**SIMULASI PROTOKOL MANET DALAM KOMUNIKASI AUDIO PADA DAERAH PEGUNUNGAN MENGGUNAKAN ALGORITMA PA-SHORT**

**UNIVERSITAS UDAYANA**

**KOMPETENSI JARINGAN**

**SKRIPSI**



**I WAYAN ADI SAPUTRA**

**1608561017**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS UDAYANA**

**BUKIT JIMBARAN**

**2020**

# SURAT PERNYATAAN KEASLIAN KARYA ILMIAH

# LEMBAR PENGESAHAN TUGAS AKHIR

Judul : SIMULASI PROTOKOL MANET DALAM KOMUNIKASI

AUDIO PADA DAERAH PEGUNUNGAN MENGGUNAKAN ALGORITMA PA-SHORT

Nama : I Wayan Adi Saputra

NIM : 1608561017

Pembimbing : 1. I Komang Ari Mogi, S.Kom., M.Kom.

2. Luh Arida Ayu Rahning Putri, S.Kom., M.Cs.

ASBTRAK

Judul : SIMULASI PROTOKOL MANET DALAM KOMUNIKASI

AUDIO PADA DAERAH PEGUNUNGAN MENGGUNAKAN ALGORITMA PA-SHORT

Nama : I Wayan Adi Saputra

NIM : 1608561017

Pembimbing : 1. I Komang Ari Mogi, S.Kom., M.Kom.

2. Luh Arida Ayu Rahning Putri, S.Kom., M.Cs.

ASBTRAK

# Kata pengantar

Penelitian dengan judul SIMULASI PROTOKOL MANET DALAM KOMUNIKASI AUDIO PADA DAERAH PEGUNUNGAN MENGGUNAKAN ALGORITMA PA-SHORT ini disusun dalam rangkaian kegiatan pelaksanaan Tugas Akhir di Jurusan Ilmu Komputer FMIPA UNUD. Penelitian ini disusun dengan harapan dapat menjadi pedoman dan arahan dalam melaksanakan penelitian di atas.

Sehubungan dengan telah terselesaikannya penelitian ini, maka diucapkan terima kasih dan penghargaan kepada berbagai pihak yang telah membantu penyusun, antara lain:

1. Bapak I Komang Ari Mogi, S.Kom., M.Kom. sebagai calon Pembimbing I yang telah banyak membantu menyempurnakan penelitian ini;
2. Ibu Luh Arida Ayu Rahning Putri, S.Kom., M.Cs. sebagai calon Pembimbing II yang telah banyak membantu menyempurnakan penelitian ini;
3. Bapak-bapak dan ibu-ibu dosen di Program Studi Teknik Informatika, yang telah meluangkan waktu turut memberikan saran dan masukan dalam penyempurnaan penelitian ini;
4. Kawan-kawan di Program Studi Teknik Informatika yang telah memberikan dukungan moral dalam penyelesaian penelitian ini.

Disadari pula bahwa sudah tentu penelitian tugas akhir ini masih mengandung kelemahan dan kekurangan. Memperhatikan hal ini, maka adanya masukan dan saran – saran penyempurnaan sangat diharapkan.

Bukit Jimbaran,6 September 2019

Penulis

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# DAFTAR LAMPIRAN

# BAB I PENDAHULUAN

1. **Latar Belakang**

Indonesia merupakan negara berbentuk kepulauan yang tersusun atas beribu-ribu pulau. Secara geografis Indonesia terletak pada wilayah cincin api dunia yang menyebabkan potensi bencana yang sangat besar (Nirwansyah & Nugroho, 2015). Luas daerah Indonesia, menyebabkan Indonesia termasuk dalam daerah yang sangat rawan atas kondisi geografis (Sulistyo, 2016) terhadap bencana, terutama gempa bumi, gunung api (Supriyono, 2014), banjir dan tanah longsor (Supriyono, Citra Sulistyo, & Barchia, 2017a). Kondisi ini juga didukung oleh banyaknya daerah pegunungan di Indonesia, terdapat 30 lebih pegunungan yang terdata di Indonesia. Dengan demikian Indonesia memiliki 23 juta penduduk yang mendiam daerah pegunungan (Sumedi, 2010).

Pada bencana Gunung Agung tahun 2017, relawan yang berpartisipasi menggunakan komunikasi radio dengan memasang 3 buah unit *repeater* radio komunikasi (Aini, 2017). 28 desa terdampak ini menghandalkan komunikasi radio sehari-hari untuk mengetahui kondisi terkini Gunung Agung. Namun pada tahun 2019, komunikasi radio terancam karena izin dari pemerintah (Beritabali, 2019). Pada hasil rapat terakhir, relawan diharuskan untuk membentuk jaringan komunikasi mandiri. Dalam beberapa kasus evakuasi, terdapat permasalahan komunikasi disebabkan karena minimnya jaringan telekomunikasi (Yulistyo, 2012). Hal ini menyebabkan koordinasi antara tim evakuasi terhambat yang menyebabkan terancamnya kondisi terhadap korban bencana yang belum terevakuasi. Perkembangan pada bidang jaringan, dapat membantu pembuatan jaringan mandiri yang dapat mengakomodasi komunikasi audio, teknologi pembuatan jaringan mandiri sering disebut dengan jaringan *Ad-Hoc*.

*Voice Over Internet Protocols* (VoIP) merupakan salah satu teknologi penting yang memungkinkan kita melakukan komunikasi suara melalui koneksi internet. Komunikasi digunakan untuk berinteraksi dan menyampaikan informasi. Proses komunikasi pada masa ini, sudah tidak lagi terbatas pada jarak. Teknologi jaringan nirkabel (*wireless*) dapat membuat perangkat tersebut bergerak dan berada di dalam suatu area dimana fasilitas nirkabel tersebut dapat saling terhubung dan saling menjangkau, akan sangat mungkin membentuk sebuah jaringan yang bersifat sementara. Jaringan semacam ini disebut sebagai MANET.

*Mobile Ad-hoc Network* (MANET) merupakan salah satu teknologi jaringan komunikasi. MANET memungkinkan perangkat nirkabel memiliki kemampuan untuk berkomunikasi tanpa merujuk pada infrastruktur. Dalam MANET, setiap *node* dapat bergerak secara bebas, sehingga jaringan dapat mengalami perubahan topologi dengan cepat. Karena *node* dalam MANET memiliki jangkauan transmisi yang terbatas, *node* kemungkinan tidak dapat berkomunikasi secara langsung dengan *node* lainnya. Pada MANET jalur *routing* mengandung *hop* dan setiap *node* berfungsi sebagai *router* untuk menentukan ke arah mana tujuan atau rute yang dipilih (Murahlishankar & George, 2014).

Berdasarkan uraian diatas, akan dilakukan penelitian terkait penerapan MANET dalam komunikasi audio pada daerah pegunungan dengan optimasi protokol *routing* menggunakan algoritma *Path Aware Self-Healing and Optimizing Routing Techniques*. Tujuan dari algoritma *Path Aware* SHORT adalah mencoba untuk mempersingkat panjang rute bila memungkinkan. Rute yang lebih pendek tidak hanya mengurangi *latency*, tetapi juga diharapkan dapat meningkatkan *throughput*. Penerapan *Path Aware* SHORT diterapkan pada protokol *routing* AOMDV (*Ad-Hoc on Demand Distance Vector*). Penelitian dilakukan secara simulasi menggunakan Network Simulator 2.35. Pemodelan daerah pegunungan dilakukan dengan menggunakan SUMO (*Simulation of Urban Mobility*). Untuk komunikasi audio menggunakan modul Ns2Voip++ pada Network Simulator 2.35. Diharapkan pada penelitian ini diketahui bagaimana pengaruh jumlah *node* terhadap kualitas komunikasi *audio* di model daerah pegunungan pada MANET menggunakan protokol *routing* AOMDV dan PA-AOMDV. Pengujian kualitas komunikasi dilakukan dengan perhitungan *delay, packet loss* dan MOS E-Model.

1. **Rumusan Masalah**

Berdasarkan latar belakang diatas, maka dapat dirumuskan permasalahan sebagai berikut:

* 1. Bagaimana penerapan MANET atau Mobile Ad-hoc Network dalam komunikasi di daerah pegunungan.
  2. Bagaimana pengaruh jumlah *node* terhadap kualitas komunikasi *audio* di daerah pegunungan pada MANET menggunakan protokol *routing* AOMDV dan PA-AOMDV ditinjau dari *delay, packet loss* MOS E-Model.
  3. Pengaruh pemilihan codec?

1. **Batasan Masalah**

Beberapa Batasan masalah yang akan dijadikan acuan pada penelitian ini sebagai berikut:

1. Simulasi dilakukan terbatas hanya dengan protocol *routing* pada MANET yakni protokol (AOMDV) serta hanya menggunakan algoritma PA-SHORT (*Path Aware Self-Healing and Optimizing Routing Techniques*).
2. Penelitian akan dilakukan secara simulasi menggunakan program NS2 atau Network Simulator 2.35.
3. Parameter kinerja yang diukur dan diamati adalah berdasarkan *delay, packet loss* dan MOS dengan E-Model.
4. Model lokasi yang dibangkitkan berdasarkan wilayah Desa Sebudi , Kecamatan Selat, Kabupaten Karangasem dengan luas wilayah yang diuji 2000m X 2000m.
5. Lama waktu simulasi adalah 120 detik dengan waktu komunikasi 60 detik.
6. Model komunikasi audio yang dibangkitkan satu arah dengan pembangkit model komunikasi audio Ns2VoIP++.
7. **Tujuan Penelitian**

Berdasarkan rumusan masalah tersebut, tujuan penelitian ini adalah sebagai berikut:

1. Untuk mengetahui bagaimana penerapan MANET atau *Mobile Ad-hoc Network* dalam komunikasi *audio* di daerah pegunungan.
2. Untuk mengetahui bagaimana pengaruh jumlah *node* terhadap kualitas komunikasi *audio* di daerah pegunungan pada MANET menggunakan protokol *routing* AOMDV dan PA-AOMDV ditinjau dari *delay, packet loss* dan MOS E-Model.
3. untuk mengetahui pengaruh pemilihan codec???
4. **Manfaat Penelitian**

Beberapa manfaat yang dapat diambil dari penelitian ini adalah sebagai berikut:

1. Para pembaca diharapkan mengetahui bagaimana penerapan MANET atau *Mobile Ad-hoc Network* dalam komunikasi *audio* di daerah pegunungan.
2. Para pembaca diharapkan mengetahui bagaimana pengaruh jumlah *node* terhadap kualitas komunikasi *audio* di daerah pegunungan pada MANET menggunakan protokol *routing* AOMDV dan PA-AOMDV ditinjau dari *delay, packet loss,* dan MOS E-Model.
3. para pembaca diharapkan mengetahui pergaruh pemilihan codec??????
4. **Metodologi Penelitian**
5. **Desain Penelitian**

Pada penelitian ini menggunakan sebuah simulasi yang merupakan suatu teknik untuk meniru operasi-operasi atau proses-proses yang terjadi dalam suatu sistem dengan bantuan perangkat komputer serta dilandasi dengan beberapa asumsi-asumsi tertentu sehingga sistem tersebut dapat dipelajari secara ilmiah (W. David Kelton, 1991). Pada penelitian ini juga menggunakan metode penelitian eskperimental.

Menurut (Sugiyono, 2010) metode penelitian eksperimen didefinisikan sebagai metode penelitian yang digunakan untuk mencari pengaruh dari perlakukan tertentu terhadap sesuatu dalam kondisi yang dikendalikan. Jadi, melalui penelitian ini dapat disimpulkan bahwa penelitian ini akan mencari hubungan sebab-akibat dari suatu percobaan, dan hasil yang didapatkan akan bervariasi sesuai dengan kondisi yang ditentukan. Maka dari itulah penelitian ini menggunakan metode penelitian *experimental*.

1. **Variabel Penelitian**

Menurut pengertiannya, variable merupakan segala yang berkaitan dengan kondisi, keadaan, factor, perlakuan, atau tindakan yang diperkirakan dapat mempengaruhi hasil penelitian. Variable juga berkaitan langsung untuk mengetahui suati keadaan tertentu dan diharapkan mendapatkan dampat/akibat dari eksperimen sering disebut variable eksperimental (*treatment variable*).

Dalam penelitian ini terdapat variable bebas dan variable terikat, dimana dalam penelitian ini, variable tersebut dijabarkan sebagai berikut:

1. Variable Bebas

Merupakan variable yang mempengaruhi hasil dari variable lain jika variable ini berubah, dalam penelitian ini variabel bebas dalam penelitian ini adalah jumlah *node,* protokol *routing,* dan jenis *codec.* Variasi jumlah *node* adalah dengan menggunakan 25 *node* sebagai *node* awal, kemudian 35, 45, 55, 65 dan 75 *node.* Untuk protokol *routing* yang digunakan adalah AOMDV dan PA-AOMDV serta untuk pemilihan *codec* adalah G.711 dan G.723.1.

1. Variable terikat

Variable terikat dapat didefinisikan sebagai hasil yang didapatkan dalam penelitian yang dipengaruhi oleh variable bebas. Pada penelitian ini, variable terikat antara lain:

* 1. *Delay*
  2. *Packet Loss*
  3. MOS E-Model

1. **Skenario Tahapan Penelitian**

Gambar dibawah menunjukkan bagaimana tahapan-tahapan dalam penelitian. Pada gambar, terdapat alur yang memperlihatkan tahapan dari penelitian, penjelasan mengenai alur sebagai berikut:

1. Pengumpulan data

Tahap ini merupakan proses mengumpulkan data-data dan referensi yang mendukung serta yang dibutuhkan dalam penelitian yang akan dilakukan.

1. Instalasi *Software* Simulasi

Pada tahap ini akan dilakukan proses instalasi *software* simulasi yang digunakan dalam proses penelitian, dimana *software* simulasi yang digunakan adalah Network Simulator 2.35.

1. Memodelkan Daerah Pegunungan SUMO

Tahap ini memodelkan lokasi pegunungan melalui *openstreetmap* dan SUMO (*Simulation of Urban Mobility*) sehingga menghasilkan berkas *mobility* yang digunakan pada Network Simulator 2.35.

1. Memodelkan Protokol AOMDV dan PA-AOMDV pada Simulator

Pada tahapan ini mulai dilakukan penerapan protokol *routing* AOMDV dan PA-AOMDV pada Network Simulator 2.35.

1. Penerapan *Traffic* Komunikasi *Audio* dengan Codec G711 dan G723.1

Tahap ini akan diterapkan *traffic* komunikasi *audio* pada penelitian. Pembangkitan *traffic audio* menggunakan patch Ns2Voip++ dengan Codec G711 dan G723.1 sebagai pembanding, dan decoder H.263.

1. Pembuatan Kode Program

Pada tahapan ini merupakan tahapan pembuatan kode program untuk Protokol *Routing* PA-AOMDV, kode scenario simulasi, kode pengujian seperti *Delay* dan *Packet Loss* serta kode program perhitungan MOS E-Model.

1. Pengumpulan Data Simulasi

Tahap ini merupakan tahapan pengumpulan data-data hasil simulasi. Pengumpulan data ini akan dilakukan untuk semua scenario yang telah ditentukan.

1. Analisis

Data hasil simulasi yang telah dikumpulkan tersebut akan diamati bagaimana pengaruh variable yang diteliti dengan kualitas komunikasi yang dihasilkan. Data hasil simulasi akan dicatat dan dituangkan kedalam bentuk tabel dan grafik. Dengan membandingkan komunikasi audio menggunakan protokol *routing* AOMDV dan PA-AOMDV serta pengaruh kualitas komunikasi berdasarkan Codec G711 dan G723.1.

1. Penarikan kesimpulan dan saran

*g*Setelah dilakukan analisis data yang diperoleh dari proses simulasi, kemudian akan disimpulkan berdasarkan nilai-nilai yang diperoleh dari analisis data simulasi yang telah dilakukan pada penelitian.

Gambar . Tahapan Penelitian

1. **Skenario Pengujian**

Pada penelitian ini dilakukan beberapa scenario pengujian untuk mengetahui kualitas komunikasi *audio* dengan menggunakan protokol routing AOMDV dan PA-AOMDV serta pengaruh codec dan jumlah node yang terlibat dalam proses simulasi.

Tabelin aja dulu gan

|  |  |
| --- | --- |
| Node | 25, 35, 45, 55, 65, 75 |
| Codec | G711, G723.1 |
| Protokol Routing | AOMDV, PA-AOMDV |
| *Backgroud traffic* | ON, OFF |

# BAB II TINJAUAN PUSTAKA

Pada bagian ini, akan dipaparkan sejumlah tinjauan empiris dan tinjauan teoritis yang dijadikan acuan dalam penelitian menyenai simulasi protokol MANET dalam komunikasi audio di daerah pegunungan yang dilakukan ini.

1. **Tinjauan Empiris**

Penelitian terdahulu yang berkaitan dengan penelitian yang penulis baik dari segi metode maupun jenis penelitian yang dilakukan, akan diperjelas pada tinjauan empiris. Beberapa penelitian terdahulu yang berkaitan yaitu:

1. ***Routing Protocol for MANET: A Literature Survey*** (Muralishankar et al, 2014)

Mobile Ad-hoc Network (MANET) merupakan jaringan yang jaringan yang dapat dibuat tanpa dukungan infrastruktur jaringan. Pada jaringan MANET, node bergerak dengan cepat. Batasan transmisi pada MANET menyebabkan rentangan komunikasi pada setiap node tidak selalu dapat dilakukan secara langsung, sehingga pada MANET, setiap node dapat berperan sebagai router atau bersifat multihop.

1. **Meningkatkan Efisiensi Rute Pada Protokol Routing AOMDV Menggunakan Metode PA-SHORT di Jaringan MANET** (Wulandari, Jatmika, & Bimantoro, 2019)

Penelitian ini dilatarbelakangi oleh permasalahan routing pada jaringan MANET, dimana penemuan jalur paling efektif antara node yang terus bergerak. Maka diperlukan protokol routing yang dapat menangani dengan baik segala perubahan tersebut. Wulandari memilih menggunakan AOMDV atau *Ad-hoc On-demand Multipath Distance Vector* karena AOMDV ketika mengalami kerusakan dalam proses pencarian rute, maka akan menggunakan rute cadangan yang dimilikinya tanpa harus mencari lagi. Selain itu AOMDV menyediakan dua layanan yaitu route discovery dan route maintenance, AOMDV berkeja mencari rute dengan cara memperhitungkan jarak berdasarkan jumlah hop. Perubahan topologi dapat dimanfaatkan untuk mengurangi jumlah hop. Pengurangan jumlah hop dapat meningkatkan throughput dan menurunkan delay. Algoritma Path Aware SHORT digunakan untuk mempersingkat hop sehingga diharapkan dapat meningkatkan QoS.

Pada penelitian Wulandari dkk ini mengefisiensikan pencarian rute pada protokol AOMDV menggunakan PA-SHORT sehingga menjadi PA-AOMDV. Menggunakan algoritma Path Aware SHORT mengurangi jumlah hops pada rute dan meningkatkan optimasi dengan monitoring rute serta update mengenai rute terbaik. Pada penelitian ini menggunakan jumlah node sebanyak 50 dan 100 node pada luas area simulasi 500m2 x 500 m2 dan 1000m2 x 1000m2, waktu simulasi 600 detik dengan jenis mobilitas Random Way Point serta model propagasi Two Ray Ground. Penelitian ini menggunakan Network Simulator 2 dengan menggunakan bahasa TCL (Tool Command Language) dalam pembuatan kode AOMDV dan PA-AOMDV. Pada penelitian ini dihasilkan kesimpulan yaitu PA-AOMDV menghasilkan peningkatan sebesar 16.04% throughput pada 50 node dan 11,55% untuk 100 node dengan aera 500 x 500 m2 dan peningkatan sebesar 61,84% pada 50 node dan 45,2% untuk 100 node untuk luas area 1000 x 1000 m2. QoS delay mengalami penurunan pada PA-AOMDV serta peningkatan PDR dan penurunan routing overhead. Kinerja protokol PA-AOMDV meningkatkan optimalisasi routing dengan memantau rute secara bertahap mengalihkan jalur setiap kali short-cut ditemukan.

1. **Analisis Kualitas VoIP pada SCTP Menggunakan ECN dan AQM** (Surimi & Pulungan, 2015)

Penelitian ini dilatar belakangi oleh penggunaan protokol UDP yang tidak memiliki *congestion control* yang menyebabkan peluang terjadinya *congestion* sangat besar dalam aplikasi VoIP. Penelitian dari mengajukan modifikasi pada protokol alternative SCTP menggunakan mekanisme ECN dan AQM sehingga dapat menurunkan *delay* dan *jitter* pada proses komunikasi. Simulasi dilakukan dengan menggunakan Network Simulator 2.29. Komunikasi VoIP yang dibangkitkan pada penelitian Surimi & Pulungan ini menggunakan modul Ns2VoIP++ dan dengan bentuk topologi dan pergerakan yang statis. Komunikasi VoIP yang dibentuk menggunakan *codec* G.711 dan dengan protokol aplikasi H.323. Parameter yang diujikan dalam penelitian Surimi & Pulungan ini adalah scenario penggunaan protokol SCTP dengan dan tanpa mekanisme ECN+AVQ. Pengujian kualitas dilakukan dengan perhitungan MOS E-Model melalui nilai rentang R factor. Hasil pengujian menunjukkan kualitas komunikasi lebih baik ketika menggunakan protokol SCTP dengan mekanisme ECN+AVQ daripada SCTP tanda ECN+AVQ. Protokol komunikasi yang diajukan dapat mengungguli kualitas komunikasi protokol TCP namun lebih rendah dari kualitas komunikasi dibanding protokol UDP.

1. ***Performance Analysis of VoIP in Wireless Networks*** (Wheeb, 2017)

Penelitian ini dilatar belakangi oleh kualitas dan kapasitas dari komunikasi VoIP. Penelitian dari Whaab ini bertujuan untuk menganalisis kualitas dari komunikasi VoIP pada jaringan nirkabel dengan *transport protocol* dan *codec* yang berbeda. Protokol *transport* yang digunakan yakni UDP, SCTP dan RTP dengan *codec* G.711 dan G.726. Penelitian dilakukan dengan simulasi, simulasi dilakukan dengan menggunakan Network Simulator 2 dengan protokol jaringan DSDV, jumlah *node* 50, dengan topologi statis dan luas area simulasi 4000m2. Masing-masing *node* melakukan komunikasi 2 arah, untuk menghitung kualitas komunikasi yang terjadi, penelitian ini menggunakan parameter kinerja *throughput, end-to-end delay, jitter* dan *packet loss.*

Untuk hasil pengujian parameter kinerja *throughput* penggunaan protokol transfer SCTP menghasilkan kualitas komunikasi lebih bagus daripada UDP dan RTP pada penggunaan *codec* G.711 dan G.726, untuk parameter kinerja *end-to-end delay* protokol *transport* UDP dan RTP menghasilkan kinerja lebih baik daripada protokol *transport* SCTP, kemudian parameter kinerja *jitter* menghasilkan UDP dan RTP memiliki kualitas kinerja yang lebih baik daripada SCTP dan parameter kinerja PDR (*Packet Delivery Ratio*) menghasilkan protokol SCTP dan UDP memiliki kinerja yang lebih baik daripada protokol RTP dengan perbandingan 0.99918 berbanding 0.97183 dan 0.76132.

1. ***VOIP APPLICATIONS OVER MANET: CODEC PERFORMANCE ENHANCEMENT BY TUNING ROUTING PROTOCOL PARAMETERS*** (El Brak, Bouhorma, El Brak, & Boudhir, 2013)

Penelitian dilator belakangi dengan bagaimana MANET dapat mendukung komunikasi VoIP dengan kelebihan dan kekurangan dari jaringan MANET itu sendiri. Pendekatan penelitian yang dilakukan berkonsentrasi kepada karakteristik dari MANET yang *multi hop* dan mempertahankan kualitas komunikasi pada MANET melalui simulasi. Untuk karena itu evaluasi kinerja dari berbagai *codec* serta dampaknya terhadap kualitas komunikasi akan dianalisis pada penelitian ini. Penelitian dari El Brak & Boundhir ini juga mengajukan pengaturan parameter *routing* protokol untuk memperoleh kualitas kinerja yang lebih baik. Penelitian ini menggunakan simulasi pada Network Simulator 2, dengan protokol *routing* OLSR. Luas area simulasi dibagi menjadi dua, yakni 1000m2 dan 2000m2. Dengan jumlal *node* yang bergerak pada luas area 1000m2 10, 30 dan 50 *node* sedangkan di luas area 2000m2 dengan jumlah *node* 40, 60, dan 80 *node*. Pembangkitan pergerakan *node* menggunakan perintah setdest pada Network Simulator 2. Sedangkan pembangkitan komunikasi VoIP menggunakan modul Ns2VoIP++ yang akan menghasilkan paket dengan *transport layer* RTP/UDP. Penggunaan *codec* pada pembangkitan komunikasi VoIP Ns2VoIP++ yakni dengan *codec* G.711, G.726, G.728, G.729A dan G.723.1 *bit rate* 5.3kbps. Lama waktu simulasi adalah 200 detik dengan waktu komunikasi VoIP 60 detik. Pergerakan *node* dalam rentangan nilai 0-50Km/h selama waktu simulasi. Pengujian kualitas yang dilakukan berupa perhitungan *delay, packet loss,* MOS, *throughput,* dan NRL (*Normalized Routing Load*).

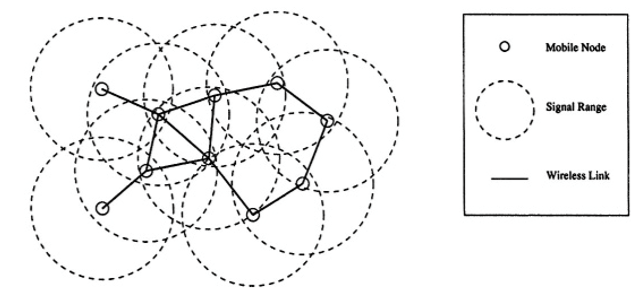
Hasil pengujian dari scenario satu dan dua menunjukkan pada parameter kinerja *delay* penggunaan *codec* G.723.1 dengan *bit rate* yang rendah memiliki *delay* yang paling sedikit diantara *codec* lainnya. Hal ini dipengaruhi oleh ukuran paket, semakin besar ukuran paket maka semakin banyak waktu yang diperlukan untuk memproses paket tersebut. Dengan *payload* yang relative kecil (20 bytes) *codec* G.723.1 memiliki *delay* paling rendah di kedua scenario. Sedangkan G.711 dengan *payload* 160 bytes memiliki *delay* yang paling tinggi. Untuk pengujian *packet loss* dipengaruhi oleh *traffic* yang padat dari *codec* serta proses *routing discovery* dan adanya *routing loops*. Ketika *node* menghabiskan waktu untuk mencari rute, paket VoIP akan terus degenerate, ketika rute belum ditemukan dan antrian paket sudah penuh, maka paket-paket tersebut akan dibuang. *Codec* dengan *bit rate* yang rendah memiliki nilai *packet loss* yang lebih sedikit, karena *traffic* yang diciptakan *codec* lebih lambat. Codec G.723.1 memiliki nilai *packet loss* yang paling rendah dibandingkan *codec* yang lainnya. Pengujian MOS yang dipengaruhi oleh *delay* dan *packet loss* menghasilkan *codec* G.723.1 sebagai *codec* dengan MOS tertinggi yakni 2.8, dimana nilai ini dikatakan sudah cukup diterima.

1. **Tinjauan Teoritis**

Beberapa teori yang dijadikan acuan pada penelitian yang penulis lakukan adalah sebagai berikut:

1. ***Mobile Ad-Hoc Network* (MANET)**

Definisi *Ad-*(Bello, 2013)*Hoc Network* adalah desentralisasi dari jaringan nirkabel*,* disebut dengan *Ad-Hoc­* karena tidak bergantung pada infrastruktur yang sudah ada seperti *router* dalam jaringan kabel maupun *access point* pada jaringan nirkabel. Setiap *node* dapat bertugas untuk menrutekan data kepada *node* lainnya, jadi penentuan *node* yang mana mengirimkan data dibuat secara dinamis berdasarkan konektivitas pada jaringan itu sendiri (Nishani & Biba, 2016). Namun tidak benar-benar dapat dikatakan bahwa MANET tidak dapat memiliki infrastruktur jaringan. Beberapa mediasi antar *node* dapat dipilih untuk bertindak sebagai *base station* dan *node* tetangga. Penerapan MANET sangat diperlukan di daerah seperti jasa penyelamatan darurat, medan perang dan tempat lainnya dimana penyebaran infrastruktur jaringan menjadi sangat sulit. Dalam MANET, setiap *mobile node* bertindak sebagai router serta *base station* untuk menemukan dan mempertahankan rut eke *mobile node* lainnnya untuk berkomunikasi melalui jaringan (Bello, 2013).



Gambar .. Mobile Ad-Hoc Network  
*Sumber: (Gui & Mohapatra, 2008)*

Pergerakan dari *node* dapat menyebabkan perubahan topologi yang tidak bisa diprediksikan, karena itu tugas untuk menemukan dan mempertahankan rute merupakan tugas penting dari protokol penemuan rute MANET. Parameter yang mempengaruhi kinerja protokol penentuan rute jaringan MANET adalah beberapa *node* di dalam area, perilaku pergerakan *node* yang mengubah konektivitas (bergerak secara bebas) dan kualitas rute. Penggunaan MANET memberikan beberapa keuntungan, diantaranya adalah (Purba, Primananda, & Amron, 2018):

1. Tidak memerlukan dukungan *backbone* infrastruktur, hal ini menyebabkan lebih mudahnya untuk diimplementasikan dan sangat berguna ketika infrastruktur tidak berfungsi bahkan tidak ada.
2. *Node* yang selalu bergerak atau dinamis dapat mengakses informasi sevara *real-time* ketika berhubungan dengan *node* lainnya. Sehingga pertukaran data dapat menjadi acuan dalam pengambilan keputusan dan segera dilakukan.
3. Fleksibel, karena jaringan bersifat sementara.
4. Dapat dikonfigurasikan dalam bermacam-macam topologi, baik dengan jumlah *node* kecil hingga besar sesuai dengan aplikasi dan instalasi.

Sedangkan kekurangan dari penggunaan MANET sendiri adalah sebagai berikut:

1. Packet loss akan terjadi bila transmisi mengalami kesalahan.
2. Memungkinkan seringnya terjadi konektivitas yang terputus karena tidak selalu berada dalam cakupan area (*network area*).
3. *Bandwidth* komunikasi terbatas.
4. *Lifetime* baterai yang singkat.
5. **Karakteristik *Mobile Ad-Hoc Network***

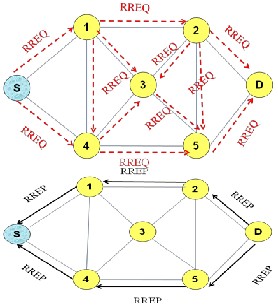
Berikut ini merupakan karakteristik dari jaringan *Mobile Ad-Hoc Network* (MANET) adalah sebagai berikut (Pathan, 2016):

1. *Mobility*: *node* dapat bergerak cepat dengan penebaran di daerah yang tdak memiliki infrastruktur. Pada MANET dapat memiliki individual random mobility, group mobility, bergerak sepanjang rute yang telah direncanakan sebelumnya dan lain sebagainya. Mobilitas memiliki dampak besar pada pemilihan skema routing dan dapat mempengaruhi kinerja.
2. *Multihopping*: jaringan multihop adalah jaringan di mana jalur dari sumber ke tujuan melintasi beberapa node lainnya. Jaring ad hoc sering menunjukkan banyak hop untuk negosiasi hambatan, penggunaan kembali spektrum, dan konservasi energi.
3. Self-organisasi: jaringan ad hoc secara mandiri dapat menentukan parameter konfigurasi sendiri termasuk: pengalamatan, routing, clustering, identifikasi posisi, kontrol daya, dan lain-lain. Dalam beberapa kasus, node khusus misalnya, mobile backbone nodes dapat mengkoordinasikan gerakan dinamis dan mendistribusikan di wilayah geografis untuk menyediakan cakupan dari daerah yang terputus.
4. ***Ad Hoc On Demand Multipath Distance Vector* (AOMDV)**

*Ad-Hoc On Demand Multipath Distance Vector* (AOMDV) merupakan protokol *routing* reaktif pengembangan dari protokol *routing* *Ad-hoc On Demand Distance Vector* (AOMDV) yang dikembangan untuk meminimalisir terjadinya kegagalan hubungan dan rute yang terputus. AOMDV dalam prosesnya menghitung beberapa jalur *loop free* dan *link disjoint path* (Humolong Bonartua Nainggolan, 2012). AOMDV menggunakan *link disjoint path* karena menggunakan perutean *multipath* untuk mengurangi *routing overhead sitasi*. Perbedaan antara AOMDV dan AOMDV adalah pada jumlah rute yang ditemukan dalam tiap kali proses pencarian rute atau *route discovery*. AOMDV dalam pencarian rute tidak seperti AOMDV yang hanya memilih satu RREQ, pada AOMDV semua RREQ yang datang akan dipertimbangan dalam upaya agar mendapatkan beberapa rute alternatif.

AOMDV memiliki konsep berbasis vector dan menggunakan pendekatan *hop by hop*. AOMDV memiliki dua fitur utama yang mirip dengan AOMDV yakni pencarian jalur/rute (*route discovery*) dan pemeliharaan jalur/rute (*route maintenance*) (Bahteradi Putra & Anggoro, 2016).

1. ***Route Discovery***

AOMDV melakukan penemuan rute (*route discovery*) hanya ketika dibutuhkan. AOMDV dalam penemuan rute (*route discovery*) memiliki perbedaan dengan AOMDV, AOMDV akan mempertimbangkan setiap RREP oleh *node* sumber, sehingga akan menghasilkan lebih dari satu temuan rute pada proses penemuan rute (*route discovery*), sedangkan AOMDV hanya akan menerima satu RREP. Adanya lebih dari satu rute yang menuju ke *node* tujuan menyebabkan ketika terjadinya kegagalan disuatu rute, maka AOMDV akan menggunakan rute yang lainnya, sehingga tidak perlu mengulangi proses pencarian rute (*route discovery*) (Wulandari et al., 2019). Penemuan rute hanya akan dilakukan apabila semua rute yang telah ditemukan sebelumnya mengalami kegagalan. Proses penemuan rute pada protokol *routing* AOMDV ditujukkan pada Gambar XX.

Gambar . Proses pencarian rute protokol routing AOMDV

Sumber: (Wulandari et al., 2019)

Berikut merupakan langkah-langkah dari protokol *routing* AOMDV dalam penemuan rute (*route discovery*):

* + 1. Ketika *node* sumber akan melakukan komunikasi dengan *node* tujuan, maka *node* sumber akan mulai melakukan *flooding* paket *route request* (RREQ) pada *node-node* tetangganya di jaringan.
    2. Ketika *flooding* RREQ dilakukan, maka sebuah *node* akan memungkinkan menerima lebih dari satu RREQ. Jika pada protokol *routing* AOMDV, hanya RREQ yang pertama yang digunakan dalam pembuatan *reverse path*, lain halnya pada protokol *routing* AOMDV.
    3. AOMDV akan mempertimbangkan semua RREQ yang tersedia untuk pembentukan *reverse path* ke *node* sumber. Beberapa RREQ yang dipertimbangkan akan menghasilkan beberapa *reverse path* yang berbeda. RREQ yang dipertimbangkan adalah RREQ yang dapat mempertahankan *loop-free* dan *disjointness* dari *node* sumber.
    4. Ketika *intermediate node* menerima *reverse path* melalui salinan RREQ, *node*  akan melakukan pengecekan terhadap apakah ada satu atau lebih *forward path* ke *node* tujuan yang valid, jika ada *node intermediate* ini akan membuat paket RREP dan mengirim kembali melalui *reverse path* ke *node* sumber.
    5. Saat *node* tujuan menerima paket RREQ, *node* tersebut juga akan membuat *reverse path* seperti cara yang dilakukan *intermetiate node* pada langkah sebelumnya.

1. ***Route Maintenance***

Merupakan proses yang bergantung pada perubahan topologi. Untuk menjaga rute, setiap *node* akan mencoba untuk mendeteksi kesalahan koneksi atau kegagalan rute secara terus menerus. Ketika ada *node* yang keluar dari jangkauan transmisi, atau beberapa peristiwa yang membatasi komunikasi antar *node* pada jaringan. Pemeliharaan rute (*route maintenance*) pada protokol *routing* AOMDV merupakan pengembangan sederhana dari pemeliharaan rute (*route maintenance*) protokol *routing* AOMDV. Sama dengan AOMDV, AOMDV juga menggunakan pesan *route error* (RRER), ketika sebuah *node* mendeteksi adanya kesalahan koneksi atau menerima data paket yang tidak memiliki rute tujuan. *Node* tersebut akan membuat paket *route error* (RRER) untuk memberitahukan ke *node* yang lain apabila rut eke *node* tujuan rusak. AOMDV memiliki beberapa rute untuk menjaga agar komunikasi dapat tetap berlangsung yaitu ketika sebuah *node* menemui *link* atau rute yang rusak, maka *node* tersebut akan memilih rute cadangan.

Proses pengiriman paket RRER pada AOMDV ditujukkan pada Gambar xx. Pada saat terjadinya kerusakan rute yang menghubungkan *node* I dan *node* D, maka *node* I akan mengirimkan paket RRER ke *node* H. seterusnya, *node* H akan mengirimkan paket RRER ke *node* S. Setelah *node* S menerima paket RRER, maka komunikasi akan dilakukan kembali dengan rute cadangan apabila masih tersedia. Apabila rute cadangan tidak tersedia, maka *node* S akan melakukan proses pencarian rute baru (*route discovery*).

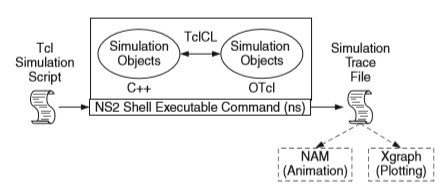
Gambar . Proses perbaikan rute (route maintenance) protokol routing AOMDV

Sumber: (Bahteradi Putra & Anggoro, 2016)

1. ***Network Simulator 2 (NS2)***

Network Simulator 2 (versi 2.35) merupakan sebuah alat simulasi event-driven yang berguna untuk mempelajari sifat dinamis dari komunikasi pada jaringan. Simulasi wired maupun wireless serta protokol (misalnya algoritma routing, TCP/UDP) dapat dilakukan menggunakan NS2. NS2 memberikan kebebasan untuk menentukan jenis komunikasi, protokol jaringan dan simulasi yang sesuai.

NS2 dapat dijalankan oleh pengguna dengan mengekseskusi “ns” yang mengambil satu argument input, nama dari file Tcl. Dalam kebanyakan kasus, simulasi file Tcl dibuat dan digunakan untuk memplot grafik dan animasi dari komunikasi jaringan.

NS2 menggunakan dua bahasa utama yakni C++ dan Object Tool Command Language (OTCL). C++ berfungsi untuk mendefinisikan backend dari simulasi, OTCL membentuk simulasi dengan membuat dan mengkonfigurasi objek serta penjadwalan diskrit dari event yakni antarmuka. C++ dan OTCL dihubungkan menggunakan TclCL. C++ dan OTCL dipetakan ke objek C++ dan variable dalam domain OTCL yang disebut dengan handles. Handles merupakan sebuah string (misalnya “\_o10”) dalam domain OTCL dan tidak mengandung fungsi apapun. Sedangkan pada C++ fungsi untuk menerima paket didefinisikan dengan class “connector”. Pada domain OTCL handle berfungsi sebagai antarmuka yang berinteraksi dengan pengguna dan objek OTCL lainnya.

Gambar .. Arsitektur Network Simulation 2.

Sumber: (Issariyakul, Teerawat & Introduction, 2013)

Dalam NS2 dapat dibangun banyak class dengan C++. Class pada C++ ini dapat digunakan untuk mengatur simulasi melalui script simulasi Tcl. Setelah simulasi melalui NS2, output yang dihasilkan berbasis teks, untuk merepresentasikan hasil grafis dan interaktif, digunakan alat seperi NAM (Network Animator) dan XGraph (Issariyakul, Teerawat & Introduction, 2013).

1. **Komunikasi *Audio***

Merupakan komunikasi yang terjadi antar sistem dalam bentuk suara, yang ditransmisikan kedalam bentuk jaringan. Komunikasi bisa terjadi dalam 2 perangkat atau lebih, namun biasanya terjadi antara 2 perangkat yang saling terhubung oleh jaringan. Dalam teknologi komunikasi, komunikasi audio merupakan satu hal yang sangat penting, karena pada saat ini komunikasi audio dianggap komunikasi yang paling praktis. Hal ini menyebabkan hadirnya teknologi pemrosesan sinyal digital yang berbasis teknologi IP (Internet Protocol) yang diintegrasikan antara komunikasi data dan suara. Salah satu teknologi komunikasi suara yang berjalan pada teknologi IP adalah VoIP.

VoIP menjadikan internet (IP) sebagai media untuk bisa melakukan komunikasi suara jarak jauh secara langsung. Dimana sinyal suara analog diubah menjadi sinyal digital dan dikirimkan melalui jaringan berupa paket-paket data secara *real time.* Komponen dalam komunikasi VoIP meliputi standar kompresi atau codec, protokol jaringan VoIP dan besaran paket dari VoIP (Dudi, 2016).

1. **Standar Kompresi Suara**

Pengkodean sinyal suara merupakan teknik yang menjelaskan bagaimana aliran sinyal analog atau sinyal suara didigitalisasi dan dikompresi menjadi bentuk sinyal *digital,* sinyal tersebut kemudian dikompresi sehingga didapat ukuran yang lebih padat. Proses pengkodean ini biasa dikenal dengan nama *codec.* Beberapa *codec* telah distandarisasi oleh ITU-T seperti G.711, G.723 dan G.729. Setiap *codec* memiliki metode kompresi, waktu tunda untuk *code* dan *decode* serta *bitrate* yang berbeda-beda. Pemilihan *codec* yang tepat dapat mempengaruhi kualitas layanan komunikasi secara keseluruhan. *Codec* dengan *bitrate* yang besar akan menghasilkan kualitas suara yang lebih baik disbanding *codec* dengan *bitrate* yang lebih rendah, namun *codec* dengan *bitrate* yang tinggi akan membutuhkan kapasitas jaringan yang lebih besar. Maka pemilihan *codec* juga akan berpengaruh kepada kebutuhan *bandwidth* pada jaringan.

1. **Codec Standar ITU-T**

ITU-T (*International Telecommunication Union*) merupakan sebuah organisasi internasional yang bertugas untuk membuat dan membakukan standar dalam bidang telekomunikasi. ITU-T merekomendasikan penggunaan *codec* dalam komunikasi *audio* melalui jaringan internet menjadi *codec* G.711, G.723 dan G.729 merupakan *codec* standar dan yang paling sering digunakan dalam komunikasi *audio.* Pada penelitian ini *codec* yang digunakan adalah G.711 dan G.723.1 sebagai parameter scenario.

* + 1. *Codec* G.711

Merupakan standar yang dikeluarkan oleh ITU-T. *Codec* ini menggunakan *encoding* PCM (*Pulse Code Modulation*)dengan menggunakan *sample rate* 8000 sample/detik. Dimana dalam setiap detik data suara akan disampling sebesar 8000 sample/detik, dengan representasi 8 *bit* setiap samplenya. Sehingga besaran *bitrate codec* ini adalah 64 kbps. Dalam melakukan pemaketan data suara, *codec* G.711 akan memaketkan data suara setiap 10 ms, yang artinya dalam melakukan sekali pemaketan dengan besaran *bitrate* 64 kbps maka data suara yang dipaketkan setiap 10 ms adalah sebesar 80 *bytes. Codec* G.711 merupakan *codec* yang memiliki *bitrate* paling tinggi (Dudi, 2016).

* + 1. *Codec* G.723.1

Pengkodean sinyal suara G.723.1 merupakan jenis pengkodean data suara yang direkomendasikan untuk terminal *multimedia* dengan *bit rate* rendah. G.723.1 memiliki *dual rate speech coder* yang dapat menswitch pada batas 5.3 kbps dan 6.3 kbps. Dengan memiliki *dual rate speech coder* maka G.723.1 dilengkapi dengan fasilitas memperbagus sinyal suara hasil pemaketan. Pada bagian *encoder* dilengkapi dengan *formant perceptual weighting filter* dan *harmonic noise sharping filter* sementara dibagian *decoder*nya memiliki *pitch postfilter* dan *formant postfilter* sehingga suara hasil rekonstruksi sangat mirip dengan suara aslinya. Sinyal eksitasi untuk *bit rate* rendah dikodekan dengan *Algebraic Code Excited Linier Prediction* (ACELP) sedangkan *bit rate* tinggi dikodekan dengan menggunakan *Multipulse Maximum Likelihood Quantization* (MP-MLQ). *Rate* yang lebih tinggi menghasilkan kualitas suara yang lebih baik dan *bit rate* yang lebih rendah memberikan kualitas suara baik dan fleksibilitas. *Codec* G.723.1 dengan *bit rate* 6.3 memiliki ukuran paket sampel 24 bytes dan ukuran payload 24 bytes.

1. **Protokol Jaringan VoIP**

Berdasarkan fungsinya, protokol jaringan VoIP dibagi menjadi dua, yakni persinyalan dan media transfer (Dudi, 2016). Protokol persinyalan digunakan untuk membangun dan menjaga suatu sesi komunikasi yang sedang berlangsung serta memutus suatu koneksi. Terdapat beberapa protokol persinyalan pada komunikasi *audio* yakni H.323, SIP, SCCP, MGCP, MEGACO dan SIGTRAN. Tetapi yang paling umum digunakan adalah H.323 dan SIP. H.323 dikembangkan oleh ITU-T (*International Telecommunication Union*) sedangkan SIP dikembangkan oleh IETF (*Internet Engineering Task Force*). Dalam penelitian tugas akhir ini, protokol yang digunakan adalah H.323.

Protokol H.323 merupakan standar ITU-T yang digunakan mendefunisikan komunikasi *multimedia real time* dan konferensi melalui jaringan *packet-based* yang tidak menyediakan *Quaranteed QoS* seperti jaringan LAN (Dudi, 2016). Standar ini merupakan standar yang terdiri dari beberapa komponen dalam membangun layanan komunikasi *multimedia* yang menerangkan *set voice, video* dan konferensi data, diantaranya:

1. Terminal: menyediakan komunikasi dua arah secara *real time*
2. Gateway: berfungsi menyambungkan terminal H.323 dengan terminal non-H.323
3. Gatekeeper: menyediakan layanan *call control,* bekerja sama dengan terminal, *Multi Point Control Unit, gateway* atau *multi point controller.* Sebagai mekanisme standar yang terintegrasi dalam protokol H.323, *gatekeeper* menyediakan fungsi *authentication, authorization, accounting, address translation call control* dan *call routing.*
4. *Multi Point Control Unit*:semua terminal yang akan melakukan komunikasi dengan terminal secara konferensi harus melakukan koneksi ke MCU. MCU melakukan *conference resource,* negosiasi antar terminal dalam menentukan codec yang digunakan serta menangani *media stream.*

*Media transfer* berfungsi untuk mengatur komunikasi saat *transfer data* baik *voice, video* maupun data secara *real time* agar berlangsung dengan baik. Tiap paket VoIP terdiri atas dua bagian, yakni *header dan payload* (beban). *Header* ini terdiri atas *IP Header, Real time Transport Protocol (RTP) header, User Datagram Protocol (UDP) header* dan *link header.*

*RTP Header* adalah *header* yang dimanfaatkan untuk melakukan segmentasi data *real time.* Sama seperti UDP, RTP tidak menjamin realibilitas paket hingga sampai di tujuan. RTP menggunakan RTCP (*Real Time Control Protocol*) yang mengendalikan QoS dan sinkronisasi *media stream* yang berbeda. UDP *Header* memiliki ciri akan mengirimkan paket ke tujuan secepat-cepatnya namun tidak menjamin realibilitas dari paket tersebut sampai di tujuan. UDP cocok digunakan pada aplikasi komunikasi *audio* yang sangat peka terhadap *delay* dan *latency.* IP *Header* bertugas menyimpan informasi *routing* untuk mengirimkan paket-paket sesuai rute hingga tujuan.

1. **Besaran Paket Data Suara**

Dalam proses pemaketan suara pada komunikasi VoIP, menggunakan besaran *bit rate.* Besaran *bit rate* tergantung dari *codec* yang digunakan. Setelah dilakukan pemaketan data suara, data suara tersebut akan melalui beberapa protokol yakni *IP Header, UDP Header* serta *RTP Header.* *Voice Payload* merupakan data suara yang sudah dipaketkan, selanjutnya paket akan melalui protokol *RTP Header RTP Header* dilakukan penyusunan paket suara yang memiliki besaran data proses 12 byte, selanjutnya paket akan melalui protokol *UDP*, protokol ini digunakan untuk jenis aplikasi yang memerlukan kecepatan pengiriman data, besaran *header* dari *UDP* adalah 8 byte. Selanjutnya paket melalui *IP Header* akan dilakukan klasifikasi jenis aplikasi yang sesuai dari layanan tersebut. Besaran proses data pada protokol ini adalah 20 byte. Berikut merupakan susuan *header* dari komunikasi VoIP.

|  |  |  |  |
| --- | --- | --- | --- |
| IP Header | UDP Header | RTP Header | Voice Payload |
| 20 Byte | 8 Byte | 12 Byte | x Byte |

Sumber gambar: (Dudi, 2016).

1. **Daerah Pegunungan**

Merupakan daerah dataran yang menjulang lebih tinggi dari sekelilingnya. Dalam pengertian lain, pegunungan adalah perbukitan yang berketinggian antara 500 m hingga 600 m dari permukaan laut (Sahab, 2017).

1. ***Simulation of Urban Mobility (SUMO)***

SUMO merupakan simulator lalu lintas yang bersifat *open source* dikembangkan di *Institute of Transportation System* Jerman. SUMO adalah simulasi lalu lintas *amicroscopic* yang berarti setiap kendaraan dan pedestrian dimodelkan secara eksplisit (Michael Behrisch, Krajzewicz, & Weber, 2013). Fitur utama dari SUMO termasuk pergerakan kendaraan bebas tabrakann, perbedaan tipe kendaraan, multi jalur dan lain-lain. Dengan mengkombinasikan SUMO dengan *Open Street Map* (OSM) dapat dilakukan simulasi dengan lokasi yang diinginkan sesuai keadaan di dunia nyata. Karena SUMO merupakan murni *traffic generator*, jalur yang dihasilkan ini tidak dapat langsung digunakan pada simulator jaringan seperti NS2 karena memerlukan representasi *road network* (Pradana, Negara, & Dewanta, 2017).

Untuk menghasilkan *road network* atau jalur di SUMO, menggunakan tools pada SUMO seperti *netconvert, plyconvert,* dan *randomTrips.py*. Berikut ini penjelasan dari masing-masing *tools* yang digunakan pada penelitian (Deshmukh & Dorle, 2016):

|  |  |
| --- | --- |
| Tools | Fungsi |
| Netconvert | Ekstrak data jaringan dari map OSM dan menghasilkan jaringan jalan yang digunakan di SUMO. Fungsi ini mengidentifikasi *node*, persimpangan dan lainnya. Juga digunakan untuk membangun file *network* pada SUMO |
| Polyconvert | Impor bentuk geometris (polygon - bangunan) dari berbagai sumber dan mengonversinya menjadi representasi yang divisualisasikan dalam SUMO-GUI. |
| randomTrips.py | Digunakan untuk membangkitkan *node*, termasuk posisi dan pergerakan *node* pada *edges.* |
| traceExporter.py | Digunakan untuk menghasilkan *file* yang dapat digunakan diluar ekosistem SUMO. Beberapa opsi memberikan pilihan untuk menghasilkan *file* pergerakan untuk Network Simulator 2/3. |

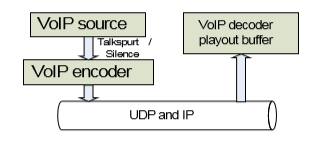
1. ***Open Street Map* (OSM)**

*Open Street Map* (OSM)adalah proyek *open source* untuk membuat contoh peta dunia yang dapat diubah secara bebas oleh siapa saja. Dua factor pendukung dalam OSM untuk digunakan dan dikembangan adalah kurangnya ketersediaan informasi peta pada sebagian besar wilayah didunia dan gabungan dari alat navigasi yang terjangkau. OSM adalah contoh utama dari informasi geografis yang ditetapkan secara bebas. Yang paling penting adalah peta OSM dapat disimpan di internet dan siapa saja dapat mengkases peta kapan saja secara gratis (Dian Khumara, Fauziyyah, & Kristalina, 2019).

OSM digambarkan sebagai “*Map of Wikipedia*” yang tidak dapat dipisahkan dari ketersediaan mekanisme dimana sukarelawan atau siapa pun dapat berkontribusi secara langsung untuk mengubah atau memperbarui data geografis untuk membuat peta yang lebih akurat, terperinci, dan terkini. Karena keterbukaan akses ini, peta OSM dapat digunakan sebagai pemodelan lokasi pada proses penelitian. Pemodelan lokasi dilakukan dengan mengekstrak peta OSM menghasilkan file berformat osm, kemudian diproses menggunakan fungsi dari SUMO, sehingga menghasilkan file input yang akan menjadi acuan lokasi pada penelitian.

1. ***NS2Voip++***

NS2Voip++ merupakan modul pembentukan *traffic* pada Network Simulator 2.35. Modul ini dikembangkan oleh University of Pisa di Italy menyempurnakan modul sebelumnya, yakni Ns2Voip (Andreozzi, Migliorini, Stea, & Vallati, 2010). NS2Voip++ dikembangkan dalam bahasa C++, NS2Voip++ dapat melakukan pembentukan *traffic* komunikasi *audio* dengan jenis codec yang berbeda dan model VAD (Voice Activity Detection). Jenis codec yang tersedia pada NS2Voip++ meliputi G.711, G.723.1, G.729A dan GSM.AMR. VAD (Voice Activity Detection) pada NS2Voip++ berupa *one-to-one, exponential,* dan *Weibull-custom.* VAD (Voice Activity Detection) meliputi periode *talk* dan *silence*.

*Voip Source* merupakan model penggunaan komunikasi *audio*. *Voip Source* meliputi pengaturan periode *talkspurt* dan *silence.* Ketika proses *talkspurt* *encoder* akan menghasilkan data sesuai dengan codec yang dipilih meliputi besaran paket dan frekuensi pembuatan paket. Di sisi penerima, *decoder* VoIP menerima paket dari jaringan dan menggunakan *buffer playout* untuk mempercepat *playout* mereka (Andreozzi et al., 2010).

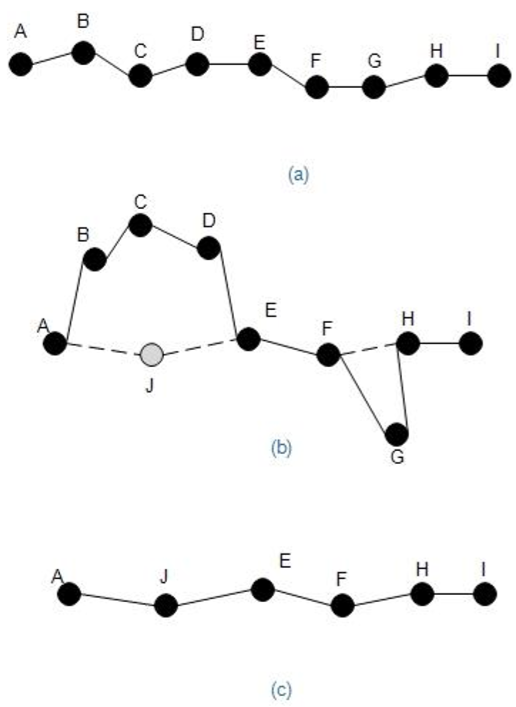
Gambar . Arsitektur modul Ns2Voip++

Sumber: (Andreozzi et al., 2010)

Pada versi terbaru NS2Voip++, pengembang menambahkan *buffer playout* yang lebih realistis dan terkenal, yakni H323 dan eEM (Causal) (Andreozzi et al., 2010). H323 dikenal sebagai *buffer playout* dinamis dan digunakan dalam berbagai perangkat lunak yang terkenal seperti *open source* VoIP Client Ekiga. Dengan mengimport kode asli codec H323 dari aplikasi Ekiga ke dalam modul NS2Voip++.

1. ***Path Aware-SHORT***

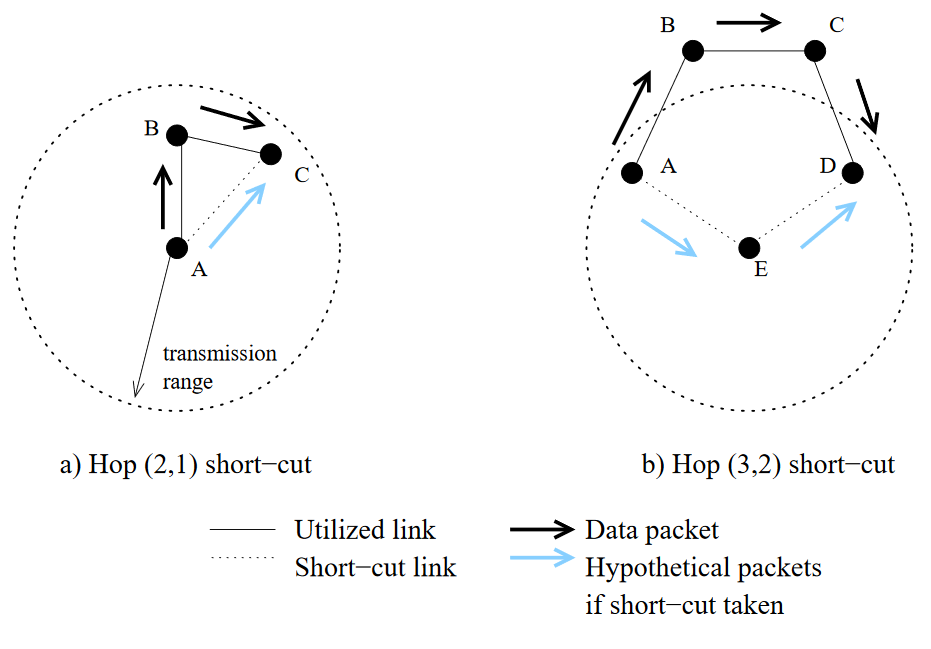
Bentuk jalur rute dapat berubah secara signifikan ketika konektivitas utuh karena pergerakan dari *node*. Perubahan bentuk ini dapat menimbulkan jalur yang lebih baik ataupun sebaliknya. Perubahan bentuk dapat dieksploitasi untuk mendapatkan jalur-jalur pengaturan rute yang lebih baik jika dapat menghindari setiap *overhead* yang signifikan (setidaknya menghindari proses-proses penemuan rute ekstra).

Gambar 6.7 menunjukkan perubahan rute, Gambar xx (a) menunjukkan jalur awal yang ditentukan melalui proses penemuan jalur, dimana jarak antara sumber dan tujuan adalah jarak terpendek menurut jumlah hop, maka sesuai jalur awal, node A membutuhkan 8 hop untuk menuju ke node I. MANET selama perjalanan waktu, mobilitas node dapat membuat jalur rute menjadi berubah, perubahan jalur diperlihatkan pada Gambar xx (b) dengan bentuk baru, node J ada dalam tingkatan transmisi node F, idealnya, jalur terpendek dari A ke node I hanya membutuhkan hop seperti pada Gambar xx (c).

Gambar . Proses perubahan topologi *node* pada jaringan

Sumber: (Fahriani, Djanali, & Shiddiqi, 2012)

Tujuan utama dari penggunaan PA-SHORT adalah untuk menemukan rute pintas ketika diperlukan atau ketika terjadi perubaha topologi. Scenario dasar dari proses penemuan rute pintas ditujukkan pada Gambar xx. Pada Gambar xx (a) , jaluar A-B-C dapat direduksi menjadi A-C karena C berada dalam jangkauan transmisi *node* A. Pembentukkan jalur pintas tersebut disebut reduksi (2,1). Gambar xx (b) menunjukkan bahwa jalur perutean A-B-C-D dapat disingkat menjadi A-E-D karena E berada dalam jangkauan transmisi A, dan D berada dalam jangkauan transmisi E. Pembentukkan jalur pintas ini disebut reduksi (3,2). Dengan demikian, reduksi (n,2) menyiratkan bahwa n *hop* disepanjang penemuan rute dapat dikurangi hanya menjadi 2 *hop*. Secara umum reduksi (n,k) menyiratkan bahwa n *hop* dapat direduksi menjadi k *hop*, dimana k<n. Untuk menghindari kompleksitas, maka nilai k menjadi 1 atau 2 pada algoritma PA-SHORT. Semakin tinggi perbedaan antara n dan k, maka semakin baik kinerja dari PA-SHORT (Gui & Mohapatra, 2008).

 Sumber: (Gui & Mohapatra, 2003)

Gambar . Penemuan rute *short-cut*

1. **MOS E-Model**

*Mean Opinion Score* (MOS) merupakan satuan kualitas suara yang biasanya digunakan untuk mengukur kualitas suara yang dihasilkan dari proses komunikasi dalam jaringan. Terdapat 2 standar yang digunakan dalam menentukan kualitas yakni ITU-T p.800 dan E-Model ITU-T G.107. Pada standar ITU-T P.800 kualitas ditentukan secara subjektif menggunakan pendapat orang-perorang yang pengujiannya dapat menggunakan *conversation opinion test* dan *listening test*. Metode ini dirasa kurang efektif untuk menentukan kualitas suara untuk komunikasi audio karena ketidaktepanan nilai dan standar orang yang berbeda-beda (Surya, 2011).

Kemudian ITU menciptakan standar pengukuran, sehingga MOS dapat dilakukan sebagai pengujian yang bersifat objektif dengan menggunakan E-Model yang menghasilkan nilai R factor. Standar baru tersebut tertuang pada ITU-T.G.107. E-Model memetakan metrik jaringan ke estimasi nilai MOS. E-Model menghitung nilai R factor berkisar 0 hingga 100. Variable yang dipertimbangkan dalam pengujian khususnya komunikasi audio adalah Id dan Ie (yang scn2 aja pakai). R factor diitung dengan rumusan sebagai berikut:

Variabel R pada persamaan X merupakan factor kualitas transmisi. Id merupakan factor penurunan kualitas yang disebabkan oleh delay atau d. Nilai Id dijabarkan pada persamaan berikut:

Variable H merupakan fungsi heavyside ditunjukkan pada persamaan X. Sedangkan Ie merupakan factor penurunan kualitas yang diakibatkan presentasi kehilangan paket. Nilai Ie dijabarkan pada persamaan X.

Setelah nilai R factor ditemukan, maka nilai tersebut akan diestimasikan kedalam MOS (ITU-T.P.800). Hasil estimasi akan dikorelasikan kedalam tingkat kualitas komunikasi. Untuk mengubah R factor kedalam MOS (ITU-T.P.800) menggunakan ketentuan berdasarkan rentang nilai R factor yang ditunjukkan pada persamaan berikut.

Tabel x Mapping MOS dan E-Model (Mohd & Ong, 2007)

|  |  |  |
| --- | --- | --- |
| R | *User Satisfaction* | MOS |
| 90-100 | *Very Satisfied* | 4.3-4.5 |
| 80-90 | *Satisfied* | 4.0-4.3 |
| 70-80 | *Some user dissatisfied* | 3.6-4.0 |
| 60-70 | *Many users dissatisfied* | 3.1-3.6 |
| 50-60 | *Nearly all user dissatisfied* | 2.6-3.1 |
| 0-50 | *Not recommended* | 1-2.6 |

Untuk mendapatkan perhitunga *Id* dan *Ie* maka diperlukan nilai dari *delay* dan *packet loss* dari komunikasi yang dilakukan. Semakin kecil nilai dari *delay* dan *packet loss* memungkinkan semakin tinggi nilai R yang didapatkan. Melalui nilai *delay* dan *packet loss* juga dapat memperlihatkan kondisi dari kualitas komunikasi yang dilakukan.

1. ***Average End to End Delay***

Merupakan rata-rata selang waktu yang diperlukan oleh *node* sumber mengirimkan paket menuju ke *node* tujuan. Perhitungan *Average End to End Delay* diperlihatkan pada persamaan ()

(2)

Dimana :

Tr = Waktu penerimaan paket (detik)

Ts = Waktu pengiriman paket(detik)

Berdasarkan kategori, delay dapat diklasifikasikan sebagai berikut:

Tabel . Rentang nilai *delay*

|  |  |  |
| --- | --- | --- |
| **Kategori delay** | **Besar *delay*** | **Indeks** |
| Sangat bagus | < 150 ms | 4 |
| Bagus | 150 – 300 ms | 3 |
| Sedang | 300 – 450 ms | 2 |
| Jelek | > 450 ms | 1 |

1. ***Packet Loss***

Suatu parameter yang menunjukkan kondisi jumlah total paket yang hilang karena collision dan congestion pada jaringan. Di dalam jaringan, nilai packet loss ini diharapkan memiliki nilai yang minimum. Berikut merupakan rumusan untuk mencari nilai packet loss:

()

*Packet loss* dapat dikategorikan dalam rentangan tabel sebagai berikut:

Tabel . Rentang nilai *packet loss*

|  |  |  |
| --- | --- | --- |
| **Kategori Degredesi** | **Paket Loss** | **Indeks** |
| Sangat Bagus | 1 % | 4 |
| Bagus | 3 % | 3 |
| Sedang | 15 % | 2 |
| Jelek | 25 % | 1 |

# BAB III ANALISIS DAN PERANCANGAN SISTEM

1. **Tahapan Analisis**

Dalam pengerjaan penelitian tugas akhir ini dilakukan dalam sebuah simulasi, simulasi pada penelitian ini menggunakan *tools* simulator Network Simulator 2.35 atau NS-2.35. Dalam proses penelitian ini, akan diimplementasikan protokol routing AOMDV (*Ad hoc On-Demand Distance Vector*) dan penggunaan algoritma Path Aware SHORT pada protokol AOMDV sehingga menghasilkan protokol *routing* baru, yakni PA-AOMDV (*Path Aware-Ad hoc On-Demand Distance Vector*). Kedua protokol *routing* tersebut akan digunakan pada simulasi MANET atau Mobile Ad-Hoc Network.

*Node* yang berkomunikasi pada simulasi yang dilakukan akan bergerak berdasarkan kondisi geografis dari pegunungan yang ditentukan. Hal ini dapat dicapai dengan menggunakan data map geografis dari *OpenStreetMap* dan dikonversi menjadi posisi dan pergerakan *node* menggunakan SUMO (*Simulation of Urban Mobility*). Posisi dan rute pergerakan *node* dibuat secara acak mengikuti *egde* yang terbangun dari data map *OpenStreetMap.* Kecepatan dibuat dengan rentangan yang disesuaikan dengan kecepatan pergerakan manusia. Komunikasi yang terjadi antar *node* akan melibatkan satu *node* sumber dengan satu *node* tujuan. Besaran paket komunikasi yang dikirimkan akan tergantung dengan *codec* yang ditentukan serta waktu paket dikirimkan akan ditentukan melalui parameter simulasi. Jenis paket komunikasi yang digunakan ada paket UDP (*User Datagram Protocol*).

Dalam penelitian ini menggunakan beberapa parameter simulasi, berikut merupakan parameter simulasi yang digunakan:

|  |  |
| --- | --- |
| **Parameter** | **Nilai** |
| Tipe Kanal | *Wireless Channel* |
| Model Propagasi | *TwoRayGround* |
| Tipe *Network Interface* | *WirelessPhy* |
| Tipe MAC | IEEE 802.11 |
| *Max* Paket dalam Antrian | 50 |
| Tipe Paket | UDP |
| Mobilitas *Node* | SUMO |
| Kecepatan *Node* | 0 m/s sampai 7 m/s |

Dari tabel diatas, terdapat beberapa nilai *default* untuk parameter Tipe Kanal, Model Propagasi, Network Interface, Tipe MAC dan *Max* paket dalam antrian.

1. **Perancangan Simulasi**

Dalam penelitian ini terdapat beberapa scenario yang akan dijadikan acuan dalam pengujian dan analisis data hasil simulasi. Scenario diberikan dalam simulasi dengan tujuan untuk mengukur kualitas komunikasi *audio* pada simulasi yang dipenaruhi oleh variable yang berubah ubah. Berikut merupakan scenario yang digunakan dalam proses simulasi:

|  |  |
| --- | --- |
| **Parameter** | **Nilai** |
| Luas Area | 4000m2 |
| Waktu Simulasi | 120 detik |
| Lama Waktu Komunikasi *Audio* | 60 detik |
| Jumlah *Node* | 25, 35, 45, 55, 65, 75 |
| Protokol *Routing* | AOMDV, PA-AOMDV |
| Codec *Audio* | G.711, G.723.1 |

Simulasi dilakukan dengan luas wilayah simulasi 4000m2. Setiap *node*, baik itu *node* sumber, *node* penerima dan *node* perantara diletakkan secara acak dalam *edge* dan rentangan luas wilayah tersebut. *Node* sumber merupakan node yang mengirimkan paket, *node* pengantar merupakan *node* yang digunakan untuk meneruskan paket hingga menuju ke *node* tujuan. Sedangkan *node* tujuan merupakan *node* yang menerima paket dari *node* sumber. Jumlah node yang digunakan dalam simulasi adalah 25, 35, 45, 55, 65 dan 75 node. Hal ini dilakukan untuk mendapatkan nilai dari penambahan jumlah node serta acuan dan perbandingan dalam menentukan jumlah *node* yang sesuai ketika diimplementasikan dalam luas wilayah simulasi tertentu.

Setiap *node* diletakkan secara acak pada *edges* serta mobilitas *node* dibuat mengikuti *edges* secara acak dengan kecepatan dalam rentangan 0 m/s hingga 7 m/s. Kondisi *node* yang diletakkan secara acak pada *edges* serta mobilitas *node* dibuat menggunakan randomTrips.py pada SUMO, dimana randomTrips.py ini akan membangkitkan *node* pada *edges* dan memberikan mobilitas pada *nodes* dengan menelusuri *edges* pada data lokasi *OpenStreetMap*. Hasil randomTrips.py akan menghasilkan *file* dengan format xml. *File* xml ini merupakan bagian dari file konfigurasi sumo dengan format sumo.cfg, *file* berformat sumo.cfg ini mengandung parameter lain dari *file net-file, route-files dan additional-files*. Berikut merupakan isi dari *file* konfigurasi sumo.cfg.

<configuration>

<input>

<net-file value='"$HOME/$proj/$netfiles"'/>

<route-files value='"$HOME/$proj/$rout/$i.rou.xml"'/>

<additional-files value='"$HOME/$proj/$poly/$i.poly.xml"'/>

</input>

<time>

<begin value='"0"'/>

<end value='"120"'/>

<step-length value='"0.1"'/>

</time>

</configuration>

Untuk menghasilkan *file* mobilitas untuk Network Simulator 2.35 melalui SUMO, maka setelah *file* konfigurasi SUMO dibuat, maka proses selanjutnya adalah menghasilkan *file* output dari *file* konfigurasi. Melalui perintah sumo dan dengan parameter *file* konfigurasi maka didapatkan *file* output berformat \*.sumo.xml. Format perintah sumo sebagai berikut:

sumo -c <*file* konfigurasi \*.sumo.cfg> --fcd-output <*file* *output* \*.sumo.xml>

Kemudian dengan menggunakan fungsi traceExporter pada SUMO. Fungsi ini akan mengkonversi *file* \*.sumo.xml menjadi *file* dengan format \*.tcl, yang pada Network Simulator 2.35 akan digunakan sebagai *file* mobilitas dengan menggunakan fungsi *source* pada fungsi pemrograman tcl. Untuk parameter mobilitas Network Simulator 2.35, pada perintah traceExporter, didefinisikan tipe *output* dengan –-ns2mobility-output. Berikut merupakan format fungsi dan traceExporter.py

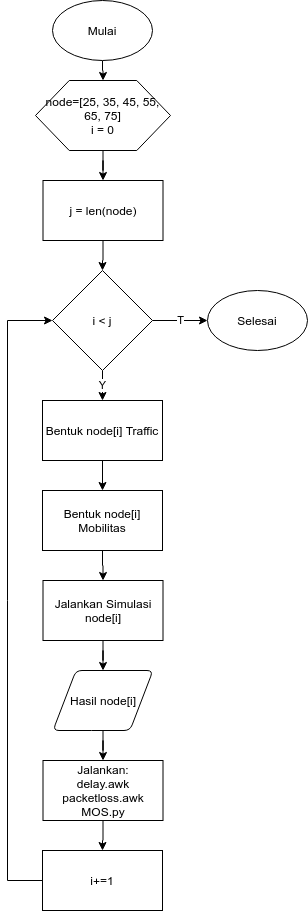
python $SUMO\_HOME/tools/traceExporter.py --fcd-input <file \*.sumo.xml> <tipe *output*> <*output file* \*.tcl>

Selanjutnya adalah pembuatan *traffic* komunikasi *audio*. Pembuatan *traffic* komunikasi *audio* menggunakan modul Ns2Voip++ (<http://cng1.iet.unipi.it/wiki/index.php/Ns2voip%2B%2B>). Ns2Voip++ merupakan sebuah patch yang akan menghasilkan satu *file* berformat tcl sebagai *file* yang menciptakan *traffic* komunikasi *audio.* Pada *file* tcl ini pengaturan beberapa parameter untuk disesuaikan dengan kondisi simulasi. *File* tcl dari Ns2Voip++ akan digunakan sebagai sumber paket komunikasi *audio*, jenis paket dari Ns2Voip++ adalah bertipe UDP (*User Datagram Protocol*). Paket ini akan dikirimkan untuk mensimulasikan komunikasi *audio* di Network Simulator 2.35.

Langkah selanjutnya adalah menjalankan simulasi di Network Simulator 2.35 dengan menjalankan program tcl, simulasi akan menghasilkan *output file trace* dan NAM *file.* Berdasarkan *output* tersebut, akan diketahui kualitas dari komunikasi *audio* yang dijalankan pada simulasi.

1. **Tahapan Kerja Simulasi**

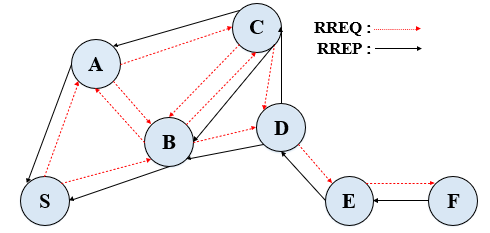
Tahap pengerjaan simulasi ditunjukkan pada diagram alir pada gambar dibawah, berikut penjelas mengenai diagram alir tahapan pengerjaan simulasi:



Gambar . Proses pengerjaan simulasi

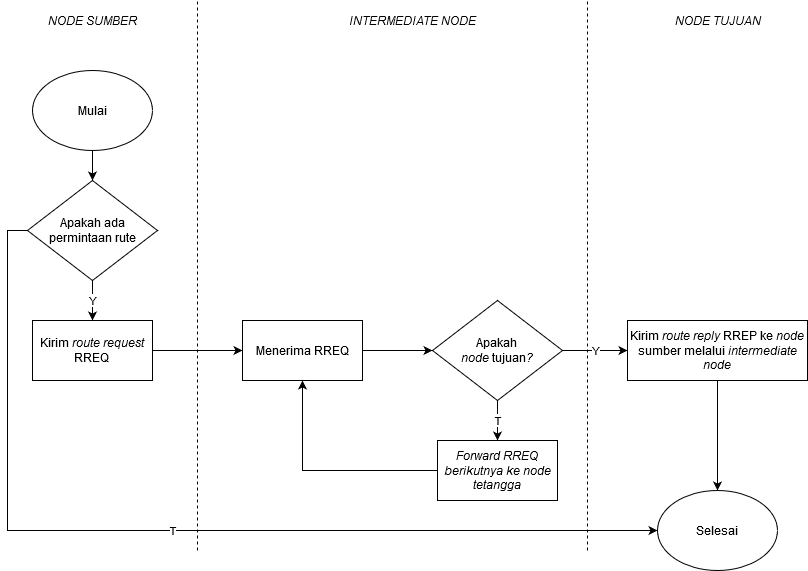
Jumlah *node* pada proses simulasi mengalami penambahan konstan dari node 25, 35, 45, 55, 65, dan terakhir 75 *node*. Pengerjaan simulasi akan dilakukan sebanyak jumlah *node* pada list.

1. Konfigurasi dan pembentukan *node* pada jaringan *wireless* pada Network Simulator 2.35 dengan penggunaan protokol *routing* AOMDV dan PA-AOMDV.
2. Konfigurasi *traffic* komunikasi *audio* pada *file* generator *traffic*. Proses ini juga akan menentukan *node* pengirim dan penerima.
3. Menjalankan simulasi untuk mendapatkan *file trace*.
4. Menjalankan kode perhitungan *delay, packet loss* dan perhitungan MOS untuk mendapatkan nilai kualitas komunikasi *audio* pada simulasi.
5. Jika nilai i tidak lebih dari j, maka simulasi dilanjutkan pada node indeks i+1.
6. **Penemuan Rute (*Route Discovery*) Protokol *Routing* AOMDV**

Protokol *routing* AOMDV merupakan protokol *routing* yang menggunakan konsep *distance vector* yakni pencarian rute berdasarkan jarak terdekat dan menggunakan pendekatan *hop by hop routing.* AOMDV menghasilkan lebih dari satu rute atau jalur dalam proses penemuan rute apabila memungkinkan. Adanya rute alternative menyebabkan AOMDV tidak perlu melakukan proses pencarian rute apabila terjadi kegalalan rute pada rute sebelumnya. Proses penemuan rute akan dilakukan kembali, apabila rute alternative tiak tersedia.

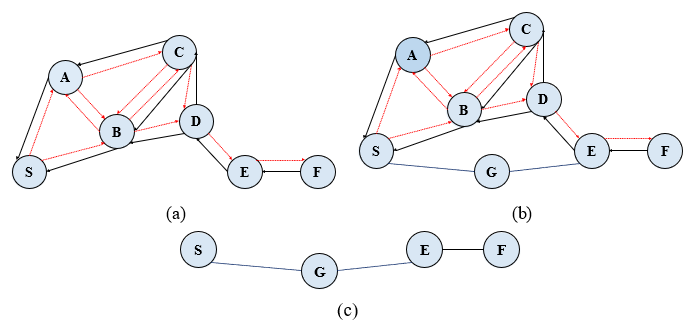
Gambar . Proses penemuan rute protokol *routing* AOMDV

Proses penemuan rute (*route discovery*) pada AOMDV ditujukkan pada Gambar xx. Pada kasus tersebut, *node* S merupakan *node* sumber yang akan mengirimkan paket ke *node* F. *Node* S akan melakukan *flooding* paket RREQ ke *node-node* tetangganya yakni *node* A dan *node* B. Karena *node* A dan *node* B bukan *node* tujuan, maka kedua *node* akan meneruskan paket RREQ ke *node-node* tetangganya namun tidak ke *node* sumber. Setelah itu *node* A dan *node B* akan melakukan pengaturan untuk rute cadangan. Ketika *node* F (*node* tujuan) menerima paket RREQ maka *node* F akan mengirimkan paket RREP sebagai balasan paket RREQ. *Node* penerima paket RREP akan melakukan pengaturan untuk *forward path*. Proses *route discovery* pada protokol *routing* AOMDV dapat dilihat pada Gambar xx.



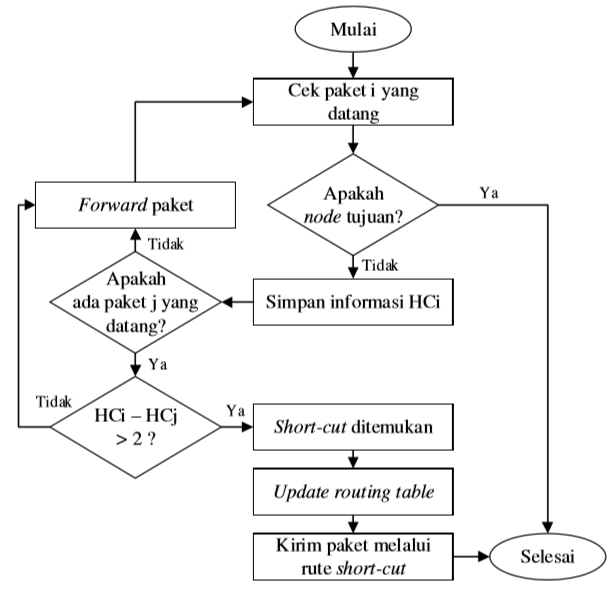
Gambar . Diagram alir proses pencarian rute protokol *routing* AOMDV

1. **Penemuan Rute (*Route Discovery*) Protokol Routing Modifikasi PA-AOMDV**

Mobilitas atau pergerakan dari *node* dapat merubah jalur rute melalui perubahan topologi. Perubahan bentuk topologi dapat dimanfaatkan untuk mendapatkan jalur yang lebih baik. Pertimbangan jalur dari *node* S menuju *node* F ditujukkan oleh Gambar xx (a) dan jalur terpendek yang ditemukan adalah rute dari *node* S-B-D-E-F. Jalur yang ditemukan berdasarkan jarak yang dihitung melalui jumlah *hop* dari *node* S dan *node* F yakni 4 *hop.* Pergerakan *node* dapat membuat topologi dan jalur rute berubah serta penambahan atau pengurangan *node* dalam jangkauan transmisi yang ditunjukkan pada Gambar xx (b). Dalam bentuk topologi baru, *node* G berada dalam rentang transmisi *node* S, sehingga *node* S dapat menjangkau *node* G, dan *node* E masuk dalam jangkauan transmisi *node* G. Dengan demikian perubahan rute jalur terpendek dari *node* S ke *node* F hanya membutuhkan 3 *hop* seperti pada Gambar xx (c).

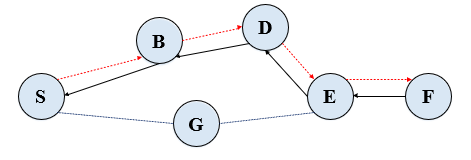
Gambar . Ilustrasi perubahan topologi dan penemuan rute *Path Aware*

Ketika *node* mengirimkan paket, *node* penerima akan memeriksa apakah paket ditujukkan untuk *node* tersebut atau tidak. Apabila paket ditujukkan kepada *node* tersebut, maka paket akan langsung dikirim, jika *node* tersebut bukan sebagai *node* tujuan, maka paket akan diteruskan sesuai rute melalui *intermediate node* dan informasi HC disimpan (HCi). Ketika terdapat paket lain datang untuk dikirimkan, paket akan dikirim melalui *intermediate node* (HCj). HC merupakan banyaknya *hop* yang dilewati pada rute dalam pengiriman paket. Jika selisih antara HCi dan HCj tidak lebih besar daripada 2 maka tidak ditemukan rute *short-cut.* Apabila selisih lebih besar daripada 2 maka *short-cut* ditemukan dan *node* akan mengirim pesan ke *node* sebelumnya untuk memperbarui tabel *routing*, sehingga rute yang digunakan adalah rute *short-cut*. Pencarian rute dengan PA-SHORT ditujukkan pada Gambar xx.

Algoritma *Path Aware* SHORT mengoptimalkan rute berdasarkan dengan *hop* terpendek. Maka dari itu setiap *header* tiap paket membawa *field hop-count* (HC). HC diinisialisasikan ke nol pada *node* sumber dan bertambah satu setiap *hop* yang dilalui paket. Pada paket *header* juga terdapat informasi mengenai alamat tujuan (DA), alamat sumber (SA), *hop-count* (HC) dan NA yaitu alamat tetangga dalam transmisi *node* yang menyiarkan paket. Format setiap entri adalah <SA, DA, HC, NA>.

Gambar . Diagram alir penemuan *short-cut* pada *Path Aware*

Pencarian rute (*route discovery*) dari *node* S ke *node* F ditunjukkan pada Gambar xx (a). Rute yang dihasilkan dalam pencarian rute menghasilkan rute dengan 4 *hop* untuk menuju ke *node* F, yakni S-B-D-E-F. Pergerakan *node* menyebabkan perubahan topologi diperlihatkan pada Gambar xx (b) dengan tetap mempertahankan konektivitas *node*. Dalam bentuk baru *node* G berada dalam jangkauan transmisi *node* S, sehingga dapat dilakukan perubahan rute seperti pada Gambar xx.

Pada Gambar xx akan dianalisis dalam bentuk langkah-langkah, dimana akan dijelaskan proses pemilihan rute dalam perubahan topologi dalam jaringan. Berikut langkah-langkah *node* saat perubahan rute:

Gambar . Contoh perubahan rute

* + - 1. *Node* S akan mengirimkan paket ke *node* B. Paket ini diinisialisasi *node* S sebagai (S, F, 0, S). *Hop-count* awal akan bernilai 0. *Node* B dan *Node* G berada dalam jangkauan transmisi *node* S. Kedua dari *node* tersebut akan menerima paket dari *node* S. *Node* B dan *node* S bukan *node* tujuan, sehingga *node* B dan *node* G mencatat (S, F, 0, S).
      2. Meneruskan paket ke *node* D, *node* B akan mengirimkan paket dengan *hop count* 1. *Node* D mencatat (S, F, 1, B).
      3. *Node* D mengirimkan paket ke *node* E dengan *hop count* 2. *Node* B dan *node* E berada dalam jangkauan transmisi *node* D, maka *node* D akan mengirimkan paket ke *node* E dengan *header* (S, F, 2, D).
      4. Ketika meneruskan paket ke *node* F, *node* E mengirimkan paket dengan *hop count* 3. *Node* D, F, dan *node* G berada dalam jangkauan transmisi *node* E. *Node* F akan menerima langsung paket dari *node* E tanpa meneruskannya kembali, karena *node* F merupakan *node* tujuan. *Node* F dan *node* G akan mencatat paket dengan *header* (S, F, 3, E). *Node* G akan membandingkan dengan paket *header* sebelumnya (S, F, 0, S) dengan *header* paket yang baru dari *node* E yakni (S, F, 3, E). Perbedaan *hop count* pada *header* lebih dari 2 yakni 3. Maka, *node* G akan mengirimkan pesan ke *node* S untuk memperbarui tabel *routing* sehingga rute dari *node* S ke *node* F mulai dari *node* S akan dilanjutkan ke *node* G. *Node* G akan memperbarui tabel *routing*-nya untuk mengirimkan paket ke *node* tujuan yakni *node* F. Sehingga *node* G akan mengirimkan paket ke *node* E dengan *hop count* 2. Rute S-G-E dibuat sebagai pintasan dari rute S-B-D-E sebelumnya. Jadi rute dari *node* S ke *node* F adalah S-G-E-F.

1. ***Traffic* Komunikasi *Audio***

Traffic komunikasi *audio* dibangun dengan modul pembangkitan *traffic* NS2Voip++. Pembangkitan *traffic* komunikasi *audio* pada proses simulasi mengikuti parameter pada NS2Voip++ sebagai berikut:

|  |  |
| --- | --- |
| Lama Waktu Komunikasi | 60 detik |
| *Talkspurt* | Weibull distribution f(x; 0.412, 0.824) |
| *Silence* | Weibull distribution f(x; 0.899, 1.089) |
| Codec | G.711, G.723.1 |
| Tipe Paket | UDP |

Komunikasi *audio* yang dirancang akan dilakukan selama 60 detik antara penerima dan pengirim. Komunikasi *audio* yang dilakukan merupakan komunikasi satu arah, dengan periode *talkspurt* dan *silence* dibangkitkan dengan distribusi Weibull pada modul NS2Voip++. Ukuran data komunikasi *audio* akan ditentukan berdasarakan ukuran *payload* dari codec yang digunakan. Spesifikasi codec yang digunakan untuk membangkitkan ukuran data atau paket komunikasi *audio* diperlihatkan pada Gambar xx.

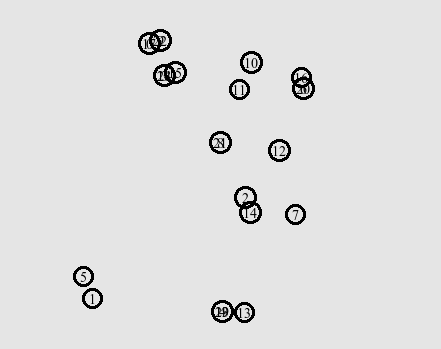
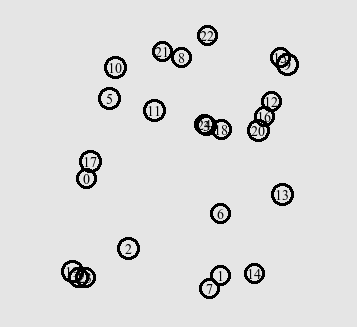
Sumber tabel (El Brak, Bouhorma, & Boudhir, 2012).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Codec | Bit rate (kbps) | Sample size (bytes) | Packet per second | Payload size (bytes) |
| G.711 | 64 | 80 | 50 | 160 |
| G.723.1 | 6.3 | 24 | 34 | 24 |

1. **Topologi Jaringan**

Bentuk topologi pada jaringan *ad-hoc* dibuat berdasarkan data geologis pegunungan *openstreetmap.* Peletakan *node* pengirim, penerima dan pengantar diletakkan secara acak pada *edges* yang dihasilkan dari data geologis pegunungan *openstreetmap.* Hasil dari setiap scenario yang dijalankan memiliki pergerakan *node* penerima, pengirim dan pengantar yang berbeda.

Gambar . Pergerakan *node* dan perubahan topologi



Gambar diatas menunjukkan posisi awal *node* yang diletakkan secara acak pada *edges. Node* pada MANET bergerak dan berpindah secara acak mengikuti jalur *edges* pada saat simulasi dijalankan, pada gambar dapat dicontohkan *node* 1 dan *node* 7 bergerak berpindah dan membentuk topologi baru.

1. **Parameter Kinerja**

Pada penelitian ini, yang menjadi tolak ukur kualitas komunikasi *audio* pada simulasi yang dilakukan adalah nilai MOS E-Model. Perhitungan MOS E-Model ini diperoleh dengan beberapa parameter tambahan, yakni *delay* dan *packet loss.* Untuk menghitung *delay* dan *packet loss* dapat dilakukan melalui *file trace*. *File trace* merupakan keluaran yang mencatat segala perilaku *node* pada saat simulasi. Melalui *file* ini perhitungan *delay* dan *packet loss* dapat dilakukan, sehingga mendapatkan nilai MOS E-Model pada akhirnya. Gambar dibawah ini merupakan contoh dari *file trace* proses simulasi.

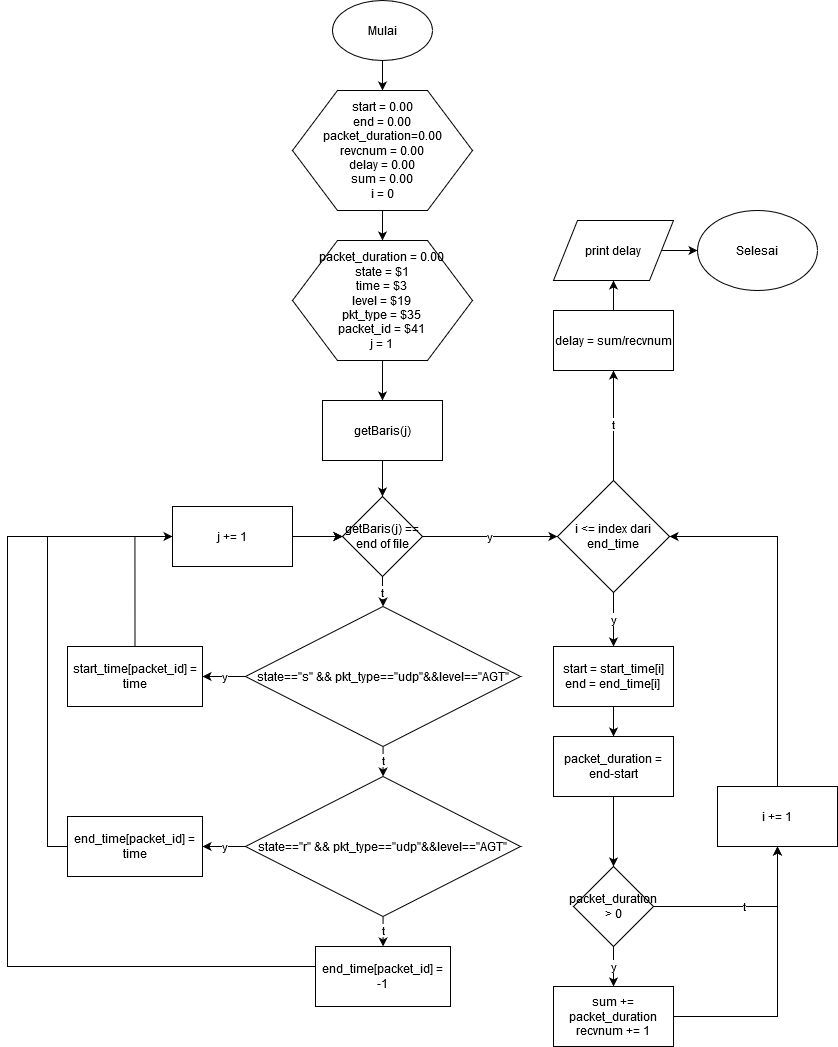
|  |  |
| --- | --- |
| 1 | s -t 10.010000000 -Hs 1 -Hd -2 -Ni 1 -Nx 2116.31 -Ny 2677.17 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 1.1 -Id 24.0 -It udp -Il 163 -If 1 -Ii 99 -Iv 32 |
| 2 | s -t 10.033660465 -Hs 7 -Hd -2 -Ni 7 -Nx 2227.70 -Ny 2787.52 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 7.2 -Id 9.0 -It cbr -Il 512 -If 0 -Ii 101 -Iv 32 -Pn cbr -Pi 23 -Pf 0 -Po 0 |
| 3 | r -t 10.034419350 -Hs 10 -Hd -2 -Ni 10 -Nx 2176.96 -Ny 2958.10 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md ffffffff -Ms 9 -Mt 800 -Is 9.255 -Id -1.255 -It AOMDV -Il 48 -If 0 -Ii 0 -Iv 28 -P aodv -Pt 0x2 -Ph 3 -Pb 2 -Pd 24 -Pds 0 -Ps 1 -Pss 6 -Pc REQUEST |

*File trace* mencatat aktivitas yang terjadi pada saat simulasi seperti *send, drop, receive,* waktu, jumlah paket yang dikirimkan. Karena komunikasi yang dilakukan merupakan komunikasi wireless atau nirkabel, maka hal tersebut juga akan mempengaruhi parameter pada *file trace.* Penjelasan mengenai parameter yang terdapat pada *file trace* diperlihatkan pada Gambar xx.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Event | Singkatan | *Flag* | Tipe | *Value* |
| Koneksi *Wireless* (Nirkabel) | s: send  r: receive  d: drop  f: forward | -t | double | Waktu |
| -Ni | int | ID *Node* |
| -Nx | double | Koordinat X pada *node* |
| -Ny | double | Koordinat Y pada *node* |
| -Nz | double | Koordinat Z pada *node* |
| -Ne | double | *Node* energi |
| -Nl | string | Level *trace* jaringan (AGT, RTR, MAC dan lainnya) |
| -Nw | string | Sebab paket *drop* |
| -Hs | int | ID *node* dari *Hop Source* |
| -Hd | int | ID *node* dari *Hop Destination* |
| -Ma | hexadecimal | Durasi |
| -Ms | hexadecimal | Alamat *ethernet* sumber |
| -Md | hexadecimal | Alamat *ethernet* tujuan |
| -Mt | hexadecimal | Tipe *ethernet* |
| -P | string | Tipe paket (arp, dsr, imep, tora, dan lain-lain) |
| -Pn | string | Tipe paket (cbr, tcl) |
| IP *Trace* |  | -Is | int.int | Alamat sumber dan *port* |
| -Id | int.int | Alamat tujuan dan *port* |
| -It | string | Tipe paket |
| -Il | int | Ukuran paket |
| -If | int | ID *flow* |
| -Ii | int | *Unique ID* |
| -Iv | int | Nilai TTL |
| Protokol *Trace* (AOMDV/PA-AOMDV) |  | -Pt | hexadecimal | tipe |
| -Ph | int | Jumlah *hop* |
| -Pb | int | *Broadcast* ID |
| -Pd | int | Tujuan |
| -Pds | int | Nomor urut tujuan |
| -Ps | int | Sumber |
| -Pss | int | Nomor urut sumber |
| -Pl | double | *Lifetime* |
| -Pc | string | Operasi (REQUEST, REPLY, ERROR, HELLO) |
| CBR *Trace* |  | -Pi | int | Nomor urut |
| -Pf | int | Berapa kali paket telah di*forward* |
| -Po | int | Jumlah optimal *forward* |

Simulasi yang dilakukan merupakan simulasi dengan perangkat atau *interface* nirkabel (*wireless*)maka *file trace* hasil *trace* akan selalu melibatkan *event* dari Koneksi *Wireless* (nirkabel). Pada baris 1 Gambar xx, merupakan aktivitas dari IP *trace*, membawa tipe paket UDP yang berarti merupakan aktivitas dari komunikasi *audio*. Kemudian pada baris 2 merupakan aktivitas dari pengiriman paket CBR dan baris 3 merupakan aktivitas dari protokol *routing* yang digunakan pada proses simulasi.

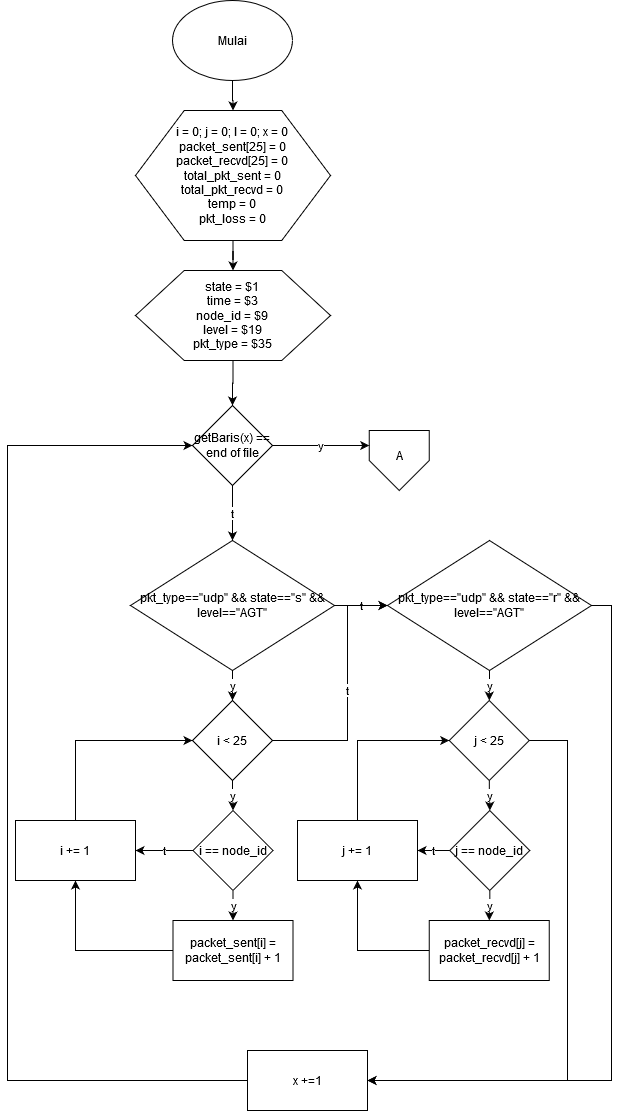
Parameter yang akan dianalisis melalui *file trace* adalah nilai *delay, packet loss,* R factor dan MOS. Setiap scenario menghasilkan *file trace* yang berbeda, dan melalui *file* tersebut akan ditentukan nilai *delay, packet loss* dan R factor. Melalui hal ini akan dianalisa pengaruh jumlah *node* dan jenis codec terhadap kualitas komunikasi *audio* ditinjau dari *delay*, *packet loss* dan kemudian ditentukan melalui rentangan nilai R factor dan MOS. Adapun *flowchart* dari perhitungan *Delay, Packet loss* dan R factor dijelaskan pada Gambar xx, Gambar xx, Gambar xx, Gambar xx dan Gambar xx.



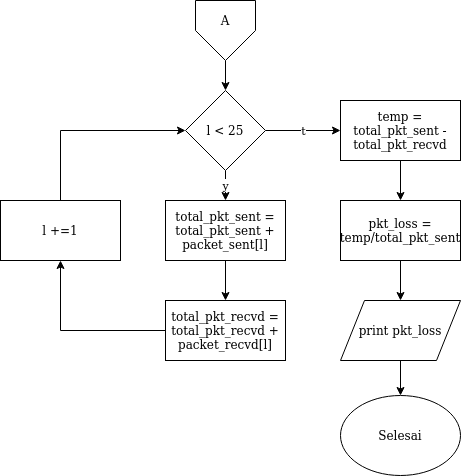
Gambar . Diagram alir perhitungan *delay*

Gambar kinerja delay

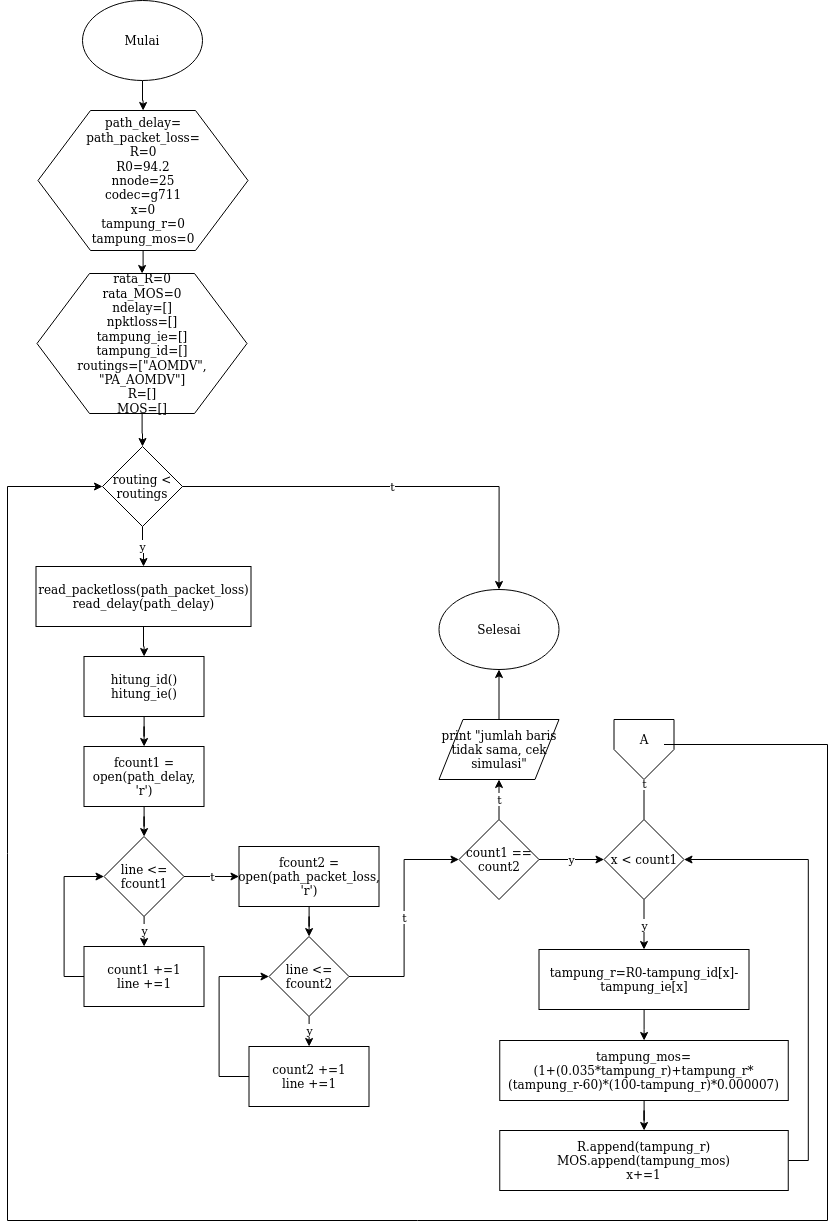
Pada proses perhitungan *delay*, Gambar xx menjelaskan proses perhitungan *delay.* Pada *flowchart* perhitungan *delay* tahap awal yang dilakukan adalah inisialisasi variable penampung nilai dan variable yang menampung nilai kolom pada *file trace*. Nilai $1 akan merepresentasikan nilai pada kolom 1 dan begitu seterusnya. Kemudian dilakukan perulangan sebanyak jumlah baris pada *file trace*. Ketika proses masih dalam perulangan kemudian akan masuk pada proses pemilihan, ketika *state* sama dengan ‘s’ atau send dan tipe paket sama dengan UDP, serta *level* aktivitas AGT (agent) maka nilai time akan masuk ke indeks ke packet\_id pada variable start\_time. Ketika nilai state sama dengan ‘r’ atau receive dan tipe paket UDP serta level aktivitas *trace* AGT (agent) maka nilai time akan disimpan pada indeks ke packet\_id pada variable end\_time. Ketika perulangan hingga akhir dari *file* telah dilakukan, proses selanjutnya akan dilakukan perulangan sebanyak indeks dari variable end\_time, perulangan akan memproses variable start dan end menyimpan nilai dari variable start\_time dan end\_time pada indeks ke i, kemudian variable packet\_duration akan menyimpan nilai selisih dari end dengan start, ketika nilai packet\_duration lebih besar daripada 0, maka nilai variable sum akan dijumlahkan dengan nilai variable packet\_duration dan variable recvnum akan ditambah 1. Ketika perulangan telah dilakukan sebanyak indeks dari variable end\_time maka nilai delay akan menyimpan nilai dari lamanya durasi komunikasi dengan banyaknya proses komunikasi dilakukan.

Perhitungan untuk nilai *packet loss* diperlihatkan pada Gambar xx dan dilanjutkan pada Gambar xx. Proses awal merupakan inisialisasi variable penampung nilai dan variable untuk menampung nilai dari kolom pada *file trace.* Perulangan pertama dilakukan sebanyak baris pada *file trace*. Kemudian ketika *event* sama dengan ‘s’ atau *send* dengan tipe paket UDP dan *level trace* adalah AGT (Agent) maka akan dilakukan perulangan sebanyak jumlah node yang disimulasikan, nilai variable i akan sama dengan nilai node­­\_id pada baris yang diproses. Kemudian nilai array packet\_sent pada indeks ke node\_id akan ditambahkan satu, itu berarti packet\_sent pada indeks node\_id akan bertambah satu ketika terjadi *event sent* pada baris *file trace* node\_id tersebut. Hal ini juga berlaku pada *event received*, ketika proses *receviced* terjadi maka array packet\_recvd indeks ke node\_id akan ditambahkan satu ketika baris *file trace* dengan *event* ‘r’ dan node\_id yang sama. Ketika proses perulangan hingga baris terakhir pada *file trace* selesai maka proses selanjutnya adalah perulangan sejumlah *node* pada proses simulasi. Perulangan ini akan menjumlahkan paket yang terkirim dari array packet\_sent dan menjumlahkan paket yang diterima dari array packet\_recvd. Hasil pernjumlahan ini akan disimpan pada variable total\_pkt\_sent dan total\_pkt\_recvd. Perhitungan selanjutnya adalah mencari paket yang terbuang (*packet loss*) dengan selisih jumlah paket yang terkirim dengan jumlah paket yang diterima dibagi dengan jumlah paket yang dikirim.

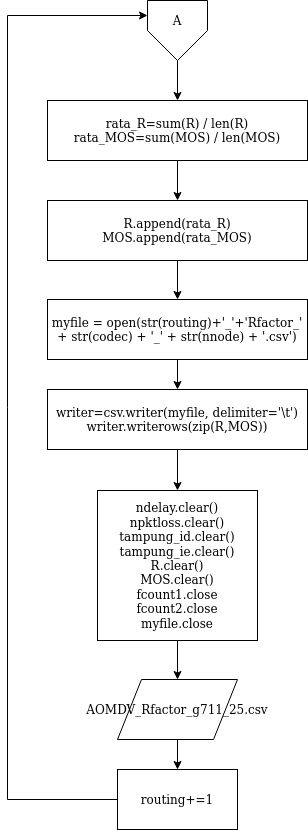
Gambar . Diagram alir perhitungan *packet loss*

Dari perhitungan *delay* dan *packet loss* nilai kedua perhitungan tersebut akan digunakan pada proses pencarian nilai R factor untuk menentukan nilai MOS. Perhitungan MOS menggunakan MOS E-Model. Berikut penjelasan *flowchart* dari perhitungan R factor Gambar xx dan Gambar xx.

Gambar . Diagram alir perhitungan *packet loss* (lanjutan)

Pada tahap awal merupakan inisialisasi variable nilai. Variable tersebut juga meliputi *path* dari *file* nilai *delay* dan *file* nilai *packet loss*. Inisialiasi variable nilai R0, jumlah node dan jenis codec juga dilakukan, serta variable yang akan menampung *list* dari nilai *delay*, nilai *packet loss*, pengaruh *delay* dan *packet loss* serta *list* untuk menampung nilai R dan MOS dari replikasi ketika simulasi. Proses selanjutnya adalah membaca isi dari *file* nilai *delay* dan *file* nilai *packet loss* dilanjutkan proses perhitungan nilai id dan ie untuk nilai pengaruh *delay* dan pengaruh *packet loss.* Dilanjutkan dengan pengecekkan jumlah baris dari *file delay* dan *file packet loss*, ketika jumlah baris sama, maka proses dilanjutkan dengan perhitungan nilai R. Nilai R akan ditampung pada *list* R untuk masing-masing replikasi. Nilai R ini akan disimpan pada *file* berformat csv. Pada Gambar 3.11 dan Gambar 3.12 dicontohkan dengan jumlah *node* sebanyak 25 serta codec G.711.

Gambar . Diagram alir perhitungan nilai R factor dan MOS



Gambar . Diagram alir perhitungan nilai R factor dan MOS (lanjutan)

1. **Tahap Evaluasi**

Pada proses pengujian dengan MOS E-Model untuk mendapatkan nilai dari kualitas komunikasi *audio* yang dilakukan, maka perhitungan tersebut melibatkan perhitungan *delay* dan *packet loss.* Nilai dari MOS E-Model merupakan rentangan nilai yang dibagi berdasarkan indeks-indeks tertentu. Nilai dari rentangan indeks tersebut menunjukkan kualitas komunikasi *audio* yang terjadi pada proses simulasi.

*Delay* merupakan lama waktu yang diperlukan paket untuk sampai dari *node* pengimrim hingga *node* tujuan. Menurut (TIPHON, 1999:1) nilai minimum *delay* untuk mendapatkan komunikasi *audio* yang baik adalah tidak lebih dari 450 ms. Sedangkan *packet loss* merupakan kondisi paket yang hilang dalam proses pengiriman dari *node* sumber ke *node* tujuan karena *collision* ataupun *congestion* pada jaringan. Nilai minimum menurut (TIPHON, 1999:1) yang merepresentasikan komunikasi *audio* yang baik adalah tidak kurang dari 15% paket yang hilang.

Nilai MOS E-Model dipengaruhi oleh 2 faktor yang disebutkan sebelumnya, yakni *delay* dan *packet loss*. Nilai yang disarankan oleh (Mohd & Ong, 2007) sebagai representasi komunikasi yang baik adalah tidak kurang dari 80 untuk nilai R dan 3.6 untuk nilai MOS. Jika scenario menghasilkan nilai yang kurang dari nilai yang disarankan, maka komunikasi dikatakan kurang baik, dari pengaruh *delay* ataupun *packet loss.*

# BAB IV HASIL DAN PEMBAHASAN

1. **Tahap Implementasi**

Pada penelitian ini, komunikasi *audio* disimulasikan dalam perangkat lunak Network Simulator 2.35. Network Simulator 2.35 digunakan untuk membantu proses simulasi *routing* dan komunikasi *audio.* Protokol *routing* yang digunakan dalam proses simulasi adalah AOMDV dan PA-AOMDV sedangkan *traffic audio* yang disimulasikan, dibangkitkan melalui *patch* NS2Voip++. Untuk pembentukkan topologi dan pergerakan *node* menggunakan SUMO dan data geologis pegunungan pada *OpenStreetMap.* Network Simulator 2.35 diimplementasikan pada sistem operasi dengan spesifikasi perangkat sebagai berikut:

|  |  |
| --- | --- |
| Sistem Operasi | Ubuntu 18.04 |
| Prosesor | Intel(R) Core(TM) i7-6700HQ |
| Memory | 8192MB |
| Hardisk | 100GB |

1. **Konfigurasi**

Pada tahap konfigurasi, dilakukan langkah-langkah untuk mengimplementasikan parameter-parameter simulasi serta scenario dalam simulasi. Adapun konfigurasi yang dilakukan adalah sebagai berikut:

1. **Modifikasi PA-AOMDV**

Pada Network Simulator 2.35, protokol AOMDV telah disediakan ketika pengguna menginstall Network Simulator 2.35. Pada penelitian ini, selain menggunakan protokol AOMDV, scenario juga menggunakan protokol PA-AOMDV. Untuk dapat mengimplementasikan PA-AOMDV dilakukan dengan memodifikasi file dasar AOMDV sehingga menjadi PA-AOMDV. Pada Network Simulator 2.35 diperlukan beberapa modifikasi pada *file* packet.h, ns\_tcl.cc, Makefile.in, priqueue.cc, ns-agent.tcl, ns-lib.tcl, ns-mobilenode.tcl, ns-packet.tcl, cmu-trace.ccdan cmu-trace.h*.* Serta penambahan *folder* pa\_aomdv pada direktori ns-2.35 dan *file* pada pa\_aomdv pa\_aomdv\_logs.cc, pa\_aomdv\_packet.h, pa\_aomdv\_rqueue.h, pa\_aomdv\_rqueue.cc, pa\_aomdv\_rtable.h, pa\_rtable.cc, pa\_aomdv.hdan pa\_aomdv.cc.

Modifikasi AOMDV dengan PA-SHORT dilakukan pada *file header* dan c++ yakni pa\_aomdv.h dan pa\_aomdv.cc. Pada *file header*, ditambahkan kode untuk inisialisasi variable yang menampung nilai *hop count*, yakni hc\_i dan hc\_j.

|  |  |
| --- | --- |
| 1 | double hc\_i, hc\_j; |

Kemudian pada *file* pa\_aomdv.cc akan dlakukan inisialisasi nilai dari variable penampung *hop count* hc\_i dan hc\_j pada constructor. Kemudian pada fungsi penerimaan paket, akan ditambahkan kode untuk mencari selisih dari variable *hop count* hc\_i dan hc\_j, ketika selisih lebih besar dari 2, dan kode untuk *forward* paket. Pada fungsi pencarian rute (*route discovery*) akan ditambahkan kode untuk informasi *hop count* pada variable hc\_i. Pada fungsi penerimaan hello paket, nilai variable hc\_j akan menyimpan nilai *hop* dari reply packet. Berikut merupakan potongan modifikasi kode dari *file* pa\_aomdv.cc

|  |  |
| --- | --- |
| 1 | /\* |
| 2 | Constructor |
| 3 | \*/ |
| 4 |  |
| 5 | PA\_AOMDV::PA\_AOMDV(nsaddr\_t id) : Agent(PT\_PA\_AOMDV), |
| 6 | btimer(this), htimer(this), ntimer(this), |
| 7 | rtimer(this), lrtimer(this), rqueue() { |
| 8 | //edit |
| 9 | hc\_i=hc\_j=0.0; |
| 10 | . . . |
| 11 | } |
| 12 | void |
| 13 | PA\_AOMDV::recv(Packet \*p, Handler\*) { |
| 14 | . . . |
| 15 | if ( (u\_int32\_t)ih->daddr() != IP\_BROADCAST){ |
| 16 | rt\_resolve(p); |
| 17 | //edit |
| 18 | if (hc\_i-hc\_j>2){ |
| 19 | recvRequest(p); |
| 20 | } |
| 21 | } |
| 22 | else{ |
| 23 | forward((pa\_aomdv\_rt\_entry\*) 0, p, NO\_PA\_AOMDV\_DELAY); |
| 24 | } |
| 25 | } |
| 26 | void |
| 27 | PA\_AOMDV::recvRequest(Packet \*p) { |
| 28 | . . . |
| 29 | hc\_i = rp->rp\_hop\_count; |
| 30 | . . . |
| 31 | } |
| 32 | void |
| 33 | PA\_AOMDV::recvHello(Packet \*p) { |
| 34 | . . . |
| 35 | hc\_j = rp->rp\_first\_hop; |
| 36 | } |

1. **Pembentukan Mobilitas**

Proses pembentukan mobilitas menggunakan data geologis dari *OpenStreetMap* kemudian diproses menggunakan SUMO. Berikut merupakan kode pembuatan pergerakan *node* dari data geologis *OpenStreetMap.*

|  |  |
| --- | --- |
| 1 | nnode=(25 35 45 55 65 75) #definisikan jumlah node |

Pembuatan mobilitas untuk *node* diimplementasikan pada kode *bash* yakni generate\_mobility. Pada prosesnya, akan diinisialisasikan terlebih dahulu, *node* yang diakan dibuat pergerakannya. Pada variable nnode mengandung *list* dari *node* yang akan dibangkitkan pergerakannya.

|  |  |
| --- | --- |
| 1 | polyconvert --osm-files osmfiles/sebudi.osm --net-file $netfiles --type-file $SUMO\_HOME/data/typemap/osmPolyconvert.typ.xml -o $poly/$i.poly.xml |

Perintah polyconvert akan membentuk pola geometris – bangunan dari *file* osm. Perintah tersebut akan memerlukan parameter *netfiles* dari proses NETCONVERT dan *file* xml *typemap.* Keluaran perintah ini akan menghasilkan file *additional* pada *folder* $poly dengan nama sesuai jumlah *node* yang dibentuk pada nilai $i.

Gambar xx merupakan pembangkitan pergerakan dan posisi *node* pada *edges* menggunakan fungsi randomTrips.py dari SUMO. Pada baris kedua akan dilakukan pemberian nilai pada variable x dengan 1 per jumlah node yang akan dibentuk. Kemudian pada perintah randomTrips.py beberapa parameter seperti –attributes=”type” mendeklarasikan jenis *node* yang akan dibangkitkan, pada *file* ini dapat diinisialisasikan rentangan kecepatan pergerakan *node*. Parameter -n adalah *netfiles,* -a merupakan *additional file*, *file* dimana rentangan kecepatan pergerakan *node* seperti pada Gambar xx, -r merupakan keluaran dari perintah randomTrips.py, dimana akan menghasilkan *file route* yang berisikan informasi jalur *edges* yang dilalui dari masing-masing *node*, -e merupakan END TIME pembangkitan *node*. -e pada perintah ini diberi nilai 1 detik agar pembuatan *node* dalam 1 detik akan menghasilkan 1/$x *node*.

|  |  |
| --- | --- |
| 1 | x=0 |
| 2 | x=`awk -v var1=1 -v var2=$i 'BEGIN { print ( var1 / var2 ) }'` |
| 3 | python $SUMO\_HOME/tools/randomTrips.py --trip-attributes="type=\"myType\"" -n $netfiles -a $type -r $rout/$i.rou.xml -e 1 -p $x -l #akan membuat 1/$x node dalam 1 detik pada file route. |

Gambar xx merupakan isi dari *file additional* mengenai id dari *node* dan kecepatan maksimum dari pegerakan *node*. Kecepatan pada SUMO didefinisikan dalam satuan meter/second.

|  |  |
| --- | --- |
| 1 | <additional> |
| 2 | <vType id="myType" maxSpeed="7.0" vClass="passenger"/> |
| 3 | </additional> |

Gambar xx merupakan kode pembentukkan *file* berformat \*.sumo.cfg. Isi dari *file* yang terbentuk menggunakan *file* dari proses sebelumnya yakni input *net-files, route-files* dan *additional-files*. Masing-masing parameter memiliki nilai *path* dari masing-masing file tersebut. Kemudian pada *file* konfigurasi SUMO ini juga didefinisikan waktu mulai dan waktu berhenti simulasi. Waktu ini akan menjadi waktu rentangan pergerakan *node*, sehingga nilai *end value* disesuaikan dengan lama waktu simulasi. Pada baris ke 16, perintah sumo -c akan menghasilkan *file* sumo.xml, *file* ini mengandung nilai pergerakan dari *node* setiap 0.10 detik. Parameter yang dibutuhkan dari perintah sumo -c yakni *path* file konfigurasi sumo.cfg, dan *path* untuk *output file*.

|  |  |
| --- | --- |
| 1 | echo "building $i sumo.cfg configuration" |
| 2 | printf "<configuration> |
| 3 | <input> |
| 4 | <net-file value='"$HOME/$proj/$netfiles"'/> |
| 5 | <route-files value='"$HOME/$proj/$rout/$i.rou.xml"'/> |
| 6 | <additional-files value='"$HOME/$proj/$poly/$i.poly.xml"'/> |
| 7 | </input> |
| 8 | <time> |
| 9 | <begin value='"0"'/> |
| 10 | <end value='"120"'/> |
| 11 | <step-length value='"0.1"'/> |
| 12 | </time> |
| 13 | </configuration>" > $sumocfg/$i.sumo.cfg |
| 14 |  |
| 15 | echo "sumo.xml files creating ....." |
| 16 | sumo -c $HOME/$proj/$sumocfg/$i.sumo.cfg --fcd-output $sumoxml/$i.sumo.xml |

Untuk menghasilkan *file* pergerakan yang dapat digunakan pada Network Simulator 2.35, SUMO menyediakan *tools* yakni traceExporter.py. Ini memerlukan parameter input berupa *file* sumo.xml dan dengan parameter –ns2mobility-output dapat menghasilkan *file* dengan format tcl yang dapat digunakan sebagai pergerakan *node* pada Network Simulator 2.35. Gambar xx merupakan penggunaan perintah traceExporter.py pada SUMO.

|  |  |
| --- | --- |
| 1 | Python $SUMO\_HOME/tools/traceExporter.py --fcd-input $sumoxml/$i.sumo.xml --ns2mobility-output $ns2mobility/mobility\_$i.tcl |

Isikan ndak ya file pergerakan .tclnya?

Berikut merupakan hasil dari proses traceExporter.py sehingga menghasilkan *file* mobilitas untuk proses peletakan dan pergerakan dari *node* pada proses simulasi. Gambar xx merupakan potongan *file* mobilitas.

|  |  |
| --- | --- |
| 1 | $node\_(0) set X\_ 2230.3 |
| 2 | $node\_(0) set Y\_ 2946.8 |
| 3 | $node\_(0) set Z\_ 0 |
| 4 | $ns\_ at 0.0 "$node\_(0) setdest 2230.3 2946.8 0.00" |
| 5 | $node\_(1) set X\_ 2123.28 |
| 6 | $node\_(1) set Y\_ 2967.12 |
| 7 | $node\_(1) set Z\_ 0 |
| 8 | $ns\_ at 0.0 "$node\_(1) setdest 2123.28 2967.12 0.00" |
| 9 | $ns\_ at 0.1 "$node\_(0) setdest 2230.28 2946.81 0.25" |
| 10 | $ns\_ at 0.1 "$node\_(1) setdest 2123.27 2967.1 0.25" |
| 11 | $node\_(2) set X\_ 2275.66 |
| 12 | $node\_(2) set Y\_ 2787.62 |
| 13 | $node\_(2) set Z\_ 0 |
| 14 | $ns\_ at 0.1 "$node\_(2) setdest 2275.66 2787.62 0.00" |
|  | . . . . |

1. **Membentuk Objek Simulasi**

Pada proses simulasi, diperlukan penentuan objek-objek yang akan menjadi tolak ukur untuk menentukan kualitas komunikasi atau jaringan baik itu jumlah *node* yang digunakan terdiri dari *node* sumber, *node* pengantar, *node* tujuan, jenis codec serta protokol *routing* MANET yang digunakan.

|  |  |
| --- | --- |
| 1 | set val(chan) Channel/WirelessChannel |
| 2 | set val(prop) Propagation/TwoRayGround |
| 3 | set val(netif) Phy/WirelessPhy |
| 4 | set val(mac) Mac/802\_11 |
| 5 | set val(ifq) Queue/DropTail/PriQueue |
| 6 | set val(ll) LL |
| 7 | set val(ant) Antenna/OmniAntenna |
| 8 | set val(x) 4594 |
| 9 | set val(y) 4864 |
| 10 | set val(ifqlen) 50 |
| 11 | set val(adhocRouting) $opt(routing) |
| 12 | set val(nn) $opt(nnode) |
| 13 | set val(stop) 120 |
| 14 | set val(vip) "./voip.tcl" |

Gambar xx menjelaskan parameter dari pembentukan jaringan *wireless*. Pada baris 1 menujukkan jenis kanal yang digunakan, baris 2 merupakan tipe radio, baris 3 menunjukkan jenis *interface*, baris 4 merupakan jenis MAC, baris 7 menunjukkan jenis antenna. Nilai val(x) dan val(y) merupakan nilai panjang dan lebar dari luas simulasi, baris 10 menunjukkan maksimum paket dalam antrian, baris 11 menunjukkan jenis protokol *routing* yang digunakan, baris 12 menunjukkan jumlah node, baris 13 merupakan nilai dari lama waktu simulasi. Kemudian val(vip) berisikan nilai *path* menuju file generator *traffic* komunikasi *audio* pada simulasi.

|  |  |
| --- | --- |
| 1 | set topo [new Topography] |
| 2 | $topo load\_flatgrid $val(x) $val(y) |
| 3 | # SET G O D |
| 4 | set god\_ [create-god $val(nn)] |

Gambar xx merupakan *script* untuk tahap pengaturan luas area simulasi. Luas area sudah ditentukan denan parameter val(x) dan val(y), yang ditunjukkan pada Gambar xx. Dilanjutkan dengan konfigurasi tiap-tiap *node*.

|  |  |
| --- | --- |
| 1 | # N O D E Config |
| 2 | $ns\_ node-config -adhocRouting $val(adhocRouting) \ |
| 3 | -llType $val(ll) \ |
| 4 | -macType $val(mac) \ |
| 5 | -ifqType $val(ifq) \ |
| 6 | -ifqLen $val(ifqlen) \ |
| 7 | -antType $val(ant) \ |
| 8 | -propType $val(prop) \ |
| 9 | -phyType $val(netif) \ |
| 10 | -channelType $val(chan) \ |
| 11 | -topoInstance $topo \ |
| 12 | -agentTrace ON \ |
| 13 | -routerTrace ON \ |
| 14 | -macTrace OFF \ |
| 15 | -movementTrace OFF |

Gambar xx menunjukkan pengaturan tiap-tiap *node* yang terbentuk. Pada tahap ini, parameter pada Gambar xx akan digunakan oleh tiap-tiap *node* sesuai dengan banyaknya *node* yang dibentuk.

1. **Pembuatan *File Trace* dan *File NAM***

*File trace* merupakan hasil dari seluruh aktivitas yang terjadi selama proses simulasi, seluruh aktivitas seperti *event,* waktu, besaran paket, *node* sumber, *node* tujuan tercatat dalam *file trace.* Berikut merupakan kode yang digunakan untuk menghasilkan *file trace.*

|  |  |
| --- | --- |
| 1 | #use new trace format |
| 2 | $ns\_ use-newtrace |
| 3 | set tracefile [open "trace/voip-$opt(codec)-$opt(nnode)-T$opt(try)-Node[lindex $vnode1 [expr $opt(try)-1]]-[lindex $vnode2 [expr $opt(try)-1]].tr" w] |
| 4 | $ns\_ trace-all $tracefile |

Gambar xx menunjukkan proses pembuatan *file trace* simulasi. Pada baris 2, merupakan kode untuk menggunakan format *file trace* yang baru. Kemudian pada baris 3, perintah set tracefile akan membuat *file* pada *folder* trace dengan format penamaan file voip-<jenis codec>-<jumlah node>-<replikasi ke>-<node pengirim>-<node tujuan> dengan format .tr. Pada baris 4 perintah untuk melakukan *trace* dan disimpan pada *file* $tracefile.

*File NAM* merupakan hasil dari proses simulasi pada Network Simulator 2.35 yang menampilkan objek-objek simulasi yang digunakan. Hal seperti jumlah *node,* pergerakan *node* serta proses transmisi yang disimulasikan dapat dilihat melalui *file NAM*. Berikut merupakan kode yang digunakan untuk menghasilkan *file NAM.*

|  |  |
| --- | --- |
| 1 | set namtrace [open "nam/voip-$opt(codec)-$opt(nnode)-T$opt(try)-Node[lindex $vnode1 [expr $opt(try)-1]]-[lindex $vnode2 [expr $opt(try)-1]].nam" w] |
| 2 | $ns\_ namtrace-all-wireless $namtrace $val(x) $val(y) |

Gambar xx memperlihatkan kode pembentukan *file NAM* pada simulasi. Baris 1 merupakan inisialisasi *file NAM* pada variable namtrace. Format penamaan *file NAM* sama dengan *file trace*. Kemudian pada baris 2 merupakan proses pembentukan objek-objek simulasi pada *file* namtrace.

Berikut ini merupakan kode yang digunakan untuk mengakhiri keseluruhan aktivitas pada *file trace* dan *file NAM.*

|  |  |
| --- | --- |
| 1 | proc finish {} { |
| 2 | global ns\_ tracefile |
| 3 | $ns\_ flush-trace |
| 4 | close $tracefile |
| 5 | exit 0 |
| 6 | } |

1. **Membentuk *Node***

Posisi dan pergerakan *node* yang sebelumnya telah dibuat menggunakan data *OpenStreetMap* dan SUMO akan digunakan pada *file* tcl. Berikut merupakan kode yang digunakan untuk membangkitkan sejumlah node sesuai scenario simulasi.

|  |  |
| --- | --- |
| 1 | for {set i 0} {$i < $val(nn) } {incr i} { |
| 2 | set node\_($i) [$ns\_ node] |
| 3 | $node\_($i) random-motion 0 |
| 4 | #Without random motion |
| 5 | } |

Gambar diatas merupakan kode pembangkitan *node*, perulangan pembangkitan *node* sejumlah *node* pada scenario.

|  |  |
| --- | --- |
| 1 | # M O B I L I T Y config |
| 2 | if {$opt(nnode) == 25} { |
| 3 | set val(mob) "./osmfiles/mobility/mobility\_25.tcl" |
| 4 | } elseif {$opt(nnode) == 35} { |
| 5 | set val(mob) "./osmfiles/mobility/mobility\_35.tcl" |
| 6 | } elseif {$opt(nnode) == 45} { |
| 7 | set val(mob) "./osmfiles/mobility/mobility\_45.tcl" |
| 8 | } elseif {$opt(nnode) == 55} { |
| 9 | set val(mob) "./osmfiles/mobility/mobility\_55.tcl" |
| 10 | } elseif {$opt(nnode) == 65} { |
| 11 | set val(mob) "./osmfiles/mobility/mobility\_65.tcl" |
| 12 | } elseif {$opt(nnode) == 75} { |
| 13 | set val(mob) "./osmfiles/mobility/mobility\_75.tcl" |
| 14 | } else { |
| 15 | puts "# tidak ada file MOBILITY!!" |
| 16 | exit 0 |
| 17 | } |

Pada Gambar xx, pergerakan *node* akan disesuaikan dengan jumlah *node* yang digunakan dalam simulasi. Jumlah pergerakan *node* sesuai dengan scenario simulasi yang ditentukan pada Gambar xx.

1. **Pembentukan *Traffic* *Audio***

Pembentukan *traffic audio* untuk komunikasi audio dilakukan pada *file* tcl dari patch NS2Voip++. Berikut merupakan kode pembentukan paket *audio* yang dikirimkan pada proses simulasi sesuai dengan parameter komunikasi *audio* pada proses pembentukkan komunikasi *audio*.

Konfigurasi pembentukan komunikasi *audio* pada baris 3 bernilai *off* berarti komunikasi yang dilakukan satu arah hanya dari *node* pengirim ke *node* tujuan. Baris 5 bernilai *nodebug* berarti informasi konfigurasi tidak akan ditampilkan pada terminal, baris 6 menunjukkan model voip yakni melingkupi lama waktu *talkspurt* dan waktu *silence*. Baris 7 bernilai codec yang digunakan, baris 8 menunjukan besar header dari paket baris 9 merupakan nilai *frame* yang dikirim per paket dan baris 10 merupakan jenis decoder yang digunakan. Baris 13 hingga 16 menunjukkan nilai pembangkit bilangan random untuk distribusi Weibull yang digunakan sebagai rentangan lama waktu *talkspurt* dan *silence.*

Kode di voip.tcl

|  |  |
| --- | --- |
| 1 | . . . . |
| 2 | # VoIP configuration |
| 3 | # |
| 4 | set opt(voip-bidirectional) "off" ;# VoIP bidirectional enable switch <on|off> |
| 5 | set opt(voip-debug) "nodebug" ;# VoIP debug options - "debug" or "nodebug" |
| 6 | set opt(voip-model) weibull-custom ;# VoIP VAD model #jika one-to-one, maka itu push to talk communication, half dulplex sifatnya. vad juga bisa weibull custom dan exponential |
| 7 | set opt(voip-codec) $opt(codec) ;#G.723.1;# VoIP codec |
| 8 | set opt(voip-comp-hdr-size) 3 ;# header size, in bytes |
| 9 | set opt(voip-aggr) 2 ;# number of frames per packet |
| 10 | set opt(voip-decoder-chain) $opt(buffer) ;# decoders |
| 11 | . . . . |
| 12 | # They are meaningful only if voip-model is set to weibull-custom. |
| 13 | set opt(voip-talk-scale) 0.4122 |
| 14 | set opt(voip-talk-shape) 0.824 |
| 15 | set opt(voip-silence-scale) 0.899 |
| 16 | set opt(voip-silence-shape) 1.089 |
| 17 | . . . . |

1. **Pembentukan *Background Traffic***

Implementasi pembentukan *background traffic* pada Network Simulator 2.35 menggunakan *file* cbrgen.tcl dari Network Simulator 2.35. Kode pembuatan traffic diimplementasikan dengan *script* *bash*, berikut perintah penggunaan pembentukan *background traffic.*

|  |  |
| --- | --- |
| 1 | ns cbrgen.tcl -type cbr -nn $i -seed 1 -mc 10 -rate 10.0 > traffic-$i.tcl |

Pembentukan *traffic* digunakan untuk merepresentasikan seolah-olah jaringan sedang berada dalam kondisi yang sibuk atau kondisi jaringan yang tidak optimal. *Traffic* yang digunakan berupa paket CBR (Constant Bit Rate) dengan maksimum koneksi antara 10 *node* dengan besaran paket 10.0 bytes serta dengan jenis paket agent yang dibentuk bertipe UDP (User Datagram Protocol). Perintah diatas akan menghasilkan *output* *file* yang berisikan pembentukan komunikasi antar *node* dengan besaran paketnya. Hasil dari pembangkitan *background traffic* diperlihatkan pada Gambar xx

|  |  |
| --- | --- |
| 1 | . . . . |
| 2 | # 1 connecting to 2 at time 2.5568388786897245 |
| 3 | # |
| 4 | set udp\_(0) [new Agent/UDP] |
| 5 | $ns\_ attach-agent $node\_(1) $udp\_(0) |
| 6 | set null\_(0) [new Agent/Null] |
| 7 | $ns\_ attach-agent $node\_(2) $null\_(0) |
| 8 | set cbr\_(0) [new Application/Traffic/CBR] |
| 9 | $cbr\_(0) set packetSize\_ 512 |
| 10 | $cbr\_(0) set interval\_ 0.10000000000000001 |
| 11 | $cbr\_(0) set random\_ 1 |
| 12 | $cbr\_(0) set maxpkts\_ 10000 |
| 13 | $cbr\_(0) attach-agent $udp\_(0) |
| 14 | $ns\_ connect $udp\_(0) $null\_(0) |
| 15 | $ns\_ at 2.5568388786897245 "$cbr\_(0) start" |
| 16 | . . . . |

Gambar xx menunjukkan penggunaan *background traffic* yang dibentuk. Penggunaan *background traffic* akan disesuaikan dengan jumlah *node* pada proses simulasi.

|  |  |
| --- | --- |
| 1 | # B G T R A F F I C config |
| 2 | if {$opt(nnode) == 25} { |
| 3 | set val(cp) "./traffic-25.tcl" |
| 4 | } elseif {$opt(nnode) == 35} { |
| 5 | set val(cp) "./traffic-35.tcl" |
| 6 | } elseif {$opt(nnode) == 45} { |
| 7 | set val(cp) "./traffic-45.tcl" |
| 8 | } elseif {$opt(nnode) == 55} { |
| 9 | set val(cp) "./traffic-55.tcl" |
| 10 | } elseif {$opt(nnode) == 65} { |
| 11 | set val(cp) "./traffic-65.tcl" |
| 12 | } elseif {$opt(nnode) == 75} { |
| 13 | set val(cp) "./traffic-75.tcl" |
| 14 | } else { |
| 15 | puts "# tidak ada file traffic!!" |
| 16 | exit 0 |
| 17 | } |

1. **Pembentukan Koneksi**

Pembentukan koneksi dilakukan pada *file* pembangkitan *traffic audio* yakni *file* voip.tcl. Pada fungsi scenario ditentukan juga waktu mulai komunikasi dilakukan, hal tersebut dilakukan pada baris 3 Gambar xx. Baris 4 memanggil *file* vnode.tcl, *file* ini berisikan *node-node* mana saja yang akan saling berkomunikasi sesuai dengan jumlah *node* yang disimulasikan. Baris 5 menset waktu berhenti untuk komunikasi *audio*. Perulangan pada baris 7 akan dilakukan sebanyak jumlah komunikasi yang dibangun. Pada penelitian ini, jumlah komunikasi yang dilakukan hanya antara 2 *node* atau 1 komunikasi. Baris 9 akan memanggil fungsi create voip dengan parameter $fid sebagai id voip dan $start sebagai waktu mulai komunikasi dan $stop sebagai waktu selesai komunikasi.

Pada proses create voip ini pembuatan VoIPSource melingkupi voip-model yakni rentangan waktu *talkspurt* dan waktu *silence*, pembentukan VoIPHeader mengatur ukuran *header* dari paket yang dibentuk dan pembentukan VoIPEncoder meliputi jenis codec yang digunakan dan pembuatan jenis decoder yang digunakan. Fungsi create\_udp pada baris ke 10 akan membentuk koneksi diantara *node*. Parameter yang digunakan yakni *node* sumber, *node* tujuan, id, dan jenis aplikasi. Fungsi ini akan menentukan ukuran paket yang dibentuk berdasarkan jenis codec, decoder dan ukuran header dari paket. Nilai $vnode1 dan $vnode2 didapatkan dari *file* vnode.tcl. Pada replikasi pertama, $vnode1 bernilai index ke-0 dari list vnode1, begitu juga nilai $vnode2 bernilai index ke-0 dari list vnode2 pada replikasi pertama.

|  |  |
| --- | --- |
| 1 | proc scenario {} { |
| 2 | global ns\_ opt node\_ |
| 3 | set fid 1 |
| 4 | set start {10.0} #30 |
| 5 | source vnode.tcl |
| 6 | set stop [expr $opt(duration)-5] #90 |
| 7 |  |
| 8 | for { set i 0} { $i < $opt(voipflows)} { incr i } { |
| 9 | create\_voip $fid $start $stop |
| 10 | set f [create\_udp $node\_([lindex $vnode1 [expr $opt(try)-1]]) $node\_([lindex $vnode2 [expr $opt(try)-1]]) $fid "voip"] |
| 11 | } |
| 12 | } |

Gambar xx menunjukkan list dari vnode1 dan vnode2 yang akan menjadi *node* sumber dan *node* tujuan pada proses komunikasi dan *traffic audio* yang ditbentuk.

|  |  |
| --- | --- |
| 1 | #set source node dan destination node |
| 2 | if { $opt(nnode) == 50 } { |
| 3 | set vnode1 {1 2 3 4 5 6 7 8 9 10} |
| 4 | set vnode2 {49 48 47 46 45 44 43 42 41 40} |
| 5 | } elseif { $opt(nnode) == 25 } { |
| 6 | set vnode1 {1 2 3 4 5 6 7 8 9 10} |
| 7 | set vnode2 {24 23 22 21 20 19 18 17 16 15} |
| 8 | } elseif { $opt(nnode) == 35 } { |
| 9 | set vnode1 {1 2 3 4 5 6 7 8 9 10} |
| 10 | set vnode2 {34 33 32 31 30 29 28 27 26 25} |
| 11 | } elseif { $opt(nnode) == 45 } { |
| 12 | set vnode1 {1 2 3 4 5 6 7 8 9 10} |
| 13 | set vnode2 {44 43 42 41 40 39 38 37 36 35} |
| 14 | } elseif { $opt(nnode) == 55 } { |
| 15 | set vnode1 {1 2 3 4 5 6 7 8 9 10} |
| 16 | set vnode2 {54 53 52 51 50 49 48 47 46 45} |
| 17 | } elseif { $opt(nnode) == 65 } { |
| 18 | set vnode1 {1 2 3 4 5 6 7 8 9 10} |
| 19 | set vnode2 {64 63 62 61 60 59 58 57 56 55} |
| 20 | } elseif { $opt(nnode) == 75 } { |
| 21 | set vnode1 {1 2 3 4 5 6 7 8 9 10} |
| 22 | set vnode2 {74 73 72 71 70 69 68 67 66 65} |
| 23 | } elseif { $opt(nnode) == 70 } { |
| 24 | set vnode1 {1 2 3 4 5 6 7 8 9 10} |
| 25 | set vnode2 {69 68 67 66 65 64 63 62 61 60} |
| 26 | } else { |
| 27 | set vnode1 {2 6 10 14 18} |
| 28 | set vnode2 {3 7 11 15 19} |
| 29 | } |

1. **Pengujian dan Evaluasi Sistem**

Proses simulasi akan menghasilkan keluaran berupa *file trace* dengan format .tr yang menyimpan informasi segala *event* atau aktivitas dalam proses simulasi. *File trace* ini digunakan untuk melakukan analisis bagaimana pengaruh peningkatan jumlah *node* terhadap kualitas komunikasi dengan *codec* G.711 menggunakan protokol *routing* AOMDV dan PA\_AOMDV. Analisis hasil simulasi ditinjau dari rentang nilai MOS E-Model dari hasil pengujian setiap scenario. Perhitungan nilai MOS E-Model dipengaruhi oleh nilai *delay* dan *packet loss.*

1. **Pengujian Protokol AOMDV dan PA\_AOMDV dengan *codec* G.711**

Tabel 4.1 merupakan hasil dari pengujian *delay* dan *packet loss* dari scenario dengan protokol *routing* AOMDV dan PA\_AOMDV. Masing-masing skenario direplikasi sebanyak 10 kali. Pada penggunaan *codec* G.711 menghasilkan rata-rata *delay* yang dihasilkan oleh protokol AOMDV sebesar 3.782 millisecond sedangkan protokol PA\_AOMDV menghasilkan rata-rata *delay* sebesar 4.307 millisecond, perbedaan nilai *delay* ini disebabkan karena protokol PA\_AOMDV akan melakukan pengecekan terhadap jumlah *hop count i* dan *j* setiap ada paket yang datang. Ukuran nilai *delay* yang dihasilkan pada scenario ini termasuk dalam kategori bagus mengacu pada Tabel 3.x mengenai rentangan kategori nilai *delay.*

Tabel 4.1 juga menunjukkan nilai *packet loss*. Rata-rata *packet* *loss* dari skenario pada Tabel 4.1 menghasilkan besaran 7.6% dan 6.2% untuk protokol AOMDV dan PA\_AOMDV. *Packet loss* pada simulasi ini dipengaruhi oleh perubahan topologi sehingga menyebabkan kerusakan pada rute pengiriman paket. Hal ini diperlihatkan pada *file trace* protokol AOMDV pada replikasi ke-4, dimana terdapat paket dengan tipe UDP yang di*drop* oleh *node* karena tidak terdapat rute yang cadangan ketika rute yang lama rusak karena perubahan topologi. Hal ini diperlihatkan pada Gambar xx, *event* d atau *drop* pada detik ke 79.010987059 paket dari *node* 4 sebagai *node* sumber menuju ke *node* 21 sebagai *node* tujuan tidak dapat diteruskan, dengan error NRTE atau “DROP\_RTR\_NO\_ROUTE” yang berarti tidak ada rute untuk mencapai *node* tersebut. Pada baris 2 diperlihatkan bahwa *node* terakhir pada rute tersebut akan mengirimkan pesan *broadcast error*, sehingga *node* sumber akan melakukan proses penemuan rute kembali.

|  |  |
| --- | --- |
| 1 | d -t 79.010987059 -Hs 10 -Hd 10 -Ni 10 -Nx 2059.89 -Ny 2705.21 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw NRTE -Ma 13a -Md a -Ms 4 -Mt 800 -Is 4.0 -Id 21.0 -It udp -Il 183 -If 1 -Ii 942 -Iv 29 |
| 2 | s -t 79.010987059 -Hs 10 -Hd 0 -Ni 10 -Nx 2059.89 -Ny 2705.21 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 10.255 -Id -1.255 -It AOMDV -Il 32 -If 0 -Ii 0 -Iv 1 -P aomdv -Pt 0x8 -Ph 1 -Pd 21 -Pds 0 -Pl 0.000000 -Pc ERROR |

Gambar . *Packet drop* pada *file trace* error NRTE.

Selain error NRTE, pada simulasi ini *error* lain juga mempengaruhi nilai *packet loss* disebabkan karena CBK atau “DROP\_RTR\_MAC\_CALLBACK”. Hal ini diperlihatkan pada Gambax xx, error CBK disebabkan karena lapisan MAC tidak dapat mengirimkan paket dan selanjutnya menginformasikan kelapisan atas bahwa terjadi kegagalan transmisi. Hasil nilai *packet loss* yang didapatkan pada scenario Tabel 4.1 berdasarkan Tabel 3.x mengenai rentangan nilai *packet loss* termasuk dalam rentangan yang bagus.

|  |  |
| --- | --- |
| 1 | d -t 71.057908780 -Hs 4 -Hd 21 -Ni 4 -Nx 2218.06 -Ny 2689.02 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw CBK -Ma 13a -Md 15 -Ms 4 -Mt 800 -Is 4.0 -Id 21.0 -It udp -Il 183 -If 1 -Ii 722 -Iv 30 |

Gambar . *Packet drop* pada *file trace* error CBK

Kode error packet loss NRTE, CBK, ARP/IFQ

Tabel . Hasil pengujian *delay* dan *packet* loss *codec* G.711

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2300m x 3000m | Replikasi | 25 *Node*  G.711 | | | |
| AOMDV | | PA\_AOMDV | |
| *Delay* (ms) | *Packet loss* | *Delay* (ms) | *Packet loss* |
| 1 | 3.05 | 0.00 | 3.08 | 0.00 |
| 2 | 3.62 | 0.10 | 4.13 | 0.16 |
| 3 | 4.35 | 0.06 | 8.49 | 0.13 |
| 4 | 3.88 | 0.21 | 3.98 | 0.16 |
| 5 | 3.08 | 0.00 | 3.06 | 0.00 |
| 6 | 3.65 | 0.02 | 4.18 | 0.03 |
| 7 | 3.10 | 0.00 | 3.09 | 0.00 |
| 8 | 3.11 | 0.00 | 3.07 | 0.00 |
| 9 | 4.94 | 0.22 | 5.14 | 0.07 |
| 10 | 5.04 | 0.15 | 4.85 | 0.07 |
| Rata-rata | | 3.782 | 0.076 | 4.307 | 0.062 |

Dari hasil pengujian pada tabel 4.1 hasil rata-rata *delay* termasuk dalam rentang nilai yang sangat baik dan *packet loss* termasuk dalam rentang nilai yang baik mengacu pada tabel 3.x dan 3.x mengenai rentang nilai *delay* dan *packet loss.* Hasil *delay* dan *packet loss* dari scenario tabel 4.1 akan mempengaruhi nilai *Mean Opinion Score* (MOS). Perhitungan MOS pada protokol AOMDV dan PA\_AOMDV dengan 25 *node* diperlihatkan pada tabel 4.2.

Pada Tabel 4.2 pengujian dengan metode MOS E-Model protokol AOMDV dan PA\_AOMDV dengan scenario 25 *node* dan *codec* G.711 menghasilkan rata-rata nilai R 76.47 dan MOS 3.75 pada penggunaan protokol AOMDV. Kemudian pada penggunaan protokol PA\_AOMDV menghasilkan nilai rata-rata R 78.09 dan MOS 3.84. Nilai R dan MOS pada penggunaan protokol AOMDV dan PA\_AOMDV tidak jauh berbeda, pada scenario ini, penggunaan protokol PA\_AOMDV memiliki hasil yang sedikit lebih baik dibandingkan penggunaan protokol AOMDV, hal ini dikarenakan nilai *packet loss* pada penggunaan protokol AOMDV lebih besar daripada protokol PA\_AOMDV. Hasil dari pengujian MOS E-Model ini sangat dipengaruhi oleh nilai *packet loss* pada proses komunikasi. Semakin tinggi nilai *packet loss* maka nilai MOS akan semakin rendah. Berdasarkan hasil pengujian pada tabel 4.2, nilai R dan MOS pada protokol AOMDV dan PA\_AOMDV dengan 25 *node* dan *codec* G.711 serta penggunaan menghasilkan nilai yang termasuk dalam kategori “*some user dissatisfied*” mengacu pada tabel 3.x mengenai rentangan nilai R dan MOS menurut ITU-T.

Tabel . Hasil pengujian MOS E-Model pada scenario 25 *node* *codec* G.711

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2300m x 3000m | Replikasi | G.711  25 *Node* | | | |
| AOMDV | | PA\_AOMDV | |
| R | MOS | R | MOS |
| 1 | 94.13 | 4.43 | 94.13 | 4.43 |
| 2 | 66.62 | 3.43 | 57.39 | 2.96 |
| 3 | 74.84 | 3.82 | 61.54 | 3.18 |
| 4 | 51.41 | 2.65 | 57.39 | 2.96 |
| 5 | 94.13 | 4.43 | 94.13 | 4.43 |
| 6 | 86.24 | 4.24 | 82.95 | 4.13 |
| 7 | 94.13 | 4.43 | 94.13 | 4.43 |
| 8 | 94.13 | 4.43 | 94.13 | 4.43 |
| 9 | 50.32 | 2.59 | 72.54 | 3.71 |
| 10 | 58.72 | 3.03 | 72.55 | 3.71 |
| Rata-rata | | 76.47 | 3.75 | 78.09 | 3.84 |

Pada Tabel 4.1 dan 4.2 diperlihatkan sebelumnya hasil pengujian *delay, packet loss* dan MOS E-Model pada scenario 25 *node* protokol AOMDV dan PA\_AOMDV dengan penggunaan *codec* G.711. Pada scenario tersebutmenghasilkan nilai MOS dalam rentangan *"some user satisfied"* untuk kedua protokol dan *codec* yang digunakan. Untuk melihat pengaruh dari jumlah *node* terhadap kualitas komunikasi yang dilakukan, maka penambahan jumlah *node* dilakukan.

Tabel 4.3 menampilkan hasil pengujian dengan penambahan jumlah node ketika simulasi pada protokol AOMDV dan PA\_AOMDV *codec* G.711. Masing-masing scenario *node* dilakukan sebanyak 10 replikasi, dengan detail tabel hasil pengujian *delay, packet loss* dan MOS E-Model akan disertakan dalam lampiran. Pengujian dengan penambahan jumlah *node* dinilai berhasil meningkatkan nilai kualitas komunikasi *audio* yang dilakukan. Hal ini terlihat pada hasil pengujian MOS pada Tabel 4.3, seiring penambahan jumlah *node*, nilai MOS meningkat.

Pada pengujian dengan protokol AOMDV dan PA\_AOMDV, nilai MOS pada PA\_AOMDV sedikit lebih baik dibandingkan dengan nilai MOS pada AOMDV. Tabel 4.3 menunjukkan PA\_AOMDV lebih unggul meskipun tidak terlalu signifikan pada pengujian dengan *codec* G.711. Pada scenario dengan jumlah *node* 45, protokol PA\_AOMDV tetap mengalami peningkatan dari scenario sebelumnya, namun pada scenario 45 *node,* nilai MOS dari AOMDV sedikit lebih tinggi daripada PA\_AOMDV. Hal ini disebabkan pada scenario 45 *node* PA\_AOMDV mengalami lebih banyak *packet drop* yang disebabkan oleh error CBK pada proses simulasi.

Tabel . Hasil pengujian MOS E-Model dengan penambahan jumlah *node* pada *codec* G.711

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2300m x 3000m | *nodes* | G.711 | | | |
| AOMDV | | PA\_AOMDV | |
| R | MOS | R | MOS |
| 25 | 76.47 | 3.75 | 78.09 | 3.84 |
| 35 | 80.45 | 3.93 | 81.38 | 3.97 |
| 45 | 83.32 | 4.03 | 83.31 | 3.99 |
| 55 | 86.71 | 4.19 | 87.04 | 4.19 |
| 65 | 88.29 | 4.21 | 90.58 | 4.29 |
| 75 | 90.72 | 4.32 | 91.07 | 4.33 |

Tabel 4.3 dan Gambar 4.3 menunjukan representasi hasil pengujian protokol AOMDV dan PA\_AOMDV dengan *codec* G.711 dalam bentuk tabel dan grafik. Hasil pengujian dengan *codec* G.711 menunjukkan peningkatan pada setiap scenario. Dalam kondisi lingkungan simulasi yang sama, dengan *codec* yang berbeda, apakah dapat menghasilkan kualitas MOS yang lebih baik. Maka dari itu pengujian dengan *codec* jenis lainnya dilakukan pada penelitian ini. Pengujian selanjutnya menggunakan *codec* G.723.1.

Gambar . Grafik perbandingan nilai MOS protokol AOMDV dan PA\_AOMDV *codec* G.711

1. **Pengujian Protokol AOMDV dan PA\_AOMDV dengan *codec* G.723.1**

Pengujian dengan *codec* G.7231 yang memiliki ukuran *payload* atau ukuran paket yang lebih kecil, namun kompresi yang lebih tinggi daripada *codec* G.711. Ukuran paket yang kecil, memungkinkan *node* memproses paket dengan lebih cepat, hal ini dapat mempengaruhi nilai *delay* pada pengujian. Tabel 4.4 memperlihatkan hasil pengujian dengan menggunakan *codec* G.723.1 pada protokol AOMDV dan PA\_AOMDV pada scenario 25 *node*. Dari 10 replikasi pada pengujian, rata-rata nilai *delay* yang dihasilkan adalah 2.71 millisecond dan 2.735 millisecond untuk protokol AOMDV dan PA\_AOMDV.

Tabel 4.4 juga menunjukkan hasil pengujian terhadap nilai *packet loss*, pengujian menghasilkan rata-rata *packet loss* sebesar 3.7% dan 6.5% untuk protokol AOMDV dan PA\_AOMDV. Tingginya nilai *packet loss* pada protokol PA\_AOMDV disebabkan karena banyaknya error CBK atau “DROP\_RTR\_MAC\_CALLBACK” yang terjadi. Dari hasil pengujian, rata-rata nilai *delay* dan *packet loss* pada scenario ini termasuk dalam kategori sangat bagus untuk nilai *delay* dan bagus untuk nilai *packet loss* mengacu pada tabel 3.x dan 3.x mengenai rentangan nilai *delay* dan *packet loss.*

Tabel . Hasil pengujian *delay* dan *packet loss* pada *codec* G.723.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2300m x 3000m | Replikasi | 25 *Node*  G.723.1 | | | |
| AOMDV | | PA\_AOMDV | |
| *Delay* (ms) | *Packet loss* | *Delay* (ms) | *Packet loss* |
| 1 | 2.15 | 0.00 | 2.10 | 0.00 |
| 2 | 2.96 | 0.05 | 2.60 | 0.11 |
| 3 | 2.67 | 0.08 | 3.01 | 0.07 |
| 4 | 3.44 | 0.11 | 3.41 | 0.09 |
| 5 | 2.11 | 0.00 | 2.14 | 0.00 |
| 6 | 2.11 | 0.00 | 2.24 | 0.00 |
| 7 | 2.13 | 0.00 | 2.18 | 0.00 |
| 8 | 2.14 | 0.00 | 2.16 | 0.00 |
| 9 | 3.81 | 0.05 | 3.83 | 0.16 |
| 10 | 3.58 | 0.08 | 3.68 | 0.22 |
| Rata-rata | | 2.71 | 0.037 | 2.735 | 0.065 |

Berdasarkan hasil pengujian *delay* dan *packet loss* pada Tabel 4.4, maka dapat dilakukan perhitungan terhadap nilai kualitas komunikasi atau nilai MOS untuk scenario 25 *node* dengan *codec* G.723.1 pada protokol AOMDV dan PA\_AOMDV. Tabel 4.5 memperlihatkan hasil perhitungan MOS menghasilkan rata-rata nilai MOS 4.06 dan 3.83 untuk protokol AOMDV dan PA\_AOMDV. Pada scenario ini AOMDV memiliki nilai MOS yang lebih tinggi daripada PA\_AOMDV, hal ini terjadi karena *packet loss* pada AOMDV lebih kecil dibandingkan dengan PA\_AOMDV, sehingga nilai MOS pada AOMDV bisa lebih tinggi. Dari hasil tersebut, rata-rata nilai MOS termasuk dalam kategori “*some user dissastified*” untuk MOS PA\_AOMDV dan “*satisfied*” untuk MOS AOMDV.

Tabel . Hasil pengujian MOS E-Model pada scenario 25 *node codec* G.723.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2300m x 3000m | Replikasi | G.723.1  25 *Node* | | | |
| AOMDV | | PA\_AOMDV | |
| R | MOS | R | MOS |
| 1 | 94.15 | 4.43 | 94.15 | 4.43 |
| 2 | 77.34 | 3.92 | 64.90 | 3.35 |
| 3 | 70.48 | 3.62 | 72.59 | 3.72 |
| 4 | 64.88 | 3.35 | 68.49 | 3.53 |
| 5 | 94.15 | 4.43 | 94.15 | 4.43 |
| 6 | 94.15 | 4.43 | 94.15 | 4.43 |
| 7 | 94.15 | 4.43 | 94.15 | 4.43 |
| 8 | 94.15 | 4.43 | 94.15 | 4.43 |
| 9 | 77.32 | 3.92 | 57.39 | 2.96 |
| 10 | 70.46 | 3.62 | 50.35 | 2.59 |
| Rata-rata | | 83.12 | 4.06 | 78.45 | 3.83 |

Untuk mengetahui pengaruh penambahan jumlah *node* terhadap kualitas komunikasi *audio* atau nilai MOS pada penelitian ini, dengan *codec* G.723.1 maka pengujian dengan penambahan *node* dilakukan. Pada Tabel 4.6 diperlihatkan hasil pengujian pada *codec* G.723.1 dengan penambahan jumlah *node*. Peningkatan nilai MOS pada protokol AOMDV terlihat dari scenario 55 *node* hingga 75 *node*. Pada scenario awal, AOMDV menghasilkan nilai MOS yang cukup tinggi, kemudian mengalami penurunan pada scenario 45 *node*. Pada scenario selanjutnya, AOMDV mengalami peningkatan nilai MOS hingga scenario terakhir. Pada protokol PA\_AOMDV peningkatan nilai MOS mulai dari scenario 25 *node* hingga 45 *node*, dan sedikit menurun di 55 *node* namun kembali meningkat hingga scenario terakhir.

Tabel . Hasil pengujian MOS E-Model dengan penambahan jumlah *node* pada *codec* G.723.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2300m x 3000m | *nodes* | G.723.1 | | | |
| AOMDV | | PA\_AOMDV | |
| R | MOS | R | MOS |
| 25 | 83.12 | 4.06 | 78.45 | 3.83 |
| 35 | 83.40 | 4.05 | 81.83 | 3.99 |
| 45 | 81.90 | 3.96 | 88.45 | 4.23 |
| 55 | 86.25 | 4.16 | 85.53 | 4.15 |
| 65 | 87.54 | 4.17 | 90.60 | 4.29 |
| 75 | 92.43 | 4.38 | 93.69 | 4.42 |

Gambar 4.4 menunjukan perbandingan protokol AOMDV dan PA\_AOMDV dengan *codec* G.723.1. Grafik perbandingan merepresentasikan nilai pada Tabel 4.6, nilai kualitas MOS pada pengujian dengan *codec* G.723.1 tidak selalu selaras dengan penambahan jumlah *node*. Pada pengujian dengan *codec* G.723.1 terdapat sedikit penurunan di beberapa scenario. Penurunan nilai MOS, sangat dipengaruhi oleh nilai *packet loss* pada proses simulasi.

Berdasarkan pengujian yang dilakukan pada 4.3.1 dan 4.3.2 maka dapat dibentuk grafik perbandingan antara *codec* G.711 dan *codec* G.723.1

Gambar . Grafik perbandingan nilai MOS protokol AOMDV dan PA\_AOMDV *codec* G.723.1

Gambar . Grafik perbandingan nilai MOS protokol AOMDV dan PA\_AOMDV *codec* G.723.1