke n5, menggunakan perintah di bawah ini:

```
$ns attach-agent $n0 $tcp
$ns attach-agent $n4 $tcp
$ns attach-agent $n1 $udp
$ns attach-agent $n5 $udp
```

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

```
$ftp attach-agent $tcp
$cbr 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 FTP pada n0 dan CBR pada n1 mulai mengirimkan data dan kapan selesai mengirimkan data digunakan perintah berikut:

```
$ns at 0.1 "$cbr start"
$ns at 0.1 "$ftp start"
$ns at 19.0 "$cbr stop"
$ns at 19.0 "$ftp stop"
```

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

```
$ns at 20.0 "finish"
```

Dengan menggunakan perintah tersebut, aplikasi TCP dan UDP akan mulai berjalan pada detik ke 0.1 dan berhenti pada detik 19.0. Pada detik ke 20 aplikasi selesai. Selanjutnya untuk memulai simulasi atau menjalankan program dapat dilakukan dengan menggunakan perintah:

```
$ns run
```

3.2.4 Menjalankan *Script* di NS-2

Setelah pembuatan *script* .tcl selesai *script* dijalankan diatas aplikasi *network simulator 2*. Dengan cara mengetik *ns voip.tcl* pada terminal. Untuk pengujian *script*, apabila hasil yang ditampilkan berupa gambar visualisasi nam. Maka *script* yang dibuat sudah benar dan sesuai dengan konfigurasi pada *script* .tcl.

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

3.2.5 Pengolahan Data

Setelah *Script* dijalankan didapat file out_voip.tr yang berisi *trace file* dari masing-masing parameter. Gambar 3.4 merupakan contoh *trace file*.

```
+ 0.1 1 2 cbr 160 ------- 2 1.0 5.0 0 0
- 0.1 1 2 cbr 160 ------- 2 1.0 5.0 0 0
+ 0.1 0 2 tcp 160 ------- 1 0.0 4.0 0 1
- 0.1 0 2 tcp 160 ------- 1 0.0 4.0 0 1
r 0.11064 1 2 cbr 160 ------- 2 1.0 5.0 0 0
+ 0.11064 2 3 cbr 160 ------- 2 1.0 5.0 0 0
- 0.11064 2 3 cbr 160 ------- 2 1.0 5.0 0 0
r 0.11064 0 2 tcp 160 ------- 1 0.0 4.0 0 1
+ 0.11064 2 3 tcp 160 ------- 1 0.0 4.0 0 1
+ 0.12 1 2 cbr 160 ------- 2 1.0 5.0 1 2
- 0.12 1 2 cbr 160 ------- 2 1.0 5.0 1 2
- 0.128926 2 3 tcp 160 ------- 1 0.0 4.0 0 1
r 0.13064 1 2 cbr 160 ------- 2 1.0 5.0 1 2
```

Gambar 3.4. *trace file*

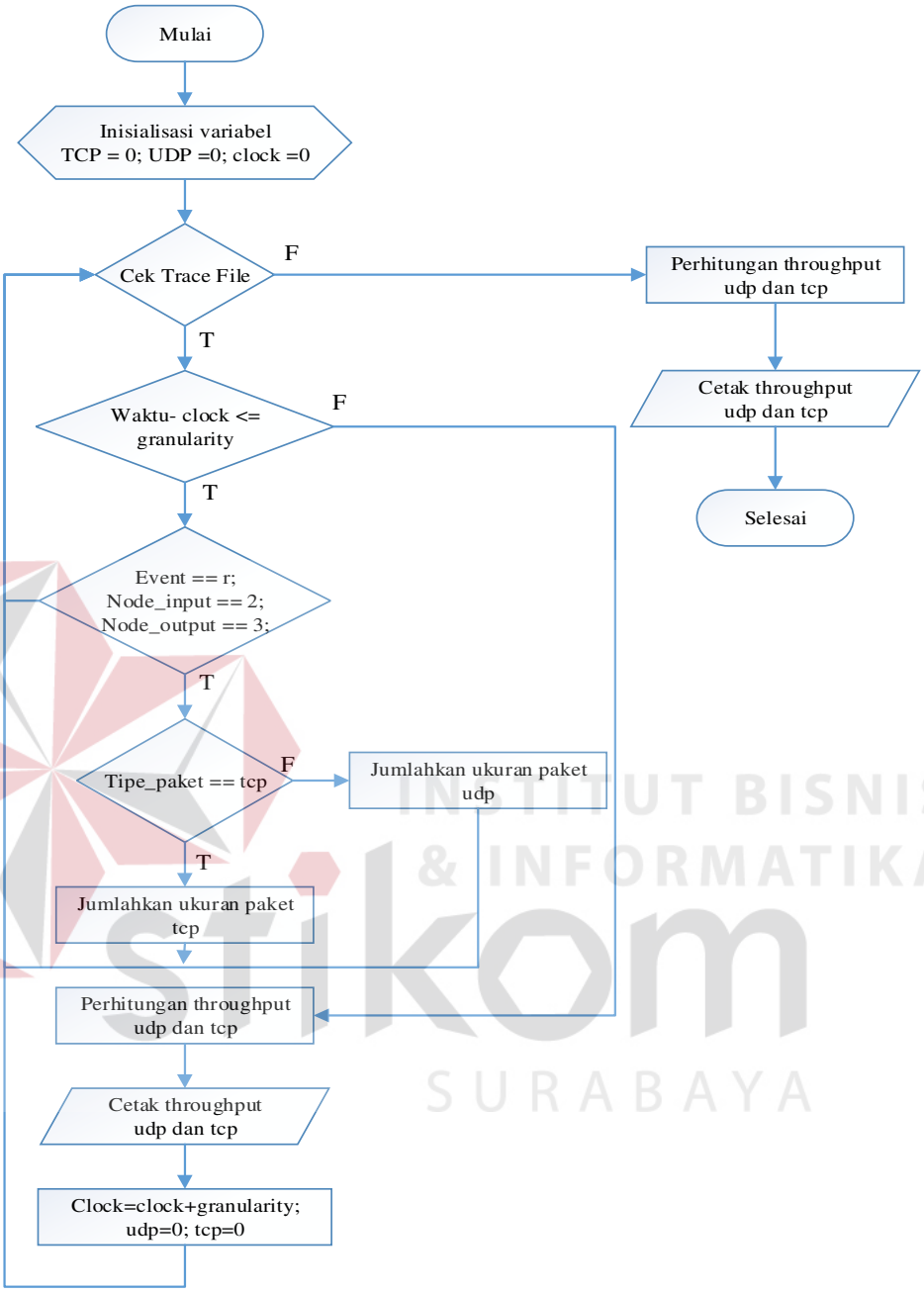
Setelah didapatkan *trace file* (yaitu out.tr) pada Gambar 3.4, selanjutnya dilakukan pengolahan data dengan menggunakan pemrograman *perl*. Untuk memanggil pemograman perl dapat didefinisikan dengan perintah

```
perl <nama file>.pl <trace file> <required node> <granularity> > output file
```

Selanjutnya dilakukan pengolahan data dengan *LibreOffice calc* untuk mendapatkan hasil yang diinginkan. Pengolahan data dengan *perl* dan *LibreOffice calc* dapat dijelaskan dalam bentuk *flowchart* sebagai berikut ini:

1. *Utilisasi bandwidth*

Untuk mendapatkan nilai *utilisasi bandwidth*, yang dicari lebih dahulu adalah nilai *throughput* dengan pemograman *perl*. Gambar 3.5 merupakan *flowchart* untuk menghasilkan nilai *throughput* pada video dan VOIP.



Gambar 3.5. *Flowchart Throughput*

Penjelasan *flowchart* pada Gambar 3.5:

a. Mulai

b. Inisialisasi *variabel* TCP, UDP dan clock.

c. Pengecekan data *trace file,* dimana *trace file* berisi data keluaran dari hasil simulasi. Jika masih ada data pada baris, maka proses dilanjutkan pada proses berikutnya. Jika tidak ada dilakukan perhitungan *throughput*. Dan hasilnya dicetak.

d. Jika waktu yang dimasukkan kurang dari sama dengan *granularity* maka dilanjutkan pada proses selanjutnya. Jika tidak sama dengan *granularity* maka dilanjutkan ke proses h.

e. Pengecekan kolom *event* yaitu x[0] (kolom ke-0) dengan status "r", x[2] (kolom ke-2) adalah node sumber, dan x[3] (kolom ke-3) adalah node tujuan. Jika ketiga kondisi terpenuhi maka dilanjutkan ke proses berikutnya, jika tidak kembali ke proses c.

f. Jika kolom ke-4 sama dengan TCP maka dilanjutkan ke proses berikutnya jika bukan TCP maka dijumlahkan dan hasilnya disimpan dalam *variabel* UDP. Dan proses kembali ke c.

g. Menjumlahkan data TCP dan menyimpan hasilnya kedalam *variabel* TCP. Dilanjutkan dengan kembali ke c.

h. Dilakukan perhitungan *throughput*.

i. Dicetak hasil perhitungan.

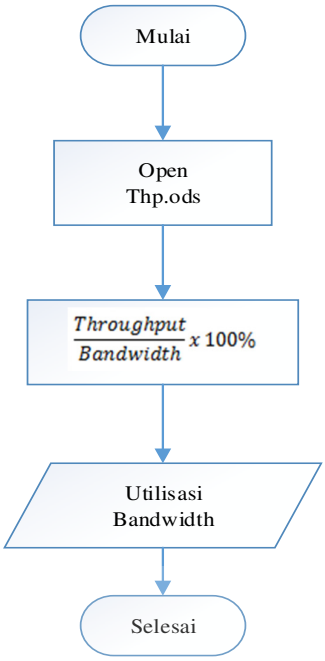
j. Proses dilanjukan dengan menambah *variabel clock* dengan nilai *granularity* dan disimpan pada *variabel clock*. Selanjutnya proses dikembalikan ke c.

Setelah program *throughput* dijalankan, didapatkan nilai dari *throughput* seperti Tabel 3.4.

Tabel 3.4. Hasil *Throughput bandwidth* 70Kb

Waktu	Throughput	
	TCP Vegas	UDP
1,001783	1280	5920
2,0016	2080	6560
3,005566	160	8640
4,00064	640	8000
5,006354	960	7680
6,004423	800	7840
7,004354	800	8000
8,005211	960	7680
9,00328	800	7840
10,000583	800	8000
11,004069	800	7840
12,002137	800	7840
13,002069	800	8000
14,002926	800	7840
15,000994	800	7840
16,000926	480	8320
17,001783	960	7680
18,007497	640	8000
19,005566	800	7840
19,848114	640	2560

Setelah didapatkan nilai *throughput* selanjutnya dilakukan perhitungan menggunakan *LibreOffice calc* untuk mendapatkan nilai dari *utilisasi bandwidth.* Gambar 3.6 merupakan *flowchart* dari *utilisasi bandwidth.*



Gambar 3.6. *Flowchart Utilisasi bandwidth*

Penjelasan Gambar 3.6 *flowchart utilisasi bandwidth*:

a. Mulai

b. Membuka file thp.ods. thp berisi file pemrosesan *throughput* dengan pemograman *perl*.

c. *Utilisasi bandwidth* dihitung dengan menggunakan persamaan 2.1. Sebagai contoh perhitungan *utilisasi bandwidth* pada detik pertama:

Bandwidth 70 Kb $= \frac{\mathbf{70 \times 1000}}{\mathbf{8}} = 8750$ B

Utilisasi Bandwidth $= \frac{\boldsymbol{throughput}}{\boldsymbol{bandwidth}}$ x100%

$= \frac{\mathbf{1280\ B}}{\mathbf{8750\ B}}$ x 100%

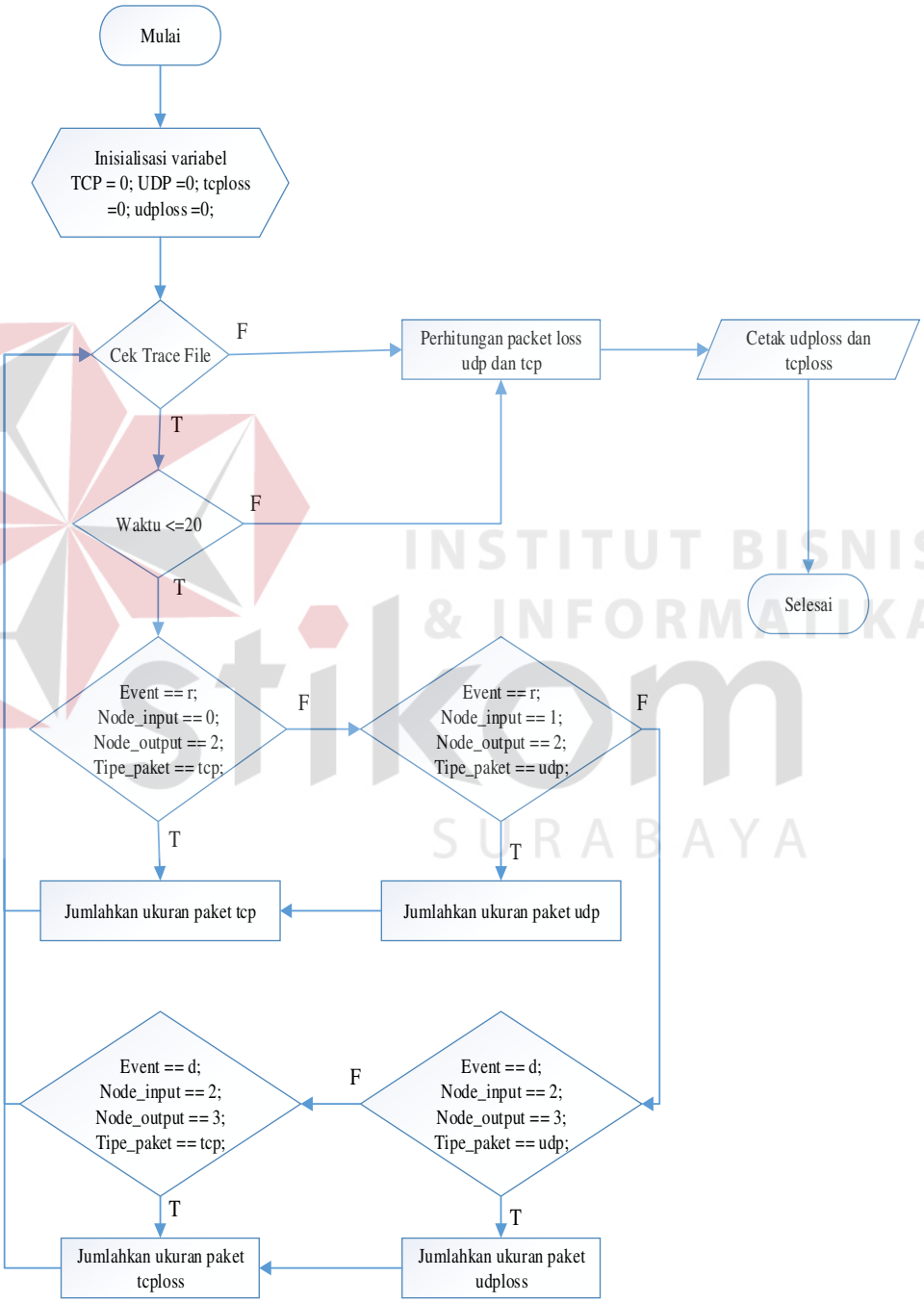
= 14,63 %

Perhitungan dilakukan sampai dengan detik ke-20 setelah itu di rata-rata untuk mendapatkan nilai utilisasi pada setiap percobaan.

d. Didapatkan nilai *utilisasi bandwidth*.

e. Selesai

2. *Packet loss*



Gambar 3.7. *Flowchart Packet loss*

Gambar 3.7 merupakan *flowchart* keseluruhan untuk mendapatkan nilai *packet loss* dengan mengunakan pemrograman *perl*. Program ini digunakan untuk mencari nilai *packet loss* pada video dan voip.

Penjelasan Gambar 3.7 *flowchart packet loss*:

a. Mulai
b. Inisialisasi *variabel* TCP, UDP, udploss, dan tcploss
c. Pengecekan data *trace file* dimana *trace file* berisi data keluaran dari hasil simulasi. Jika masih ada data pada baris, maka proses dilanjutkan pada proses berikutnya. Jika tidak ada dilakukan perhitungan *packet loss*. Dan hasilnya dicetak.
d. Jika waktu yang dimasukkan kurang dari sama dengan 20 maka dilanjutkan pada proses selanjutnya. Jika lebih dari 20 maka dilakukan perhitungan *packet loss*. Dan hasilnya dicetak.
e. Pengecekan kolom *event* yaitu x[0] (kolom ke-0) dengan status "r", x[2] (kolom ke-2) adalah node sumber, x[3] (kolom ke-3) adalah node tujuan. x[4] (kolom ke-4) jenis protokol adalah TCP. Jika kondisi terpenuhi maka data TCP dijumlahkan dan disimpan hasilnya kedalam *variabel* sum1. Setelah itu dilanjutkan dengan kembali ke c.
f. Jika bukan TCP maka dilakukan pengecekan kolom *event* yaitu x[0] (kolom ke-0) dengan status "r", x[2] (kolom ke-2) adalah node sumber, x[3] (kolom ke-3) adalah node tujuan. x[4] (kolom ke-4) jenis protokol adalah UDP, jika kondisi terpenuhi maka dijumlahkan dan hasilnya disimpan dalam *variabel* sum2. Dan proses kembali ke c.

g. Jika kondisi diatas tidak terpenuhi maka dilakukan pengecekan kolom *event* yaitu x[0] (kolom ke-0) dengan status "d", x[2] (kolom ke-2) adalah node sumber, x[3] (kolom ke-3) adalah node tujuan. x[4] (kolom ke-4) jenis protokol adalah UDP, jika kondisi terpenuhi maka dijumlahkan dan hasilnya disimpan dalam *variabel* drop2.

h. Jika kondisi diatas tidak terpenuhi maka dilakukan pengecekan kolom *event* yaitu x[0] (kolom ke-0) dengan status "d", x[2] (kolom ke-2) adalah node sumber, x[3] (kolom ke-3) adalah node tujuan. x[4] (kolom ke-4) jenis protokol adalah TCP, jika kondisi terpenuhi maka dijumlahkan dan hasilnya disimpan dalam *variabel* drop1. Dan proses dikembalikan ke langkah c.

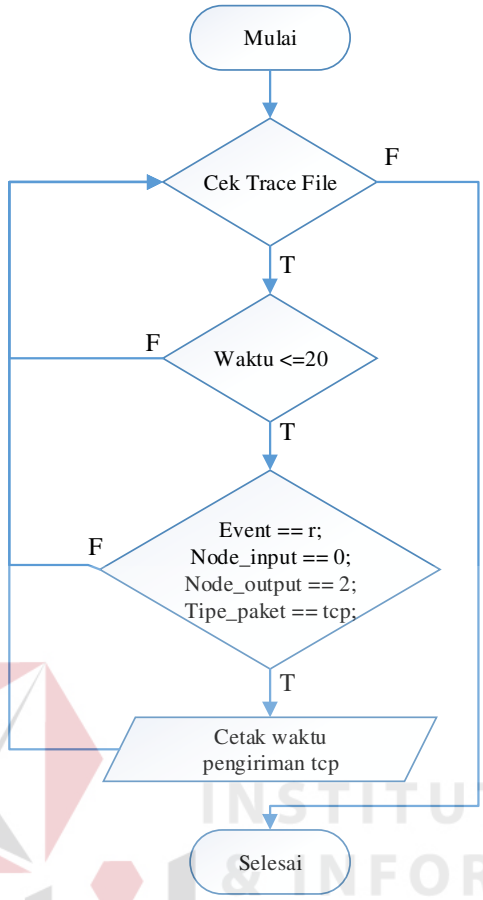
Setelah program *perl packet loss* dijalankan, didapatkan nilai dari *packet loss* seperti pada Tabel 3.5:

Tabel 3.5. Hasil *Packet Loss bandwidth* 70 Kb

	TCP Vegas	UDP
Packet Loss	6,25%	0,11%

3. *Latency*

Untuk mendapatkan nilai *latency*, yang dicari lebih dahulu adalah nilai paket terkirim dan paket diterima. Gambar 3.8 merupakan *flowchart* paket terkirim sedangkan Gambar 3.9 merupakan *flowchart* paket diterima pada pemrograman *perl*.



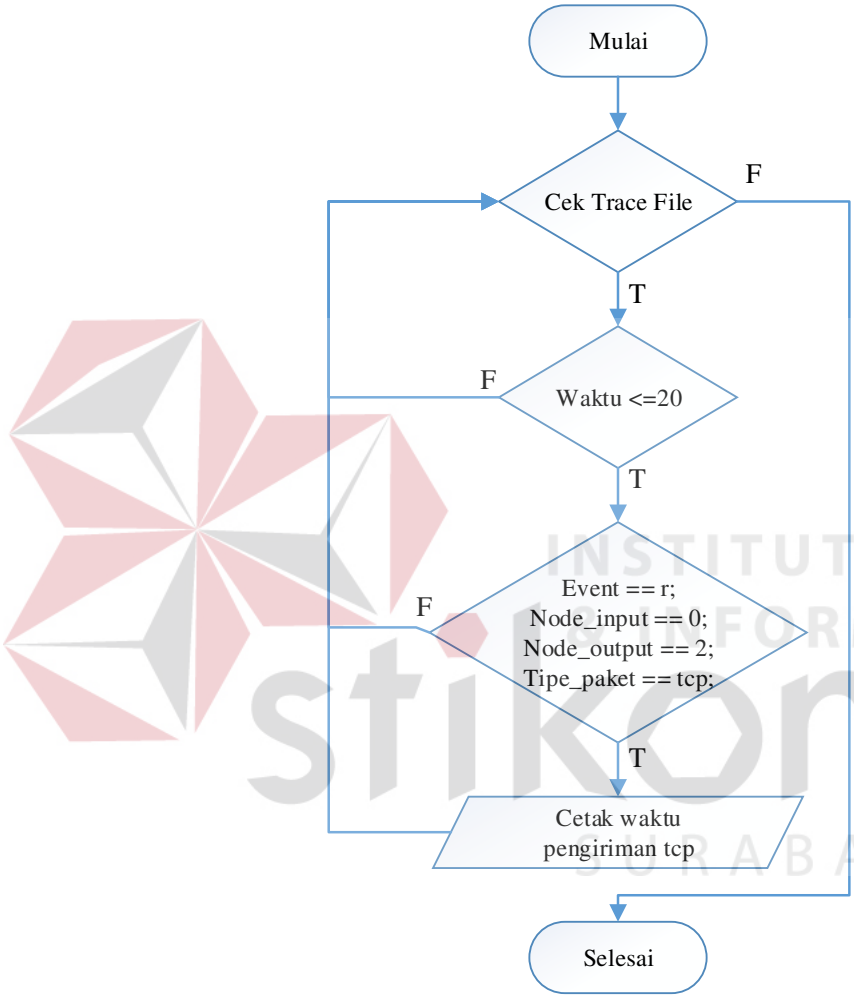
Gambar 3.8. *Flowchart* paket terkirim

Penjelasan Gambar 3.8 *flowchart* paket terkirim

a. Mulai
b. Pengecekan data out.tr dimana out.tr berisi data keluaran dari hasil simulasi. Jika masih ada data pada baris, maka proses dilanjutkan pada proses berikutnya. Jika tidak ada proses selesai.
c. Jika waktu kurang dari sama dengan 20 maka dilanjutkan pada proses selanjutnya. Jika tidak lebih dari 20 maka kembali pada proses b.
d. Pengecekan kolom *event* yaitu x[0] (kolom ke-0) dengan status ”r”, x[2] (kolom ke-2) adalah node sumber, dan x[4] (kolom ke-4) adalah jenis

protokol. Jika ketiga kondisi terpenuhi maka dilanjutkan ke proses berikutnya, jika tidak kembali ke proses b.

e. Mencetak waktu dan id unik. Proses dilanjutkan ke langkah b.



Gambar 3.9. *Flowchart* paket diterima

Penjelasan Gambar 3.9 *flowchart* paket diterima:

a. Mulai

b. Pengecekan data out.tr dimana out.tr berisi data keluaran dari hasil simulasi. Jika masih ada data pada baris, maka proses dilanjutkan pada proses berikutnya. Jika tidak ada proses selesai.

c. Jika waktu kurang dari sama dengan 20 maka dilanjutkan pada proses selanjutnya. Jika tidak lebih dari 20 maka kembali pada proses b.

d. Pengecekan kolom *event* yaitu x[0] (kolom ke-0) dengan status "r", x[3] (kolom ke-3) adalah node tujuan, dan x[4] (kolom ke-4) adalah jenis protokol. Jika ketiga kondisi terpenuhi maka dilanjutkan ke proses berikutnya, jika tidak kembali ke proses b.

e. Mencetak waktu dan id unik. Proses dilanjutkan ke langkah b.

Setelah program dijalankan didapatkan hasil seperti Tabel 3.6 dan Tabel 3.7. Tabel 3.6 dan Tabel 3.7 merupaka sebagian data yang ada.

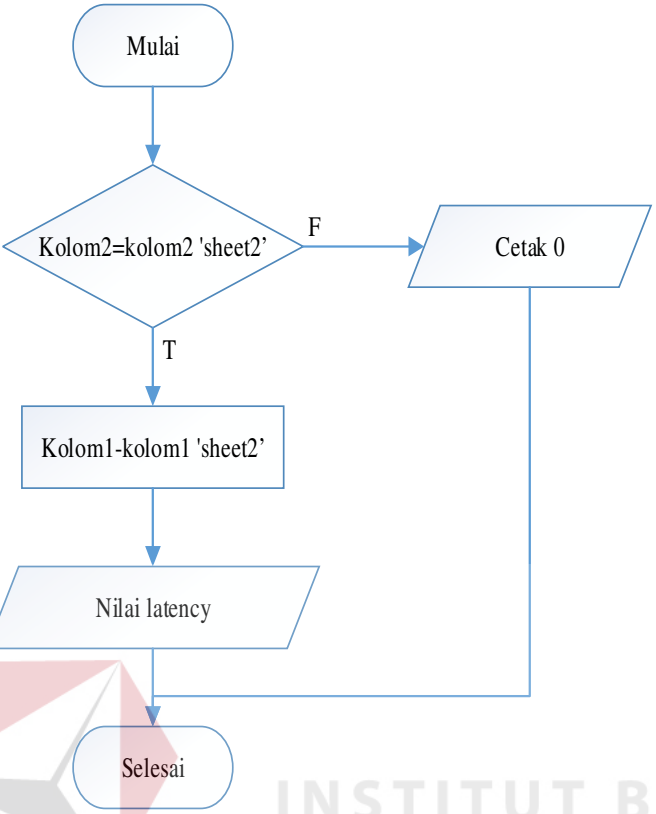
Tabel 3.6. Hasil Paket Diterima

Event	Waktu	From node	To node	Pkt Type	Packet Size	Packet Id
r	0,207851	3	4	tcp	160	1
r	0,390709	3	4	tcp	160	12
r	0,573566	3	4	tcp	160	23
r	0,591851	3	4	tcp	160	24
r	0,610137	3	4	tcp	160	25
r	0,792994	3	4	tcp	160	38
r	0,829566	3	4	tcp	160	40
r	0,866137	3	4	tcp	160	42
r	1,048994	3	4	tcp	160	55
r	1,06728	3	4	tcp	160	56
r	1,122137	3	4	tcp	160	59
r	1,140423	3	4	tcp	160	60
r	1,19528	3	4	tcp	160	63
r	1,213566	3	4	tcp	160	64
r	1,396423	3	4	tcp	160	77
r	1,432994	3	4	tcp	160	80
r	1,487851	3	4	tcp	160	84
r	1,524423	3	4	tcp	160	87

Tabel 3.7. Hasil Paket Terkirim

Event	Waktu	From node	To node	Pkt Type	Pkt Size	Pkt Id
r	0,11064	0	2	tcp	160	1
r	0,293383	0	2	tcp	160	12
r	0,47624	0	2	tcp	160	23
r	0,47688	0	2	tcp	160	24
r	0,47752	0	2	tcp	160	25
r	0,659097	0	2	tcp	160	38
r	0,677383	0	2	tcp	160	40
r	0,695669	0	2	tcp	160	42
r	0,878526	0	2	tcp	160	55
r	0,879166	0	2	tcp	160	56
r	0,915097	0	2	tcp	160	59
r	0,915737	0	2	tcp	160	60
r	0,951669	0	2	tcp	160	63
r	0,952309	0	2	tcp	160	64
r	1,134526	0	2	tcp	160	77
r	1,152811	0	2	tcp	160	80
r	1,207669	0	2	tcp	160	84
r	1,225954	0	2	tcp	160	87
r	1,280811	0	2	tcp	160	91
r	1,518526	0	2	tcp	160	108

Setelah didapatkan waktu terkirim dan waktu diterima selanjutnya dilakukan perhitungan menggunakan *LibreOffice calc* untuk mendapatkan nilai *latency*. Perhitungan ini digunakan untuk mendapatkan nilai dari latency. Gambar 3.11 merupakan *flowchart* dari *latency*.



Gambar 3.10. *Flowchart* Latency

Penjelasan Gambar 3.10 *flowchart latency*:

a. Mulai
b. Jika nilai kolom2 pada sheet 1 sama dengan kolom2 pada sheet 2 maka dilanjutkan pada proses berikutnya, kolom2 adalan ID unik dari paket data. Tetapi jika nilai kolom 2 tidak sama dengan kolom2 pada sheet2 maka hasil dicetak 0 dan proses selesai .
c. Kolom1 pada sheet 1 – kolom1 pada sheet 2. Kolom1 pada sheet 1 adalah waktu dimana paket diterima. Sedangkan kolom1 pada sheet 2 adalah waktu dimana paket dikirim. Sebagai contoh perhitungan latency pada detik ke-0 seperti pada Gambar 3.11. Perhitungan ini dilakukan sampai pada detik ke-20

Paket diterima 'sheet1'		Paket dikirim 'sheet2'	
Waktu	Id unik	Waktu	Id unik
0,207851	1	0,11064	1

latency = Tr-Ts

= 0,207851 -0,11064

=0,097211 s

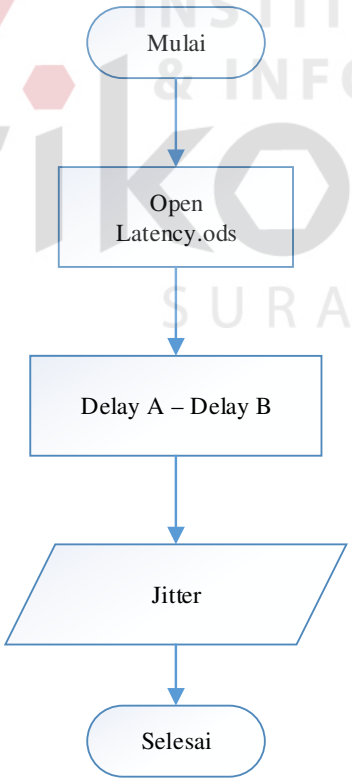
Gambar 3.11. Perhitungan *Latency*

d. Didapatkan nilai *latency*.

e. Selesai.

4. Jitter

Setelah didapatkan nilai *latency*. Selanjutnya dilakukan perhitungan nilai *jitter*. Gambar 3.12 merupakan *flowchart* dari *jitter*.



Gambar 3.12. Flowchart *Jitter*

Pada Gambar 3.12 perhitungan nilai jitter didapatkan dengan cara nilai *latency* dari paket sekarang dikurangi dengan *latency* paket sebelumnya (persamaan 2.4).

3.2.6 Plotting

Setelah mendapatkan nilai *utilisasi bandwidth, jitter, latency dan packet loss,* selanjutnya adalah menggambarkan kedalam grafik dengan menggunakan *LibreOffice calc*. *LibreOffice calc* digunakan untuk memudahkan dalam melakukan perbandingan.