TUGAS AKHIR - KI141502

Analisis Kinerja Model Propagasi FreeSpace dan TwoRayGround pada Ad Hoc On Demand Vector(AODV) Routing pada MANET

Yosua Nove Pratama
NRP 05111340000071

Dosen Pembimbing
Dr.Eng. RADITYO ANGGORO, S.Kom., M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI141502

Analisis Kinerja Model Propagasi Free Space dan TwoRayGround pada Ad Hoc On Demand Vector(AODV) Routing pada MANET

Yosua Nove Pratama
NRP 05111340000071

Dosen Pembimbing
Dr.Eng. RADITYO ANGGORO, S.Kom., M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknik Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – KI141502

Analysis FreeSpace and Tworayground Propagation Model with Ad Hoc On Demand Vector (AODV) on MANET

Yosua Nove Pratama
NRP 05111340000071

Advisor
Dr.Eng. RADITYO ANGGORO, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Electronics Technology and Informatics Intelligent
Institut Teknologi Sepuluh Nopember

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

Analisis Kinerja Model Propagasi FreeSpace dan TwoRayGround pada Ad Hoc On Demand Vector(AODV) pada MANET

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Arsitektur Jaringan Komputer
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informasi Cerdas
Institut Teknologi Sepuluh Nopember

Oleh :
Yosua Nove Pratama
NRP : 05111340000071

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Dr.Eng. RADITYO ANGGORO, S.Kom., M.Sc.
NIP: 198410162008121002 ..... (pembimbing 1)

SURABAYA
JANUARI 2020

[Halaman ini sengaja dikosongkan]

Analisis Kinerja Model Propagasi FreeSpace dan TwoRayGround pada Ad Hoc On Demand Vector (AODV) pada MANET

Nama Mahasiswa : Yosua Nove Pratama
NRP : 05111340000071
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing 1 : Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Dosen Pembimbing 2 : -

ABSTRAK

Mobile Ad Hoc Network (MANET) merupakan jaringan wireless dengan node yang tidak memiliki router yang tetap atau berpindah-pindah. Adanya node yang bergerak menyebabkan topologi serta konektivitas setiap node pada jaringan MANET berubah secara dinamis. Oleh karena itu diperlukan routing protokol khusus yang digunakan untuk meng-cover konektivitas setiap node pada jaringan supaya dapat memberi jalur routing yang optimal.

Ada beberapa routing protocol yang ada pada MANET, salah satunya adalah *Ad Hoc On Demand Vector* atau AODV. Ad Hoc on Demand Vector merupakan kategori reactive routing protocol yang hanya menyimpan informasi routing seputar path atau host yang aktif. Didalam AODV, apabila node asal tidak menemukan rute yang tersedia, maka node tersebut akan melakukan route discovery.

Implementasi tersebut menggunakan Network Simulator 2(NS-2). Hasil akhir dari Penelitian ini adalah hasil analisis kinerja dari model propagasi *TwoRayGround* dan *FreeSpace* pada protokol routing AODV di lingkungan MANET berdasarkan *Packet Delivery Ratio*(PDR), *End-to-End Delay*(E2E) serta *Routing Overhead*(RO). Kemudian hasil dari dua jenis propagasi tersebut dibandingkan.

***Kata kunci**: MANET, AODV, NS-2, TwoRayGround, FreeSpace*

[Halaman ini sengaja dikosongkan]

Analysis Tworayground and Freespace Propagation Model with Ad Hoc On Demand Vector (AODV) on MANET

Student Name	: Yosua Nove Pratama
Student ID	: 05111340000071
Major	: Teknik Informatika FTIf-ITS
Advisor 1	: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.
Advisor 2	: -

ABSTRACT

MANET is a Wireless Network with node that do not have a fixed router. The existence of a moving node causes the topology and connectivity of each node on the MANET to change dinamically. Therefore we need a special routing protocol that is used to cover the connectivity of each node on the network in order to provide optimal routing path.

Ther are many routing protocol that implementation on MANET, such as Ad Hoc On Demand Vector(AODV). Ad Hoc on Demand Vector is a reactive routing protocol category that only stores routing information about the active path or host. In AODV, if the origin node does not find an available route, then the node will do a route discovery.

Implementation on MANET was base on simulation. Simulation will be using Network Simulator 2 with TwoRayGround and FreeSpace propagation model on AODV Protocol and we will know the Packet Delivery Ratio(PDR), Routing Overhead(RO), and End to End Delay.

Keyword : MANET, AODV, NS-2, TwoRayGround, FreeSpace

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul :

"Analisis Kinerja Model Propagasi TwoRayGround dan FreeSpace pada Ad Hoc On Demand Vector (AODV) di lingkungan MANET"

Harapan dari penulis semoga apa yang tertulis di dalam buku Tugas Akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini, serta dapat memberikan kontribusi yang nyata.

Dalam pelaksanaan dan pembuatan Tugas Akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT atas segala nikmat dan rizki yang telah di berikan kepada penulis.
2. Keluarga yang senantiasa memberikan dukungan penuh baik secara moriil maupun materiil.
3. Bapak Dr.Eng Radityo Anggoro S.Kom., M,Sc. selaku dosen pembimbing pertama yang telah bersedia meluangkan waktu untuk membimbing dan memotivasi penulis serta memberi arahan dalam mengerjakan tugas akhir.
4. Ibu Dr. Eng. Nanik Suciati, S.Kom, M.Kom. selaku dosen wali dari penulis yang memberikan arahan selama menjalani perkuliahan.
5. Bapak/Ibu dosen, staf dan karyawan Jurusan Teknik Informatika ITS yang telah banyak memberikan dukungan, ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
6. Admiral Budi Arviansyah Wilarsani dan Nadia Zhafirah Usna yang selalu membantu penulis dalam mengerjakan tugas akhir ini.

7. Teman Teman Panitia SCHEMATICS HMTC 2014, departemen Media Informasi HMTC 2014/2015 yang telah memberikan pengalaman berarti selama kuliah.
8. Teman Teman angkatan 2013 dan angkatan 2014 yang sudah membantu penulis selama di kampus.
9. Serta pihak-pihak lain yang tidak dapat disebutkan disini yang telah banyak membantu penulis dalam penyusunan Tugas Akhir ini.

Penulis telah berusaha sebaik-baiknya dalam menyusun Tugas Akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, 4 Januari 2020
Penulis

Yosua Nove Pratama

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1. Latar Belakang

Pada saat ini kemajuan teknologi informasi berkembang semakin pesat. Teknologi nirkabel(wireless) yang berkembang inilah yang menyebabkan perangkat mobile seperti handphone, tablet, notebook, serta perangkat lain juga ikut berkembang. Hal tersebut menyebabkan perangkat komunikasi dapat berhubungan satu sama lain tanpa infrastruktur jaringan yang tetap. Maka dari itu manusia menggunakan salah satu infrastruktur komunikasi yang disebut MANET.

MANET merupakan jaringan wireless yang memiliki node yang tidak memiliki router yang tetap atau berpindah-pindah. Setiap node yang ada dalam jaringan ini melakukan routing dan pengiriman data. Setiap node bertanggung jawab atas proses routing discovery dan routing maintenance untuk setiap jalur pengiriman data ke node yang lain. Hal ini disebabkan karena setiap node yang ada pada jaringan selalu bergerak. Sehingga menyebabkan topologi serta konektivitas setiap node pada jaringan MANET berubah secara dinamis. Oleh karena itu diperlukan routing protokol khusus yang digunakan untuk meng-cover konektivitas setiap node pada jaringan supaya dapat memberi jalur routing yang optimal. [1]

Ada beberapa routing protocol yang ada pada MANET, salah satunya adalah AODV. Ad Hoc on Demand Vector merupakan kategori reactive routing protocol yang hanya menyimpan informasi routing seputar path atau host yang aktif. Didalam AODV, apabila node asal tidak menemukan rute yang tersedia, maka node tersebut akan melakukan route discovery.

Setiap node yang ada dalam jaringan ini bertanggung jawab menyimpan rute didalam routing table. Pada saat pengiriman data jika terjadi perubahan topologi yang mengakibatkan suatu node tidak bisa dituju berdasarkan informasi rute yang ada di routing table, maka node tersebut akan mengirimkan Request Error(RRER) ke node sebelumnya.Setiap node yang memperoleh RRER akan menghapus informasi rute yang ada di routing table.Jika informasi itu sampai di node asal, maka node asal akan melakukan routing discovery. [2]

Pada tugas akhir ini akan diimplementasikan Ad Hoc On Distance Demand Vector (AODV) routing di lingkungan MANET dengan membandingkan dua jenis propagasi Free Space dan Two Ray Ground. Implementasi tersebut menggunakan Network Simulator 2(NS-2). Hasil akhir dari tugas akhir ini adalah hasil analisis kinerja dari model propagasi Two Ray Ground dan Free Space pada protokol routing AODV di lingkungan MANET berdasarkan Packet Delivery Ratio(PDR), End-to-End Delay serta Routing Overhead. Kemudian hasil dari dua jenis propagasi tersebut dibandingkan.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana cara analisis kinerja mode propagasi Free Space pada Routing Ad Hoc On Demand Vector Routing pada MANET?
2. Bagaimana cara analisis kinerja mode propagasi Two Ray Ground pada Routing Ad Hoc On Demand Vector Routing pada MANET?
3. Bagaimana cara membandingkan Kinerja Model Propagasi Free Space dan Two Ray Ground pada Ad Hoc On Demand Vector(AODV) Routing pada MANET?

1.3. Batasan Permasalahan

Batasan masalah pada pembuatan tugas akhir ini adalah sebagai berikut :

1. Protokol Routing hanya dijalankan dan diujicobakan pada aplikasi Network Simulator 2 (NS 2).
2. Protokol Routing yang diujicobakan adalah Ad Hoc On Demand Vector(AODV).
3. Lingkungan jaringan yang digunakan untuk uji coba adalah Mobile Ad Hoc Network (MANET).
4. Model Propagasi yang digunakan dalam pengerjaan ini adalah Two Ray Ground dan Free Space.

1.4. Tujuan

Tujuan dari pembuatan tugas akhir ini adalah untuk memberikan hasil analisis kinerja model propagasi Two Ray Ground dan Free Space pada routing AODV di lingkungan MANET dengan menggunakan aplikasi Network Simulator 2 (NS-2).

1.5. Manfaat

Dengan dibuatnya tugas akhir ini dapat memberikan manfaat berupa hasil studi perbandingan performansi antara model propagasi Two Ray Ground dan Free Space pada routing AODV pada lingkungan MANET dengan parameter Packet Deilvery Ratio (PDR), End-to-End Delay dan Routing Overhead (RO).

1.6. Metodologi

Dalam pengerjaan tugas akhir ini langkah-langkah yang dikerjakan yaitu:

1.6.1. Penyusunan proposal Tugas Akhir

Proposal ini berisi mengenai deskripsi tugas akhir mengenai Analisis Kinerja Model Propagasi Free Space dan Two

Ray Ground pada Ad Hoc On Demand Vector(AODV) Routing pada MANET.

1.6.2. Studi literatur

Bagian studi literatur akan dipelajari sejumlah referensi dan tool yang digunakan untuk pengerjaan tugas akhir ini yaitu mengenai propagasi Free Space dan Two Ray Ground, Mobile Ad Hoc Network (MANET), routing protocol AODV, Network Simulator 2 (NS-2), Generator File Node-Movement dan Traffic-Connection Pattern serta AWK.

1.6.3. Implementasi

Pada tahap ini akan dibuat prototype yang merupakan rancangan dasar serta dan sistem yang akan dibuat. Selain itu akan dibuat desain sistem dan desain proses yang ada.

1.6.4. Pengujian dan Evaluasi

Pada tahap ini dilakukan uji coba dari sistem yang sudah dirancang. Pengujian dilakukan dengan pembuatan skenario node-movement yang menggunakna routing Ad Hoc On Demand Vector (AODV) serta Traffic Connection Pattern Generation. Kemudian routing tersebut dijalankan pada NS-2 dan akan menghasilkan trace file. Kemudian trace file tersebut akan dianalisis Packet Deilvery Ratio (PDR), End-to-End Delay dan Routing Overhead (RO) untuk mengetahui performa dari routing protocol yang telah dibuat serta untuk membandingkan dua jenis propagasi yang dirancang.

1.6.5. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas

akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7. Sistematika Penulisan

Secara garis besar, sistematika buku Tugas Akhir akan disusm seperti berikut:

Bab I Pendahuluan

Bab ini berisi penjelasan tentang latar belakang masalah, tujuan dan manfaat pembuatan Tugas Akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

Bab II Tinjauan Pustaka

Bab ini membahas beberapa teori yang akan menunjang pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan metode yang nantinya akan diimplementasikan dan dilakukan pengujian dari aplikasi yang dibangun.

Bab IV Implementasi

Bab ini berisi implementasi dari rancangan sistem yang sudah dilakukan pada tahap perancangan. Penjelasan scenario yang dirancang menggunakan file *node-movement* dan *traffic-pattern*, serta menggunakan kode sumber analisis untuk menguji performa dari dua protokol routing.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian dan evaluasi dari sistem dan performa dalam skenario pada *routing protocol* AODV. Penguji ini dilakukan dengan skenario yang dibuat dalam *network simulator* untuk mendapatkan hasil uji dari PDR, E2E dan RO.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan serta saran-saran untuk melakukan pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk literatur dalam mengembangkan Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi kode sumber dari program yang dipakai secara keseluruhan.

BAB II
TINJAUAN PUSTAKA

Bab ini menjelaskan tentang tinjauan pustaka yang menjadi dasar pembuatan Tugas Akhir. Beberapa teori, pustaka, dan teknologi yang mendasari pengerjaan Tugas Akhir ini. Serta memberi gambaran terhadap alat, protocol *routing* serta definisi yang digunakan dalam pembuatan Tugas Akhir.

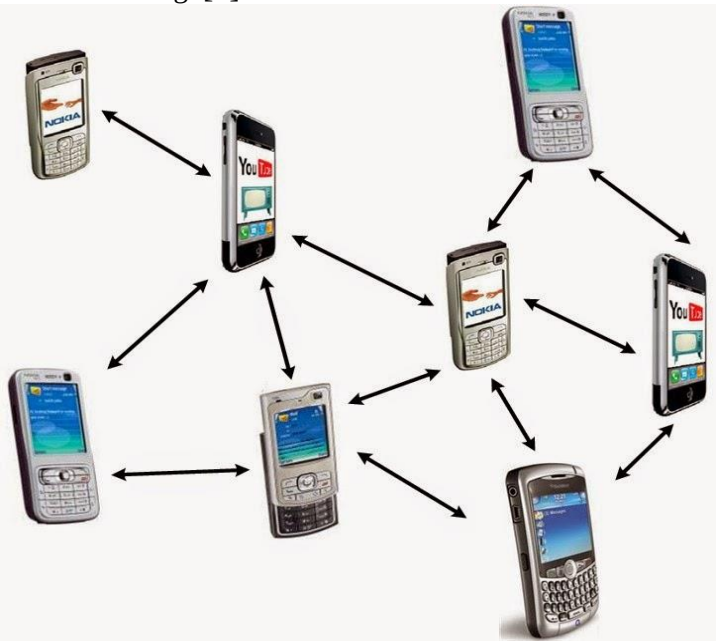
2.1. *Mobile Ad Hoc Network* (MANET)

Mobile Ad Hoc Network adalah kumpulan dari beberapa wireless node yang dapat di set-up secara dinamis dimana saja dan kapan saja tanpa menggunakan infrastruktur yang ada. MANET juga merupakan jaringan sementara yang dibentuk oleh beberapa mobile node tanpa adanya pusat administrasi dan infrastruktur kabel. Pada MANET, mobile house yang terhubung dengan wireless dapat bergerak bebas dan juga berperan sebagai router. [3]

Pada lapis protokol MANET yang punya kemiripan dengan layer TCP/IP tampak ada perbedaan di layer network-nya. Mobile node menggunakan ad hoc protokol routing untuk merutekan paket-paket. Layer network MANET dibagi menjadi dua bagian yaitu layer network dan layer ad hoc ruting. Protokol yang digunakan pada bagian network layer MANET adalah **Internet Protokol (IP)** dan protokol yang dipakai pada bagian ad hoc ruting layer adalah Ad Hoc On-Demand Distance Vector (AODV). Protokol ruting ad hoc yang lainpun bisa juga dipakai pada layer ini. Pada layer transport digunakan User Diagam Protokol. Karakteristik yang dimiliki UDP diantaranya toleran terhadap loss dan tidak toleran terhadap delay. Pengiriman paket ini tidak memerlukan **paket** balasan yang berfungsi untuk memastikan sampainya sebuah paket. Meskipun demikian paket-paket UDP mempunyai keunggulan yaitu lebih efektif dalam penggunaan bandwidth, karena mampu

meneruskan paket ke jalur lain apabila terjadi suatu kemacetan dalam pengiriman paket**.** [4]

Tujuan Routing pada MANET adalah untuk mencari jalur yang paling tepat dari satu node ke node lainnya dalam suatu jaringan, baik jaringan berkabel maupun wireless. Pada MANET, proses routing tidaklah semudah seperti yang terjadi pada jaringan berkabel. Ini dikarenakan setiap node berperan sebagai router yang menerima aliran data dan menghitung jalur yang tepat untuk selanjutkan dikirimkan ke jalur tersebut sampai tujuan. Jadi, setiap node tidak dipengaruhi oleh node lain (otonomi) dalam menentukan routing. [5]



Gambar 1. 1 skema MANET

Terdapat berbagai jenis protokol routing untuk MANET yang secara keseluruhan yang dapat dibagi menjadi beberapa kelompok, antara lain:

a. Proactive Routing

Algoritma ini akan mengelola daftar tujuan dan rute terbaru masing-masing dengan cara mendistribusikan, sehingga jalur lalu lintas (traffic) akan sering dilalui oleh routing table tersebut. Hal ini akan memperlambat aliran data jika terjadi restrukturisasi routing table.

b. Reactive Routing

Tipe ini akan mencari rute (on demand) dengan cara membanjiri jaringan dengan paket router request. Sehingga dapat menyebabkan jaringan akan penuh (clogging).

c. Flow Oriented Routing

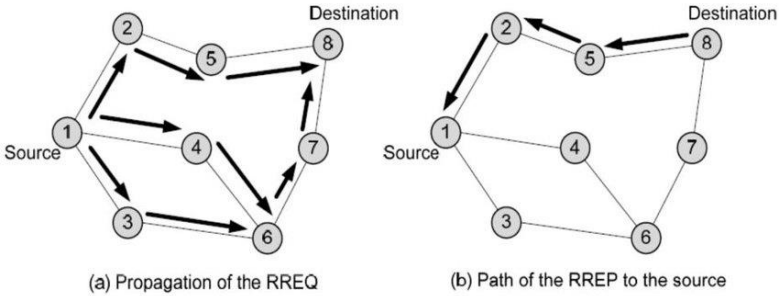
Tipe protokol ini mencari route dengan mengikuti aliran yang disediakan. Salah satu pikiran adalah dengan unicast secara terus-menerus ketika meneruskan data saat mempromosika link baru. Beberapa kekurangan tipe protokol ini adalah membutuhkan waktu yang lama untuk mencari rute baru.

d. Hybrid Routing

Tipe protokol yang menggabungkan antara proactive routing dengan reactive routing.

2.2. *Ad Hoc On Demand Vector* (AODV)

Ad Hoc On Demand Vector adalah protokol routing yang dirancang untuk jaringan ad hoc nirkabel dan seluler. Protokol ini menetapkan rute ke tujuan berdasarkan permintaan dan mendukung rute unicast dan multicast. Protokol AODV dikembangkan bersama oleh Nokia Research Center, University of California, Santa Barbara dan University of Cincinnati pada tahun 1991. [6]



Gambar 2.1 Proses *routing discovery* pada AODV

AODV adalah routing protocol yang termasuk kategori reactive routing protocol. Seperti reactive routing protocol lainnya, AODV hanya menyimpan informasi routing seputar path dan host yang aktif. Didalam AODV, informasi routing disimpan di semua node. Setiap node menyimpan tabel routing next-hop, dimana menyimpan informasi tujuan ke hop berikutnya dimana node tersebut memiliki route tertentu. Didalam AODV, ketika node asal ingin mengirim packet ke tujuan namun tidak ada route yang tersedia, node tersebut akan memulai proses route discovery.

Didalam proses route discovery, node asal membroadcast route request (RREQ) packets dimana disetakan nomor sequence tujuan. Ketika node tujuan atau node yang memiliki route ke tujuan menerima packet RREQ, node tersebut akan memeriksa nomor sequence tujuan yang sampai pada nodenya ketika packet tiba dan nomor sequence sama didalam RREQ. Untuk memastikan bahwa packet tersebut masih bersifat baru, node tujuan membalas paket RREQ dengan route reply (RREP) packet. RREP dibuat dan dikirim kembail ke node asal hanya jika nomor sequence tujuan sama dengan atau lebih besar dari yang dispesifikasikan di RREQ. AODV hanya menggunakan link yang bersifat simetris dan RREP mengikuti path sebaliknya dari path yang dihasilkan oleh RREQ. Ketika menerima RREP, setiap node diantara asal dan tujuan mengupdate routing table next-hop dengan RREP ke tujuannya. Node asal kemudian memilih route

dengan jumlah hop paling sedikit untuk mengirimkan paket tujuannya. [2]

2.3. Model Propagasi TwoRayGround

Two-Rays Ground Reflected Model adalah model propagasi radio yang memprediksi kehilangan jalur antara antena pemancar dan antena penerima ketika mereka berada di LOS (line of sight). Secara umum, kedua antena masing-masing memiliki ketinggian yang berbeda. Sinyal yang diterima memiliki dua komponen, komponen LOS dan komponen multipath yang dibentuk terutama oleh gelombang pantulan tunggal. [7]

Model refleksi tanah dua-ray mempertimbangkan jalur langsung dan jalur refleksi tanah. Hal itu ditunjukkan bahwa model ini memberikan prediksi yang lebih akurat pada jarak jauh daripada model ruang bebas. Kekuatan yang diterima pada jarak d diprediksi oleh

$$P_r(d) = \frac{P_t G_t G_r {h_t}^2 {h_r}^2}{d^4 L}$$

Pada.persamaan diatas, P_t adalah power yang dibutuhkan dalam propagasi. G_t dan G_r meruapakan tegangan yang ada pada *transmiter* dan *receiver*, lalu untuk ht dan har merupakan ketinggian dari antena *transmitter* dan *receiver*, nilai L diasumsikan L=1 pada NS-2

Model dua-ray tidak memberikan hasil yang baik untuk jarak pendek karena osilasi yang disebabkan oleh kombinasi konstruktif dan destruktif dari kedua sinar. Sebaliknya, model *freespace* masih dapat digunakan ketika nilai d diperkecil. [8]

2.4. Model Propagasi FreeSpace

Model propagasi Free Space digunakan untuk memprediksi kekutan sinyal yang diterima ketika pemancar dan penerima memiliki kejernihan, jalur line-of-sight terhalang antara mereka. Sistem komunikasi satelit dan microwave line-of-sight

link radio biasanya menjalani propagasi ruang bebas. Seperti kebanyakan model propagasi radio gelombang skala besar, model ruang bebas memprediksi bahwa seluruh daya terima sebagai fungsi dari jarak T-R separation akan naik beberapa pangkat. Kekuatan ruang bebas yang diterima oleh anten penerima yang dipisahkan dari antena pemancar memancr dengan jarak de, ditunjukan oleh persamaan Friis free space dibawah ini.

$$\Pr(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L}$$

dimana Pt adalah kekuatan sinyal yang ditransmisikan, Gt dan Gr kekuatan dari antena pemancar dan penerima masing-masing. L (L □1) adalah kehilangan sistem, dan λ adalah panjang gelombang. Pada umumnya diasumsikan Gt = Gr = 1 dan L = 1 ketika pada simulasi ns. [9]

Gambar 2.2 Model propagasi *freespace*

2.5. Network Simulator 2 (NS-2)

. Network Simulator (Versi 2), secara luas dikenal sebagai NS2, hanyalah sebuah event-driven alat simulasi yang telah terbukti berguna dalam mempelajari sifat dinamis dari jaringan komunikasi. Simulasi fungsi jaringan kabel dan nirkabel dan protokol (mis., algoritma perutean, TCP, UDP) dapat dilakukan menggunakan NS2. Secara umum, NS2 memberi pengguna cara untuk menentukan protokol jaringan dan mensimulasikannya perilaku yang sesuai.

Karena sifatnya yang fleksibel dan modular, NS2 telah mendapatkan popularitas yang konstan di komunitas riset jejaring

sejak kelahirannya pada tahun 1989. Sejak saat itu, beberapa revolusi dan revisi telah menandai semakin matangnya alat ini, terima kasih untuk kontribusi substansial dari para pemain di lapangan. Di antara ini adalah Universitas California dan Universitas Cornell yang mengembangkan jaringan NYATA simulator, 1 fondasi tempat NS ditemukan. Sejak 1995 Pertahanan Badan Proyek Penelitian Lanjutan (DARPA) mendukung pengembangan NS melalui proyek Virtual InterNetwork Testbed (VINT) .2 Saat ini National Science Foundation (NSF) telah bergabung dalam perjalanan dalam pengembangan. Terakhir tapi tidak paling tidak, kelompok peneliti dan pengembang di komunitas terus-menerus bekerja untuk menjaga NS2 kuat dan serbaguna. [10]

2.6. AWK

Perintah awk adalah sebuah command Linux yang dapat berfungsi sebagai alat penyaringan (filtering tools) yang fungsinya hampir sama dengan perintah grep. Perintah awk juga biasanya dipakai untuk mengolah dan analisis file log yang isinya sangat panjang. Perintah awk mendukung fitur regex (regular expressions) karna fungsinya yang mirip perintah grep.

Perlu diketahui bahwa AWK adalah sebuah singkatan dari pembuat algoritma pengurai ini. AWK diambil dari inisial ketiga pembuatnya yaitu "*Aho, Weinberger, and Kernighan*". Awk paling sering digunakan untuk me-scan dan mem-proses sebuah pola. Pencarian pada sebuah file yang cocok dengan pola yang dibuat lalu menyaring pencarian tersebut kedalam sebuah file baru.

Berikut ini adalah fitur-fitur yang dimiliki Awk :

1. Awk menjadikan text file sebagai records dan fields
2. Seperti bahasa pemrograman lainnya, Awk mengandung variabel, kondisi, dan looping
3. Awk mempunyai operator aritmatika dan string
4. Awk bisa digenerate menjadi laporan yang berformat

Pada pengerjaan Tugas Akhir ini AWK digunakan sebagai *tools* untuk membuat *script* dalam perhitungan *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2E) dan *Routing Overhead* (RO) dari hasil *trace* NS-2. Kode sumber PDR, E2E dan RO tercantum pada lampiran.

2.7. *Generator File Node Movement*

Untuk menghasilkan *random movement* dari *node* dalam jaringan nirkabel digunakan *tools* yang disebut 'setdest' yang dikembangkan oleh CMU (*Carnegie Mellon University*). Kecepatan gerak yang spesifik menuju lokasi acak atau lokasi spesifik yang berada dalam kawasan yang telah ditentukan menghasilkan sebuah *node movement*. Ketika *node* tiba di lokasi pergerakan, *node* tersebut bisa diatur untuk berhenti sementara waktu. Selanjutnya *node* akan terus bergerak menuju lokasi berikutnya. Lokasi 'setdest' berada pada direktori '`~ns/indep-utils.cmu-scen-gen/setdest/`'.

Pengguna harus menjalankan program 'setdest' sebelum menjalankan program simulasi. Format *command line* 'setdest' ditunjukan pada Gambar 2.3 dan keterangannya ditunjukan pada Table 2.1.

```
1. ./setdest [-v version] [-n num_of_nodes] [-
   p pausetime] [-M maxspeed] [-t simtime] [-
   x maxx] [-y maxy] > [outdir/movement-file]
```

Gambar 2.3 Format *Command Line* 'setdest'

Table 2.1 Penjelasan Command Line 'setdest'

Parameter	Keterangan
-v version	Versi 'setdest' simulator yang digunakan.
-n num	Jumlah node dalam skenario.
-p pausetime	Durasi ketika sebuah node tetap diam

	setelah tiba di lokasi pergerakan. Jika nilai ini diatur ke 0, maka node tidak akan berhenti ketika tiba di lokasi pergerakan dan akan terus bergerak.
-M maxspeed	Kecepatan maksimum sebuah node. Node akan bergerak pada kecepatan acak dalam rentang[0,maxspeed].
-t simtime	Waktu simulasi.
-x max x	Panjang maksimum area simulasi.
-y max y	Lebar maksimum area simulasi.

File output dihasilkan oleh c*ommand line* 'setdest' yang berisi jumlah *node* dan mobilitas yang akan digunakan dalam *file* tcl selama simulasi. Selain mengandung skrip pergerakan *file output* juga mengandung beberapa statistik lain tentang perubahan *link* dan rute.

Untuk membuat skenario *node movement* yang terdiri dari 50 *node*, bergerak dengan kecepatan maksimum 10 m/s dengan jeda rata-rata antar gerakan sebesar 2 detik, simulasi akan berhenti setelah 100 detik dengan batas topologi yang diartikan sebagai 500 x 500 meter2, maka *command line*-nya seperti pada Gambar 2.4.

```
1.  ./setdest -v 1 -n 50 -p 10 -M 5 -t 100 -x 500 -
    y 500 > scenario.txt
```

Gambar 2.4 Contoh *command line* 'setdest'

2.8. *File Traffic Connection Pattern*

Skrip *traffic-scenario pattern generator* digunakan sebagai konfigurasi antara mobilitas *node Random traffic connection* pada TCP dan CBR. Skrip Tcl yang disebut "cbrgrn" dapat menghasilkan alur *traffic* yang acak. Skrip ini membantu untuk menghasilkan *traffic load*. *Traffic load* dapat berupa TCP atau

CBR. Skrip ini disimpan dalam file ‘CMU-scen-gen’ yang terletak dalam direktori ~ns/indep-utils/cmu-scen-gen. Program “cbrgen.tcl” digunakan sesuai dengan command line pada Gambar 2.5 dan dengan keterangan yang diunjukan pada Table 2.2.[8]

```
1. ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-
   seed seed] [-mc connections] [-
   rate rate] > traffic-
```

Gambar 2.5 Format Command Line ‘cbrgen.tcl’

Table 2.2 Penjelasan command line ‘cbrgen.tcl’

Parameter	Keterangan
-type cbr|tcp	Jenis traffic yang digunakan TCP atau CBR
-nn nodes	Jumlah total node
-s seed	Random seed
-mc connection	Jumlah Koneksi
-rate rate	Jumlah paket per detik

Command line pada Gambar 2.6 merupakan cara membuat sebuah *file* koneksi CBR antara 2 *node* yang memiliki maksimal 1 koneksi dengan nilai *seed* 1.0 dan jumlah paket per detik sebanyak 0.25.

```
1. ns cbrgen.tcl -type cbr -nn 2 -seed 1.0 -mc 1 -
   rate 0.25 > cbr.txt
```

Gambar 2.6 Contoh *command line* ‘cbrgen.tcl’

BAB III
ANALISIS DAN PERANCANGAN SISTEM

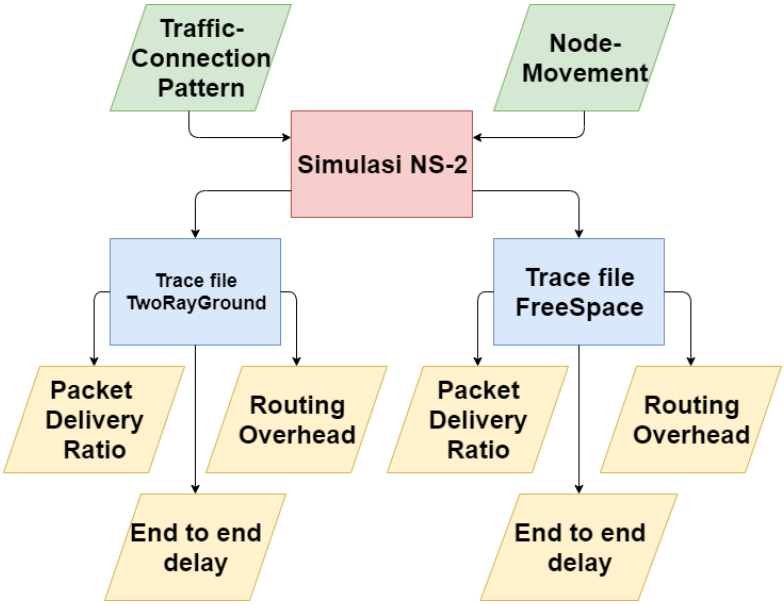
Pada bab ini akan dijelaskan mengenai dasar perancangan dari perangkat lunak yang akan dibangun dalam tugas akhir ini. Perancangan tersebut mencakup deskripsi umum sistem perancangan skenario, arsitektur sistem, model fungsional, diagram alur sistem serta gambaran implementasi.

3.1. Deskripsi Umum Sistem

Pada Tugas Akhir ini akan dilakukan analisis tentang performa model transmisi *TwoRayGround* dan *FreeSpace* pada MANET. Pada pembuatan skenario penulis menggunakan *mobility generator* yang bersifat random way point dan telah ada pada *Network Simulator-2* (NS-2) yaitu dengan cara men-*generate file node movement* dan membuat koneksi antar *node* dengan cara men-*generate file traffic connection pattern*. Rancangan Simulasi yang dibuat dapat dilihat pada Gambar 3.1.

Dalam penelitian ini penulis akan menganalisa performa dari model transmisi *TwoRayGround* dan *FreeSpace* pada simulasi skenario yang dijalankan pada NS-2 menggunakan *routing protocol* AODV pada system operasi Linux.

Pada setiap skenario, kecepatan maksimum dari satu *node* ke *node* lainnya dibuat bervariasi yaitu 5, 10 dan 15 m/s. Hasil proses uji coba dari tiap skenario akan menghasilkan sebuah *trace file* yang nantinya akan dilakukan analisis perhitungan metrik *packet delivery Ratio* (PDR), *End-to-End Delay* (E2E), dan *Routing Overhead* (RO). Dari hasil metrik tersebut dianalisis performa dari transmisi *TwoRayGround* dan *FreeSpace.*



Gambar 3.1 Flowchart rancangan umum sistem

3.2. Perancangan Skenario

Perancangan skenario uji coba mobilitas MANET diawali dengan melakukan pembuatan skenario pada *mobility generation* yang bersifat *Random Way Point* dengan cara men-*generate file node-movement* kemudian membuat koneksi dari node-node yang akan dipakai dalam simulasi dengan menggunakan *file traffic-connection* yang sudah ada pada NS-2. Pada tugas akhir ini pembuatan skenario untuk melihat pergerakan *node* dibedakan berdasarkan tiga kecepatan maksimum yaitu 5, 10 dan 15 m/s. Kemudian untuk koneksinya digunakan dua *node* untuk menentukan *node* pengirim dan *node* penerima paket. Pembuatan scenario tadi dibuat dengan menggunakan fungsi *setdest* yang sudah ada pada NS2. Perancangan skenario tadi adalah sebagai berikut:

3.2.1. Perancangan Skenario Node Movement (Mobility Generation)

Pada perancangan skenario mobility generation dibuat dengan cara men-*generate file node movement* yang telah ada pada NS-2 atau *tools*-nya biasa disebut 'setdest' yang nantinya akan menghasilkan *output* dalam bentuk .txt dan digunakan dalam file tcl selama simulasi pada NS-2 sebagai bentuk pergerakan *node* yang berpindah pindah. Isi parameter dan spesifikasi dari perintah *setdest* dapat dilihat pada Table 3.1.

Table 3.1 Penjelasan skenario node movement

No.	Parameter	Spesifikasi
1	Jumlah *Node*	50
2	Waktu Simulasi	100 detik
3	Area	510m x 510m
4	Kecepatan Maksimal	-5m/s -10m/s -15m/s
5	Sumber *Traffic*	CBR
6	Waktu Jeda (Dalam Detik)	10
7	Ukuran Paket	512 bytes
8	*Rate* Paket	0.25 paket per detik
9	Jumlah maksimal koneksi	1
10	Model mobilitas yang digunakan	*Random way point*

Dari Table 3.1 dapat dilihat bahwa digunakan 3 kecepatan maksimal. Sehingga dalam perancangannya akan dilakukan 3 kali perintah *setdest* dengan spesifikasi dari parameter kecepatan maksimal yang berbeda.

3.2.2. *Traffic-Connection Pattern*

Traffic-Connection dibuat dengan menjalankan program *cbrgen.tcl* yang telah ada pada NS-2 dan mengisi spesifikasi dari perintah yang sudah ada. Kemudian akan menghasilkan *output*

dari hasil program tersebut dalam bentuk .txt. Lalu file output tadi dapat digunakan sebagai konfigurasi dari koneksi untuk menghubungkan antar *node* yang ada pada skenario selama simulasi pada NS-2. Parameter dan Spesifikasi dari Traffic-Connection Pattern tadi dapat dilihat pada Table 3.2

Table 3.2 Penjelasan Traffic-connection pattern

No.	Parameter	Spesifikasi
1	-type cbr|tcp	CBR
2	-nn nodes	2
3	-s seed	1.0
4	-mc connection	1
5	-rate rate	0.25

Pada Table 3.2 dapat dilihat bahwa jenis traffic yang digunakan adalah jenis CBR. Jenis tersebut dipilih karena sifat yang dipakai dalam traffic-connection pattern adalah random way point.

3.3. Perancangan Simulasi NS-2

Pada perancangan kode NS-2 dengan konfigurasi MANET, dilakukan dengan cara menggabungkan skenario mobilitas dan *script* TCL yang berisi konfigurasi mengenai informasi dari lingkungan simulasi. Konfigurasi lingkungan simulasi MANET pada NS-2 dapat dilihat di Table 3.3

Table 3.3 Penjelasan simulasi NS-2 dengan propagasi TwoRayGround

No	Parameter	Spesifikasi
1	*Network Simulator*	NS-2, 2.35
2	Routing Protocol	AODV
3	Waktu Simulasi	100 detik
4	Area Simulasi	510m x 510m
5	Banyak *node*	50
6	Radius Transmisi	100m

7	Tipe Data	*Constant Bit Rate* (CBR)
8	*Source/Destination*	Statis
9	*Packet Rate*	512 bytes
10	Tipe Koneksi	UDP
11	Protokol MAC	IEEE 802.11p
12	Propagasi Sinyal	- *TwoRayGround* - *FreeSpace*
13	Tipe Kanal	*Wireless Channel*
14	Tipe Interface Queue	DropTail/PriQueue
15	Tipe Antena	OmniAntenna
16	Tipe Peta	MANET (*random way point*)

Hasil dari menjalankan simulasi ini adalah berupa trace file yang nantinya akan dianalisis pada metrik analisis.

3.4. Perancangan Metriks Analisis

Beberapa parameter yang dianalisis dalam Tugas Akhir ini adalah *Packet Delivery Ratio* (PDR), *End - to end Delay* (E2D), dan *Routing Overhead* (RO). Penjelasan dari ketiga matriks analisis adalah sebagai berikut:

3.4.1. Paket Delivery Ratio (PDR)

Packet delivery ratio merupakan perbandingan dari jumlah paket komunikasi yang diterima dengan jumlah paket komunikasi yang dikirimkan. *Packet delivery ratio* dihitung menggunakan persamaan 3.1

$$Packet\ Delivery\ Ratio = \frac{Received}{Sent} x100\% \quad (1)$$

Nilai dari PDR merupakan nilai dari keberhasilan paket yang dikirimkan. Hasil yang didapatkan dari Packet Delivery Ratio dijadikan dalam persen. Jadi, semakin tinggi *packet delivery ratio* semakin berhasil pengiriman paket yang dilakukan.

3.4.2. End-to-End Delay

Rata-rata *end-to-end delay* merupakan rata-rata *delay* atau waktu yang dibutuhkan untuk mengirim paket dari node asal ke node tujuan. *End to End Delay* dihitung dengan persamaan 3.2, dimana *delay* didapatkan dari rentang waktu antara node asal dengan node tujuan. *Delay* tiap paket tersebut dijumlahkan kemudian dibagi dengan jumlah paket yang berhasil diterima.

$$End\ to\ End\ Delay = \frac{\sum_{i=0}^{recvnum} t^{received(i)} - t^{sent(i)}}{recvnum} \quad (2.2)$$

3.4.3. Routing Overhead (RO)

Routing overhead adalah jumlah paket routing yang dihasilkan dibagi jumlah dari packet data yang diterima dalam proses routing selama simulasi. Perhitungan routing overhead didapatkan dari jumlah semua paket kontrol routing yang ditransmisikan, baik itu *route request* (RREQ), *route reply* (RREP), maupun *route error* (RRER). Perhitungan Routing Overhead dapat dilihat dengan presamaan 3.3.

$$Routing\ Overhead = \sum_{i=0}^{sent\ and\ forwarded} \frac{packet\ routing}{packet\ data} \quad (3.3)$$

BAB IV
IMPLEMENTASI

Bab ini akan membahas implementasi dari perancangan yang sudah dibahas sebelumnya. Implementasi itu meliputi lingkungan pembangunan perangkat lunak, implementasi skenario, implementasi simulasi NS-2, dan implementasi matiks analisis.

4.1. Lingkungan Implementasi Protokol

Sub bab ini menjelaskan tentang lingkungan implementasi protokol yang dilakukan pada lingkungan:

Table 4.1 Spesifikasi Lingkungan Implementasi

Perangkat Keras	- Laptop HP R202TX o Prosesor Intel(R) Core(TM) i5-5200U CPU @ 2.70GHz o RAM 6 GB o HDD 500 GB
Perangkat Lunak	- Laptop HP R202TX o Sistem Operasi Ubuntu 16.04 o Network Simulator 2, 2.35

4.2. Implementasi Skenario

Dalam bab ini mengimplementasi skenario mobilitas MANET dengan kondisi yang berbeda pada beban *traffic*-nya dan mobilitas/pergerakan *node*-nya. Pada implementasi ini ada dua model yang akan digunakan untuk studi simulasi pada jaringan MANET yaitu *mobility generation* yang digunakan untuk mempelajari pengaruh mobilitas dari *node* pada kinerja seluruh jaringan dan *traffic connection* yang digunakan untuk mempelajari pengaruh beban *traffic* pada jaringan implementasi skenarionya adalah sebagai berikut.

4.2.1. Skenario File Node-Movement(Mobility Generation)

Dalam implementasi skenario *mobility generation* akan digunakan *tools generate default* yang dimiliki oleh NS-2 yaitu ‘setdest’. File skenario *node movement* (*mobility generation*) digunakan untuk setiap simulasi yang ditandai dengan jeda waktu. Simulasi dilakukan dengan membuat pola gerakan yang dihasilkan dari kecepatan maksimal menjadi berbeda, dengan tujuan untuk mempelajari efek mobilitas. Pengaturan pada file skenario pergerakan *node*, dibuat dengan memvariasikan kecepatan maksimal program ‘setdest’ pada NS-2 yang digunakan untuk menghasilkan *file node movement* dengan menggunakan algoritma *Random Way Point*. Format *command line* pada Gambar 4.1 yang digunakan untuk menghasilkan gerakan acak pada *node* adalah sebagai berikut:

```
1. ./setdest [-v version] [-n num_of_nodes] [-
   p pausetime] [-M maxspeed] [-t simtime] [-
   x maxx] [-y maxy] > [outdir/movement-file]
```

Gambar 4.1 format kode ‘setdest’

Ada ketentuan yang diujicobakan pada skenario ini adalah ‘setdest’ simulator yang digunakan adalah versi 1, jumlah node dalam skenario yaitu 50, waktu jeda (*pause time*) yaitu 10 detik,kecepatan maksimal pada skenario A sebesar 5m/s, skenario B sebesar 10m/s dan skenario C sebesar 15m/s, waktu simulasi yaitu 100 detik. File mobilitas yang dihasilkan tadi disimpan dalam direktori “ ~ns/indep-utils/cmu-scen-gen/setdest/”. Pada Gambar 4.2 merupakan contoh dari *command line* pada ‘setdest’ dengan berbagai kecepatan maksimal yang berbeda sesuai dan node sebanyak 50. Lalu untuk setiap kecepatan maksimal tersebut dibuat 10 buah file untuk satu protokol routing dan satu model transmisi.

```
1. ./setdest -v 1 -n 50 -p 10 -M 5 -t 100 -x 510 -
   y 510 > scenario.txt
2. ./setdest -v 1 -n 50 -p 10 -M 10 -t 100 -x 510 -
   y 510 > scenario.txt
3. ./setdest -v 1 -n 50 -p 10 -M 15 -t 100 -x 510 -
   y 510 > scenario.txt
```

Gambar 4.2 implementasi 'setdest'

Hasil dari pemakaian 'setdest' menunjukan skenario acak serta posisi awal dari setiap node dengan sumbu X, Y, dan Z yang merupakan titik dari sumbu yang digunakan. Hasil dari 'setdest' dapat dilihat pada Gambar 4.3.

```
#
# nodes: 50, pause: 10.00, max speed: 5.00, max x: 510.00, max y: 510.00
#
$node_(0) set X_ 494.320078750275
$node_(0) set Y_ 227.449934547671
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 433.155725418215
$node_(1) set Y_ 71.338539206985
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 234.816130218505
$node_(2) set Y_ 337.876205627865
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 409.755501206765
$node_(3) set Y_ 216.915206783070
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 0.034648798687
$node_(4) set Y_ 75.546624081605
$node_(4) set Z_ 0.000000000000
$node_(5) set X_ 219.357711918746
$node_(5) set Y_ 193.475881242888
$node_(5) set Z_ 0.000000000000
$node_(6) set X_ 302.943903783414
$node_(6) set Y_ 187.722536773954
$node_(6) set Z_ 0.000000000000
```

Gambar 4.3 Posisi awal setiap node pada sumbu X, Y, dan Z

```
$ns_ at 10.000000000000 "$node_(0) setdest 430.594614984859
251.789156224612 2.327024838659"
$ns_ at 10.000000000000 "$node_(1) setdest 28.965941033074
150.753078525104 1.173931734911"
$ns_ at 10.000000000000 "$node_(2) setdest 476.494052149533
394.581430212813 2.975027445659"
$ns_ at 10.000000000000 "$node_(3) setdest 311.114520128309
460.669755977418 4.798016055251"
$ns_ at 10.000000000000 "$node_(4) setdest 210.945774548855
397.749127657576 0.068050226931"
$ns_ at 10.000000000000 "$node_(5) setdest 265.208012611435
456.195794962981 0.871416610026"
$ns_ at 10.000000000000 "$node_(6) setdest 219.800602756097
389.037134149048 3.236343789184"
$ns_ at 10.000000000000 "$node_(7) setdest 432.496490329334
178.241326160309 3.177925964354"
$ns_ at 10.000000000000 "$node_(8) setdest 153.442845548529
306.143873135542 3.465941493846"
$ns_ at 10.000000000000 "$node_(9) setdest 59.984176383984
69.800103809317 1.100910765582"
```

Gambar 4.4 Pergerakan node

Gambar 4.4 Hasil dari implementasi juga yang menunjukan data pergerakan node. Informasi posisi awal serta pergerakan node telah diatur dalam satu *file* skenario yang nantinya digunakan pada simulasi. Pada potongan dari *file* scen.txt menunjukan bahwa *node* (1) pada detik ke 10 awal melakukan perpindahan ke arah tujuan (28.97, 150.75) dengan menggunakan kecepatan 1.17m/s.

```
$god_ set-dist 0 1 1
$god_ set-dist 0 2 2
$god_ set-dist 0 3 1
$god_ set-dist 0 4 3
$god_ set-dist 0 5 2
$god_ set-dist 0 6 1
$god_ set-dist 0 7 1
$god_ set-dist 0 8 1
$god_ set-dist 0 9 2
$god_ set-dist 0 10 1
$god_ set-dist 0 11 1
$god_ set-dist 0 12 2
$god_ set-dist 0 13 2
```

Gambar 4.5 GOD bentuk array

GOD yang merupakan singkatan dari General Operation Director memiliki fungsi sebagai tempat penyimpan informasi global tentang jumlah serta pergerakan node. Pada Gambar 4.5 Fungsi dari GOD adalah untuk memberitahu array terpendek dari hop yang dibutuhkan untuk perindahan satu node ke node lain.

```
$ns_ at 10.239885684559 "$god_ set-dist 4 31 2"
$ns_ at 10.239885684559 "$god_ set-dist 5 31 1"
$ns_ at 10.266377519971 "$god_ set-dist 2 49 1"
$ns_ at 10.333935143309 "$god_ set-dist 14 31 2"
$ns_ at 10.333935143309 "$god_ set-dist 31 41 1"
$ns_ at 10.375823362495 "$god_ set-dist 27 31 2"
$ns_ at 10.375823362495 "$god_ set-dist 31 49 1"
$ns_ at 10.581392155929 "$god_ set-dist 4 38 2"
$ns_ at 10.581392155929 "$god_ set-dist 12 38 1"
$ns_ at 10.666611162372 "$god_ set-dist 9 19 1"
$ns_ at 10.715632772586 "$god_ set-dist 15 40 1"
$ns_ at 10.913070734887 "$god_ set-dist 4 30 2"
$ns_ at 10.913070734887 "$god_ set-dist 12 30 1"
$ns_ at 11.020086662480 "$god_ set-dist 15 34 1"
$ns_ at 11.020086662480 "$god_ set-dist 25 34 2"
$ns_ at 11.112981121338 "$god_ set-dist 2 18 1"
```

Gambar 4.6 Penyimpanan *access point* pada GOD

Lalu informasi yang dimasukkan pada objek GOD merupakan jalur paling pendek dari satu node ke node yang lain. Pada Gambar 4.6 merupakan contoh dari hasil objek GOD. Dapat dilihat pada baris ke-4 memperlihatkan informasi dari objek GOD berupa perubahan perpindahan dari node 14 ke node 31 menjadi 2 hop pada detik ke 10.333935143309.

4.2.2. File traffic-connection pattern

Implementasi *random traffic-connection* untuk TCP dan CBR menggunakan skrip *traffic-scenario generator*. Skrip tersebut digunakan untuk mengatur pergerakan antar *node*. Skrip *traffic generator* ini terdapat pada direktori ~ns/indep-utils/cmu-scen-gen dan disimpan dalam bentuk file cbrgen.tcl. File ini dapat digunakan untuk membuat *traffic connection* CBR ataupun TCP pada jaringan pergerakan antar *node*. Pada saat menjalankan perintah pada *file traffic-connection cbrgen.tcl* ini kita harus mendefinisikan tipe *traffic connection*-nya (CBR atau TCP), banyaknya *node* dan koneksi maksimal yang ada pada jaringan tersebut, *random seed* dan misalkan pada koneksi CBR, rate yang nilai kebalikannya digunakan untuk menghitung waktu interval antar paket CBR yang kemudian disimpan dalam sebuah *file traffic*. Format *command line* pada Gambar 4.7 yang digunakan untuk menghasilkan gerakan acak pada *node* adalah sebagai berikut.

```
1. ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-
   seed seed] [-mc connections] [-
   rate rate] > traffic-
```

Gambar 4.7 Format command line ‘cbrgen.tcl’

Gambar 4.8 merupakan implementasi untuk menjalankan cbrgen.tcl untuk membuat file koneksi CBR diantara 2 node memiliki maksimal 1 koneksi dengan nilai seed 1.0 dan jumlah paket per detik sebanyak 0.25 yang disimpan dalam cbr.txt yang nantinya akan digunakan pada saat simulasi.

```
1. ns cbrgen.tcl -type cbr -nn 2 -seed 1.0 -mc 1 -
   rate 0.25 > cbr.txt
```

Gambar 4.8 Implementasi kode ‘cbrgen.tcl’

```
#
# nodes: 2, max conn: 1, send rate: 0.25, seed:
1.0
#
#
# 1 connecting to 2 at time 2.5568388786897245
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.25
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 2.5568388786897245 "$cbr_(0) start"
#
#Total sources/connections: 1/1
#
```

Gambar 4.9 ***File traffic connection pattern***

Gambar 4.9 Merupakan potongan dari file cbr.txt. File tersebut merupakan output yang dihasilkan setelah menjalankan *file* cbrgen.tcl. Pada set UDP, koneksi CBR menggunakan UDP *Agent*. Koneksi dari UDP itu merupakan koneksi antara node 1 dan node 2. Untuk pengiriman paket data dilakukan setiap satu detik dengan ukuran paket 512 *byte* dengan interval 0.25 dan dengan jumlah maksimal paket sebesar 10.000.

4.3. Implementasi Simulasi pada NS-2

Simulasi akan dijalankan menggunakan dua model transmisi, yaitu *FreeSpace* dan *TwoRayGround.* Kemudian skenario *node-movement* dan *traffic-connection generation* digunakan untuk pendeskripsian lingkungan simulasi dan disimpan pada sebuah file dengan *ekstensi* .txt. File ini berisi konfigurasi setiap *node* dan proses yang dilakukan selama simulasi berjalan.

Pada bagian awal simulasi MANET pada NS-2 merupakan konfigurasi dibutuhkan dalam *file* simulasi. Pada baris pertama menunjukan tipe channel yang dipakai adalah *WirelessChannel.* Kemudian pada baris kedua adalah konfigurasi dari jenis propagasi yang akan digunakan. Simulasi ini menggunakan 2 file dengan konfigurasi propagasi yang berbeda. Untuk file simulasi yang pertama, menggunakan setting propagasi *FreeSpace* dan file simulasi yang kedua menggunakan setting propagasi *TwoRayGround.* Pada baris kelima, untuk AODV konfigurasi *interface type* adalah "Queue/DropTail/PriQueue". Kemudian 4 baris berikutnya berturut-turut adalah adalah *Link layer, antenna,* jumlah paket maksimal, serta jumlah node. Lalu pada baris 10 sampai 16 *routing protocol* dengan jenis AODV, panjang (*x axis)* dan (*y axis)* dengan menggunakan 500meter pada keduanya, simulasi berakhir pada detik ke 100, jumlah seed 0.0, *connection pattern* yang digunakan 'scen.txt', scenario yang digunakan 'scen.txt'.

Potongan konfigurasi pada simulasi dengan setting propagasi *FreeSpace* dapat dilihat pada Gambar 4.10, sedangkan dengan setting propagasi *TwoRayGround* dapat dilihat pada Gambar 4.11.

```
set val(chan)           Channel/WirelessChannel    ;
set val(prop)           Propagation/FreeSpace      ;
set val(netif)          Phy/WirelessPhy            ;
set val(mac)            Mac/802_11                 ;
set val(ifq)            Queue/DropTail/PriQueue    ;
set val(ll)             LL                         ;
set val(ant)            Antenna/OmniAntenna        ;
set val(ifqlen)         50                         ;
set val(nn)             50                         ;
set val(rp)             AODV                       ;
set val(x)              510                        ;
set val(y)              510                        ;
set val(seed)           1.0                        ;
set val(stop)           100                        ;
set val(cp)             "cbr.txt"
set val(sc)             "coba10scen15.txt"
Phy/WirelessPhy set RXThresh_ 1.42681e-08          ;
```

Gambar 4.10 Konfigurasi parameter pada NS-2 dengan propagasi *Freespace*

```
set val(chan)           Channel/WirelessChannel    ;
set val(prop)           Propagation/TwoRayGround   ;
set val(netif)          Phy/WirelessPhy            ;
set val(mac)            Mac/802_11                 ;
set val(ifq)            Queue/DropTail/PriQueue    ;
set val(ll)             LL                         ;
set val(ant)            Antenna/OmniAntenna        ;
set val(ifqlen)         50                         ;
set val(nn)             50                         ;
set val(rp)             AODV                       ;
set val(x)              510                        ;
set val(y)              510                        ;
set val(seed)           1.0                        ;
set val(stop)           100                        ;
set val(cp)             "cbr.txt"
set val(sc)             "coba10scen15.txt"
Phy/WirelessPhy set RXThresh_ 1.42681e-08          ;
```

Gambar 4.11 Konfigurasi parameter pada NS-2 dengan propagasi TwoRayGround

Gambar 4.12 merupakan pengaturan dari *transmission range* yang digunakan pada simulasi. Nilai yang diubah ialah *RXThresh_(receiver Sensitivity Threshold).* Nilai 1.42681e-08 pada variabel tersebut memiliki artian bahwa *range* atau jangkauan yang dapat dicapai ialah sejauh 100 meter.

```
1.  Phy/WirelessPhy set RXThresh_ 1.42681e-08;
```

Gambar 4.12 Pengaturan *Transmission range*

```
set ns_ [new Simulator]
set tracefd [open test10fs15.tr w]
# set windowVsTime2 [open win.tr w]
set namtrace [open test10fs15.nam w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace
$val(x) $val(y)

#set up topography object
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
set god_ [create-god $val(nn)]

global ns_
```

```
# configure the nodes
$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType $val(netif) \
                 -channelType $val(chan) \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace ON \
                 -movementTrace OFF

    for {set i 0} {$i < $val(nn) } {incr i} {
    set node_($i) [$ns_ node ]
    $node_($i) random-motion 0        ;
    }
```

Gambar 4.13 Pengaturan variabel global, konfigurasi trace File dan NAM serta pengaturang Pergerakan *Node* pada NS-2

Skrip yang ditunjukan pada Gambar 4.13 dipakai untuk kedua jenis file dengan konfigurasi propagasi yang berbeda. Skrip itu merupakan skrip untuk pengaturan variabel global yang diawali dengan set ns untuk pembuatan simulator baru. Set *tracefd* merupakan pengaturan untuk nama *trace file* berekstensi .tr. Set *namtrace* merupakan pengaturan untuk nama *nam file* berekstensi *.nam.* Set topo merupakan pengaturan untuk objek topografi berdasarkan pada luas koordinat yang telah dikonfigurasi sebelumnya. *Set god* dan *node config – channelType* merupakan konfigurasi yang dilakukan pada *node-node* yang akan dibuat. Terakhir dilakukan perulangan untuk

membuat pergerakan dari *node node*. *Node-node* yang dibuat tidak dapat melakukan pergerakan secara acak karena pergerakan *node* merupakan *trace file* yang dihasilkan oleh *mobility generator*.

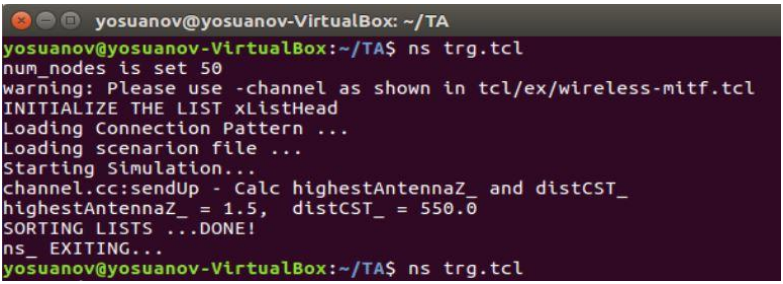
Skrip pada Gambar 4.14 merupakan bagian akhir dari keseluruhan skrip yang digunakan untuk menginisiasi penempatan awal *node-node* yang dibuat pada skenario *node-movement* (*mobility generation*), pergerakan *node* tersebut selama waktu simulasi dilakukan dan melakukan konfigurasi pengiriman paket data yang dilakukan nantinya dihasilkan pada *file output.tr* pada potongan skrip tersebut, akan dipanggil file skenario *node-movement* (*mobility generation*) dan *traffic-connection pattern* kemudian pengiriman paket data dimulai pada detik ke-0 dan diberhentikan pada detik ke 100 seperti yang telah di konfigurasi sebelumnya.

```
puts "Loading Connection Pattern ..."
source $val(cp)

puts "Loading scenario file ..."
source $val(sc)

#define node initial position in nam
        for {set i 0} {$i < $val(nn)}
        {incr i} {
                $ns_ initial_node_pos
                $node_($i) 20
        }

#tell nodes when the simulation ends
        for {set i 0} {$i < $val(nn) }
        { incr i } {
                $ns_ at $val(stop).0
                "$node_($i) reset";
        }
```

```
$ns_ at $val(stop).0002 "puts \
"ns_ EXITING...\";$ns_ halt"

puts $tracefd "M 0.0 nn $val(nn) x
$val(x) y $val(y) rp $val(rp)"
puts $tracefd "M 0.0 sc $val(sc) cp
$val(cp) seed $val(seed)"
puts $tracefd "M 0.0 prop $val(prop)
ant $val(ant)"

puts "Starting Simulation..."

$ns_ run
```

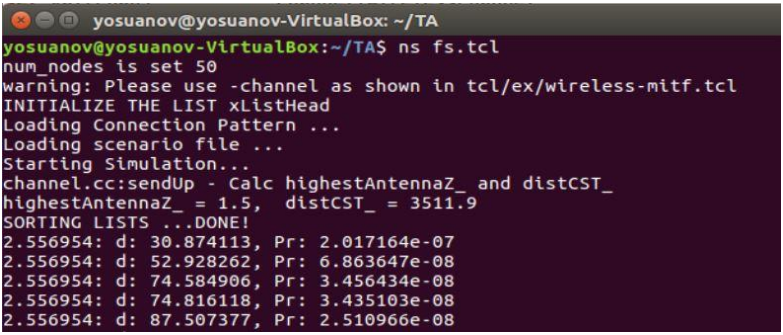
Gambar 4.14 Menginisiasi penempatan awal node

Gambar 4.15 Dan Gambar 4.16 adalah tampilan dari terminal saat mengeksekusi *file* tcl dari NS-2 menggunakan transmisi *TwoRayGround* dan *Freespace* pada *routing protocol* AODV. Jumlah node yang dipakai 50 bersifat acak sesuai skenario *node-movement* yang sudah di-*generate.* Simulasi dilakukan sebanyak 10 kali pada setiap penambahan kecpatan. Kemudian dari 10 kali percobaan tadi diambil rata-rata yang digunakan untuk menarik kesimpulan dari perobaan tadi.



Gambar 4.15 Tampilan eksekusi simulasi dengan *Routing Protocol* AODV pada model Proapagasi *TwoRayGround*

Pada saat menjalankan simulasi dari NS-2, terdapat keterangan jumlah node sebanyak 50. Lalu inisialisasi list untuk baris yang akan digunakan. Kemudian memproses *file* skenario dan pola dari koneksi. Lalu menngkalkulasi dan mencari jarak jangkau. Dibawahnya didapatkan hasil dari antenna tertinggi dan jarak jangkau. Lalu pemberitahuan bahwa sorting list selesai dan simulasi selesai. Maka dihasilkan file berekstensi *.tr* dan *.nam.*



Gambar 4.16 Tampilan eksekusi simulasi dengan *Routing Protocol* AODV pada model Propagasi *Freespace*

Dari eksekusi *file* TCL menghasilkan trace *file* dengan ekstensi .tr dan *file* animasi dari pengiriman paket dengan ekstensi .nam. Kemudian isi dari *trace file* akan dianalisis parameter-parameternya untuk menghasilkan data yang dibutuhkan untuk menghitung metrik analisis seperti PDR, E2D dan RO. Gambar 4.17 menunjukkan tampilan dari hasil *file* .tr dan Gambar 4.18 menunjukkan tampilan dari *file* .nam.

```
s 2.556953879 _1_ MAC  --- 0 AODV 106
[0 ffffffff 1 800] ------- [1:255
-1:255 30 0] [0x2 1 1 [2 0] [1 4]]
(REQUEST)
r 2.557801967 _22_ MAC  --- 0 AODV 48
[0 ffffffff 1 800] ------- [1:255
-1:255 30 0] [0x2 1 1 [2 0] [1 4]]
(REQUEST)
r 2.557802063 _21_ MAC  --- 0 AODV 48
[0 ffffffff 1 800] ------- [1:255
-1:255 30 0] [0x2 1 1 [2 0] [1 4]]
(REQUEST)
r 2.557802161 _26_ MAC  --- 0 AODV 48
[0 ffffffff 1 800] ------- [1:255
-1:255 30 0] [0x2 1 1 [2 0] [1 4]]
(REQUEST)
```

```
r 2.557802250 _34_ MAC  --- 0 AODV 48
[0 ffffffff 1 800] ------- [1:255
-1:255 30 0] [0x2 1 1 [2 0] [1 4]]
(REQUEST)
r 2.557826967 _22_ RTR  --- 0 AODV 48
[0 ffffffff 1 800] ------- [1:255
-1:255 30 0] [0x2 1 1 [2 0] [1 4]]
(REQUEST)
r 2.557827063 _21_ RTR  --- 0 AODV 48
[0 ffffffff 1 800] ------- [1:255
-1:255 30 0] [0x2 1 1 [2 0] [1 4]]
(REQUEST)
r 2.557827161 _26_ RTR  --- 0 AODV 48
[0 ffffffff 1 800] ------- [1:255
-1:255 30 0] [0x2 1 1 [2 0] [1 4]]
(REQUEST)
r 2.557827200 _18_ RTR  --- 0 AODV 48
[0 ffffffff 1 800] ------- [1:255
-1:255 30 0] [0x2 1 1 [2 0] [1 4]]
(REQUEST)
r 2.557827250 _34_ RTR  --- 0 AODV 48
[0 ffffffff 1 800] ------- [1:255
-1:255 30 0] [0x2 1 1 [2 0] [1 4]]
(REQUEST)
s 2.558708194 _22_ RTR  --- 0 AODV 48
[0 ffffffff 1 800] ------- [22:255
-1:255 29 0] [0x2 2 1 [2 0] [1 4]]
(REQUEST)
```

Gambar 4.17 Tampilan dari file .tr

Table 4.2 Parameter Trace *File*

No.	Parameter	Keterangan
1	Event	Identifikasi dari kondisi yang terjadi
2	Time	Waktu dari string penelusurang paket dibuat
3	From node	Id dari node asal saat tracing
4	To node	Id dari node tujun saat tracing
5	Packet type	Tipe dari paket
6	Packet size	Ukuran dari paket
7	Flags	7 angka flag string
8	Fid	Flow id
9	Source address	Lokasi node sumber
10	Destination address	Lokasi node tujuan
11	Sequence number	Nomor urut
12	Packet id	Identitas dari paket

```
n -t * -s 0  -x 494.320078750275 -y
227.44993454767101 -Z 0 -z 20  -v
circle -c black
n -t * -s 1  -x 433.15572541821501 -y
71.338539206985004 -Z 0 -z 20  -v
circle -c black
n -t * -s 2  -x 234.81613021850501 -y
337.87620562786498 -Z 0 -z 20  -v
circle -c black
n -t * -s 3  -x 409.75550120676502 -y
216.91520678307 -Z 0 -z 20  -v circle
-c black
n -t * -s 4  -x 0.034648798687 -y
75.546624081605003 -Z 0 -z 20  -v
circle -c black
```

```
n -t * -s 5  -x 219.35771191874599 -y
193.47588124288799 -Z 0 -z 20  -v
circle -c black
n -t * -s 6  -x 302.94390378341399 -y
187.722536773954 -Z 0 -z 20  -v
circle -c black
n -t * -s 7  -x 320.62579983616399 -y
373.27790022636799 -Z 0 -z 20  -v
circle -c black
n -t * -s 8  -x 362.056983636658 -y
345.83390418753299 -Z 0 -z 20  -v
circle -c black
n -t * -s 9  -x 192.20256831919201 -y
337.25647025260702 -Z 0 -z 20  -v
circle -c black
n -t * -s 10  -x 422.77973544974498 -
y 202.48874796499999 -Z 0 -z 20  -v
circle -c black
n -t * -s 11  -x 346.85375074985899 -
y 157.67471346686099 -Z 0 -z 20  -v
circle -c black
n -t * -s 12  -x 212.17231815490899 -
y 122.31959451513301 -Z 0 -z 20  -v
circle -c black
n -t * -s 13  -x 38.144999089221997 -
y 227.96674709699201 -Z 0 -z 20  -v
circle -c black
```

Gambar 4.18 Isi dari nam *file*

Table 4.3 Parameter nam *file*

No.	Parameter	Keterangan
1	N node	Node
2	-t time	Waktu
3	-s node id	Id pada node
4	-x x location	Posisi x node
5	-y y location	Posisi y node
6	-Z node start	Id node awal
7	-z size of node	Jumlah node
8	-v shape	Bentuk node (*Circle, Box* dan *Hexagon)*
9	-c color	Warna pada *node*

4.4. *Implementasi Metriks Analisis*

Simulasi yang telah dijalankan oleh NS-2 menghasilkan output berupa *trace file* bernama trace.tr yang berisikan data mengenai apa saja yang terjadi selama simulasi dalam bentuk *plain text*. Dari data *trace file*, dapat dianalisa performa dari *routing protocol* yang dijalankan. Metrik yang akan dianalisa adalah *packet delivery ratio*, *end-to-end delay* dan *Routing Overhead.* Implementasi dari metrik-metrik tersebut menggunakan bahasa AWK.

4.4.1. Implementasi Packet Delivery Ratio

Packet Delivery Ratio didapatkan dengan membandingkan antara jumlah data yang diterima dengan jumlah data yang dikirim oleh agen pada suatu *trace file*. Pada Persamaan 3.1 telah dijelaskan rumus bagaimana menghitung *packet delivery ratio*. Kemudian rumus tersebut diimplementasikan kedalam skrip awk. Didalam skrip awk tersebut dilakukan penambahan jumlah paket data yang dikirim dengan syarat pada baris trace yang bersangkutan memiliki kondisi pada kolom pertama ditandai dengan huruf “s” yang artinya *send packet,* dengan kolom ketiga menunjukan *node* yang melakukan pengiriman merupakan node

1, lalu pada kolom ke-4 dan ke-7 berturut-turut memiliki kode "AGT" dan "CBR" yang menunjukan kalau pengiriman paket yang berlangsung merupakan pengiriman paket data. Kemudian penambahan jumlah paket data yang diteriam dengan syarat pada baris trace yang bersangkutan memiliki kondisi pada kolom pertama ditandai dengan huruf "r" yang artina *receive packet*, dengan kolom ketiga menunjukan *node* yang melakukan pengiriman merupakan node 1, lalu pada kolom ke-4 dan ke-7 berturut-turut memiliki kode "AGT" dan "CBR" yang menunjukan kalau pengiriman paket yang berlangsung merupakan pengiriman paket data.

Pseudocode awk untuk menghitung *packet delivery ratio* berdasarkan kedua informasi tersebut dapat dilihat pada Gambar 4.19 cara menjalankan skrip awk dapat dilihat pada perintah awk di bawah ini.

```
BEGIN {
        sendLine = 0;
        recvLine = 0;
        forwardLine = 0;
}
$0~/^s.*AGT/{
        sendLine++
}
$0~/^r.*AGT/{
        recvLine++
}
$0~/^f.*RTR/{
        forwardLine++
}
END{
printf (recvLine/sendLine)*100;
}
```

Gambar 4.19 *Pseudocode* dari Packet Delivery Ratio

Hasil dari PDR inimenghasilkan output dalam persen(%). Hasil dari perintah yang dijalankan adalah *packet delivery ratio* dari simulasi yang telah dijalankan dapat dilihat pada Gambar 4.20.

```
yosuanov@yosuanov-VirtualBox:~/TA$ awk -f PDR.awk test1TRG5.tr
Ratio:98.2278
yosuanov@yosuanov-VirtualBox:~/TA$ awk -f PDR.awk test1TRG10.tr
Ratio:98.7277
yosuanov@yosuanov-VirtualBox:~/TA$ awk -f PDR.awk test1TRG15.tr
Ratio:92.1671
```

Gambar 4.20 Contoh kode perintah dan hasil nilai PDR

4.4.2. Implementasi End-to-End Delay

Persamaan *End to end delay* didasarkan pada Persamaan 3.2. *End-to-End Delay* adalah waktu yang dibutuhkan oleh paket dari node asal ke node tujuan dalam satuan detik. Persamaan itu diterapkan dalam skrip awk. Skrip awk untuk menghitung *end to end delay* berdasarkan kedua informasi tersebut dapat dilihat pada lampiran. Untuk skrip dari *end to end delay*, data kolom ke-6 dimasukkan kedalam variabel "seqno" dengan syarat kondisi kolom pertama menunjukkan huruf "s" dan kolom keempat adalah "AGT". Pencatatan waktu mulai dari paket data yang dikirim, dengan data pada kolom kedua dimasukan kedalam array *start_time*, dengan array sebanyak variabel "seqno" dengan syarat kondisi kolom pertama menunjukkan huruf "s" serta kolom keempat adalah "AGT". Kemudian untuk menghitung waktu dari akhir proses, data pada kolom kedua dari baris trace dengan kondisi kolom pertama menunujukan huruf "r" yang artinya *receive* dan kolom ketujuh ada "CBR akan dimasukan kedalam *array* bernama *end_time*, dengan *array* sebanyak ukuran dari paket pada kolom keenam. Apabila ada baris trace dengan kondisi kolom kesatu adalah "D" yang artinya drop dan array *end_time* akan diisi "-1". Lalu array *end_time* akan dikurangi dengan *start_time* dan dimasukkan dalam array baru bernama *delay_time* sejumlah variabel *seqno* menggunakan iterasi, serta ada variabel

count untuk menghitung jumlah dlay time yang masuk, dengan kondisi variabel *end_time* lebih besar dari 0. Lalu apabila variabel *end_time* berisi data kurang dari 0 maka *array delay_time* akan diisi dengan nilai -1 yang artinya proses pengiriman paket tersebut tidak dipakai. Kemudian seluruh *array delay_time* dijumlahkan dan dimasukkan kedalam variabel baru bernam *n_to_n_delay*. Kemudian variabel *n_to_n_delay* tersebut dibagi dengan variabel *count* yang hasilnya merupakan *output* dari analisis *end to end delay*. Gambar 4.21 merupakan *pseudocode* algoritma E2D. Untuk implementasi bisa dilihat pada lampiran.

```
BEGIN{
        seqno = -1;
        count = 0;}
{
        if($4 == "AGT" and $1 == "s"
        and seqno < $6){
                seqno = $6;}
        if($4 == "AGT" and $1 == "s"){
                start_time[$6] = $2;}
        else if(($7 == "cbr") and
        ($1 == "r")){
                end_time[$6] = $2;}
        else if($1 == "D" and
        $7 == "cbr"){
                end_time[$6] = -1;}
}
END{ for(i=0; i<=seqno; i++) {
        if(end_time[i] > 0){
                delay[i] = end_time[i] -
                start_time[i];
                count++;
        }
        else{
                delay[i] = -1;
        }
}
```

```
for(i=0; i<=seqno; i++){
        if(delay[i] > 0){
                n_to_n_delay =
                n_to_n_delay + delay[i];
        }
}
n_to_n_delay = n_to_n_delay/count;
print n_to_n_delay * 1000 ;
}
```

Gambar 4.21 Pseudocode *end to end delay*

Contoh dan hasil dari perintah yang dijalankan adalah *end-to-end delay* dari simulasi yang telah dijalankan dapat dilihat pada Gambar 4.22.

```
yosuanov@yosuanov-VirtualBox:~/TA$ awk -f e2e.awk test1TRG5.tr
end to end delay : 37.4826
yosuanov@yosuanov-VirtualBox:~/TA$ awk -f e2e.awk test1TRG10.tr
end to end delay : 15.8907
yosuanov@yosuanov-VirtualBox:~/TA$ awk -f e2e.awk test1TRG15.tr
end to end delay : 1057.83
```

Gambar 4.22 Contoh hasil nilai E2E

4.4.3. Implementasi Routing Overhead

Perhitungan RO didasarkan pada Persamaan 3.3. Kemudian persamaan itu diterapkan dalam skrip awk. Skrip awk untuk menghitung RO berdasarkan kedua informasi tersebut dapat dilihat pada lampiran.

Implementasi dari *routing overhead* adalah menghitung jumlah paket yang diterima dan jumlah dari besaran paket yang ada pada setiap baris. Paket yang diterima dihitung apabila kondisi kolom 1 menunjukkan huruf “s” atau “f” serta pada kolom keempat adalah “RTR” yang artinya paket *routing*, kemudian semua paket tadi dijumlahkan. Lalu untuk besaran paket dihitung dengan menjumlahkan kolom ke-8 dari setiap baris

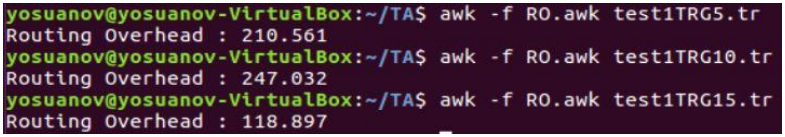
tracing dengan kondisi kolom ke-1 menunjukkan huruf “s” atau “f” serta pada kolom keempat adalah “RTR”. Kemudian jumlah paket yang diterima dibagi dengan jumlah dari besaran paket yang ada dan hasilnya merupakan hasil dari analisis *routing overhead*.

Contoh perintah untuk memanggil skrip tcl untuk menganilisis *trace file* dan *outputnya* dapat dilihat pada perintah awk di Gambar 4.23.

```
BEGIN{
recvs = 0;
besaran_paket = 0;
}
{
if (($1 == "s" or $1 == "f") and
$4 =="RTR" and $7 == "AODV"){
recvs++;
}
if (($1 == "s" or $1 == "r") and
$4 =="RTR")
{
besaran_paket = besaran_paket + $8;
}
}
END{
print besaran_paket/recvs;
}
```

Gambar 4.23 Pseudocode file ro.awk

Hasil dari perintah yang dijalankan adalah *end-to-end delay* dari simulasi yang telah dijalankan dapat dilihat pada Gambar 4.24.



Gambar 4.24 Contoh hasil nilai RO

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada dari skenario NS-2 yang telah dibuat dan hasil dari perhitungan awk yang sudah dijalankan. Pengujian fungsionalitas akan dibagi ke dalam beberapa skenario pengujian.

5.1. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan pada lingkungan dan alat kakas seperti yang tertera pada

Table 5.1 Lingkungan Pengujian Sistem

Perangkat Keras	- Laptop HP R202TX o Prosesor Intel(R) Core(TM) i5-5700u CPU @ 2.70GHz o RAM 6 GB o HDD 500 GB
Perangkat Lunak	- Laptop HP R202TX o Sistem Operasi Windows 10 pro, Linux Ubunru 16.04 LTS 64 bit (NS-2, AODV, *Mobility Generation, Traffic Connection Genneration, TwoRayGround, FreeSpace*.)

5.2. Kriteria Pengujian

Pengujian pada skenario yang dihasilkan oleh *mobility generator* default dari NS-2 menggunakan beberapa kriteria. Pada Table 5.2 menunjukan kriteria-kriteria yang ditentukan didalam skenario.

.

Table 5.2 Penjelasan skenario pengujian

Kriteria	Spesifikasi
Skenario	MANET

Kecepatan Maksimal Perpindahan node (m/s)	5, 10, 15
Jumlah percobaan	10
Jarak node 1 dan node 2	Acak
Posisi awal node	Acak
Pergerakan	Acak
Protokol Routing	AODV
Jenis Prpagasi	*FreeSpace, TwoRayGround*
Pengiriman Paket Data	0 – 100 Detik

5.3. *Analisis Packet Delivery Ratio (PDR)*

Trace file dihasilkan setelahmenjalankan program skenario *node- movement* (*mobility generation*) dan *traffic-connection pattern* menggunakan 2 model transmisi *TwoRayGround* dan *FreeSpace.* Kemudian *trace file* tersebut dianalisis nilai PDR menggunakan Skrip pdr.awk. Hasil tiap perhitungan PDR untuk skenario ditabulasikan dan dirata ratakan. Hasil dari perhitungan untuk transmisi *TwoRayGround* dapat dilihat pada Table 5.3.

Table 5.3 Hasil uji coba PDR selama 10 kali dengan menggunakan model transmisi *TwoRayGround*

Percobaan TwoRayGround	PDR		
	5m/s	10m/s	15m/s
P1	98.2287	98.7277	92.1671
P2	91.6877	97.7330	94.6154
P3	93.3482	93.8144	69.0722
P4	96.1637	98.4925	95.9391
P5	97.1649	94.4162	87.0801
P6	88.2353	97.6501	92.4051
P7	90.4145	96.1240	86.0759
P8	87.5000	96.0733	96.0630
P9	97.2010	94.6154	97.9221
P10	97.1939	82.9517	92.4051
Rata-rata	93.7138	95.0598	90.3745

Table 5.4 Rata-rata hasil PDR dengan propagasi TwoRayGround

Kecepatan Maksimal Perpindahan Node (m/s)	PDR (%)
5	93.7138
10	95.0598
15	90.3745

Pada Table 5.3 yang merupakan hasil uji coba PDR denggan menggunakan transmisi *TwoRayGround* Percobaan pertama pada kecepatan 5m/s menghasilkan nilai 98.2287%, kecepatan 10m/s menghasilkan nilai 98.7727%, kecepatan 15m/s mengahasilkan 92.1671% yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Percobaan kedua pada kecepatan 5m/s menghasilkan nilai 91.6877%, kecepatan 10m/s menghasilkan nilai 97.7330%, kecepatan 15m/s mengahasilkan 94.6154% yang artinya nilai tersebut bersifat fluktuatif cenderung meningkat. Percobaan ketiga pada kecepatan 5m/s menghasilkan nilai 93.3482%, kecepatan 10m/s menghasilkan nilai 93.8144%, kecepatan 15m/s mengahasilkan 69.0722% yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Percobaan keempat pada kecepatan 5m/s menghasilkan nilai 96.1637%, kecepatan 10m/s menghasilkan nilai 98.4925%, kecepatan 15m/s mengahasilkan 95.9391% yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Percobaan kelima pada kecepatan 5m/s menghasilkan nilai 97.1649%, kecepatan 10m/s menghasilkan nilai 94.4162%, kecepatan 15m/s mengahasilkan 87.0801% yang artinya nilai tersebut mengalami penurunan pada tiap pertambahan maksimal kecepatan. Percobaan keenam pada kecepatan 5m/s menghasilkan nilai 88.2353%, kecepatan 10m/s menghasilkan nilai 97.6501%, kecepatan 15m/s mengahasilkan 92.4051% yang artinya nilai tersebut bersifat fluktuatif cenderung meningkat pada tiap pertambahan kecepatan maksimal. Percobaan ketujuh pada kecepatan 5m/s menghasilkan nilai 90.4145%, kecepatan 10m/s menghasilkan nilai 96.1240%, kecepatan 15m/s mengahasilkan 86.0759% yang artinya nilai tersebut bersifat fluktuatif cenderung meningkat pada tiap

penambahan kecepatan maksimal. Percobaan kedelapan pada kecepatan 5m/s menghasilkan nilai 87.5000%, kecepatan 10m/s menghasilkan nilai 96.0733%, kecepatan 15m/s mengahasilkan 96.0630% yang artinya nilai tersebut bersifat fluktuatif cenderung meningkat pada setiap pertambahan kecepatan maksimal. Percobaan kesembilan pada kecepatan 5m/s menghasilkan nilai 97.2010%, kecepatan 10m/s menghasilkan nilai 94.6154%, kecepatan 15m/s mengahasilkan 97.9221% yang artinya nilai tersebut bersifat fluktuatif cenderung meningkat pada tiap pertambahan kecepatan maksimal. Percobaan kesepuluh pada kecepatan 5m/s menghasilkan nilai 97.1939%, kecepatan 10m/s menghasilkan nilai 82.9517%, kecepatan 15m/s mengahasilkan 92.4051% yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Dari sepuluh data hasil percobaan itu ada 5 hasil fluktuatif cenderung meningkat, 4 hasil bersifat fluktuatif cender menurun dan hasil yang bersifat menurun.

Table 5.5 Hasil uji coba PDR selama 10 kali dengan menggunakan model transmisi *Freespaee*

Percobaan Freespace	PDR		
	5m/s	10m/s	15m/s
P1	97.4555	99.2424	97.201
P2	98.7277	97.9275	98.4536
P3	99.75	97.1503	95.9184
P4	96.8338	97.416	95.8656
P5	98.1865	96.124	96.3158
P6	97.9747	98.7113	94.9109
P7	98.7593	95.8974	97.1722
P8	98.4252	98.2143	97.201
P9	98.9848	97.1722	97.9644
P10	98.6911	95.9494	92.287
Rata-rata	98.37886	97.38048	96.32899

Table 5.6 Rata-rata hasil PDR dengan propagasi *FreeSpace*

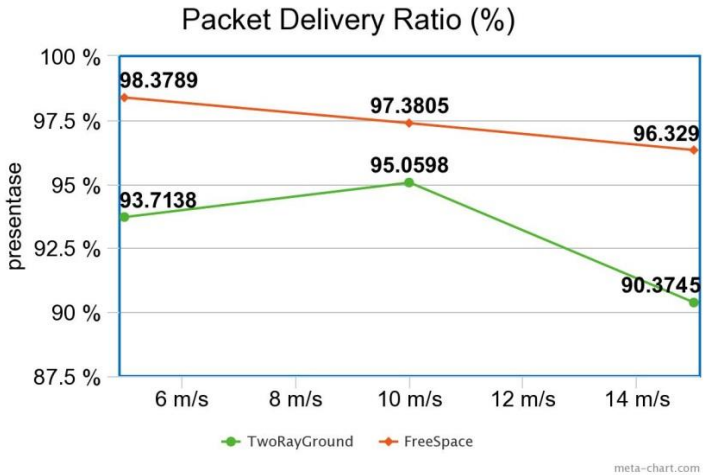
Kecepatan Maksimal Perpindahan	PDR (%)

Node (m/s)	
5	98.37886
10	97.38048
15	96.32899

Pada Table 5.5 yang merupakan hasil uji coba PDR denggan menggunakan transmisi *FreeSpace* Percobaan pertama pada kecepatan 5m/s menghasilkan nilai 97.4555%, kecepatan 10m/s menghasilkan nilai 99.2424%, kecepatan 15m/s mengahasilkan 97.2010% yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Percobaan kedua pada kecepatan 5m/s menghasilkan nilai 98.7277%, kecepatan 10m/s menghasilkan nilai 97.9275%, kecepatan 15m/s mengahasilkan 98.4536% yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Percobaan ketiga pada kecepatan 5m/s menghasilkan nilai 99.7500%, kecepatan 10m/s menghasilkan nilai 97.1503%, kecepatan 15m/s mengahasilkan 95.9184% yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Percobaan keempat pada kecepatan 5m/s menghasilkan nilai 96.8338%, kecepatan 10m/s menghasilkan nilai 97.4160%, kecepatan 15m/s mengahasilkan 95.8656% yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Percobaan kelima pada kecepatan 5m/s menghasilkan nilai 98.1865%, kecepatan 10m/s menghasilkan nilai 96.1240%, kecepatan 15m/s mengahasilkan 96.3158% yang artinya nilai tersebut bersifat fluktuatif cenderung menurun pada tiap pertambahan maksimal kecepatan. Percobaan keenam pada kecepatan 5m/s menghasilkan nilai 97.9747%, kecepatan 10m/s menghasilkan nilai 98.7113%, kecepatan 15m/s mengahasilkan 94.9109% yang artinya nilai tersebut mengalami penurunan pada tiap pertambahan kecepatan maksimal. Percobaan ketujuh pada kecepatan 5m/s menghasilkan nilai 98.7593%, kecepatan 10m/s menghasilkan nilai 95.8974%, kecepatan 15m/s mengahasilkan 97.1722% yang artinya nilai tersebut bersifat fluktuatif cenderung menurun pada tiap penambahan kecepatan maksimal. Percobaan kedelapan pada kecepatan 5m/s menghasilkan nilai 98.4252%, kecepatan 10m/s

menghasilkan nilai 98.2143%, kecepatan 15m/s mengahasilkan 97.2010% yang artinya nilai tersebut mengalami penurunan pada setiap pertambahan kecepatan maksimal. Percobaan kesembilan pada kecepatan 5m/s menghasilkan nilai 98.9848%, kecepatan 10m/s menghasilkan nilai 97.1722%, kecepatan 15m/s mengahasilkan 97.9644% yang artinya nilai tersebut bersifat fluktuatif cenderung menurun pada tiap pertambahan kecepatan maksimal. Percobaan kesepuluh pada kecepatan 5m/s menghasilkan nilai 98.6911%, kecepatan 10m/s menghasilkan nilai 95.9494%, kecepatan 15m/s mengahasilkan 92.2870% yang artinya nilai tersebut mengalami penurunan.

Gambar 5.1 menunjukkan hasil dari PDRyang berasal dari model transmisi *TwoRyGround* dan *FreeSpace* pada jaringan MANET dengan skenario *node-movement (mobility generation)* dan *traffic-connection pattern* dengan sifat *Random Way Point.* Pada output PDR dari routing AODV yang dihasilkan untuk model transmisi *TwoRayGround* bersifat fluktuatif dan cenderung menurun, untuk model transmisi *FreeSpace* mengalami penurunan pada setiap pertambahan kecepatan maksimum.



Gambar 5.1 Grafik nilai PDR

Nilai dari *Packet Delivery Ratio* dengan transmisi *TwoRayGround* menunjukan nilai yang fluktuatif dan cenderung menurun. Nilai yang fluktuatif tadi disebabkan karena pergerakan dari node yang acak. Bisa dilihat bahwa nilai rata rata pada kecepatan Maksimal 5m/s adalah 93.7138% sementara pada kecepatan maksimal 10 memiliki nilai 95.0598% dan pada kecepatan maksimal 15 bernilai 90.3745%.

Lalu nilai dari Packet Delivery Ratio dengan transmisi *FreeSpace* menunjukan nilai menurun. Dapat dilihat nilai PDR pada kecepatan maksimal 5m/s adalah 98.3789% sementara pada kecepatan maksimal 10m/s adalah 97.3805% dan pada kecepatan maksimal 15m/s adalah 96.329%.

Hal ini menunjukan bahwa semakin besar kecepatan maksimal yang dipunya maka semakin kecil pula paket data yang berhasil dikirimkan. Selain itu nilai PDR dengan transmisi *FreeSpace* lebih besar daripada transmisi *TwoRayGround*. Hal itu menunjukan bahwa saluran transmisi mempengaruhi paket data yang berhasil dikirimkan, serta nilai fluktuatif dari hasil rata-rata dari PDR dari model transmisi *TwoRayGround* disebabkan oleh posisi dan pergerakan node yang acak.

5.4. *Analisis Routing Overhead(RO)*

Hasil *Trace file* dari menjalankan skenario *node-movement* (*mobility generation*) yang menggunakan model transmisi *TwoRayGround* dan *FreeSpace* yang menghasilkan nilai *Routing Overhead* dianalisis menggunakan skrip RO.awk. Hasil tiap perhitungan RO skenario ditabulasikan dan dirata-ratakan menjadi seperti pada Table 5.5 dan Table 5.6.

Table 5.7 Hasil uji coba RO selama 10 kali dengan menggunakan model transmisi *TwoRayGround*

Percobaan TwoRayGround	RO		
	5m/s	10m/s	15m/s
P1	210.561	247.032	118.897
P2	264.832	161.997	183.46
P3	231.91	129.436	115.222

P4	176.286	180.817	153.743
P5	242.475	157.039	145.766
P6	232.835	240.292	115.872
P7	157.879	150.899	139.407
P8	222.664	193.723	139.074
P9	207.405	127.873	196.953
P10	187.057	175.267	137.374
Rata-rata	213.3904	176.4375	144.5768

Table 5.8 Nilai hasil RO dengan propagasi TwoRayGround

Kecepatan Maksimal Perpindahan Node (m/s)	RO (paket)
5	213.390
10	176.438
15	144.577

Table 5.7 yang merupakan hasil uji coba *routing overhead* denggan menggunakan transmisi *TwoRayGround* Percobaan pertama pada kecepatan 5m/s menghasilkan nilai 210.561 paket, kecepatan 10m/s menghasilkan nilai 247.032 paket, kecepatan 15m/s mengahasilkan 118.897 paket yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Percobaan kedua pada kecepatan 5m/s menghasilkan nilai 264.832 paket, kecepatan 10m/s menghasilkan nilai 161.997 paket, kecepatan 15m/s mengahasilkan 183.460 paket yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Percobaan ketiga pada kecepatan 5m/s menghasilkan nilai 231.910 paket, kecepatan 10m/s menghasilkan nilai 129.436 paket, kecepatan 15m/s mengahasilkan 115.222 paket yang artinya nilai tersebut mengalami penurunan. Percobaan keempat pada kecepatan 5m/s menghasilkan nilai 176.286 paket, kecepatan 10m/s menghasilkan nilai 180.817 paket, kecepatan 15m/s mengahasilkan 153.743 paket yang artinya nilai tersebut mengalami penurunan. Percobaan kelima pada kecepatan 5m/s menghasilkan nilai 242.475 paket, kecepatan 10m/s menghasilkan nilai 157.039

paket, kecepatan 15m/s mengahasilkan 145.766 paket yang artinya nilai tersebut mengalami penurunan pada tiap pertambahan maksimal kecepatan. Percobaan keenam pada kecepatan 5m/s menghasilkan nilai 232.835 paket, kecepatan 10m/s menghasilkan nilai 240.292 paket, kecepatan 15m/s mengahasilkan 115.872 paket yang artinya nilai tersebut bersifat fluktuatif cenderung menurun pada tiap pertambahan kecepatan maksimal. Percobaan ketujuh pada kecepatan 5m/s menghasilkan nilai 157.879 paket, kecepatan 10m/s menghasilkan nilai 150.899 paket, kecepatan 15m/s mengahasilkan 139.407 paket yang artinya nilai tersebut mengalami penurunan pada tiap penambahan kecepatan maksimal. Percobaan kedelapan pada kecepatan 5m/s menghasilkan nilai 222.664 paket, kecepatan 10m/s menghasilkan nilai 193.723 paket, kecepatan 15m/s mengahasilkan 139.074 paket yang artinya nilai tersebut mengalami penurunan pada setiap pertambahan kecepatan maksimal. Percobaan kesembilan pada kecepatan 5m/s menghasilkan nilai 207.405 paket, kecepatan 10m/s menghasilkan nilai 127.873 paket, kecepatan 15m/s mengahasilkan 196.953 paket yang artinya nilai tersebut bersifat fluktuatif cenderung menurun pada tiap pertambahan kecepatan maksimal. Percobaan kesepuluh pada kecepatan 5m/s menghasilkan nilai 187.057 paket, kecepatan 10m/s menghasilkan nilai 175.267 paket, kecepatan 15m/s mengahasilkan 137.374 paket yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Dari sepuluh data hasil percobaan itu ada 5 hasil bersifat fluktuatif cender menurun dan 5 hasil yang bersifat menurun.

Table 5.9 Hasil uji coba RO selama 10 kali dengan menggunakan model transmisi *Freespace*

Percobaan Freespace	RO		
	5m/s	10m/s	15m/s
P1	148.6	213.073	132.107
P2	239.453	145.807	200.21
P3	377.062	147.228	134.916
P4	173.02	148.406	127.753

P5	202.529	138.187	125.552
P6	192.273	261.981	112.162
P7	169.673	128.593	161.518
P8	217.97	180.859	137.409
P9	268.48	123.407	176.973
P10	193.064	145.745	119.516
Rata-rata	218.2124	163.3286	142.8116

Table 5.10 Rata-rata hasil RO dengan propagasi *FreeSpace*

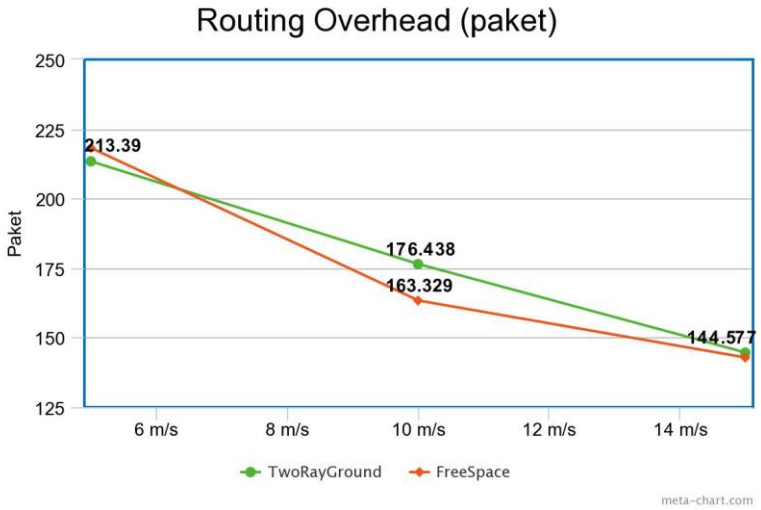
Kecepatan Maksimal Perpindahan Node (m/s)	RO (paket)
5	218.212
10	163.329
15	142.812

Table 5.9 yang merupakan hasil uji coba *routing overhad* denggan menggunakan transmisi *FreeSpace* Percobaan pertama pada kecepatan 5m/s menghasilkan nilai 148.600 paket, kecepatan 10m/s menghasilkan nilai 213.073 paket, kecepatan 15m/s mengahasilkan 132.107 paket yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Percobaan kedua pada kecepatan 5m/s menghasilkan nilai 239.453 paket, kecepatan 10m/s menghasilkan nilai 145.807 paket, kecepatan 15m/s mengahasilkan 200.210 paket yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Percobaan ketiga pada kecepatan 5m/s menghasilkan nilai 377.062 paket, kecepatan 10m/s menghasilkan nilai 147.228 paket, kecepatan 15m/s mengahasilkan 134.916 paket yang artinya nilai tersebut mengalami penurunan. Percobaan keempat pada kecepatan 5m/s menghasilkan nilai 173.020 paket, kecepatan 10m/s menghasilkan nilai 148.406 paket, kecepatan 15m/s mengahasilkan 127.753 yang artinya nilai tersebut mengalami penurunan pada setiap kenaikan kecepatan maksimal. Percobaan kelima pada kecepatan 5m/s menghasilkan nilai 202.529 paket, kecepatan 10m/s menghasilkan nilai 138.187, kecepatan 15m/s mengahasilkan

125.552 yang artinya nilai tersebut mengalami penurunan pada tiap pertambahan maksimal kecepatan. Percobaan keenam pada kecepatan 5m/s menghasilkan nilai 192.273 paket, kecepatan 10m/s menghasilkan nilai 261.981, kecepatan 15m/s mengahasilkan 112.162 paket yang artinya nilai tersebut bersifat fluktuatif cenderung menurun pada tiap pertambahan kecepatan maksimal. Percobaan ketujuh pada kecepatan 5m/s menghasilkan nilai 169.673 paket, kecepatan 10m/s menghasilkan nilai 128.593 paket, kecepatan 15m/s mengahasilkan 161.518 paket yang artinya nilai tersebut bersifat fluktuatif cenderung menurun pada tiap penambahan kecepatan maksimal. Percobaan kedelapan pada kecepatan 5m/s menghasilkan nilai 217.970 paket, kecepatan 10m/s menghasilkan nilai 180.859 paket, kecepatan 15m/s mengahasilkan 137.409 paket yang artinya nilai tersebut mengalami penurunan pada setiap pertambahan kecepatan maksimal. Percobaan kesembilan pada kecepatan 5m/s menghasilkan nilai 268.480 paket, kecepatan 10m/s menghasilkan nilai 123.407 paket, kecepatan 15m/s mengahasilkan 176.973 paket yang artinya nilai tersebut bersifat fluktuatif cenderung menurun pada tiap pertambahan kecepatan maksimal. Percobaan kesepuluh pada kecepatan 5m/s menghasilkan nilai 193.064 paket, kecepatan 10m/s menghasilkan nilai 145.745 paket, kecepatan 15m/s mengahasilkan 119.516 paket yang artinya nilai tersebut mengalami penurunan. Dari sepuluh data hasil percobaan itu ada 5 hasil fluktuatif cenderung meningkat dan 5 hasil yang bersifat menurun.

Pada Gambar 5.2 merupakan hasil pengujian model transmisi *TwoRayGround* dan *FreeSpace.* Hasil rata-rata dari 10 kali pengujian *Routing Overhead* model transmisi *TwoRayGround* pada kecepatan 5m/s adalah 213.390 paket, lalu untuk hasil pada kecepatan 10m/s adalah sebesar 176.438 paket, serta hasil pada kecepatan 15m/s adalah 144.577 paket. Kemudian untuk Hasil rata-rata dari 10 kali pengujian *Routing Overhead* model transmisi *FreeSpace* pada kecepatan 5m/s adalah 218.212 paket, lalu untuk hasil pada kecepatan 10m/s adalah sebesar 163.329 paket, serta hasil pada kecepatan 15m/s adalah 142.812

paket



Gambar 5.2 Grafik nilai RO

Nilai dari *Routing Overhead* dengan transmisi *TwoRayGround* menunjukan nilai yang menurun. Nilai Bisa dilihat bahwa nilai rata rata pada kecepatan Maksimal 5m/s adalah 213.390 paket sementara pada kecepatan maksimal 10m/s memiliki nilai 176.438 paket dan pada kecepatan maksimal 15 m/s bernilai 144.577 paket.

Lalu nilai dari *Routing Overhead* dengan transmisi *FreeSpace* menunjukan nilai menurun. Dapat dilihat nilai PDR pada kecepatan maksimal 5m/s adalah 218.212 paket sementara pada kecepatan maksimal 10m/s adalah 163.329 paket dan pada kecepatan maksimal 15m/s adalah 142.812 paket.

Hasil analisis dari skenario bisa dilihat bahwa nilai rata rata dari *Routing Overhead* pada kedua model transmisi mengalami penurunan. Hal itu terjadi akibat mobilitas yang semakin tinggi. Mobilitas yang semakin meningkat memungkinkan terjadinya rute putus saat pengiriman paket data sehinga pengiriman paket dimasukkan ke dalam antrian dan

menunggu rute baru terbentuk sebelum pengiriman paket data dilanjutkan kembali.

Selain itu hasil analisis *Routing Overhead* dengan menggunakan transmisi *TwoRayGorund* memiliki hasil yang hampir sama dengan transmisi *FreeSpace*. Hal itu disebabkan oleh posisi dan pergerakan node yang dikonfigurasi secara acak sehingga dapat menjadi pengaruh pada model transmisi dan *routing protocol* tertentu.

5.5. *Analisis End-to-End Delay (E2E)*

Trace file hasil menjalankan program skenario *node-movement* (*mobiility generation*) menggunakan model transmisi *TwoRayGround* dan *FreeSpace* kemudian dianalisis nilai *End to End Delay* melalui Skrip e2e.awk. Hasil tiap perhitungan E2E skenario ditabulasikan dan dirata ratakan menjadi Table 5.7.

Table 5.11 Hasil uji coba end to end delay selama 10 kali dengan menggunakan model transmisi *TwoRayGround*

Percobaan Freespace	End to end delay		
	5m/s	10m/s	15m/s
P1	37.4826	15.9807	1057.83
P2	644.717	36.9672	557.945
P3	318.811	28.5568	616.507
P4	73.8968	70.5561	35.223
P5	37.5455	100.851	777.998
P6	678.214	29.5547	38.0872
P7	1348.35	128.019	777.498
P8	45.659	307.457	78.2049
P9	32.0481	715.231	298.789
P10	1389.82	109.575	23.3413
Rata-rata	460.6544	154.2749	426.1423

Table 5.12 Rata-rata nilai E2E dengan transmisi TwoRayGround

Kecepatan Maksimal Perpindahan Node (m/s)	End to End delay (Detik)
5	460.6544
10	154.2749
15	426.1423

Table 5.12 yang merupakan hasil uji coba *end to end delay* denggan menggunakan transmisi *TwoRayGround* Percobaan pertama pada kecepatan 5m/s menghasilkan nilai 37.4826 detik, kecepatan 10m/s menghasilkan nilai 15.9807 detik, kecepatan 15m/s mengahasilkan 1057.83 detik yang artinya nilai tersebut bersifat fluktuatif cenderung meningkat. Percobaan kedua pada kecepatan 5m/s menghasilkan nilai 664.717 detik, kecepatan 10m/s menghasilkan nilai 36.9672 detik, kecepatan 15m/s mengahasilkan 557.945 detik yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Percobaan ketiga pada kecepatan 5m/s menghasilkan nilai 318.811 detik, kecepatan 10m/s menghasilkan nilai 28.5568 detik, kecepatan 15m/s mengahasilkan 616.507 detik yang artinya nilai tersebut bersifat fluktuatif cenderung meningkat. Percobaan keempat pada kecepatan 5m/s menghasilkan nilai 73.8968 detik, kecepatan 10m/s menghasilkan nilai 70.5561 detik, kecepatan 15m/s menghasilkan 35.223 detik yang artinya nilai tersebut mengalami penurunan. Percobaan kelima pada kecepatan 5m/s menghasilkan nilai 37.5455 detik, kecepatan 10m/s menghasilkan nilai 100.851 detik, kecepatan 15m/s mengahasilkan 777.998 detik yang artinya nilai tersebut mengalami peningkatan pada tiap pertambahan maksimal kecepatan. Percobaan keenam pada kecepatan 5m/s menghasilkan nilai 678.214 detik, kecepatan 10m/s menghasilkan nilai 29.5547 detik, kecepatan 15m/s mengahasilkan 38.0872 detik yang artinya nilai tersebut bersifat fluktuatif cenderung menurun pada tiap pertambahan kecepatan maksimal. Percobaan ketujuh pada kecepatan 5m/s menghasilkan nilai 1348.35 detik,

kecepatan 10m/s menghasilkan nilai 128.019 detik, kecepatan 15m/s mengahasilkan 777.598 detik yang artinya nilai tersebut bersifat fluktuatif cenderung menurun pada tiap penambahan kecepatan maksimal. Percobaan kedelapan pada kecepatan 5m/s menghasilkan nilai 45.659 detik, kecepatan 10m/s menghasilkan nilai 307.457 detik, kecepatan 15m/s menghasilkan 78.2049 detik yang artinya nilai tersebut bersifat fluktuatif cenderung meningkat pada setiap pertambahan kecepatan maksimal. Percobaan kesembilan pada kecepatan 5m/s menghasilkan nilai 32.0481 detik, kecepatan 10m/s menghasilkan nilai 715.231, kecepatan 15m/s mengahasilkan 298.789 detik yang artinya nilai tersebut bersifat fluktuatif cenderung meningkat pada tiap pertambahan kecepatan maksimal. Percobaan kesepuluh pada kecepatan 5m/s menghasilkan nilai 1389.82 detik, kecepatan 10m/s menghasilkan nilai 109.575 detik, kecepatan 15m/s mengahasilkan 23.3413 detik yang artinya nilai tersebut bersifat fluktuatif cenderung meningkat. Dari sepuluh data hasil percobaan itu ada 5 hasil fluktuatif cenderung meningkat, 3 hasil bersifat fluktuatif cender menurun, 1 hasil yang bersifat menurun dan 1 hasil yang meningkat.

Table 5.13 Hasil uji coba *end to emd delay* selama 10 kali dengan menggunakan model transmisi *Freespace*

Percobaan Freespace	End to End Delay		
	5m/s	10m/s	15m/s
P1	35.3385	14.2343	420.07
P2	149.709	27.0901	12.9515
P3	37.3322	19.5955	31.7933
P4	69.0072	69.0604	169.086
P5	58.5735	251.694	29.4006
P6	21.3249	25.491	35.0565
P7	34.3225	29.211	930.311
P8	24.9852	23.9606	40.6634
P9	26.4644	82.5478	19.9494
P10	24.8455	84.6346	18.9467

Rata-rata	48.19029	62.75193	170.8228

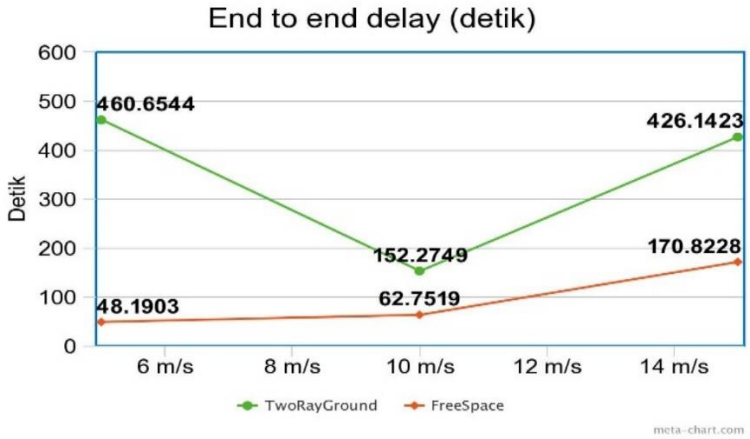
Table 5.14 Rata-rata E2D dengan transmisi Freespace

Kecepatan Maksimal Perpindahan Node (m/s)	End to End Delay (Detik)
5	48.1903
10	62.7519
15	170.8228

Table 5.14 yang merupakan hasil uji coba PDR denggan menggunakan transmisi *TwoRayGround* Percobaan pertama pada kecepatan 5m/s menghasilkan nilai 35.3385 detik, kecepatan 10m/s menghasilkan nilai 14.2343 detik, kecepatan 15m/s mengahasilkan 420.0700 detik yang artinya nilai tersebut mengalami peningkatan. Percobaan kedua pada kecepatan 5m/s menghasilkan nilai 149.709 detik, kecepatan 10m/s menghasilkan nilai 27.0901 detik, kecepatan 15m/s mengahasilkan 12.9515 detik yang artinya nilai tersebut mengalami penurunan. Percobaan ketiga pada kecepatan 5m/s menghasilkan nilai 37.3322 detik, kecepatan 10m/s menghasilkan nilai 19.5955 detik, kecepatan 15m/s mengahasilkan 31.7933 detik yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Percobaan keempat pada kecepatan 5m/s menghasilkan nilai 69.0072 detik, kecepatan 10m/s menghasilkan nilai 69.0604 detik, kecepatan 15m/s mengahasilkan 169.086 yang artinya nilai tersebut bersifat fluktuatif cenderung meningkat. Percobaan kelima pada kecepatan 5m/s menghasilkan nilai 58.5735 detik, kecepatan 10m/s menghasilkan nilai 251.694 detik, kecepatan 15m/s mengahasilkan 29.4006 detik yang artinya nilai tersebut bersifat fluktuatif cenderung menurun pada tiap pertambahan maksimal kecepatan. Percobaan keenam pada kecepatan 5m/s menghasilkan nilai 21.3249 detik, kecepatan 10m/s menghasilkan nilai 25.4910 detik, kecepatan 15m/s menghasilkan 35.0565 detik yang artinya nilai tersebut mengalami peningkatan pada tiap pertambahan kecepatan maksimal. Percobaan ketujuh pada kecepatan 5m/s

menghasilkan nilai 34.3225 detik, kecepatan 10m/s menghasilkan nilai 29.2110 detik, kecepatan 15m/s mengahasilkan 930.311 detik yang artinya nilai tersebut bersifat fluktuatif cenderung meningkat pada tiap penambahan kecepatan maksimal. Percobaan kedelapan pada kecepatan 5m/s menghasilkan nilai 24.9851 detik, kecepatan 10m/s menghasilkan nilai 23.9606 detik, kecepatan 15m/s mengahasilkan 40.6634 detik yang artinya nilai tersebut bersifat fluktuatif cenderung meningkat pada setiap pertambahan kecepatan maksimal. Percobaan kesembilan pada kecepatan 5m/s menghasilkan nilai 26.4644 detik, kecepatan 10m/s menghasilkan nilai 82.5478 detik, kecepatan 15m/s mengahasilkan 19.9494 detik yang artinya nilai tersebut bersifat fluktuatif cenderung menurun pada tiap pertambahan kecepatan maksimal. Percobaan kesepuluh pada kecepatan 5m/s menghasilkan nilai 24.8455 detik, kecepatan 10m/s menghasilkan nilai 84.6346 detik, kecepatan 15m/s mengahasilkan 18.9467 detik yang artinya nilai tersebut bersifat fluktuatif cenderung menurun. Dari sepuluh data hasil percobaan itu ada 4 hasil fluktuatif cenderung menurun, 3 hasil bersifat fluktuatif cender meningkat, 2 hasil yang bersifat meningkat serta 1 hasil mengalami penurunan.

Pada model transmisi *TwoRayGround* memiliki performa yang sangat buruk dibanding *FreeSpace*. Karena *TwoRayGround* memiliki waktu yang cukup tinggi. Untuk hasil dari *TwoRayGround* pada kecepatan 5m/s adalah 460.6554 detik, kecepatan 10 m/s adalah 152.2749 detik, serta pada kecepatan 15m/s adalah 426.1423 detik. Kemudian pada model transmisi FreeSpace mengalami peningkatan yang sangat signifikan pada kecepatan maksimal 15m/s. Hasil dari *end to end delay* dengan model tranmisi *FreeSpace* pada kecepatan 5m/s adalh 46.1903, lalu pada kecepatan 10m/s adalah 62.7519 detik, pada kecepatan 15m/s adalah 170.8228. Dari sepuluh data hasil percobaan itu ada 5 hasil fluktuatif cenderung meningkat, 3 hasil bersifat fluktuatif cender menurun, 1 hasil yang bersifat menurun dan 1 hasil yang meningkat.



Gambar 5.3 Grafik hasil nilai E2E dari dua jenis transmisi

Table 5.12 nilai dari *End-to-End Delay* dengan transmisi *TwoRayGround* menunjukan nilai yang fluktuatif dan cenderung menurun. Nilai yang fluktuatif tadi disebabkan karena pergerakan dari node yang acak. Bisa dilihat bahwa nilai rata rata pada kecepatan Maksimal 5m/s adalah 460.6544 detik sementara pada kecepatan maksimal 10m/s memiliki nilai 152.2749 detik dan pada kecepatan maksimal 15m/s bernilai 426.1423 detik.

Table 5.14 nilai dari Packet Delivery Ratio dengan transmisi *FreeSpace* menunjukan nilai menurun. Dapat dilihat nilai PDR pada kecepatan maksimal 5m/s adalah 48.1903 detik sementara pada kecepatan maksimal 10m/s adalah 62.7519 detik dan pada kecepatan maksimal 15m/s adalah 170.8228 detik.

Gambar 5.3 menunjukan pengujian model transmisi *TwoRayGround* untuk nilai rata rata dari *End to End Delay* bersifat fluktuatif pada jumlah waktu yang dibutuhkan untuk mengirimkan paket data yang dikirimkan. Nilai yang fluktuatif tadi disebabkan karena pergerakan dari node yang acak. Sedangkan hasil dari pengujian dengan transmisi *FreeSpace* mengalami kenaikan pada jumlah waktunya. Tiap *node* yang telah dibuat secara acak dapat menghasilkan situasi berbagai macam dan sangat mungkin terjadi penundaan pengiriman paket

karena rute terputus. Selain itu media transmisi *FreeSpace* mengalami waktu tunda dari pengiriman paket karena selama proses pengiriman bisa dipengaruhi oleh lingkungan.

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan Tugas Akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap Tugas Akhir ini di masa yang akan datang.

6.1. Kesimpulan

Kesimpulan yang dapat diambil dari hasil uji coba dan evaluasi dalam tugas akhir ini adalah sebagai berikut:

1. Skenario Manet yang dihasilkan oleh *node-movement*(*mobility generation*) dan dijalankan menggunakan model transmisi *TwoRayGround* dengan penambahan kecepatan maksimal perpindahan *node* memiliki performa sebagai berikut.
   - Performa *Packet Delivery Ratio* yang dihasilkan bersifat fluktuatif cenderung turun mulai dari 93.7138% pada kecepatan maksimal 5m/s menjadi 95.0598% pada kecepatan maksimal 10m/s dan menjadi 90.3745% pada kecepatan maksimal 15m/s.
   - Performa *Routing Overhead* yang dihasilkan memiliki nilai semakin turun mulai dari 213.390 paket pada kecepatan maksimal 5m/s menjadi 176.438 paket pada kecepatan maksimal 10m/s dan berubah menjadi 144.577 paket pada kecepatan maksimal 15m/s.
   - Performa *End to End Delay* yang dihasilkan memiliki nilai fluktuati cenderung turun mulai dari 460.6544 detik pada kecepatan maksimal 5m/s menjadi 154.2749 detik pada kecepatan maksimal 10m/s dan berubah menjadi 426.1423 detik pada kecepatan maksimal 15m/s.
2. Skenario Manet yang dihasilkan oleh *node-movement*(*mobility generation*) dan dijalankan

menggunakan model transmisi *FreeSpace* dengan penambahan kecepatan maksimal perpindahan *node* memiliki performa sebagai berikut.

- Performa *Packet Delivery Ratio* yang dihasilkan mengalami penurunan mulai dari 98.3789% pada kecepatan maksimal 5m/s menjadi 97.3804% pada kecepatan maksimal 10m/s dan menjadi 96.3290% pada kecepatan maksimal 15m/s.
- Performa *Routing Overhead* yang dihasilkan memiliki nilai semakin turun mulai dari 218.212 paket pada kecepatan maksimal 5m/s menjadi 163.329 paket pada kecepatan maksimal 10m/s dan berubah menjadi 142.811 paket pada kecepatan maksimal 15m/s.
- Performa *End to End Delay* yang dihasilkan mengalami kenaikan mulai dari 48.1903 detik pada kecepatan maksimal 5m/s menjadi 62.7519 detik pada kecepatan maksimal 10m/s dan berubah menjadi 170.8228 detik pada kecepatan maksimal 15m/s.

3. Hasil analisis metrik PDR dan E2E menunjukan model transmisi *FreeSpace* memiliki keunggulan daripada analisis metrik PDR dan E2E model transmisi *TwoRayGround*. Sementara untuk hasil analisis metrik RO dari model transmisi *FreeSpace* memiliki perbedaan sedikit dengan model transmisi *TwoRayGround*. Hal disebabkan oleh:
    - Model transmisi *FreeSpace* tidak menggunakan media apapun dalam transmisinya sehingga menghasilkan PDR dan waktu dari E2E delay lebih rendah.
    - Model transmisi *TwoRayGround* menggunakan media pantulan dari *ground* dalam proses transmisinya sehingga menghasilkan PDR dan waktu E2E delay yang lebih tinggi.

4. Hal hal yang dapat mempengaruhi nilai PDR,E2E dan RO yang dihasilkan dari model transmisi *TwoRayGround* dan *FreeSpace* adalah:
    - Posisi awal *node* yang dibuat secara acak.
    - Pergerakan *node* yang dibuat secara acak.
    - Lingkungan jaringan yang digunakan
    - Kecepatan maksimal pada routing
    - Transmisi yang digunakan
    - Ketinggian antena

6.2. Saran

Dalam pengerjaan Tugas Akhir ini ada saran-saran yang perlu ditambahkan untuk perbaikan serta pengembangan sistem yang telah dikerjakan sebagai berikut:

1. Dapat dilakukan percobaan dengan penambahan atau pengurangan jumlah *node* dan menggunakan bermacam-macam kombinas percobaan untuk skenario *node-movement*(*mobility generation*) seperti kecapatan maksimal, pergerakan node, jumlah node, serta parameter lainnya.
2. Dapat dilakukan percobaan dengan menggunakan analisis matrik yang lain.
3. Dapat dilakukan modifikasi pada parameter parameter lain yang berhubungan dengan simulasi AODV di NS2.
4. Dapat melakukan jumlah percobaan yang lebih banyak untuk mendapatkan hasil uji coba yang lebih baik.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] A. G. L. N. P. A. W. R. H. W. P. d. E. N. F. Dwi S S, *Analisis Performansi Protokol Routing AODV dan DSR pada Manet.*.

[2] "are'ga," Routing Protocol (AODV, DSR, DSDV), [Online]. Available: https://menulicious.student.telkomuniversity.ac.id/routing-protocol-aodv-dsr-dsdv/. [Accessed 21 7 2019].

[3] "3. Mobile Ad Hoc Network (MANET)," [Online]. Available: https://affandezone.wordpress.com/2011/10/30/3-routing-pada-manet/. [Accessed 21 7 2019].

[4] H. Agus, "Jaringan Mobile Ad hoc Network (MANET)," [Online]. Available: https://hariagus.wordpress.com/2009/08/10/jaringan-mobile-ad-hoc-network-manet/. [Accessed 21 7 2019].

[5] "Routing pada MANET," [Online]. Available: https://affandezone.wordpress.com/2011/10/09/routing-pada-manet/. [Accessed 21 7 2019].

[6] "Ad Hoc On-Demand Distance Vector (AODV)," [Online]. Available: https://www.techopedia.com/definition/2922/ad-hoc-on-demand-distance-vector-aodv. [Accessed 22 7 2019].

[7] "Two-ray ground-reflection model," [Online]. Available: https://en.wikipedia.org/wiki/Two-ray_ground-reflection_model. [Accessed 22 7 2019].

[8] "Two-Ray ground reflection model," Wikipedia, [Accessed

22 7 2019]

[9] Model propagasi pada FreeSpace, " Model propagasi pada FreeSpace," Model propagasi pada FreeSpace, [Online]. Available: https://studylibid.com/doc/117783/model-propagasi-pada-free-space-model-propagasi-ruang-bebas

[10] N. M. Mittal dan S. Choudhary, "Comperaative Study of Simulators for Vehicular Ad-hoc Network (VANETs)," vol. 4, no. 4, hlm. 10, 2014.

LAMPIRAN

```
1.  $node_(12) set X_ 494.201434867899
2.  $node_(12) set Y_ 707.271543869861
3.  $node_(12) set Z_ 0.000000000000
4.  $node_(13) set X_ 374.280877469672
5.  $node_(13) set Y_ 697.754289473615
6.  $node_(13) set Z_ 0.000000000000
7.  $node_(14) set X_ 327.819973273798
8.  $node_(14) set Y_ 214.858192366200
9.  $node_(14) set Z_ 0.000000000000
10. $node_(15) set X_ 468.869029526775
11. $node_(15) set Y_ 634.212993065897
12. $node_(15) set Z_ 0.000000000000
13. $node_(16) set X_ 613.147078346133
14. $node_(16) set Y_ 474.341496886460
15. $node_(16) set Z_ 0.000000000000
16. $node_(17) set X_ 736.633350797868
17. $node_(17) set Y_ 208.417268505039
18. $node_(17) set Z_ 0.000000000000
19. $node_(18) set X_ 632.889131559278
20. $node_(18) set Y_ 678.318290293730
21. $node_(18) set Z_ 0.000000000000
22. $node_(19) set X_ 63.920350115614
23. $node_(19) set Y_ 363.024289419188
24. $node_(19) set Z_ 0.000000000000
25. $node_(20) set X_ 497.788853091207
26. $node_(20) set Y_ 622.766809709253
27. $node_(20) set Z_ 0.000000000000
28. $node_(21) set X_ 195.394585431105
29. $node_(21) set Y_ 574.751056625559
30. $node_(21) set Z_ 0.000000000000
31. $node_(22) set X_ 172.778730638785
32. $node_(22) set Y_ 628.651370750620
33. $node_(22) set Z_ 0.000000000000
34. $node_(23) set X_ 401.304790068277
35. $node_(23) set Y_ 596.845248561309
36. $node_(23) set Z_ 0.000000000000
37. $node_(24) set X_ 629.837437301452
38. $node_(24) set Y_ 575.448231700163
39. $node_(24) set Z_ 0.000000000000
40. $node_(25) set X_ 598.467498943498
```

```
$node_(25) set Y_ 797.492506233612
$node_(25) set Z_ 0.000000000000
$node_(26) set X_ 34.103795675000
$node_(26) set Y_ 410.825221392244
```

Kode Sumber 7.1 Posisi node dari potongan Skenario

```
 $god_ set-dist 0 1 2
 $god_ set-dist 0 2 3
 $god_ set-dist 0 3 5
 $god_ set-dist 0 4 4
 $god_ set-dist 0 5 3
 $god_ set-dist 0 6 5
 $god_ set-dist 0 7 4
 $god_ set-dist 0 8 2
 $god_ set-dist 0 9 4
$god_ set-dist 0 10 5
$god_ set-dist 0 11 5
$god_ set-dist 0 12 5
$god_ set-dist 0 13 4
$god_ set-dist 0 14 2
$god_ set-dist 0 15 4
$god_ set-dist 0 16 4
$god_ set-dist 0 17 4
$god_ set-dist 0 18 5
$god_ set-dist 0 19 2
$god_ set-dist 0 20 4
$god_ set-dist 0 21 3
$god_ set-dist 0 22 3
$god_ set-dist 0 23 4
$god_ set-dist 0 24 4
$god_ set-dist 0 25 5
$god_ set-dist 0 26 2
$god_ set-dist 0 27 3
$god_ set-dist 0 28 5
$god_ set-dist 0 29 2
$god_ set-dist 0 30 3
$god_ set-dist 0 31 3
$god_ set-dist 0 32 2
$god_ set-dist 0 33 2
$god_ set-dist 0 34 5
```

```
$god_ set-dist 0 35 2
$god_ set-dist 0 36 1
$god_ set-dist 0 37 3
$god_ set-dist 0 38 4
$god_ set-dist 0 39 1
$god_ set-dist 0 40 3
$god_ set-dist 0 41 6
$god_ set-dist 0 42 4
$god_ set-dist 0 43 3
$god_ set-dist 0 44 3
$god_ set-dist 0 45 3
$god_ set-dist 0 46 2
$god_ set-dist 0 47 1
$god_ set-dist 0 48 2
$god_ set-dist 0 49 5
$god_ set-dist 1 2 1
$god_ set-dist 1 3 3
$god_ set-dist 1 4 2
$god_ set-dist 1 5 2
$god_ set-dist 1 6 3
$god_ set-dist 1 7 3
$god_ set-dist 1 8 1
$god_ set-dist 1 9 3
$god_ set-dist 1 10 4
$god_ set-dist 1 11 3
$god_ set-dist 1 12 3
$god_ set-dist 1 13 2
$god_ set-dist 1 14 2
$god_ set-dist 1 15 2
$god_ set-dist 1 16 2
$god_ set-dist 1 17 3
$god_ set-dist 1 18 3
$god_ set-dist 1 19 1
$god_ set-dist 1 20 3
$god_ set-dist 1 21 1
$god_ set-dist 1 22 1
$god_ set-dist 1 23 2
$god_ set-dist 1 24 3
$god_ set-dist 1 25 3
$god_ set-dist 1 26 1
$god_ set-dist 1 27 2
$god_ set-dist 1 28 3
$god_ set-dist 1 29 2
```

```
$god_ set-dist 1 30 2
$god_ set-dist 1 31 1
$god_ set-dist 1 32 3
$god_ set-dist 1 33 1
$god_ set-dist 1 34 4
$god_ set-dist 1 35 1
$god_ set-dist 1 36 1
$god_ set-dist 1 37 1
$god_ set-dist 1 38 3
$god_ set-dist 1 39 2
$god_ set-dist 1 40 1
$god_ set-dist 1 41 4
$god_ set-dist 1 42 2
$god_ set-dist 1 43 1
$god_ set-dist 1 44 1
$god_ set-dist 1 45 1
$god_ set-dist 1 46 1
$god_ set-dist 1 47 2
$god_ set-dist 1 48 1
$god_ set-dist 1 49 3
```

Kode Sumber 7.2 Pembuatan GOD dari potongan skenario

```
#
# nodes: 2, max conn: 1, send rate: 4.0, seed: 1.0

#
#
# 1 connecting to 2 at time 2.5568388786897245
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 4.0
$cbr_(0) set random_ 1
```

```
15. $cbr_(0) set maxpkts_ 10000
16. $cbr_(0) attach-agent $udp_(0)
17. $ns_ connect $udp_(0) $null_(0)
18. $ns_ at 2.5568388786897245 "$cbr_(0) start"
19. #
20. #Total sources/connections: 1/1
21. #
```

Kode Sumber 7.3 Koneksi yang digunakan pada 'cbr.txt'

```
1.  # A 10-node example for ad-
    hoc simulation with DSR
2.
3.  # Define options
4.  set val(chan)           Channel/WirelessChannel
    ;# channel type
5.  set val(prop)           Propagation/TwoRayGround
    ;# radio-propagation model
6.  set val(netif)          Phy/WirelessPhy
    ;# network interface type
7.  set val(mac)            Mac/802_11
    ;# MAC type
8.  set val(ifq)            CMUPriQueue    ;# interface
     queue type
9.  set val(ll)             LL
    ;# link layer type
10. set val(ant)            Antenna/OmniAntenna
    ;# antenna model
11. set val(ifqlen)         50
    ;# max packet in ifq
12. set val(nn)             50
    ;# number of mobilenodes
13. set val(rp)             DSR                        ;
    # routing protocol
14. set opt(x)              800                ;# X dim
    ension of topography
15. set opt(y)              800                ;# Y dim
    ension of topography
```

```
16. set val(stop)       100             ;# time of simul
    ation end
17. set val(seed)       0                ;
18. set val(cp)                 "cbr.txt";
19. set val(sc)                 "scenario.txt";
20.
21. Phy/WirelessPhy set RXThresh_ 1.42681e-08;
22.
23. set ns_       [new Simulator]
24. set tracefd       [open trace.tr w]
25. #set windowVsTime2 [open win.tr w]
26. set namtrace      [open simwrls.nam w]
27.
28.
29. $ns_ trace-all $tracefd
30. #$ns use-newtrace
31. $ns_ namtrace-all-
    wireless $namtrace $opt(x) $opt(y)
32.
33. # set up topography object
34. set topo       [new Topography]
35.
36. $topo load_flatgrid $opt(x) $opt(y)
37.
38. set god_ [create-god $val(nn)]
39.
40. #
41. #  Create nn mobilenodes [$val(nn)] and attach them
     to the channel.
42. #
43.
44. # configure the nodes
45.         $ns_ node-config -adhocRouting $val(rp) \
46.              -llType $val(ll) \
47.              -macType $val(mac) \
48.              -ifqType $val(ifq) \
49.              -ifqLen $val(ifqlen) \
50.              -antType $val(ant) \
51.              -propType $val(prop) \
52.              -phyType $val(netif) \
53.              -channelType $val(chan) \
54.              -topoInstance $topo \
55.              -agentTrace ON \
```

```
56.                 -routerTrace ON \
57.                 -macTrace OFF \
58.                 -movementTrace ON
59.
60.     for {set i 0} {$i < $val(nn) } { incr i } {
61.         set node_($i) [$ns_ node]
62.         $node_($i) random-motion 0;
63.     }
64.
65. # Define node movement model
66. puts "Loading Conneciton Pattern ...."
67. source $val(cp)
68.
69. #Define traffice mode
70. puts "Loading scenarion file...."
71. source $val(sc)
72.
73. #define node initial position in nam
74.
75.     for {set i 0} {$i < $val(nn) } { incr i } {
76.         $ns_ initial_node_pos $node_($i) 20
77.     }
78. #tell nodes when the simulation ends
79.     for {set i 0} {$i < $val(nn) } { incr i } {
80.         $ns_ at $val(stop).0 "$node_($i) reset";
81.     }
82. $ns_ at $val(stop).0002 "puts \"NS EXITING....\";
83. $ns_ halt"
84.
85. puts $tracefd "M 0.0 nn $val(nn) x $opt(x) y $opt(y
    ) rp $val(rp)"
86. puts $tracefd "M 0.0 sc $val(sc) cp $val(cp) seed $
    val(seed)"
87. puts $tracefd "M 0.0 prop $val(prop) ant $val(ant)"

88.
89. puts "Starting Simulation...."
90. $ns_ run
```

Kode Sumber 7.4 file .tcl untuk simulasi DSR

```
1.  BEGIN{
2.      recvs = 0;
3.      besaran_paket = 0;
4.  }
5.  {
6.      if (($1 == "s" || $1 == "r") && $4 == "RTR"){
7.          recvs++;
8.      }
9.      if (($1 == "s" || $1 == "r") && $4 == "RTR")
10.     {
11.         besaran_paket = besaran_paket + $8;
12.     }
13. }
14. END{
15.     printf("Routing Overhead : %.3f\n", besaran_pak
    et/recvs);
16. }
```

Kode Sumber 7.5 Implementasi perhitungan Routing Overhead

```
1.  BEGIN {
2.      sendLine = 0;
3.      recvLine = 0;
4.      forwardLine = 0;
5.  }
6.
7.  $0~/^s.*AGT/{
8.      sendLine++
9.  }
10.
11. $0~/^r.*AGT/{
12.     recvLine++
13. }
14.
15. $0~/^f.*RTR/{
16.     forwardLine++
17. }
18. END{
19.     printf"Ratio:%.4f\n",(recvLine/sendLine)*100;
20. }
```

Kode Sumber 7.6 implementasi perhitungan PDR

```
BEGIN {
seqno = -1;
# droppedPackets = 0;
# receivedPackets = 0;
count = 0;
}
{
if($4 == "AGT" && $1 == "s" && seqno < $6) {
seqno = $6;
}
if($4 == "AGT" && $1 == "s") {
start_time[$6] = $2;
} else if(($7 == "cbr") && ($1 == "r")) {
end_time[$6] = $2;
} else if($1 == "D" && $7 == "cbr") {
end_time[$6] = -1;
}
}
END {
for(i=0; i<=seqno; i++) {
if(end_time[i] > 0) {
delay[i] = end_time[i] - start_time[i];
count++;
}
else
{
delay[i] = -1;
}
}
for(i=0; i<=seqno; i++) {
if(delay[i] > 0) {
n_to_n_delay = n_to_n_delay + delay[i];
}
}
n_to_n_delay = n_to_n_delay/count;
printf("end to end delay : ")
print n_to_n_delay * 1000 ;
}
```

Kode Sumber 7.7 implementasi perhitungan E2E

Installasi NS-2

Instalasi NS-2 dilakukan pada sistem operasi Ubuntu 14.04. Yang diperlukan untuk menggunakan NS-2 adalah melakukan instalasi dependensi dan *source code* ns-2.35. Sebelum melakukan instalasi NS-2 diperlukan beberapa dependensi agar NS-2 dapat dijalankan. Cara instalasi dependensi tersebut ditunjukkan pada perintah di bawah

```
1. sudo apt-get install build-
   essential autoconf automake
```

Kode Sumber 7.8 installasi NS-2

Setelah semua dependensi terpasang selanjutnya adalah mengunduh *source code* ns-2.35 dengan cara yang ditunjukkan pada perintah di bawah

```
1. wget http://jaist.dl.sourceforge.net/project/nsnam/
   allinone/nsallinone-2.35/ns-allinone-2.35.tar.gz
```

Kode Sumber 7.9 mengunduh kode sumber ns-2.35

Lalu mengekstrak file dengan kode dibawah ini

```
1. tar -xvf ns-allinone-2.35.tar.gz
```

Kode Sumber 7.10 Ekstrak file NS-2

BIODATA PENULIS

Yosua Nove Pratama, lahir pada tanggal 1 November 1995 di Surabaya. Saat ini sedang menempuh perguruan tinggi di Institut Teknologi Sepuluh November jurusan Teknik Informatika Fakultas Teknologi Informasi pada tahun 2013. Terlibat aktif pada organisasi mahasiswa tingkat jurusan antara lain Departemen Media Informasi Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS sebagai staff tahun 2014 – 2015, sebagai staff Divisi Persekutuan Persekutuan Mahasiswa Kristen ITS pada tahun 2015-2016, sebagai staff pada Schematics 2014, dan staff 3D Schematics 2015 .

Apabila ingin bertanya lebih lanjut, silahkan kirim *email* ke yosuanovepratama@gmail.com.