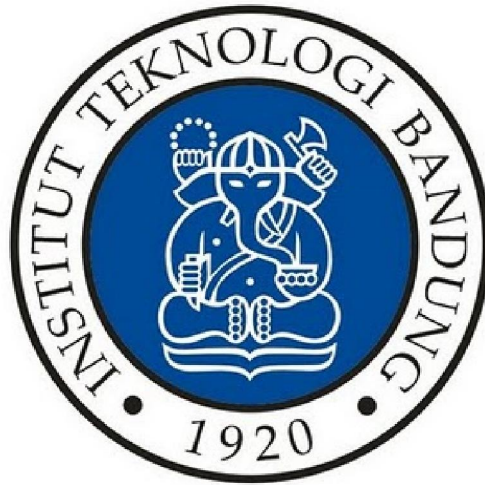


**STRESS TESTING TO THE NETWORK TOPOLOGY
USING NS2
MODELING AND SIMULATION OF NETWORK**



**OLEH : SYAIFUL AHDAN
NIM : 23215032**

**PROGRAM STUDI TEKNIK ELEKTRO
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2015**

KATA PENGANTAR

Dengan mengucapkan rasa syukur kehadiran Allah SWT atas rahmat beserta hidayah-Nya sehingga kami dapat menyusun makalah yang berjudul “STRESS TESTING TO THE NETWORK TOPOLOGY USING NS2 MODELING AND SIMULATION OF NETWORK” guna memenuhi syarat tugas akhir mata kuliah Pemodelan dan Simulasi Jaringan .

Diharapkan dengan adanya makalah ini dapat menambah pengetahuan tentang Bagaimana Melakukan Desain Jaringan berdasarkan berapa jenis topologi yang ada dan bagaimana menggunakan tools network simulator dengan NS2, penulis menyadari bahwa dalam penyusunan makalah ini masih terdapat banyak sekali kekurangan. Karena itu kami selaku penulis mengharapkan kritik dan saran yang membangun dari pembaca agar makalah ini menjadi lebih baik lagi. Semoga makalah ini bermanfaat bagi siapapun yang membacanya.

Bandung, Desember 2015

Syaiful Ahdan

BAB I

PENDAHULUAN

1.1 TUJUAN

Tujuan dari tugas akhir ini adalah membuat simulasi jaringan dengan menggunakan *software network simulator 2* (NS2). Membandingkan stress testing dengan distribusi poisson dan mengevaluasi kinerja dari topologi Star, Mesh dan ring dengan parameter throughput, rata-rata delay, probabilitas paket sukses, jitter sehingga akan mendapatkan solusi yang efisien dan optimal dari kinerja ketiga topologi tersebut.

1.2 MANFAAT

Dengan adanya proyek akhir ini kita dapat mengetahui topologi mana yang paling efisien dan optimal dalam pemilihan topologi guna kepentingan untuk implementasi jaringan terhadap kebutuhan yang sesuai.

BAB II

LANDASAN TEORI

1. Stress Testing

Stress testing adalah proses penentuan kemampuan komputer, jaringan, program atau perangkat untuk mempertahankan tingkat tertentu efektivitas dalam kondisi yang tidak menguntungkan. Proses ini dapat melibatkan tes kuantitatif dilakukan di laboratorium, seperti mengukur frekuensi kesalahan atau sistem crash. Istilah ini juga mengacu pada evaluasi kualitatif faktor-faktor seperti ketersediaan atau resistensi terhadap (DoS) serangan denial-of-service. Stress testing sering dilakukan bersamaan dengan proses yang lebih umum dari pengujian

2. NS2

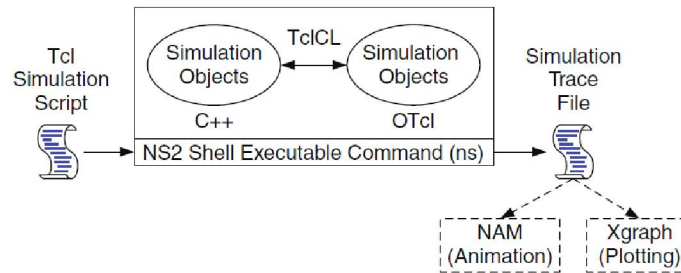
Network Simulator versi 2 atau secara umum dikenal dengan sebutan NS2 [17] adalah sebuah aplikasi simulasi yang telah terbukti berguna dalam mempelajari sifat dinamis dari komunikasi jaringan. NS2 merupakan salah satu simulator yang bersifat *open source* di bawah GPL (Gnu Public License), sehingga NS-2 dapat didownload melalui website NS-2 [18]. Pemodelan media, protokol dan network komponen lengkap dengan perilaku trafiknya sudah tersedia pada library NS-2.

Pada *Network Simulator* terdapat tampilan atau *display* baik dengan *node* yang bergerak atau *node* yang tidak bergerak, yang tentunya tidak sama dengan keadaan yang sebenarnya. Paket-paket yang membangun dalam simulasi jaringan ini antara lain :

- Tcl : *Tool Command Language*
- Tk : *Tool Kit*
- Otcl : *Object Tool Command Language*
- Tccl : *Tool Command Language / C++ Interface*
- NS2 : *Network Simulator versi 2*
- NAM : *Network Animator*

- NS-2 menyediakan sebuah perintah ns, yang merupakan sebuah perintah yang dapat dieksekusi (executable) oleh penggunanya. Dalam menjalankan simulasi pada NS-2, perintah ns tersebut membutuhkan argumen masukan berupa nama dari skrip

simulasi tel yang telah dipersiapkan untuk simulasi. NS2 akan menjalankan simulasi berdasarkan skenario yang terdapat pada file skrip simulasi tel tersebut. Simulasi tersebut akan menghasilkan sebuah file trace yang berisikan data hasil simulasi. File tersebut akan digunakan sebagai dasar dalam menampilkan grafik hasil simulasi dan menampilkan animasi simulasi. Gambar berikut menunjukkan arsitektur dari NS.



Gambar 2.1. Arsitektur dasar NS 2

Simulator NS-2 dijalankan dengan menggunakan dua bahasa pemrograman, yaitu C++ dan Object-orientes Tool Command Language (OTcl), C++ berfungsi dalam menangani mekanisme internal pada simulasi dengan NS-2. OTcl menangani interaksi langsung antara pengguna dengan simulator serta menangani interaksi antara objek-objek OTcl lainnya. C++ dan OTcl saling terhubung dengan menggunakan komponen TclCL. Variabel-variabel pada domain OTcl dipetakan pada objek C++. Variabel ini cenderung dikenal sebagai sebuah handle. Dalam domain OTcl, sebuah handle berfungsi sebagai substansi untuk menangani interaksi simulator dengan pengguna.

3. AWK (*Alfred V. Aho, Peter J. Weinberger dan Brian W*)

Awk adalah sebuah pemrograman seperti pada shell atau C yang memiliki karakteristik yaitu sebagai tool yang cocok filter/manipulasi Awk adalah penggabungan dari nama lengkap sang author, yaitu : Alfred V. Aho, Peter J. Weinberger dan Brian W. ernighan. Awk atau juga disebut Gawk (GNU awk), yaitu bahasa pemrograman umum dan utility standard POSIX 1003.2. (Portable Operating System Interface for UNIX). Jika kecepatan merupakan hal yang penting, awk adalah bahasa yang sangat sesuai.

Awk dan shell, keduanya adalah biasa dipakai untuk aplikasi yang berbeda. Awk sangat baik untuk manipulasi file teks, sedangkan shell sangat baik untuk pelaksana perintah UNIX. Secara umum bahasa pemrograman awk dapat digunakan untuk :

- Mengelola database sederhana.
- Membuat report.
- Memvalidasi data.
- Menghasilkan index and menampilkan dokumen.
- Membuat algoritma yang digunakan untuk mengubah bahasa komputer ke bahasa lainnya.

Dengan kata lain awk menyediakan fasilitas yang dapat memudahkan untuk:

- Memecah bagian data untuk proses selanjutnya.
- Mengurutkan data.
- Menampilkankomunikasi jaringan yang sederhana.

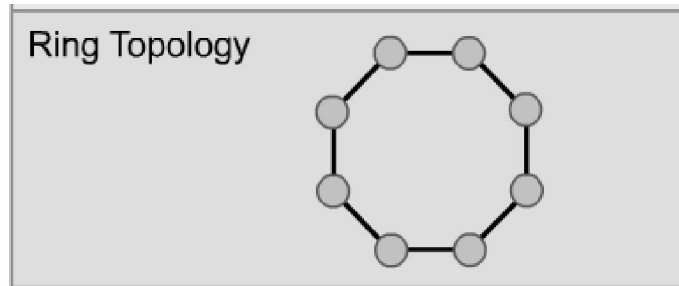
Fungsi dasar awk adalah untuk mencari file per baris (atau unit teks lain) yang berisi pola tertentu. Ketika suatu baris sesuai dengan pola, awk melakukan aksi yang khusus pada baris tersebut. awk tetap memproses baris input sedemikian hingga mencapai akhir baris input. Program pada awk berbeda dari program di kebanyakan bahasa lain, karena program awk bersifat “data-driven”; yang mana kamu diminta untuk mendeskripsikan data yang dikehendaki untuk bekerja dan kemudian apa yang akan dilakukan saat data tersebut ditemukan. Kebanyakan bahasa lainnya bersifat “procedural”; kamu harus mendeskripsikannya secara detail setiap langkah program yang harus dijalankan. Ketika bekerja dengan bahasa prosedural, biasanya sangat sulit untuk mendeskripsikan data yang hendak diproses oleh program. Oleh karena itu, program awk sering kali terasa lebih mudah untuk ditulis dan dibaca.

4. Topology jaringan

Topologi jaringan adalah bagaimana cara dan bentuk, baik secara fisik maupun secara logik dalam hubungan antar komputer. Berikut macam-macam topologi :

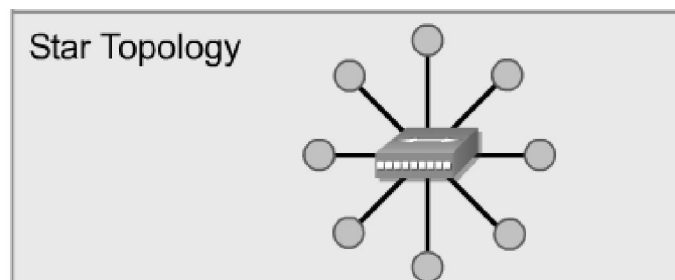
- a. Topology Ring : Dalam topologi *ring*, setiap *node* (komputer) dihubungkan dengan *node* lain sehingga membentuk lingkaran. Sistem transmisinya

menggunakan kabel yang saling menghubungkan beberapa *workstation* dengan *file server* dalam bentuk lingkaran tertutup. Topologi ini juga memiliki kelemahan, yaitu apabila salah satu hubungan ada yang putus maka keseluruhan hubungan akan terputus.



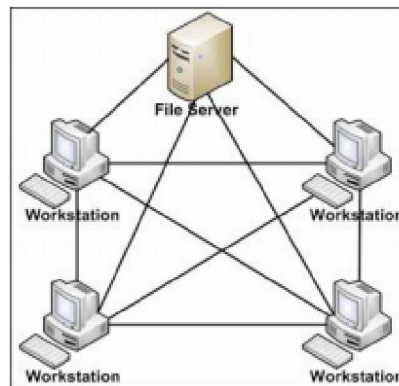
Gambar 2.2. Topology Ring

- b. **Topologi star** : paling banyak dipakai terutama dalam sistem telekomunikasi. Karena diantara topologi yang lain, topologi star dirasa paling ideal dan menguntungkan. Media transmisi yang digunakan dalam topologi star membentuk jalur tertutup (closed loop) dan setiap workstation mempunyai kabel tersendiri untuk langsung berhubungan dengan *file server*, sehingga seluruh sistem tidak akan gagal bila ada salah satu kabel pada workstation terganggu. Hal ini membuat sistem topologi *star* mudah dikembangkan, karena tiap *node* hanya memiliki kabel yang langsung terhubung ke sentral node, sehingga apabila salah satu kabel *node* terputus maka *node* lainnya tidak terganggu.



Gambar 2.3. Topology Star

- c. **Topologi mesh** : merupakan sebuah bentuk topologi jaringan dimana setiap node terhubung langsung dengan node lain pada jaringan. Hingga membentuk rangkaian menyerupai jala / jaring. Karena setiap node terhubung secara langsung dengan node yang lain maka ketika akan berkomunikasi setiap node tidak memerlukan perantara atau biasa disebut dedicated links. Berdasarkan pemaparan diatas kita bisa mengambil beberapa perhitungan dalam topologi mesh. Misalnya terdapat 6 komputer dalam jaringan maka kabel koneksi yang diperlukan agar jaringan bekerja maksimal adalah $6(6-1)/2 = 15$ koneksi dengan rumus $n(n-1)/2$. Dan masing-masing Komputer diharuskan memiliki Port I/O sejumlah $6-1= 5$ port I/O dengan rumus $n-1$.



Gambar 2.4. Topology mesh

BAB III

PERANCANGAN SISTEM

Dalam pembuatan suatu sistem harus dilakukan perencanaan dan perancangan sistem yang sesuai dengan tujuan serta permasalahan yang dihadapi. Bab ini akan membahas secara rinci mengenai perencanaan dan pembuatan sistem yang akan dibuat. Dalam proses pembuatan dan pengujian stress testing pada beberapa topologi menggunakan aplikasi NS2 sebagai simulasi jaringan.

3.1. PERANCANGAN SISTEM

Dalam perancangan dan pembuatan sistem terdapat proses yang lebih rinci antara lain :

1. Scenario

Pada tahap pertama yaitu tahap pembuatan skenario untuk topologi star, mesh dan ring, dimana dalam skenario tersebut terdiri dari file .tcl. yang berisi titik koordinat yang menyusun tiap-tiap node menjadi sebuah topologi. File tcl berisi skenario topologi yang terdiri dari variabel yang membantu mengubah skenario jaringan dan mengontrol proses simulasi.

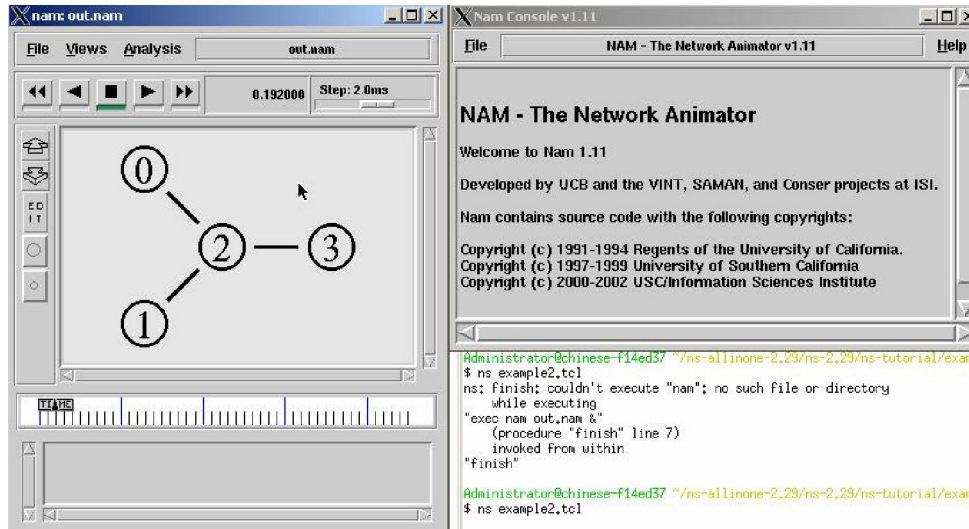
2. NS2

Network Simulator versi 2 atau secara umum dikenal dengan sebutan NS2 adalah sebuah aplikasi simulasi yang telah terbukti berguna dalam mempelajari sifat dinamis dari komunikasi jaringan. Skenario yang telah dibuat dijalankan menggunakan NS2, yang nantinya menghasilkan data yang dikemas menjadi file trace (.tr). Selain menghasilkan file trace, NS2 juga menghasilkan tampilan atau *display* berupa *node* bergerak atau *node* yang tidak bergerak, tampilan tersebut mensimulasikan skenario star dan cluster tentunya tidak sama dengan keadaan yang sebenarnya.

3. NAM : *Network Animator*

Network Animator adalah salah satu dari hasil output NS2 yang menampilkan simulasi dengan tampilan animasi, pada NAM ini dapat diketahui proses mulai dari *device* atau *node* aktif satu persatu dan proses pengiriman data antara node dengan

Koordinator PAN. Tampilan NAM dapat dilihat seperti pada gambar 3.3 dibawah ini:



Gambar 3.1 *Network Animator*

4. File trace

File trace atau file .tr berisi data hasil simulasi mulai dari waktu awal sampai waktu akhir yang nantinya akan digunakan untuk analisa numerik. Berikut adalah contoh isi dari *file trace*, hasil dari *trace* selama simulasi pada jaringan wireless :

```
s 2.000931471 _3_ MAC --- 20 tcp 112 [13a 0 1 800] ----- [4194305:2 0:0 32
4194304] [0 0] 0 0
```

- *Event* (kejadian) berisi kejadian r "*received*", s "*sent*", f "*forwarded*" dan D "*dropped*".
- *Time* (waktu).
- ID Node tempat kejadian .
- *Trace level* antara lain MAC menunjukkan jika packet berhubungan dengan MAC *layer*. Untuk AGT menunjukkan *packet transport layer*. Untuk RTR jika itu menunjukkan *packet route*.
- *Sequence number* (nomor urut packet).
- "*tcp*" adalah tipe paket (tcp, ack, udp).
- Ukuran packet.

- [13a 0 1 800] menunjukkan informasi MAC *layer*.
- [4194305:2 0:0 32 4194304] menunjukkan *IP source* dan alamat tujuan kemudian *ttl (time to live)* dari paket.
- [0 0] menunjukkan nomor urut dan pemberitahuan nomor (*tcp information*).
- 0 0 adalah format mekanisme *routing type pack*.

5. Filtering

Pada proses filtering dapat menggunakan beberapa macam pemrograman seperti python, perl, dan awk. Namun pada proyek akhir ini kami menggunakan pemrograman AWK untuk proses pemfilteran. AWK adalah bahasa pemrograman yang digunakan untuk memanipulasi data dan membuat laporan. Dalam proses filtering ini dengan input file trace (.tr) akan diproses dan menghasilkan file output yaitu throughput.txt, delay.txt, dan energy_consumption.txt. Ketiga file tersebut adalah hasil filtering dari *file trace*.

Berikut adalah cara kerja awk dalam memproses data *file traced* dari hasil simulasi :

- Awk akan membaca masukan *file* pada tiap baris.
- Pada tiap baris yang dibaca, jika ditemukan data sesuai pattern yang ada maka dilakukan proses sesuai dengan *action* yang ada.
- Jika data tidak ditemukan sesuai *pattern* yang ada, maka tidak *action* yang akan diproses.
- Jika tidak terdapat pencarian *pattern*, maka awk akan memproses *action* pada tiap baris masukan.
- Jika tidak terdapat proses *action* diberikan, maka data pada baris yang cocok dengan *pattern* akan ditampilkan pada layar sebagai *default action*.

6. Output Filter

Output filter ini adalah file yang dihasilkan dari proses filtering menggunakan pemrograman AWK. File tersebut yaitu throughput.txt, delay.txt, dan energy_consumption.txt. File throughput.txt berisi nilai throughput tiap node. Throughput adalah ukuran dari seberapa cepat proses pengiriman data melalui jaringan. Jadi data yang ditampilkan pada throughput.txt berupa kecepatan transfer data tiap node dengan PAN koordinator. Delay.txt berisi file delay yaitu waktu yang dibutuhkan data untuk menempuh jarak dari node asal ke node tujuan. Energy_consumption.txt berisi data prosentase dari energi yang dikonsumsi oleh

sebuah node berkenaan dengan energi awal. Energi awal dan energi akhir yang tersisa pada node, pada akhir simulasi akan dihitung sebagai prosentasi energi yang dikonsumsi oleh total node.

3.2. PEMBUATAN SKENARIO

Pada program simulasi topologi terdapat parameter yang dapat mempengaruhi hasil simulasi. Parameter yang digunakan dalam simulasi digolongkan menjadi 2 bagian yaitu parameter yang telah didefinisikan oleh NS2 dan parameter yang didefinisikan sendiri oleh perancang.

Table 3.1 Parameter simulasi jaringan

Parameter	Nilai
Tipe antarmuka antrian	Drop Tail
Jumlah node	8
Waktu simulasi	2500 second
Jenis Topology	Star, ring, mesh

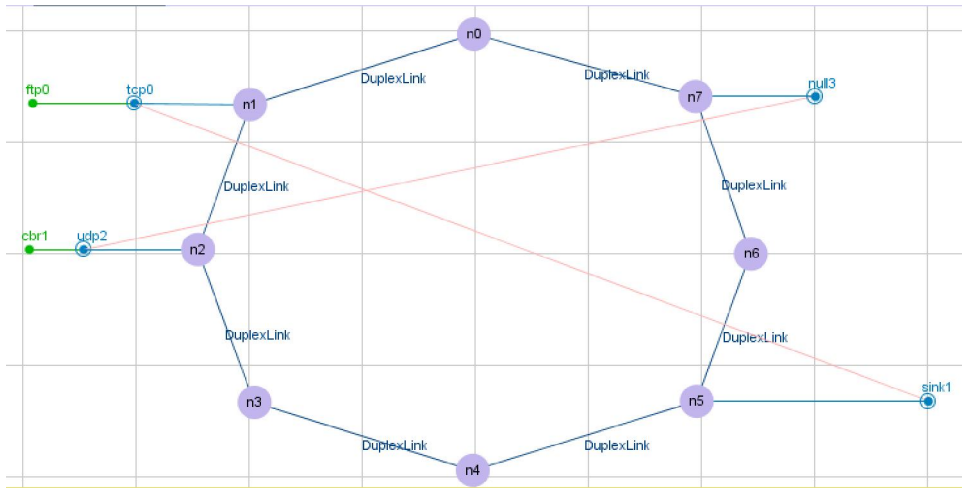
3.3. MODEL TOPOLOGI

Dalam melakukan analisa perbandingan kinerja pada beberapa topology kami menggunakan maksimal 8 node pada setiap topologi. Dimulai dari node 0 , node 1 sampai dengan node 8 Contoh model dan spesifikasi dari beberapa topologi adalah sebagai berikut :

1. Topologi Ring

Table 3.2 Desain Topologi ring

Source	Agen	App	Tipe	Destination	Agen	App
Node 0	TCP	FTP	Duplex Link	Node 5	Sink 1	FTP
Node 2	UDP	CBR	Duplex Link	Node 7	Null	CBR



Gambar 3.2 *Desain Topology Ring*

Listing Code pada NS2 untuk pembuatan Node pada topology ring

#Create 8 nodes

```
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]
```

Links Definition

```
$ns duplex-link $n0 $n1 5.0Mb 1ms DropTail
$ns queue-limit $n0 $n1 50
$ns duplex-link $n2 $n1 5.0Mb 1ms DropTail
$ns queue-limit $n2 $n1 50
$ns duplex-link $n3 $n2 5.0Mb 1ms DropTail
$ns queue-limit $n3 $n2 50
$ns duplex-link $n3 $n4 5.0Mb 1ms DropTail
$ns queue-limit $n3 $n4 50
$ns duplex-link $n4 $n5 5.0Mb 1ms DropTail
$ns queue-limit $n4 $n5 50
$ns duplex-link $n5 $n6 5.0Mb 1ms DropTail
$ns queue-limit $n5 $n6 50
$ns duplex-link $n6 $n7 5.0Mb 1ms DropTail
$ns queue-limit $n6 $n7 50
$ns duplex-link $n7 $n0 5.0Mb 1ms DropTail
$ns queue-limit $n7 $n0 50
```

Agents Definition

#Setup a TCP connection

```
set tcp0 [new Agent/TCP]
$ns attach-agent $n1 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n5 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 1500
```

#Setup a UDP connection

```
set udp2 [new Agent/UDP]
$ns attach-agent $n2 $udp2
set null3 [new Agent/Null]
$ns attach-agent $n7 $null3
$ns connect $udp2 $null3
$udp2 set packetSize_ 1500
```

#Applications Definition

```
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
#paket TCP dikirim di detik ke 0.10 sampai dengan
menit ke 1.0
$ns at 0.10 "$ftp0 start"
$ns at 1.0 "$ftp0 stop"
```

#Setup a CBR Application over UDP connection

```
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp3
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
```

#paket TCP dikirim di detik ke 0.5 sampai dengan menit ke 1.0

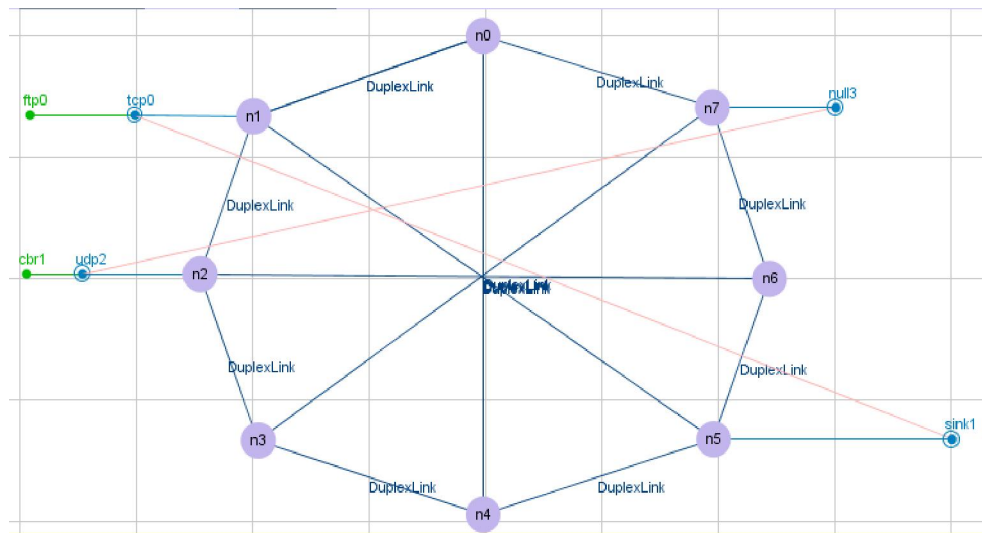
```
$ns at 0.05 "$cbr1 start"
$ns at 1.0 "$cbr1 stop"
```

link terputus di jalur antara node6 ke node7 di detik ke 30

```
$ns rtmodel-at 0.30 down $n7 $n6
# link naik kembali di jalur antara node4 ke node5 di
detik ke 3
$ns rtmodel-at 0.35 up $n7 $n6
```

Source	Agen	App	Tippe	Destination	Agen	App
Node 0	TCP	FTP	Duplex Link	Node 5	Sink 1	FTP
Node 2	UDP	CBR	Duplex Link	Node 7	Null	CBR

2. Topologi Mesh



Gambar 3.3 Desain Topology Mesh

Listing Code pada NS2 untuk pembuatan Node pada topology mesh

#Create 8 nodes

```
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]
```

Links Definition

```
$ns duplex-link $n0 $n1 5.0Mb 1ms DropTail
$ns queue-limit $n0 $n1 50
$ns duplex-link $n2 $n1 5.0Mb 1ms DropTail
$ns queue-limit $n2 $n1 50
$ns duplex-link $n3 $n2 5.0Mb 1ms DropTail
```

```
$ns queue-limit $n3 $n2 50
$ns duplex-link $n3 $n4 5.0Mb 1ms DropTail
$ns queue-limit $n3 $n4 50
$ns duplex-link $n4 $n5 5.0Mb 1ms DropTail
$ns queue-limit $n4 $n5 50
$ns duplex-link $n5 $n6 5.0Mb 1ms DropTail
$ns queue-limit $n5 $n6 50
$ns duplex-link $n6 $n7 5.0Mb 1ms DropTail
$ns queue-limit $n6 $n7 50
$ns duplex-link $n7 $n0 5.0Mb 1ms DropTail
$ns queue-limit $n7 $n0 50
```

Agents Definition

#Setup a TCP connection

```
set tcp0 [new Agent/TCP]
$ns attach-agent $n1 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n5 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 1500
```

#Setup a UDP connection

```
set udp2 [new Agent/UDP]
$ns attach-agent $n2 $udp2
set null3 [new Agent/Null]
$ns attach-agent $n7 $null3
$ns connect $udp2 $null3
$udp2 set packetSize_ 1500
```

#Setup a FTP Application over TCP connection set ftp0 [new Application/FTP]

```
$ftp0 attach-agent $tcp0
```

#paket TCP dikirim di detik ke 0.10 sampai dengan menit ke 1.0

```
$ns at 0.10 "$ftp0 start"
$ns at 1.0 "$ftp0 stop"
```

#Setup a CBR Application over UDP connection set cbr1 [new Application/Traffic/CBR]

```
$cbr1 attach-agent $udp3
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
```

#paket TCP dikirim di detik ke 0.5 sampai dengan menit ke 1.0

```
$ns at 0.05 "$cbr1 start"
$ns at 1.0 "$cbr1 stop"
```



```
#packet FTP dari node 1 ditujukan ke node 5 (secara Directly)
# link terputus di jalur antara node1 ke node5 di detik ke 30
setelah link terputus packet UDP dari node 1 ditujukan ke
node 5 packet di route kembali ke (node 1 > node 2 > node 6
> node 5)
```

```
$ns rtmodel-at 0.30 down $n5 $n1
```

```
#link naik kembali di jalur antara node5 ke node1 di detik ke
3
```

```
$ns rtmodel-at 0.90 up $n5 $n1
```

```
#packet UDP dari node 2 ditujukan ke node 7 via (node 2 >
node 3 > node 7) # link terputus di jalur antara node2 ke
node3 di detik ke 30 #setelah link terputus packet UDP dari
node 2 ditujukan ke node 7 di route kembali ke (node 2 >
node 6 > node 7)
```

```
$ns rtmodel-at 0.35 down $n2 $n3
```

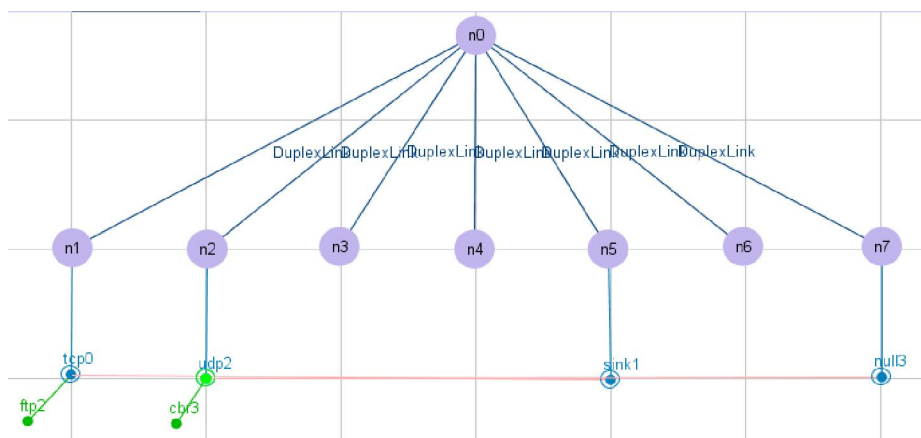
```
# link naik kembali di jalur antara node2 ke node1 di detik
ke 3
```

```
$ns rtmodel-at 0.90 up $n2 $n3
```

3. Topologi Star

Table 3.3 Desain Topologi Star

Source	Agen	App	Type	Destination	Agen	App
Node 0	TCP	FTP	Duplex Link	Node 5	Sink 1	FTP
Node 2	UDP	CBR	Duplex Link	Node 7	Null	CBR



Gambar 3.4 Desain Topology Ring

Listing Code pada NS2 untuk pembuatan Node pada topology star**#Create 8 nodes**

```
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]
```

Links Definition

```
$ns duplex-link $n0 $n1 5.0Mb 1ms DropTail
$ns queue-limit $n0 $n1 50
$ns duplex-link $n2 $n1 5.0Mb 1ms DropTail
$ns queue-limit $n2 $n1 50
$ns duplex-link $n3 $n2 5.0Mb 1ms DropTail
$ns queue-limit $n3 $n2 50
$ns duplex-link $n3 $n4 5.0Mb 1ms DropTail
$ns queue-limit $n3 $n4 50
$ns duplex-link $n4 $n5 5.0Mb 1ms DropTail
$ns queue-limit $n4 $n5 50
$ns duplex-link $n5 $n6 5.0Mb 1ms DropTail
$ns queue-limit $n5 $n6 50
$ns duplex-link $n6 $n7 5.0Mb 1ms DropTail
$ns queue-limit $n6 $n7 50
$ns duplex-link $n7 $n0 5.0Mb 1ms DropTail
$ns queue-limit $n7 $n0 50
```

Agents Definition**#Setup a TCP connection**

```
set tcp0 [new Agent/TCP]
$ns attach-agent $n1 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n5 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 1500
```

#Setup a UDP connection

```
set udp2 [new Agent/UDP]
$ns attach-agent $n2 $udp2
set null3 [new Agent/Null]
$ns attach-agent $n7 $null3
$ns connect $udp2 $null3
$udp2 set packetSize_ 1500
```

```
#Setup a FTP Application over TCP connection
  set ftp0 [new Application/FTP]
  $ftp0 attach-agent $tcp0
#paket TCP dikirim di detik ke 0.10 sampai dengan menit ke 1.0
  $ns at 0.10 "$ftp0 start"
  $ns at 1.0 "$ftp0 stop"

#Setup a CBR Application over UDP connection
  set cbr1 [new Application/Traffic/CBR]
  $cbr1 attach-agent $udp3
  $cbr1 set packetSize_ 1000
  $cbr1 set rate_ 1.0Mb
  $cbr1 set random_ null

#paket TCP dikirim di detik ke 0.5 sampai dengan menit ke 1.0
  $ns at 0.05 "$cbr1 start"
  $ns at 1.0 "$cbr1 stop"

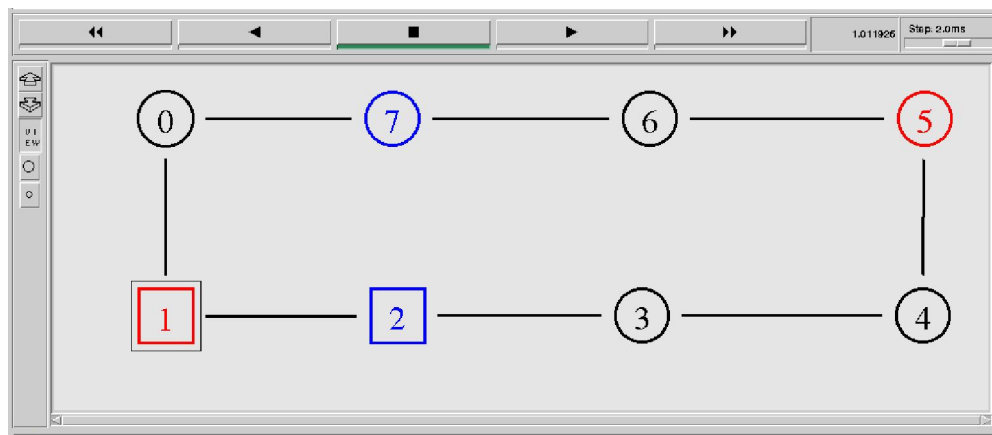
# link terputus di jalur antara node0 ke node 7 di detik ke 30
  $ns rtmodel-at 0.30 down $n0 $n7
# link naik kembali di jalur antara node0 ke node5 di detik ke 0.50
  $ns rtmodel-at 0.50 up $n0 $n7

# link terputus di jalur antara node0 ke node 5 di detik ke 30
  $ns rtmodel-at 0.40 down $n0 $n5
# link naik kembali di jalur antara node0 ke node5 di detik ke 0.40
  $ns rtmodel-at 0.55 up $n0 $n5
```

4.4 PEMBUATAN SKENARIO

Setelah Seluruh desain topologi dirancang dengan menggunakan NS3 tahap berikutnya adalah melakukan pengujian dengan menggunakan Network animator adapun setiap sekenario pada masing masing topologi berbeda beda dan disesuaikan.

1. Sekenario Topologi Ring



Gambar 3.5 *Implentasi Topology Ring*

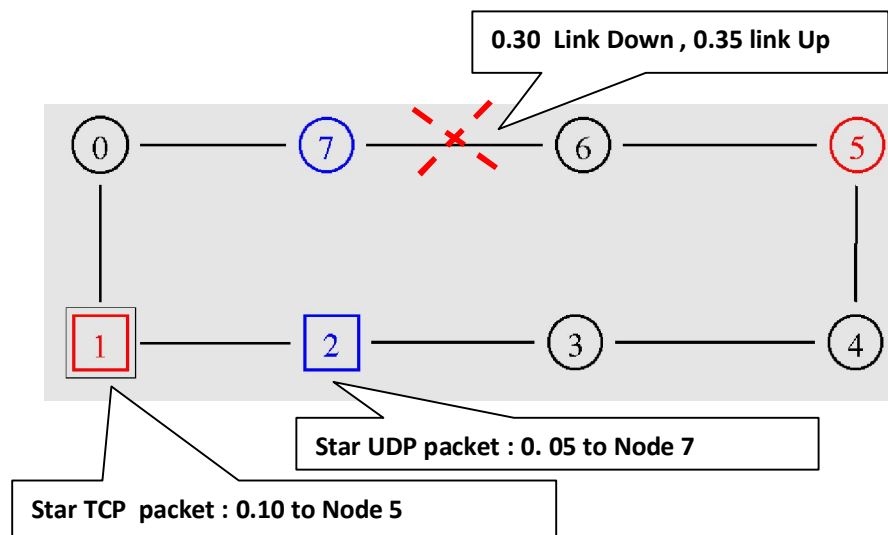
Source	Agen	Destination	App	start	stop
Node 1	TCP	Node 5	FTP	0.10	1.0
Node 2	UDP	Node 7	CBR	0.05	1.0

Table 3.4 *Desain Topologi Ring*

Pada topologi ring dibuat sekenario bahwa pada **node 1** dengan aplikasi TCP akan mengirimkan packet FTP ke **Node5** dimulai pada detik ke 10 dan berhenti pada menit ke 1.0 , dengan sekenario link akan terputus **pada link antara node 6 dan node 7**, pada saat detik **0.30** , seHINGA apakah paket yang akan dikirim apakah dapat sampai ket tujuan dengan perpindahan routing.

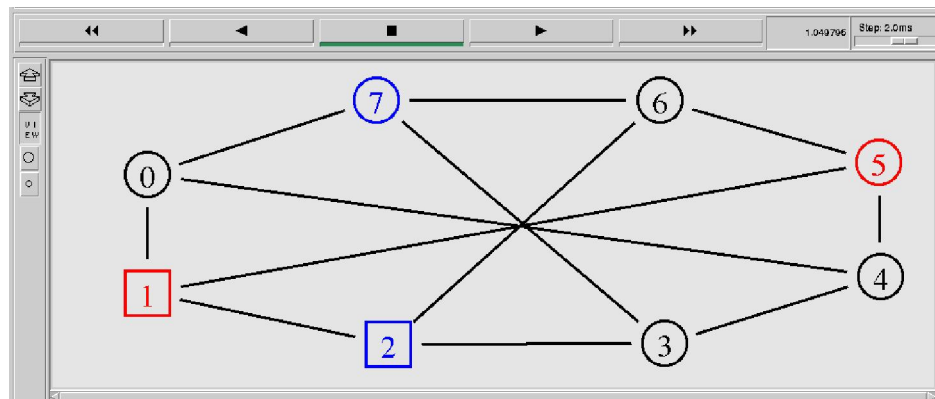
Kemudian link akan dihidupkan kembali pada saat detik ke **0.35** , sehingga pada proses perpindahan paket apakah paket yang dikirim segera berpindah ke jalur yang sudah hidup kembali apakah tetap menggunakan jalur yang lama

sedangkan pada skenario pengiriman pake UDP bahwa pada **node 2** dengan aplikasi UDP akan mengirimkan packet CBR **ke Node 7** dimulai pada detik ke 0.5 dan berhenti pada menit ke 1.0 , dengan skenario link akan terputus **pada link antara node 6 dan node 7**, pada saat detik **0.30** , sehingga apakah paket yang akan dikirim dari node 2 ke node 7 akan terpengaruh



Gambar 3.5 Skenario Topology Ring

2. Skenario Topologi mesh



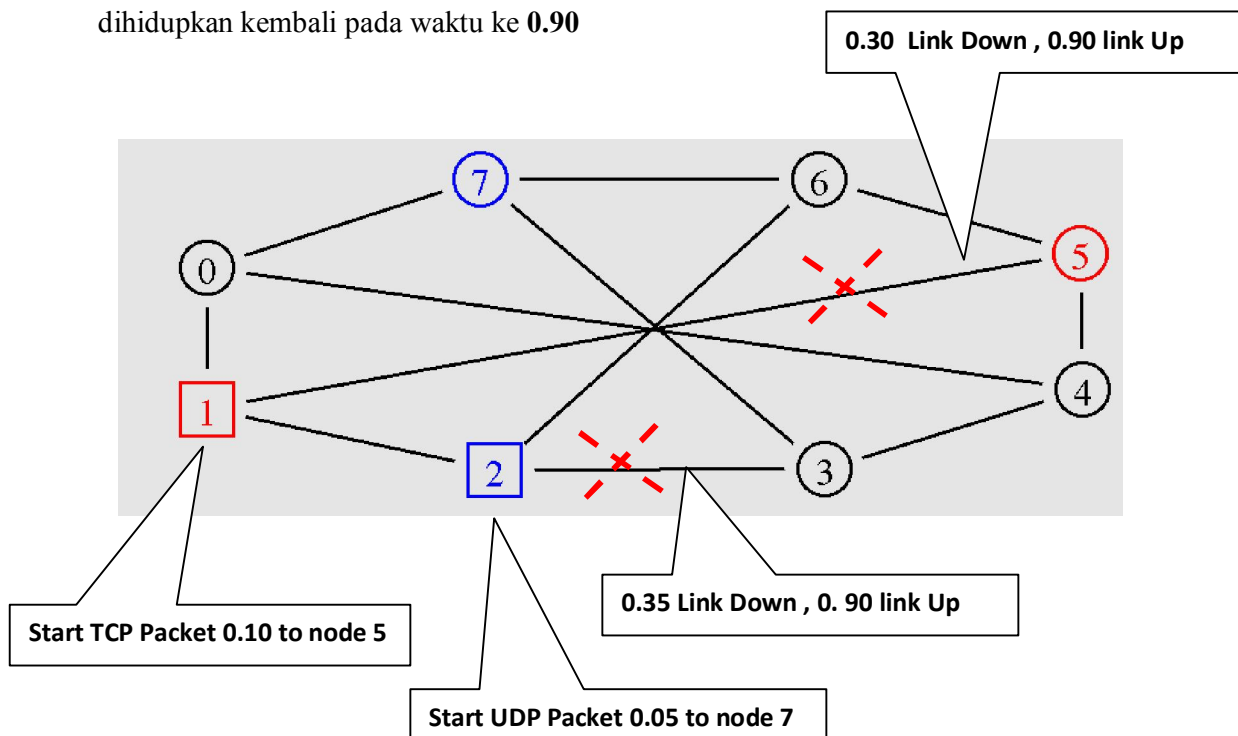
Gambar 3.7 Implementasi Topologi Mesh

Table 3.5 Sekenario Topologi Ring

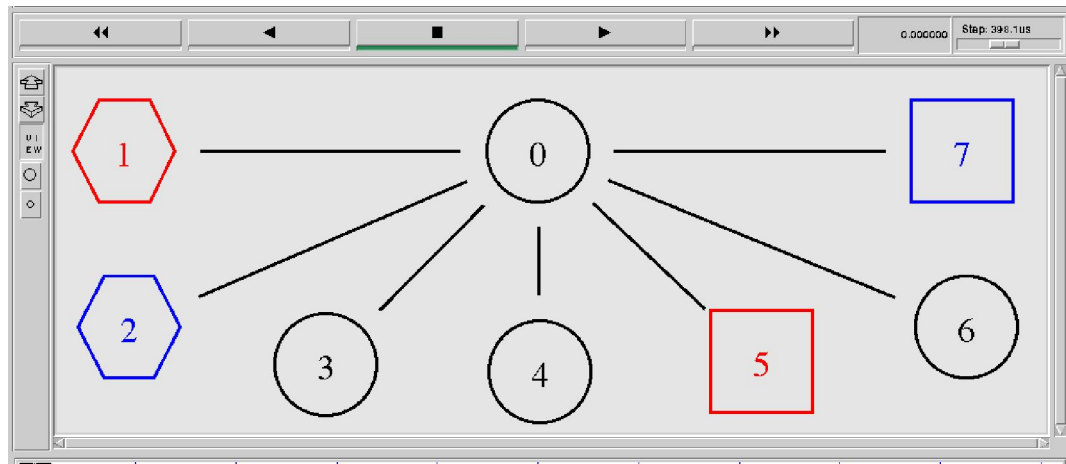
Source	Agan	Destination	App	start	stop
Node 1	TCP	Node 5	FTP	0.10	1.0
Node 2	UDP	Node 7	CBR	0.05	1.0

packet FTP dari node 1 ditujukan ke node 5 (secara Directly) kemudian link terputus di jalur antara **node1 ke node 5** di detik ke **0.30** dengan sekenario link akan terputus **pada link antara node 6 dan node 7**, pada saat detik **0.30** , seHINGA apakah paket yang akan dikirim apakah dapat sampai ket tujuan dengan perpindahan routing. Kemudian link akan dihidupkan kembali pada saat detik ke **0.90** , sehingga pada proses perpindahan paket apakah paket yang dikirim segera berpindah ke jalur yang sudah hidup kembali apakah tetap menggunakan jalur yang lama.

Sedangkan pada packet UDP dari node **2 ditujukan ke node 7** link terputus di jalur antara node **2 ke node 3** di detik ke **0.35** setelah link terputus link antara node 2 dan 3 dihidupkan kembali pada waktu ke **0.90**

**Gambar 3.8** Sekenario Topologi Mesh

3. Sekenario Topologi Star



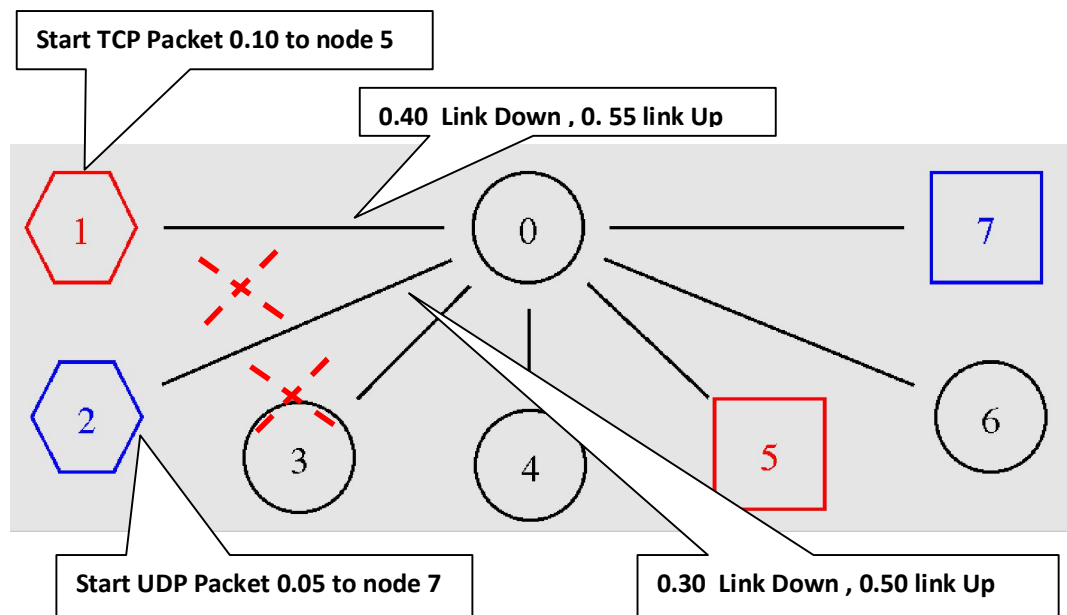
Gambar 3.9 Implementasi Topologi Star

Source	Agan	Destination	App	start	stop
Node 1	TCP	Node 5	FTP	0.10	1.0
Node 2	UDP	Node 7	CBR	0.05	1.0

Table 3.6 Sekenario Topologi Star

packet FTP dari node 1 ditujukan ke node 5 melalui perantara **Node 0** kemudian link terputus di jalur antara **node 0 ke node 1** di detik ke **0.30** dengan sekenario link akan terputus **pada link antara node 0 dan node 1** , pada saat detik **0.30** , sehingga apakah paket yang akan dikirim apakah dapat sampai ket tujuan dengan perpindahan routing. Kemudian link akan dihidupkan kembali pada saat detik ke **0.50** , sehingga pada proses perpindahan paket apakah paket yang dikirim segera berpindah ke jalur yang sudah hidup kembali apakah tetap menggunakan jalur yang lama.

Sedangkan pada packet UDP dari node **2 ditujukan ke node 7** link terputus di jalur antara node **0 ke node 2** di detik ke **40** setelah link terputus link antara node 2 dan 3 dihidupkan kembali pada waktu ke **0.55**



Gambar 3.10 *Sekenario Topologi Star*

BAB IV

PENGUJIAN SISTEM

4.5 PENGUJIAN

Pada bab ini menjelaskan mengenai uji coba simulasi scenario pada topologi star , ring dan mesh dengan menjalankannya pada PC/Laptop. Hasil uji coba yang telah dilakukan akan di analisa untuk mengetahui apakah perancangan aplikasi yang telah dibangun sesuai dengan tujuan yang diharapkan sebagaimana yang telah dipaparkan pada Bab 1.

Analisa yang perlu dilakukan adalah tentang throughput, delay, jitter dan probabilitas paket sukses pada pengujian beberapa topology. Hal ini dilakukan untuk mengetahui perbandingan kinerja antara topologi star , ring dan mesh

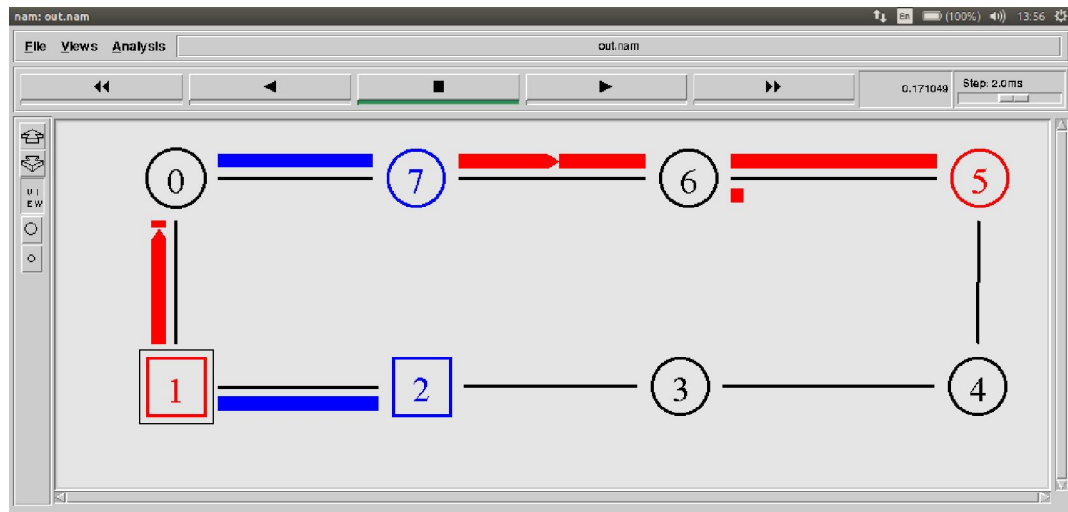
4.5.1 UJI COBA TOPOLOGY RING

Table 3.7 Sekenario Topologi Ring

Source	Agen	Destination	App	start	stop
Node 1	TCP	Node 5	FTP	0.10	1.0
Node 2	UDP	Node 7	CBR	0.05	1.0

Berdasarkan hasil pengujian pertama bahwa pengujian sesuai dengan skenario sesuai pada table diatas bahwa packet FTP dari node 1 dikirim ke Node 5 dimulai pada detik ke 0.10, dan belum dilakukan pemutusan link antara **node 6 dan node 7**, packet UDP dari node 2 dapat dikirimkan ke Node 7.

Paket FTP yang dikirimkan dari node 1 ke node 5 di route dari **node 2 > node 0 > node 7 > node 6 > node 5** , sedangkan paket UDP yang dikirimkan ke node 7 di route dari node 1 > node 0 > node 7



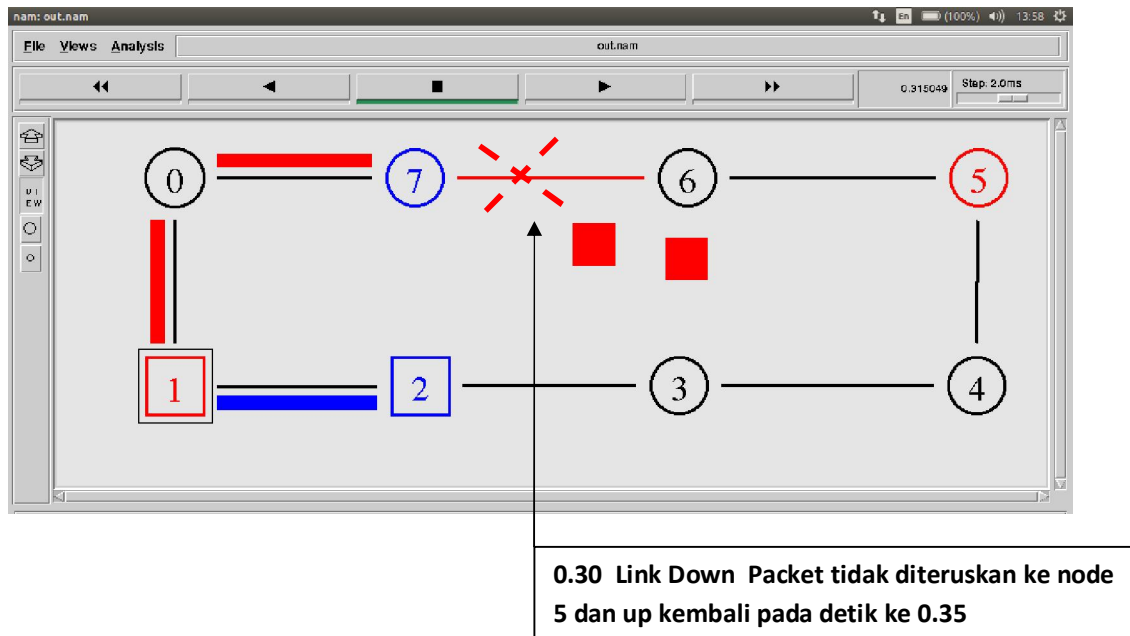
Gambar 3.11 Uji Coba Topologi Star

- Route Awal (Sebelum Link Node 6 dan Node 7 Terputus)

Table 3.8 Route Awal Topologi Ring

Source	Dstination	Directly	Hop 1	Hop2	Hop 3	Total Hop
Node 1	Node 5	Node 0	Node 7	Node 6	Node 5	5
Node 2	Node 7	Node 1	Node 0	Node 7	-	2

Kemudian sesuai dengan sekenario, penulis mencoba memutuskan link antara **node 7 dan node 6** kemudian yang terjadi adalah packet FTP yang dikirim dari Node 1 ke Node 5 terputus sedangkan packet UDP yang dikirim dari node 2 ke node 7 tetap berjalan, berdasarkan proses pengujian dapat dilihat seperti gambar dibawah ini



Gambar 3.12 Link down Node 6 dan 7

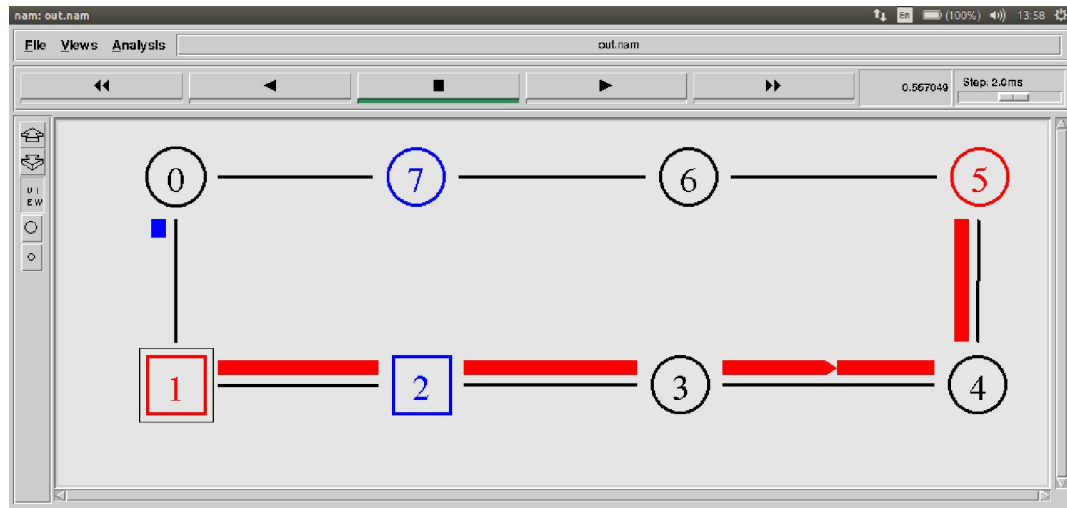
Pada detik ke 0.30 link down sehingga koneksi terputus , kemudian pengiriman dikirim melalui route baru yang secara otomatis di update dan packet diteruskan kembali melwati route yang baru seperti pada table dibawah ini

- Route baru (Setelah Link Node 6 dan Node 7 Terputus)

Table 3.9 Route Baru Topologi Ring

Source	Destination	Directly	Hop 1	Hop2	Hop 3	Total Hop
Node 1	Node 5	Node 2	Node 3	Node 4	Node 5	5
Node 2	Node 7	Node 1	Node 0	Node 7	-	2

Proses route baru dapat dilihat seperti pada gambar dibawah ini, packet FTP dari node 1 ditujukan ke node 5 melewati jalur baru yaitu dari node **1 > Node 2 > Node 4 Node 4 > node 5**



Gambar 3.13 Route Baru pada topologi Ring

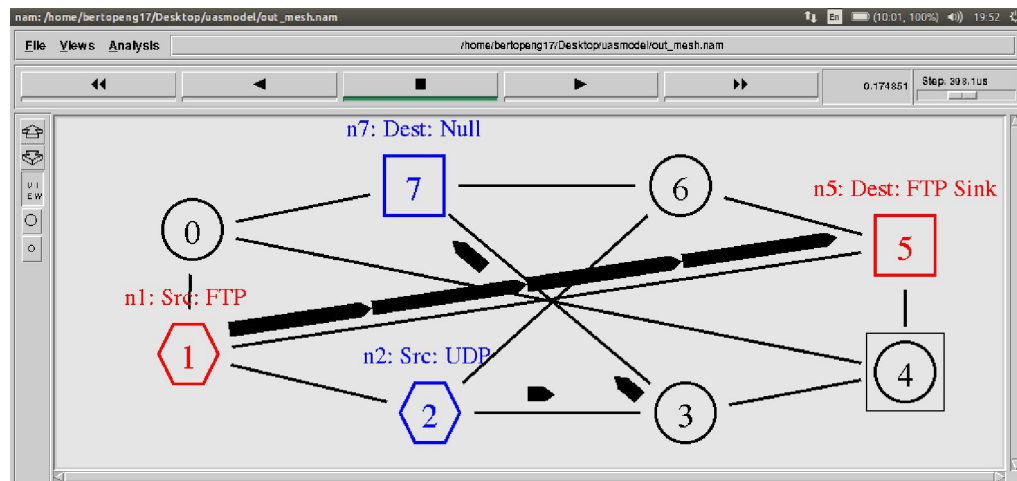
4.5.2 UJI COBA TOPOLOGY MESH

Table 3.10 Seekenario Topology Mesh

Source	Agan	Destination	App	start	stop
Node 1	TCP	Node 5	FTP	0.10	1.0
Node 2	UDP	Node 7	CBR	0.05	1.0

Berdasarkan hasil pengujian pertama bahwa pengujian sesuai dengan sekenario sesuai pada table diatas bahwa packet FTP dari node 1 dikirim ke Node 5 dimulai pada detik ke 0.10, paket UDP yang dikirim dari node 2 ke node 7 dimulai pada detik ke 0.05, dan belum dilakukan pemutusan link antara **node 2 dan node 3**, untuk pengiriman packet UDP dan Node 1 ke Node 5 untuk pengiriman paket FTP sehingga paket UDP dari node 2 dapat dikirimkan ke Node 7 , dan Paket FTP dari node 1 ke Node 5 dapat terkirim.

Paket FTP yang dikirimkan dari node 1 ke node 5 di route secara directly dari **node 1 langsung ke Node 5** , dan paket UDP yang dikirimkan ke node 7 di route dari node 2 > node 3 > node 7, dapat dilihat seperti table dibawah ini.



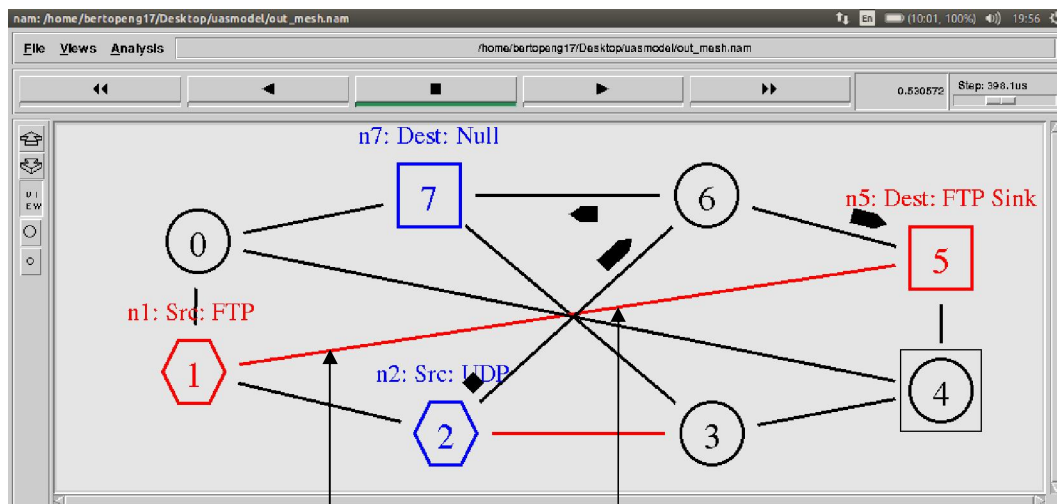
Gambar 3.14 Uji Coba Topologi Mesh

- Route Awal (Sebelum Link Node 2 ke Node 3 Terputus, dan link dari node 1 ke node 5 terputus)

Table 3.11 Route Awal Topologi Mesh

Source	Destination	Directly	Hop 1	Hop2	Hop 3	Total Hop
Node 1	Node 5	Node 5	-	-	-	0
Node 2	Node 7	Node 3	Node 7	-	-	1

Kemudian sesuai dengan skenario, penulis mencoba memutuskan link antara **node 1 dan node 5** kemudian yang terjadi adalah packet FTP yang dikirim dari Node 1 ke Node 5 terputus dan kemudian memutuskan link antara **node 2 dan node 3** kemudian yang terjadi adalah packet UDP yang dikirim dari Node 2 ke Node 7 terputus, berdasarkan proses pengujian dapat dilihat seperti gambar dibawah ini



0.30 Link Down Packet tidak diteruskan ke node 5 dan up kembali pada detik ke 0.90

0.35 Link Down Packet tidak diteruskan ke node 5 dan up kembali pada detik ke 0.90

Gambar 3.15 Link down Node 6 dan 7

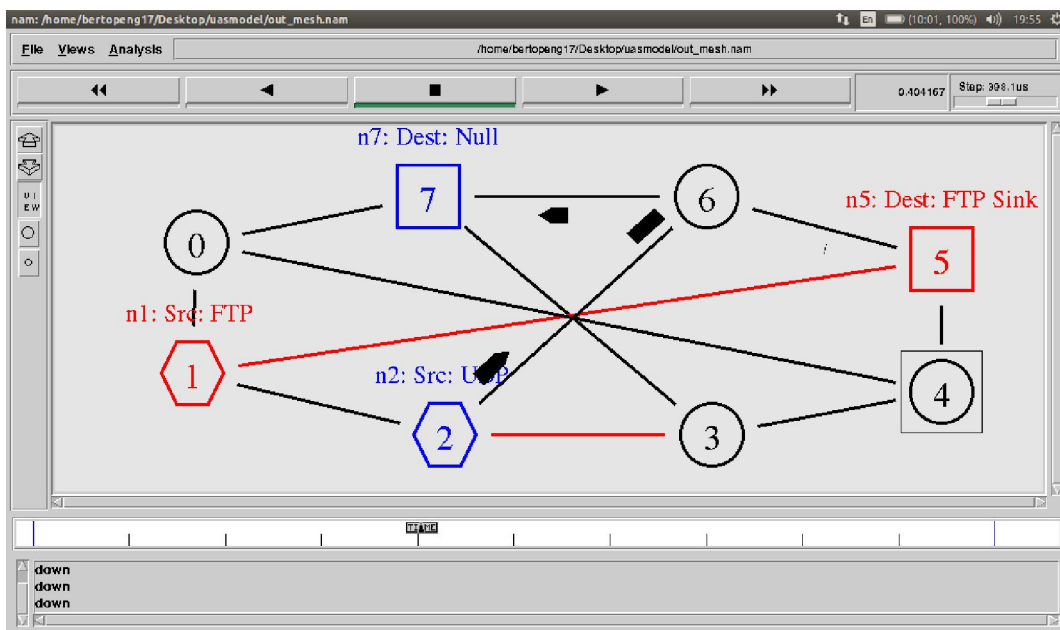
Pada detik ke 0.30 link down sehingga koneksi terputus , kemudian pengiriman dikirim melalui route baru yang secara otomatis di update dan packet diteruskan kembali melwati route yang baru seperti pada table dibawah ini

- Route baru (setelah Link Node 2 ke Node 3 Terputus, dan link dari node 1 ke node 5 terputus)

Table 3.12 Route baru Topologi Mesh

Source	Destination	Directly	Hop 1	Hop2	Hop 3	Total Hop
Node 1	Node 5	Node 2	Node 6	Node 5	-	2
Node 2	Node 7	Node 6	Node 7	-	-	1

Proses route baru dapat dilihat seperti pada gambar dibawah ini, packet UDP dari node 2 ditujukan ke node 7 melewati jalur baru yaitu dari node 2 > **Node 6** > **node 7**



Gambar 3.16 Route Baru pada topologi Mesh

4.5.3 UJI COBA TOPOLOGY STAR

Table 3.13 Sekenario Topologi Star

Source	Agen	Destination	App	start	Stop
Node 1	TCP	Node 5	FTP	0.10	1.0
Node 2	UDP	Node 7	CBR	0.05	1.0

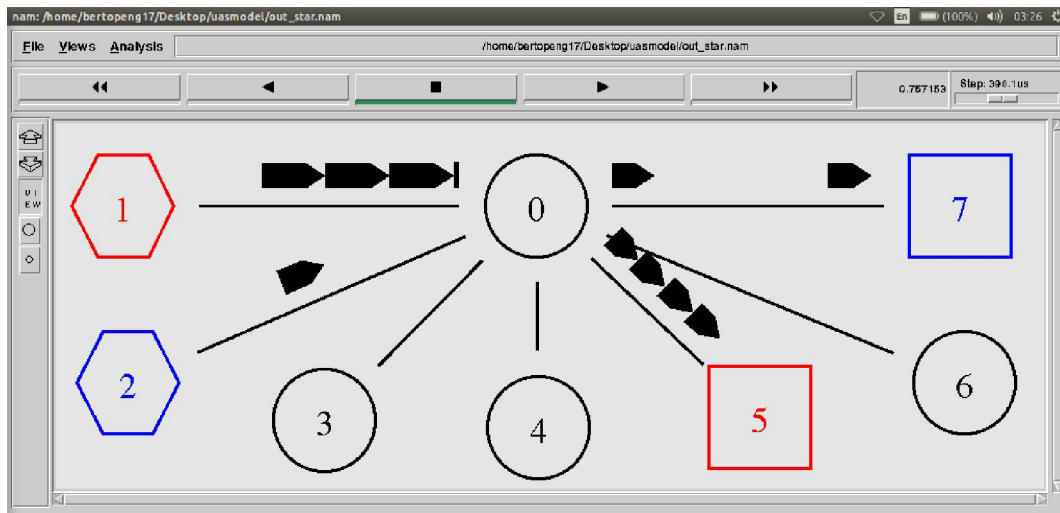
Berdasarkan hasil pengujian pertama bahwa pengujian sesuai dengan skenario sesuai pada table diatas bahwa packet FTP dari node 1 dikirim ke Node 5 dimulai pada detik ke 0.10, paket UDP yang dikirim dari node 2 ke node 7 dimulai pada detik ke 0.05, dan belum dilakukan pemutusan link antara **node 7 dan node 0**, untuk pengiriman packet UDP dan **Node 5 ke Node 0** untuk pengiriman paket FTP sehingga paket UDP dari node 2 dapat dikirimkan ke Node 7 , dan Paket FTP dari node 1 ke Node 5 dapat terkirim.

Paket FTP yang dikirimkan dari node 1 ke node 5 di route dari **node 1 langsung ke Node 5** melalui perantara node 0 , dan paket UDP yang dikirimkan ke node 7 di route dari node 2 melalui perantara node 0 sehingga dapat dilihat seperti table dibawah ini.

- Route Awal (Sebelum Link Node 2 ke Node 0 Terputus, dan link dari node 1 ke node 0 terputus

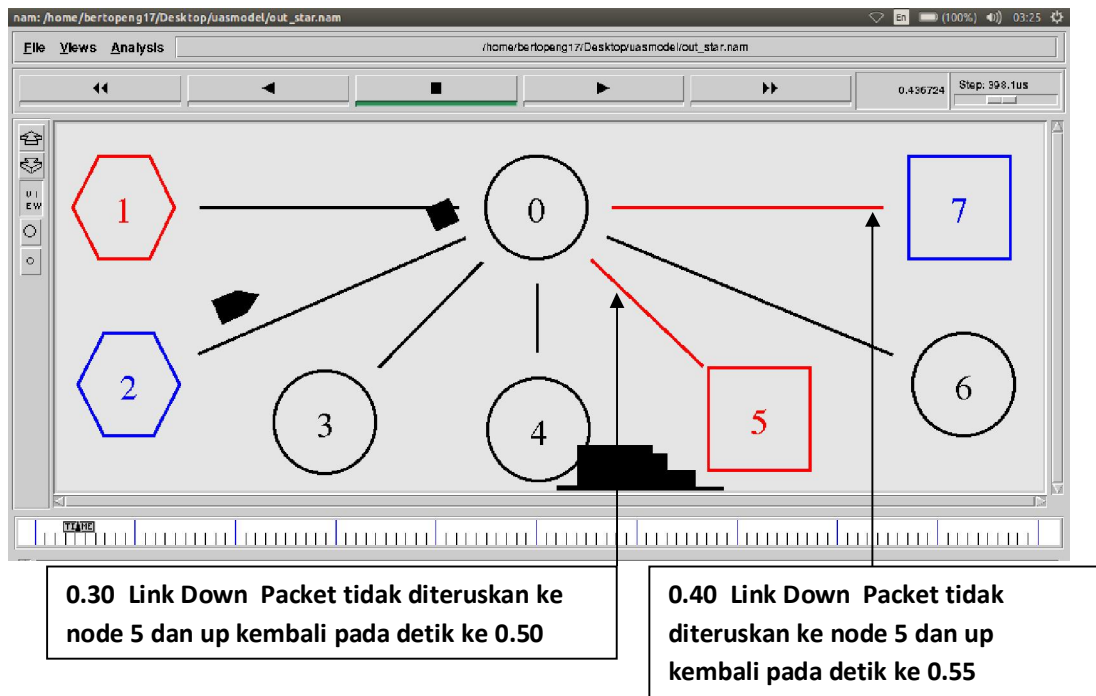
• Table 3.14 Route Awal Topologi Star

Source	Destination	Directly	Hop 1	Hop2	Hop 3	Total Hop
Node 1	Node 5	Node 0	Node 5	-	-	1
Node 2	Node 7	Node 0	Node 7	-	-	1



Gambar 3.17 Uji coba pada topology Star

Kemudian sesuai dengan skenario, penulis mencoba memutuskan link antara **node 1** dan **node 0** kemudian yang terjadi adalah packet FTP yang dikirim dari Node 1 ke Node 5 terputus dan kemudian memutuskan link antara **node 2** dan **node 0** kemudian yang terjadi adalah packet UDP yang dikirim dari Node 2 ke Node 7 terputus, berdasarkan proses pengujian dapat dilihat seperti gambar dibawah ini



Gambar 3.18 Link 5 & Link 7 Down

Karena pada topology ini node 0 adalah sebagai pusat, maka route baru tidak dapat dialihkan sehingga harus menunggu node 0 aktif kembali sampai detik ke 0.50 agar paket dapat dikirim kembali ke tujuan.

4.6 ANALISA

Pada uji coba ini setelah menjalankan simulasi pada topology star selanjutnya kita melakukan pemfilteran dimana akan didapat informasi dari file trace tersebut berupa laporan yang berisi nilai throughput, delay, konsumsi energy dan probabilitas paket sukses. Berikut data throughput, delay, konsumsi energy dan probabilitas paket sukses: adapun beberapa parameter yang akan dianalisis adalah sebagai berikut

1. Throughput

adalah bandwidth yang sebenarnya (aktual) yang diukur dengan satuan waktu tertentu dan pada kondisi jaringan tertentu yang digunakan untuk melakukan transfer file dengan ukuran tertentu. Misalnya bandwidth anda yang diketahui adalah 64 kbps, kemudian ingin mendownload file di Internet berukuran 128 kb, seharusnya file tersebut sudah sampai ke komputer hanya dengan waktu 2 detik ($128/64$), namun yang terjadi sebenarnya file tersebut tiba dalam waktu 8 detik. Jadi bandwidth yang sebenarnya atau yang disebut throughput adalah $128\text{kb}/8 \text{ detik} = 16 \text{ kbps}$. Berikut adalah faktor-faktor yang mempengaruhi bandwidth dan throughput adalah piranti jaringan, tipe data yang ditransfer, topology jaringan, banyaknya pengguna jaringan, spesifikasi komputer client/user, spesifikasi komputer server dan media transfer.

$$\text{Throughput} = \frac{\text{Besarnya Jumlah Data Dikirim}}{\text{Waktu Pengiriman}}$$

2. Packet Loss

adalah perbandingan seluruh paket IP yang hilang dengan seluruh paket IP yang dikirimkan antara pada source dan destination. Salah satu penyebab packet loss adalah antrian yang melebihi kapasitas buffer pada setiap node. Beberapa penyebab terjadinya packet loss yaitu:

1. Congestion, disebabkan terjadinya antrian yang berlebihan dalam jaringan
2. Node yang bekerja melebihi kapasitas buffer
3. Memory yang terbatas pada node
4. Policing atau kontrol terhadap jaringan untuk memastikan bahwa jumlah trafik yang mengalir sesuai dengan besarnya bandwidth. Jika besarnya trafik yang mengalir didalam jaringan melebihi dari kapasitas bandwidth yang ada maka policing control akan membuang kelebihan trafik yang ada.

$$\text{Loos Packet} = \frac{\text{Paket Dikirim} - \text{Paket yang diterima}}{\text{Waktu Pengiriman}} \times 100\%$$

3. Jitter

didefinisikan sebagai variasi dari delay atau variasi waktu kedatangan paket. Banyak hal yang dapat menyebabkan jitter, diantaranya adalah peningkatan trafik secara tiba-tiba sehingga menyebabkan penyempitan bandwith dan menimbulkan antrian. Selain itu, kecepatan terima dan kirim paket dari setiap node juga dapat menyebabkan jitter.

$$\text{Jitter} = \frac{\text{Total Variasi Delay}}{\text{Total Paket yang diterima} - 1}$$

4. Delay

didefinisikan sebagai total waktu tunda suatu paket yang diakibatkan oleh proses transmisi dari satu titik ke titik lain yang menjadi tujuannya. Delay di dalam jaringan dapat digolongkan sebagai berikut delay processing, delay packetization, delay serialization, delay jitter buffer dan delay network

```
$ awk -f [syntak file.awk] [nama file trace.tr]
```

Atau

```
$ awk -f dellay.awk out_ring.tr
```

4.6.1.1 Analisa Hasil Simulasi pada Topologi Ring

Langkah awal untuk memeriksa troughput dengan menggunakan scripting code awk Pada terminal didalam direktori tempat penyimpanan ns2 kita ketikan beris perintah

a. Throughput pada Topology Ring

```
$ awk -f [syntak file.awk] [nama file trace.tr]
```

Atau

```
$ awk -f throughput.awk out_ring.tr
```

```
Average Throughput[kbps] = -0.00
```

```
StartTime=1000000.00
```

```
bertopeng17@bertopeng17-PICO-DJH-Model: ~/Desktop/uasmodel
bertopeng17@bertopeng17-PICO-DJH-Model:~/Desktop/uasmodel$ awk -f throughput.awk out.tr
Average Throughput[kbps] = -0.00          StartTime=1000000.00  StopTime=0.00
bertopeng17@bertopeng17-PICO-DJH-Model:~/Desktop/uasmodel$
```

Membuat graphic troughput dengan cara merubah ekstensi file trace (tr) ke ekstensi file grafike dengan cara :

```
$ awk -f throughput.awk out_ring.tr > throughput_out_ring.xg
```

```
$ xgraph out_ring.xg
```

b. Jitter pada Topology Ring

Langkah awal untuk memeriksa jitter dengan menggunakan scripting code awk Pada terminal didalam direktori tempat penyimpanan ns2 kita ketikan beris perintah ;

```
$ awk -f [syntak file.awk] [nama file trace.tr]
```

Atau

```
$ awk -f jitter.awk out_ring.tr
```

```

bertopeng17@bertopeng17-PICO-DJH-Model: ~/Desktop/uasmodel
bertopeng17@bertopeng17-PICO-DJH-Model:~/Desktop/uasmodel$ awk -f jitter.awk out.tr
0.050000 0.000000
0.058000 0.000000
0.066000 0.000000
0.074000 0.000000
0.082000 0.000000
0.090000 0.000000
0.098000 0.000000
0.106000 0.005704
0.114000 -0.005704
0.122000 0.000000
0.130000 0.004744
0.138000 -0.004744
0.146000 0.006856
0.154000 0.003456
0.162000 -0.001472
0.170000 0.008384
0.178000 0.008384
0.186000 0.008384
0.194000 0.003456
0.202000 0.000992

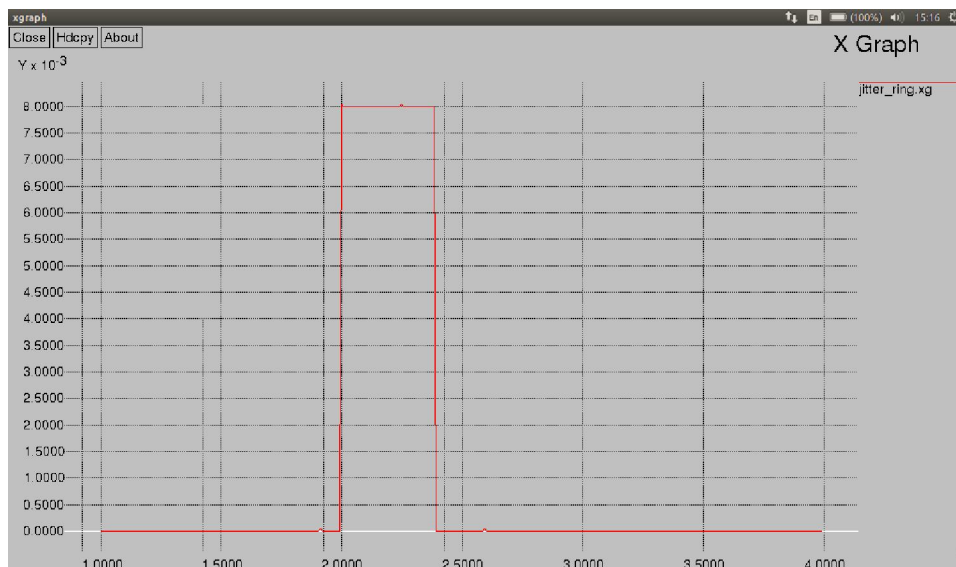
```

Gambar 3.19 *Output Jitter*

Membuat graphic jitter dengan cara merubah ekstensi file trace (tr) ke ekstensi file grafik dengan cara :

```
$ awk -f jitter.awk out_ring.tr > jitter_ring.xg
```

```
$ xgraph jitter_ring.xg
```



Gambar 3.20 *Output graph Jitter Topology Ring*

c. Memeriksa loss pada Topology Ring

Langkah awal untuk memeriksa paket loss dengan menggunakan scripting code awk Pada terminal didalam direktori tempat penyimpanan ns2 kita ketikkan beris perintah ;

```
$ awk -f [syntak file.awk] [nama file trace.tr]
```

Atau

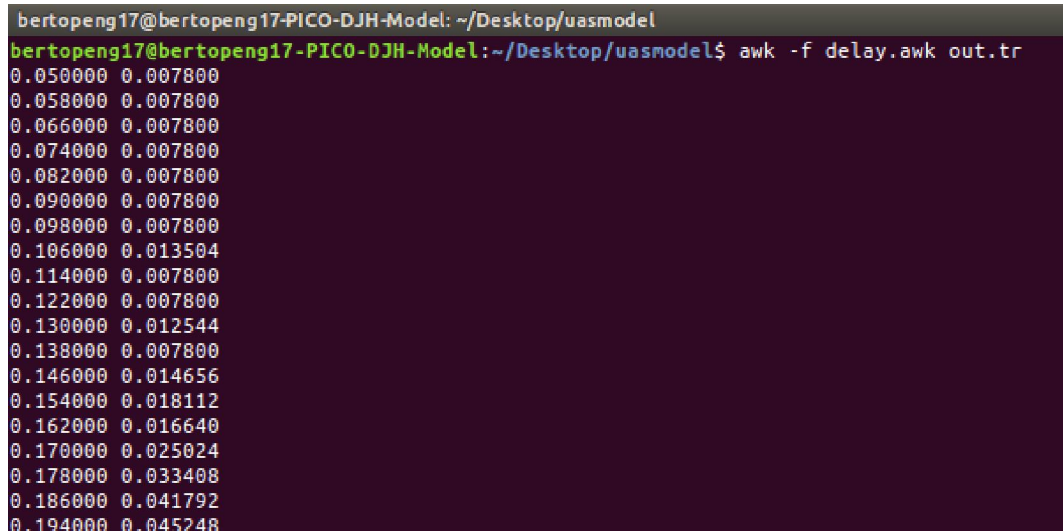
```
$ awk -f loss.awk out_ring.tr
```

d. Memeriksa Delay pada Topology Ring

```
$ awk -f [syntak file.awk] [nama file trace.tr]
```

Atau

```
$ awk -f delay.awk out_ring.tr
```



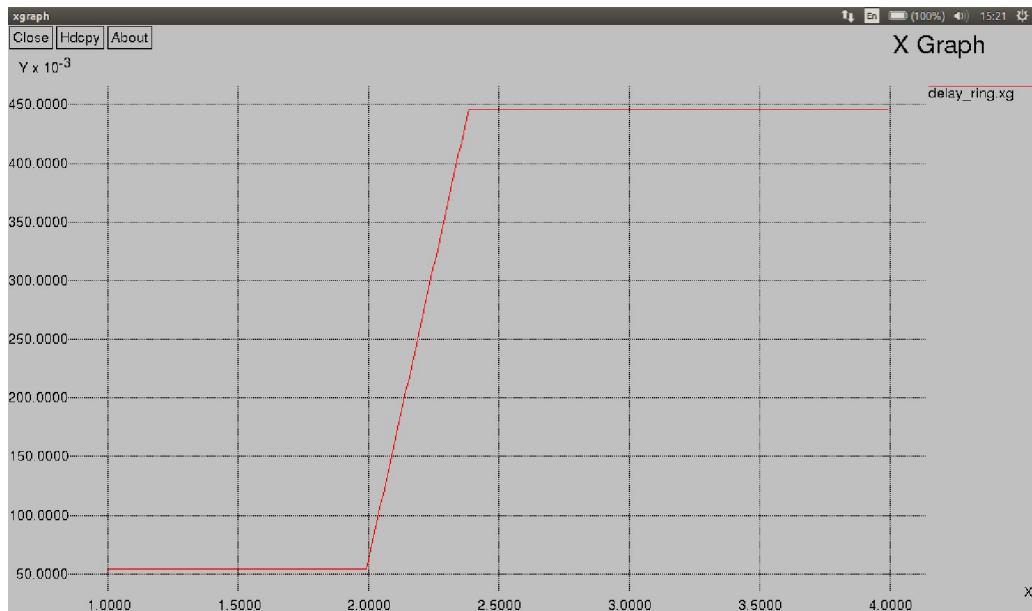
```
bertopeng17@bertopeng17-PICO-DJH-Model: ~/Desktop/uasmodel
bertopeng17@bertopeng17-PICO-DJH-Model:~/Desktop/uasmodel$ awk -f delay.awk out.tr
0.050000 0.007800
0.058000 0.007800
0.066000 0.007800
0.074000 0.007800
0.082000 0.007800
0.090000 0.007800
0.098000 0.007800
0.106000 0.013504
0.114000 0.007800
0.122000 0.007800
0.130000 0.012544
0.138000 0.007800
0.146000 0.014656
0.154000 0.018112
0.162000 0.016640
0.170000 0.025024
0.178000 0.033408
0.186000 0.041792
0.194000 0.045248
```

Gambar 3.21 *Output delay*

Membuat graphic delay dengan cara merubah ekstensi file trace (tr) ke ekstensi file grafik dengan cara :

```
$ awk -f dellay.awk out_ring.tr > dellay.xg
```

```
$ xgraph jitter.xg
```



Gambar 3.22 *Output graph delay Topology ring*

4.6.1.2 Analisa Hasil Simulasi pada Topologi Mesh

a. Throughput pada Topology Mesh

```
$ awk -f [syntak file.awk] [nama file trace.tr]
```

Atau

```
$ awk -f throughput.awk out_mesh.tr
```

```
Average Throughput[kbps] = -0.00
```

```
StartTime=1000000.00
```

```
bertopeng17@bertopeng17-PICO-DJH-Model: ~/Desktop/uasmodel
bertopeng17@bertopeng17-PICO-DJH-Model:~/Desktop/uasmodel$ awk -f throughput.awk out.tr
Average Throughput[kbps] = -0.00          StartTime=1000000.00  StopTime=0.00
bertopeng17@bertopeng17-PICO-DJH-Model:~/Desktop/uasmodel$
```

Membuat graphic troughput dengan cara merubah ekstensi file trace (tr) ke ekstensi file grafike dengan cara :

```
$ awk -f throughput.awk out_mesh.tr > troughput_out_mesh.xg
```

```
$ xgraph troughput_out_mesh.xg
```

b. Jitter pada Topology Mesh

Langkah awal untuk memeriksa jitter dengan menggunakan scripting code awk
Pada terminal didalam direktori tempat penyimpanan ns2 kita ketikan beris perintah ;

```
$ awk -f [syntak file.awk] [nama file trace.tr]
```

Atau

```
$ awk -f jitter.awk out_mesh.tr
```

```

bertopeng17@bertopeng17-PICO-DJH-Model: ~/Desktop/uasmodel
bertopeng17@bertopeng17-PICO-DJH-Model:~/Desktop/uasmodel$ awk -f jitter.awk out.tr
0.050000 0.000000
0.058000 0.000000
0.066000 0.000000
0.074000 0.000000
0.082000 0.000000
0.090000 0.000000
0.098000 0.000000
0.106000 0.005704
0.114000 -0.005704
0.122000 0.000000
0.130000 0.004744
0.138000 -0.004744
0.146000 0.006856
0.154000 0.003456
0.162000 -0.001472
0.170000 0.008384
0.178000 0.008384
0.186000 0.008384
0.194000 0.003456
0.202000 0.000992

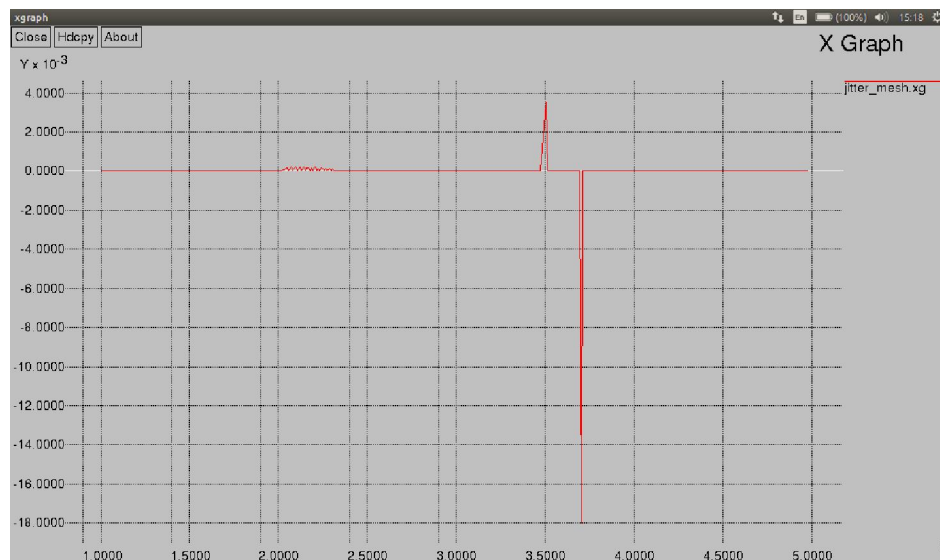
```

Gambar 3.23 *Output Jitter*

Membuat graphic jitter dengan cara merubah ekstensi file trace (tr) ke ekstensi file grafik dengan cara :

```
$ awk -f jitter.awk out_mesh.tr > jitter_mesh.xg
```

```
$ xgraph jitter_mesh.xg
```



Gambar 3.24 *Output graph Jitter topology mesh*

c. Memeriksa loss pada Topology Mesh

Langkah awal untuk memeriksa paket loss dengan menggunakan scripting code awk Pada terminal didalam direktori tempat penyimpanan ns2 kita ketikan beris perintah ;

```
$ awk -f [syntak file.awk] [nama file trace.tr]
```

Atau

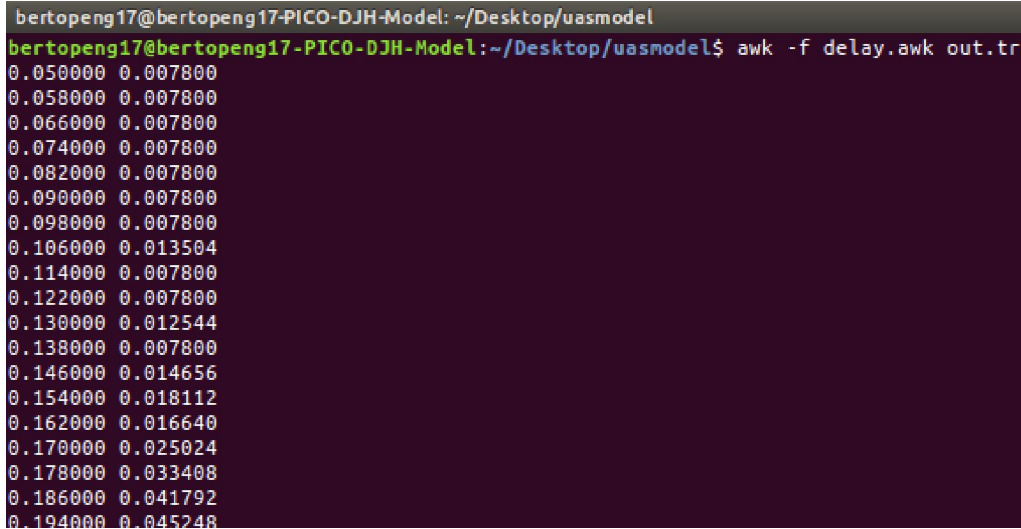
```
$ awk -f loss.awk out_Mesh.tr
```

d. Memeriksa Delay pada Topology Mesh

```
$ awk -f [syntak file.awk] [nama file trace.tr]
```

Atau

```
$ awk -f delay.awk out_ring.tr
```



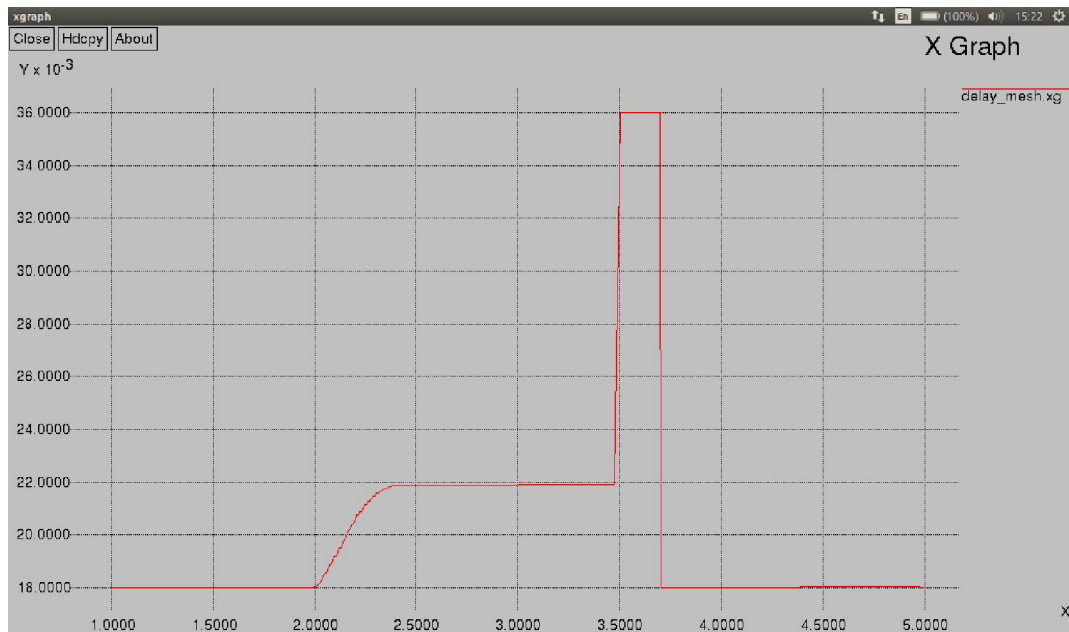
```
bertopeng17@bertopeng17-PICO-DJH-Model: ~/Desktop/uasmodel
bertopeng17@bertopeng17-PICO-DJH-Model:~/Desktop/uasmodel$ awk -f delay.awk out.tr
0.050000 0.007800
0.058000 0.007800
0.066000 0.007800
0.074000 0.007800
0.082000 0.007800
0.090000 0.007800
0.098000 0.007800
0.106000 0.013504
0.114000 0.007800
0.122000 0.007800
0.130000 0.012544
0.138000 0.007800
0.146000 0.014656
0.154000 0.018112
0.162000 0.016640
0.170000 0.025024
0.178000 0.033408
0.186000 0.041792
0.194000 0.045248
```

Gambar 3.25 *Output dellay*

Membuat graphic delay dengan cara merubah ekstensi file trace (tr) ke ekstensi file grafik dengan cara :

```
$ awk -f dellay.awk out_mesh.tr > dellay.xg
```

```
$ xgraph jitter.xg
```



Gambar 3.26 *Output graph delay topology mesh*

4.6.1.3 Analisa Hasil Simulasi pada Topologi Star

e. Throughput pada Topology Star

```
$ awk -f [syntak file.awk] [nama file trace.tr]
```

Atau

```
$ awk -f throughput.awk out_star.tr
```

```
Average Throughput[kbps] = -0.00
```

```
StartTime=1000000.00
```

```
bertopeng17@bertopeng17-PICO-DJH-Model: ~/Desktop/uasmodel
bertopeng17@bertopeng17-PICO-DJH-Model:~/Desktop/uasmodel$ awk -f throughput.awk out.tr
Average Throughput[kbps] = -0.00          StartTime=1000000.00  StopTime=0.00
bertopeng17@bertopeng17-PICO-DJH-Model:~/Desktop/uasmodel$
```

Membuat graphic troughput dengan cara merubah ekstensi file trace (tr) ke ekstensi file grafike dengan cara :

```
$ awk -f throughput.awk out_mesh.tr > troughput_out_mesh.xg
```

```
$ xgraph troughput_out_star.xg
```

f. Jitter pada Topology Star

Langkah awal untuk memeriksa jitter dengan menggunakan scripting code awk Pada terminal didalam direktori tempat penyimpanan ns2 kita ketikan beris perintah ;

```
$ awk -f [syntak file.awk] [nama file trace.tr]
```

Atau

```
$ awk -f jitter.awk out_Star.tr
```

```

bertopeng17@bertopeng17-PICO-DJH-Model: ~/Desktop/uasmodel
bertopeng17@bertopeng17-PICO-DJH-Model:~/Desktop/uasmodel$ awk -f jitter.awk out.tr
0.050000 0.000000
0.058000 0.000000
0.066000 0.000000
0.074000 0.000000
0.082000 0.000000
0.090000 0.000000
0.098000 0.000000
0.106000 0.005704
0.114000 -0.005704
0.122000 0.000000
0.130000 0.004744
0.138000 -0.004744
0.146000 0.006856
0.154000 0.003456
0.162000 -0.001472
0.170000 0.008384
0.178000 0.008384
0.186000 0.008384
0.194000 0.003456
0.202000 0.000992

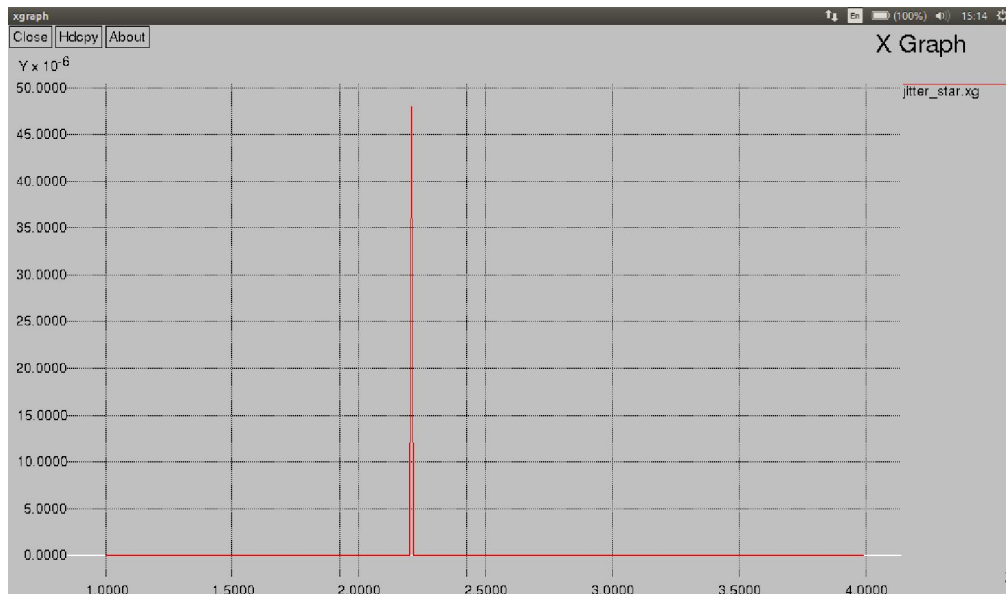
```

Gambar 3.27 *Output Jitter*

Membuat graphic jitter dengan cara merubah ekstensi file trace (tr) ke ekstensi file grafik dengan cara :

```
$ awk -f jitter.awk out_mesh.tr > jitter_mesh.xg
```

```
$ xgraph jitter_star.xg
```



Gambar 3.28 *Output graph Jitter topology mesh*

g. Memeriksa loss pada Topology Star

Langkah awal untuk memeriksa paket loss dengan menggunakan scripting code awk Pada terminal didalam direktori tempat penyimpanan ns2 kita ketikan beris perintah ;

```
$ awk -f [syntak file.awk] [nama file trace.tr]
```

Atau

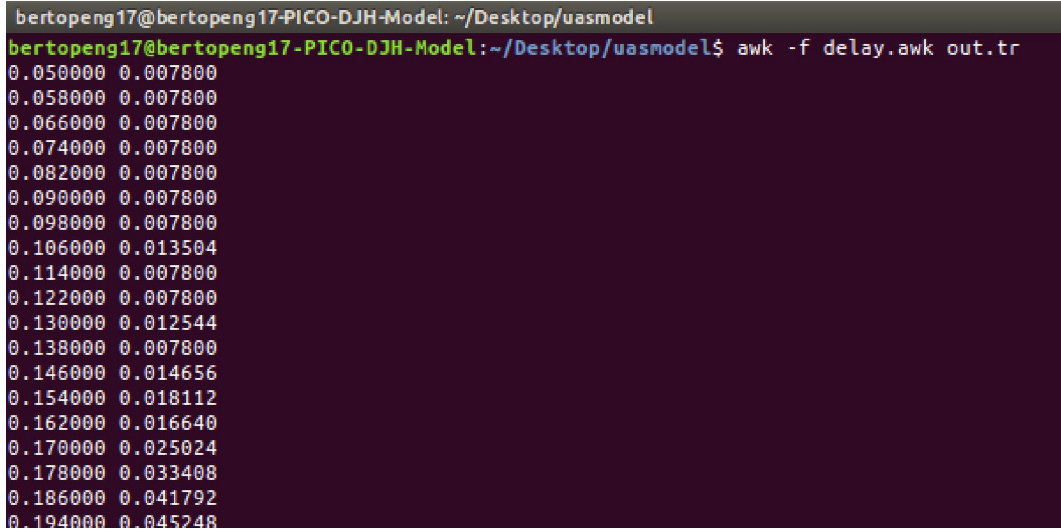
```
$ awk -f loss.awk out_star.tr
```

h. Memeriksa Delay pada Topology star

```
$ awk -f [syntak file.awk] [nama file trace.tr]
```

Atau

```
$ awk -f delay.awk out_star.tr
```



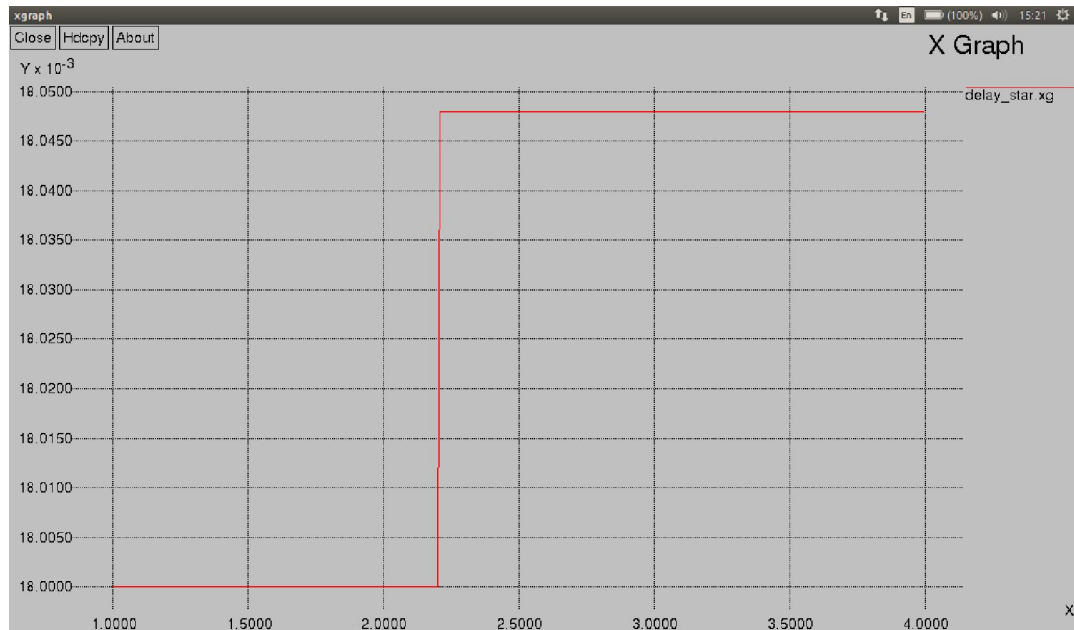
```
bertopeng17@bertopeng17-PICO-DJH-Model: ~/Desktop/uasmodel
bertopeng17@bertopeng17-PICO-DJH-Model:~/Desktop/uasmodel$ awk -f delay.awk out.tr
0.050000 0.007800
0.058000 0.007800
0.066000 0.007800
0.074000 0.007800
0.082000 0.007800
0.090000 0.007800
0.098000 0.007800
0.106000 0.013504
0.114000 0.007800
0.122000 0.007800
0.130000 0.012544
0.138000 0.007800
0.146000 0.014656
0.154000 0.018112
0.162000 0.016640
0.170000 0.025024
0.178000 0.033408
0.186000 0.041792
0.194000 0.045248
```

Gambar 3.29 *Output dellay*

Membuat graphic delay dengan cara merubah ekstensi file trace (tr) ke ekstensi file grafik dengan cara :

```
$ awk -f dellay.awk out_star.tr > dellay.xg
```

```
$ xgraph jitter.xg
```



Gambar 3.30 *Output graph delay topology star*

BAB V

KESIMPULAN

Berdasarkan hasil percobaan yang telah dilakukan pada topology star, ring dan mesh dapat disimpulkan sebagai berikut :

1. Topology star sangat terpengaruh pada node utama dimana jika node utama ini mati maka semua paket data akan loss selama perangkat utama itu mati.
2. ada topologi ring terjadi loss juga , tapi waktunya tidak lama, data kembali dikirimkan melalui rute lain setelah terjadi proses re-route, namun setelah berpindah terjadi delay dan jitter yang cukup lama , di sebabkan link yang di gunakan bercampur dengan paket data dari node sebelumnya.
3. Pada Topology Mesh walaupun node 5 di shutdown tidak berpengaruh apapun pada pengiriman paket data, namun terjadi delay dan jitter di detik ke 2.5 di sebabkan adanya proses broadcast data saat node 5 kembali hidup.

DAFTAR PUSTAKA

- https://en.wikipedia.org/wiki/Network_topology
- www.isi.edu/nsnam/ns/
- <http://nile.wpi.edu/NS/>
- <http://www.grymoire.com/Unix/Awk.html>
- <http://www.cyberciti.biz/faq/bash-scripting-using-awk/>
- <https://en.wikipedia.org/wiki/AWK>
- <http://www.tutorialspoint.com/awk/>