TUGAS AKHIR – KI141502

Studi Kinerja Protokol MAODV dengan Propagasi Model *Two Ray Ground* dan *Free space* Pada Jaringan MANET Menggunakan Network Simulator 2 (NS-2)

Wiryo Febdila
NRP 5110100705

Dosen Pembimbing
Dr. Eng. Radityo Anggoro, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya
2016

(Halaman ini sengaja dikosongkan)

FINAL PROJECT – KI141502

The study of MAODV Performance with *Two Ray Ground* Model and *Free space* Propagations on a MANET Network Using Network Simulator 2 (NS-2)

Wiryo Febdila
NRP 5110100705

Advisor
Dr. Eng. Radityo Anggoro, S.Kom., M.Sc.

DEPARTEMENT OF INFORMATICS ENGINEERING
Faculty of Information Technology
Sepuluh Nopember Intitute of Technology
Surabaya
2016

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

Studi Kinerja Protokol MAODV dengan Propagasi Model *Two Ray Ground* dan *Free space* Pada Jaringan MANET Menggunakan Network Simulator 2 (NS-2)

TUGAS AKHIR

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

Oleh :
WIRYO FEBDILA
NRP : 5110 100 705

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Dr.Eng. Radityo Anggoro, S.Kom., M.Sc. .....
NIP: 19841016 200812 1 002 (pembimbing)

SURABAYA
JULI 2016

v

(Halaman ini sengaja dikosongkan)

Studi Kinerja Protokol MAODV dengan Propagasi Model *Two Ray Ground* dan *Free space* Pada Jaringan MANET Menggunakan Network Simulator 2 (NS-2)

Nama Mahasiswa	**: Wiryo Febdila**
NRP	**: 5110100705**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing	**: Dr. Eng. Radityo Anggoro, S.Kom., M.Sc.**

ABSTRAK

Mobile Ad hoc Network (MANET) yaitu sebuah jaringan wireless dari mobile-mobile node yang tidak memiliki infrasrutkur tetap. Pada MANET ini adanya keterbatasan daya transmisi untuk mencapai node diluar jangkauan transmisi, dan disinilah terjadi peranan sebuah protokol routing yang dimana bisa membantu sebuah node mencapai node tujuan yang benar. Pada tugas akhir ini akan dibahas salah satu protokol routing multicast yaitu Multicast Ad hoc On Demand Distance Vector Routing (MAODV).

Berdasarkan permasalahan tersebut akan dilakukan pengujian performa dari MAODV dengan menggunakan dua propagasi yang berbeda yaitu Two Ray Ground dan Free Space serta diberikan kecepatan berbeda dan juga dengan perbedaan luas simulasi dengan menggunakan aplikasi Network Simulator 2 (NS2)

Dari hasil uji coba yang telah dilakukan, bahwa penambahan kecepatan pada masing-masing node untuk Two Ray Ground mendapatkan PDR, Delay dan RO yang lebih buruk dibandingkan dengan Free Space dan begitu juga dengan penambahan luas area simulasi menghasilkan nilai untuk Two Ray Ground lebih butuk dibandingkan Free Space.

Kata kunci: MANET, MAODV, TWORAYGROUND, FREESPACE, NS2

(Halaman ini sengaja dikosongkan)

The study of MAODV Performance with *Two Ray Ground* Model and *Free Space* Propagations on a MANET Network Using Network Simulator 2 (NS-2)

Student's Name	**: Wiryo Febdila**
Student's ID	**: 5110100705**
Department	**: Teknik Informatika FTIF-ITS**
Advisor	**: Dr. Eng. Radityo Anggoro, S.Kom., M.Sc.**

ABSTRACT

Mobile Ad Hoc Network (MANET) is a wireless network for several mobile nodes without any static infrastructures. MANET's limited transmitting power to reach the nodes locating outside of the transmitting range has granted an important role to a routing protocol to help reaching the correct destination with this particular characteristic. Therefore, this research evaluates the performance of one of multicast routing protocols named Multicast Ad Hoc On Demand Distance Vector Routing (MAODV).

To deal with this problem, a performance evaluation for MAODV is to be performed by using two different transmitting propagations i.e. Two Ray Ground and Free Space along with different speed and simulation area using an application named Network Simulator 2 (NS2).

The result shows that the speed increase in each node for Two Ray Ground propagation leads to worse PDR, Delay and RO values compared to those of Free Space. This is also true in the case of increasing the simulation area, which results in lower values for Two Ray Ground than those of Free Space.

Keywords: MANET, MAODV, TWORAYGROUND, FREESPACE, NS2

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan judul ***"Studi Kinerja Protokol MAODV dengan Propagasi Model Two Ray Ground dan Free space Pada Jaringan MANET Menggunakan Network Simulator 2 (NS-2)"***.

Penulis menyadari bahwa penulis tidak mungkin dapat menyelesaikan tugas akhir ini tanpa bantuan dan dukungan dari banyak pihak, baik secara langsung maupun tidak. Untuk itu, penulis ingin mengucapkan terima kasih dan penghormatan yang sebesar-besarnya kepada:

1. Kedua orang tua kandung penulis, Bapak Windra dan Ibu Almarhumah Dismareli yg telah mendidik penulis dari kecil sampai sekarang.
2. Ibu Septiawati yang telah menjadi pengganti sosok ibu kandung untuk penulis.
3. Adik penulis, Hazim Kamil yang telah mendukung penulis selama studi.
4. Bapak Dr.Eng. Radityo Anggoro, S.Kom, M.Sc. selaku dosen pembimbing, yang telah memberikan bimbingan, dukungan, dan masukan sehingga Tugas Akhir ini bisa selesai dengan tepat waktu.
5. Bapak Dr.Eng. Darlis Herumurti, S.Kom., M.Kom. selaku Ketua Jurusan Teknik Informatika ITS..
6. Ibu Dr. Nanik Suciati, S.Kom, M.Kom selaku dosen wali penulis, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya kepada penulis.
7. Teman-teman seperjuangan pengerjaan Tugas Akhir Erick,CKW,Bima dan Lala
8. Sahabat penulis di Gebang Kidul pebe,nawa,fazar,masbar,edo,Mansur.

9. Teman dari PB Gebang Kidul, yang menemani penulis olah raga adi,krisna,grezio.
10. IMAMI Surabaya 2010 Kairul, Yudha, Cuba, Tomi, Iid.
11. Seluruh teman Teknik Informatika ITS angkatan 2010, yang menjadi teman seperjuangan penulis.
12. Seluruh alumni Ar-risalah disurabaya ridho,kibe,ujang,zikrul,irsyad,raisa dan icha.
13. Seluruh teman CSSMORA ITS angkatan 2010, yang menjadi teman seperjuangan penulis.
14. Juga tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu terselesaikannya Tugas Akhir ini.

Kesempurnaan tentu sangat jauh tercapai pada Tugas Akhir ini, maka penulis mengharapkan saran dan kritik yang membangun dari pembaca.

Surabaya, Juli 2016

WIRYO FEBDILA

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

Bab ini membahas garis besar penyusunan Tugas Akhir yang meliputi latar belakang, tujuan pembuatan, rumusan dan batasan permasalahan, metodologi penyusunan Tugas Akhir, dan sistematika penulisan.

1.1. Latar Belakang

Mobile Ad hoc Network (MANET) yaitu sebuah jaringan wireless dari *mobile-mobile node* yang tidak memiliki router tetap. *Node-node* dalam jaringan ini berfungsi juga sebagai router yang bertanggung jawab untuk mencari dan menangani rute ke setiap *node* didalam jaringan, setiap *node* dapat berpindah pindah tempat mengakibatkan topologi dari jaringan ini selalu sering berubah.

Untuk aplikasi jaringan tertentu yang mengharuskan sebuah *node* mengirimkan paket data ke sebuah *group node* yang diindentifika dengan satu alamat unik adalah solusi untuk masalah diatas. Multicating sangat bermanfaat dalam hal mengurangi konsumsi bandwidth, mengurangi proses ditiap *node* dan router serta mengurangi *delay* pengiriman, hal ini akan sangat berguna dalam MANET yang memiliki keterbatasan bandwidth dan keterbatasan dalam jarak transmisi sehingga untuk menuju *node* yang jauh harus melalui beberapa *node* perantara.hal ini akan meningkatkan proses di tiap *node*.

Berdasarkan bagaimana rute dibangun antar anggota *group multicast* maka setidaknya ada dua pendekatan yang digunakan oleh prokol *multicast* di MANET yaitu *tree-based* dan *meshed-based*. Pada tugas ahkir ini kita fokus menguji sebuah protokol pada pendekatan tree-based dan tepatnya kita membahas kinerja protokol MAODV *(Multicast Ad hoc On Demand Distance Vektor)* dengan menambahkan dua propagasi sebagai perbandingan yaitu yaitu propagasi *Two Ray Ground* dan *Free space.*

Untuk dapat membandingkan penggunaan kedua propagasi tersebut, simulasi yang akan digunakan sebagai tolak ukur dalam tugas akhir ini adalah *Packet delivery ratio* (PDR),*End-to-end*

delay (E2D) *dan Routing overhead* (RO) maka kinerja dari MAODV tersebut dapat dilihat pada ketiga kondisi tersebut. Hasil dari simulasi dengan topologi yang digunakan memperlihatkan bahwa penggunaan propagasi *Two Ray Ground* dan *Free space* dan untuk mendapatkan hasil yang terbaik dari kedua propagasi tersebut.

1.2. Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana kinerja MAODV dengan propagasi *Two Ray Ground* dan *Free space* pada MANET?
2. Bagaimana perbandingan kinerja MAODV dengan menambahkan *Two Ray Ground* dan *Free space* pada MANET?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, diantaranya sebagai berikut:

1. Routing Protocol yang akan dibandingkan dalam tugas akhir ini adalah MAODV dengan *Two Ray Ground* dan *Free space* pada MANET.
2. Software yang akan digunakan untuk penyelesaian tugas akhir ini adalah *Network Simulator* 2.
3. Metrik yang digunakan untuk mengevaluasi masing-masing prokotol adalah sebagai berukut:
    - *Packet delivery ratio* (PDR)
    - *End-to-end delay* (E2D)
    - *Routing overhead* (RO)
3. Parameter yang digunakan adalah
    - Kecepatan Penpindahan *node*
    - Jumlah Pengirim
    - Jumlah penerima

`

1.4. Tujuan dan Manfaat

Tujuan dari Tugas Akhir ini adalah

1.Memberikan hasil perbandingan kinerja MAODV dengan propagasi *Two Ray Ground* dan *Free Space* pada MANET dengan menggunakan aplikasi *Network Simulator* 2.

1.5. Metodologi

Adapun langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.

   Pada tahap ini, proposal ditulis untuk mengajukan ide atas pengerjaan Tugas Akhir. Proposal ini juga mengandung proyeksi dari ide Tugas Akhir yang diajukan.

2. Studi literatur

   Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan aplikasi yaitu mengenai Networ Simulator 2, *Mobile Ad hoc Network* (MANET), dan routing protocol MAODV dengan penambahan propagasi *TwoRayGround* dan *FreeSpace.*

3. Implementasi protokol *routing*

   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini merupakan tahap yang paling penting dimana bentuk awal aplikasi yang akan diimplementasikan didefinisikan. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain suatu sistem dan desain proses-proses yang ada.

4. Uji coba dan evaluasi

   Pada tahapan ini dilakukan uji coba terhadap aplikasi yang telah dibuat. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perencanaan. Tahap ini dimaksudkan juga untuk mengevaluasi jalannya sistem,

mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

5. Penyusunan buku tugas akhir.

   Pada tahapan ini disusun buku sebagai dokumentasi dari pelaksanaan tugas akhir.

1.6. Sistematika Penulisan

Buku tugas akhir ini disusun dengan sistematika penulisan sebagai berikut:

BAB I. PENDAHULUAN

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

BAB II. TINJAUAN PUSTAKA

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang untuk mendukung pembuatan tugas akhir ini.

BAB III. PERANCANGAN

Bab ini berisi perancangan metode yang nantinya akan diimplementasikan dan dilakukan uji coba.

BAB IV. IMPLEMENTASI

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa implementasi skenario mobilitas *vehicular* yang dibuat menggunakan kedua *mobility generator*, konfigurasi sistem dan skrip analisis yang digunakan untuk menguji performa protokol *routing*.

BAB V. UJI COBA DAN EVALUASI

Bab ini menjelaskan tahap pengujian sistem dan pengujian performa dalam skenario mobilitas *vehicular* yang dibuat oleh kedua *mobility generator*.

`

BAB VI. PENUTUP

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan terhadap rumusan masalah yang ada dan saran untuk pengembangan lebih lanjut.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan tentang teori-teori yang berkaitan dengan pengimplementasian perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran atau definisi secara umum terhadap alat, protokol *routing* serta definisi yang digunakan dalam pembuatan Tugas Akhir.

2.1. *Mobile Ad hoc Network* (MANET)

Mobile Ad hoc Network (MANET) yaitu sebuah jaringan wireless dari*mobile-mobile node* yang tidak memiliki infrastruktur tetap. Karena keterbatasandaya transmisi untuk mencapai *node* diluar jangkauan transmisi, maka komunikasiantara *node* tersebut akan melewati satu atau beberapa *node* lainnya (berfungsisebagai router) sehingga MANET biasa disebut multi-*hop* network. Karena paketharus melewati beberapa *node* (multi *hop*) untuk mencapai tujuan makadiperlukan sebuah protokolrouting.Fungsi utama protokol routing adalah untukmencari dan memilih jalur yang harus dilalui sebuah paket agar sampai pada *node* tujuan yang benar.

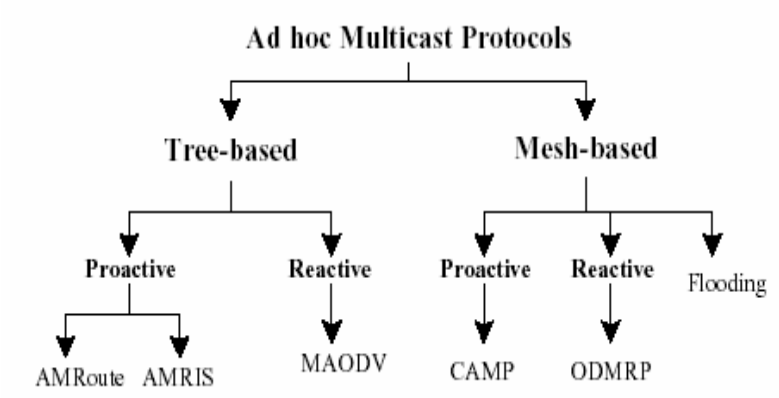
Berikut ini beberapa karakteristik dari MANET :

1. Topologi yang dinamis
   Node-node di MANET bebas bergerak kemana saja dan kapan saja, hal ini mengakibatkan topologi jaringan berubah secara acak
2. Batasan Bandwidth
   Kapasitas bandwidth yang dimiliki tiap jalur berbeda-beda.
3. Batasan Daya
   Untuk bekerja, *mobile node* menggunakan baterai sebagai sember dayanya. Daya baterai yang terbatas mengharuskan setiap operasi harus berjalan secara efisien.

.

2.2. Protokol Routing *Multicast* di MANET

*Multicast*ing memainkan peran yang sangat penting dalam proseskomunikasi di jaringan *Ad hoc*. Dengan teknik *multicast* mengirimkan paket datayang sama ke sejumlah host dapat dilakukan dengan menset alamat *node* tujuan pada paket data dengan sebuah ID unik, biasa disebut *multicast* identifier yangmenidentifikasiakan setiap *group multicast*. Dengan demikian akan mengurangikonsumsi bandwidth dan daya *node* yang terbatas. Karakteristik MANET yang memiliki topologi jaringan dinamis dimana anggota *group multicast* tersebar secaraacak dan sering berpindah-pindah tempat. Hal ini mengakibatkan prosespengiriman paket dan pemeliharaan *group multicast* (pemeliharaan rute dan keanggotaan *group*) menjadi lebih kompleks. Telah banyak protokol *multicast* untuk MANET yang diajukan, berikut ini protokol *multicast* yang menjadi dasarpengembangan protokol *multicast* lainnya.



Gambar 2.1 Prokotol *Multicast* di MANET

Berdasarkan karakteristik struktur jaringan, protokol *multicast* dapat dibedakan menjadi dua bagian yaitu tree based (berstruktur pohon) dan *mesh*.

Protokol routing *multicast* yang berbasis pohon akan membangun struktur jaringan berbentuk pohon untuk setiap

group multicast. Paket data *multicast* akan dikirmkan ke anggota *group multicast* melalui rute pohon ini. Dalam satu pohon *multicast* terdapat satu *node* yang menjadi pusat atau *group* leader yang bertanggung jawab untuk memelihara struktur pohon *multicast*. based (berstruktur *mesh*).

Protokol routing *multicast* yang berbasis *mesh* menyediakan rute redudancy, karena akan terdapat lebih dari satu rute antara pasangan anggota *group multicast*

2.3. *Multicast Ad hoc On Demand Distance Vector routing* (MAODV)

Protokol MAODV memungkinkan komunikasi secara *multicast*. Rute *multicast* dibangun secara on *demand* (proaktif). Rute yang dibangun antara *node* menggunakan struktur pohon, pohon ini terdiri dari *node* anggota *group multicast* dan *node* yang terhubung dengan anggota *group multicast*. Hal ini memungkinkan *node* yang berada lebih dari satu *hop* dari anggota *group multicast* dapat bergabung.

2.3.1. Permintaan Rute *(Route Request)*

Sebuah *node* akan membuat paket RREQ (Route Request) ketika *node* tersebut ingin bergabung ke dalam sebuah *group multicast*, atau ketika mempunyai data untuk dikirim ke *group multicast*, tetapi tidak memiliki informasi rute ke *group multicast* yang dimaksud. Paket RREQ bisa dikirim secara *broadcast* atau *unicast* tergantung dari informasi yang dimiliki *node* sumber *multicast*. *Node* sumber akan memeriksa tabel requestnya apakah terdapat *record multicast group leader* (*node* pertama untuk meminta rute ke sebuah *group multicast*), jika ada dan *node* sumber mempunyai rute ke *node* yang menjadi *group* leader maka paket RREQ akan dikirim secara *unicast*, sedangkan jika *node* sumber tidak mengetahui *node* yang menjadi *group* leader maka paket RREQ akan di *broadcast*.

Ketika *node* mengirim paket RREQ, *node* menentukan RREP_WAIT_TIMER yang nilainya minimal = (waktu *latency* dari satu *hop*) X (diameter jaringan) X (dua). Jika *node* tidak

menerima balasan setelah dua kali mengirimkan paket RREQ, maka *node* ini menjadi *node group* leader.

Paket RREQ dikirim secara *broadcast* ke jaringan, untuk mencegah terjadinya *broadcast* strom MAODV menggunakan teknik ring search dimana RREQ pertama dikirim dengan nilai Time To Live (TTL) yang kecil kemudian dinaikkan pada paket RREQ berikutnya untuk mencapai *node* yang lebih jauh.

2.3.2. Balasan Rute *(Route Reply)*

Ketika sebuah *node* menerima paket RREQ, maka akan dicek field *join-flag* di paket RREQ tersebut jika *join flag* di set, *node* boleh membalas jika *node* tersebut adalah anggota pohon *multicast* dan squence *num*ber yang tersimpan untuk *group multicast* ini lebih besar dari yang terdapat di paket RREQ. *Node* akan mengupdate tabel rute dan tabel *multicast*nya, kemudian akan mengirimkan paket RREP *(Route Reply)* secara *unicast* menuju *node* sumber *multicast*.

2.3.3. Aktifasi Rute *Multicast* (*Multicast Route Activation*)

Aktifasi rute *multicast* berhubungan dengan pemilihan jalur sebuah *node* menuju pohon *multicast* dan aktifasi jalur yang harus ditambahkan ke dalam pohon ketika sebuah *node* bergabung dengan *group multicast*. Ketika *node* sumber *multicast* mem *broadcast* RREQ, seringkali *node* menerima lebih dari satu balasan RREP, *node* akan menunggu balasan paket RREP dalam periode waktu tertentu untuk memilih paket RREP yang berisi rute dengan sequence *num*ber yang paling besar dan *hop count* (*next hop*) yang terpendek ke anggota *multicast* terdekat. *Node* kemudian akan mengirim secara *unicast* paket MACT *(Multicast Route Activation)* ke *next hop* yang telah dipilih (disimpan di tabel routingnya). Paket MACT mempunyai empat flag yang dapat di set yaitu *join flag, prune flag, group_leader flag* dan *update flag*. *Next hop* penerima paket MACT akan meyimpan ID *node* sumber ke dalam tabel routingnya, jika *node* ini anggota *group multicast* maka paket MACT tidak di kirimkan ke *node* lain. Tetapi jika *node* ini bukan anggota *group multicast*, maka

`

paket MACT akan diteruskan ke *next hop* berikutnya berdasarkan informasi di tabel rutingnya, proses ini terus berlanjut sampai *node* yang mengirimkan paket RREP (*node* penerima *multicast*) dapat dicapai.

2.3.4. Paket *Group Hello*

Node yang menjadi *group leader* bertanggung jawab mengelola *group* ini sampai *node* ini meninggalkan *group multicast* atau sampai dua partisi pohon dari satu *multicast group* kembali bergabung. Goup leader mengelola *group multicast* lewat GRPH *(Group Hello Mesaaege).* GRPH berisi data alamat *group multicast* dan sequence *num*ber *group multicast* (increment tiap *Group Hello*). *Node* menggunakan informasi yang ada di paket ini untuk mengupdate table requestnya.

2.3.5. Keluar dari Pohon *Multicast*

Keanggotaan dari *group multicast* bersifat dinamis. Tiap *node* bebas untuk bergabung atau keluar dari *group* kapan saja. Tetapi karena sebuah *node* juga bias menjadi *node* antara dalam sebuah pohon *multicast*, maka *node* ini tidak bias keluar dari pohon *multicast*. Sebuah *node* bias keluar dari pohon *multicast* :

a. Jika *node* ini adalah *node* daun

b. Jika *node* ini adalah *node* antara dan *node* daun yang terhubung dengannya keluar dari pohon

Untuk meninggalkan pohon *multicast* dilakukan dengan mengirim paket MACT dengan menset *Flag prune.*

2.3.6. Merperbaiki Jalur yang Terputus

Salah satu karakteristik jaringan *Ad hoc* adalah memiliki topologi yang dinamis (*node* bebas untuk berpindah-pindah sesering mungkin). Efek dari perubahan topologi jaringan ini akan mengakibatkan :

a. Terputusnya jalur

b. Terpartisnya pohon *multicast*

Jika sebuhah *node* tidak menerima paket *Group Hello* dari *node* tetangganyadalam interval periode *timeout* maka *link*

antara kedua *node* terputus. Yang dilakukan *node* untuk memperbaiki hal ini adalah dengan mengirimkan paket RREQ berikut *Multicast Group leader* Extension, Ekstension ini berisi jarak lama(sebelum jalur putus) antara *node* ke *group leader*. Hanya *node* anggota pohon *multicast* yang terdekat dengan *group leader* (ditunjukan dengan *hop count*) yang diijinkan membalas dengan paket RREP . Ketika *node* menerima paket RREP maka rute kembali aktif, jika *node* ini bukan *node* daun dan jarak ke *group leader* berbeda (berdasarkan informasi yang dibawa paket RREP) maka informasi ini harus disampaikan ke *node* dibawahnya dengan mengirimkan paket MACT dimana nya di set.

2.3.7. Pohon Terpartisi

Ketika *node* mencoba memperbaiki jalur yang terputus dengan mengirimkan paket RREQ tetapi tidak juga mendapat jawaban maka harus diasumsikan bahwa pohon telah terpartisi dan harus dilakukan pemilihan *group* leader yang baru. Jika *node* ini adalah anggota *group* pohon *multicast* maka *node* ini akan menjadi *group* leader yang baru, jika *node* ini adalah *node* daun maka *node* akan mengirim paket MACT dengan menset flag prune ke *node next hop*nya begitu seterusnya. Jika *node* ini bukan *node* daun maka paket MACT diset *group*_leader dan dikirim ke *node next hop*nya dan seterusnya sampai *node* terakhir yang menjadi anggota *group multicast* ditemukan. *Node* inilah yang akan menjadi *group leader* baru dan akan mem*broadcast* paket *group hello* GRPH. *Node* yang menerima paket GRPH akan mengupadate tabel routingnya.

2.3.8. Menggabungkan Pohon yang Terpartisi

Setelah jaringan terpartisi, maka akan terdapat dua *group* leader. Jika kedua partisi terhubung kembali, sebuah *node* akan menerima paket *group hello* untuk *group multicast* dimana informasi tentang *group* leadernya berbeda dengan yang telah dimilikinya. Jika *node* ini anggota *group multicast* dan *node* ini anggota dari partisi dengan *group* leader yang memiliki IP

`

address dengan nomor kecil, *node* ini dapat berinisiatif untuk menggabungkan kedua pohon dengan cara *node* mengirimkan paket RREQ dengan menset *refair* flag ke *node group* leadernya. *Node group* leader yang menerima paket tersebut akan memberikan hak untuk menggabungkan/membangun kembali pohon dengan mengirimkan kembali paket RREP ke *node* pengrim RREQ. Setelah paket RREP diterima *node* mengirimkan paket RREQ ke *group* leader pohon yang lain dengan menset *refair* flag dan *join flag* melalui *next hop* yang didapat paket GRPH sebelumnya. *Group leader* yang menerima paket akan menset/membuat nomor sequence yang lebih besar dari yang dimilikinya atau yang dibawa oleh paket RREQ. *Node group leader* pohon ini akan mengirim kembali paket RREP dengan menset *refair* flag dan menjadi *group* leader yang baru dari gabungan dua pohon yang sebelumnya terpartisi

2.4. Propagasi Refleksi *Two Ray Ground*

Dalam *mobile* radio *channel*, *single direct path* antara *base station* dan *mobile* terkadang hanya peralatan fisik biasa untuk propagasi dan rumus pada *free space* kurang akurat jika dalam penggunaannya berdiri sendiri. Model propagasi *Two Ray Ground* merupakan model yang berguna karena berdasar pada optik geometri dan dapat digunakan untuk *direct path* dan refleksi dari ground antara *transmitter* dan *receiver*. Model ini dirasa sangat akurat untuk memperkirakan kekuatan sinyal dalam skala luas dengan jarak beberapa kilometer untuk sistem *mobile* radio dengan menggunakan menara yang tinggi. *Power* yang diterima dengan jarak *d* diberikan oleh:

$$P_r(d) = P_t G_t G_r h_t^2 h_r^2 / (d^4 L)$$

dimana h_t dan h_r adalah tinggi dari antena *transmitter* dan *receiver*, nilai *L* diasumsikan sama dengan nilai L pada propagasi *free space*, L = 1. Untuk parameter yang lain, masih sama dengan parameter pada propagasi *free space*. Berdasarakan

rumus di atas, *power loss* lebih cepat hilang dibandingkan dengan rumus matematika pada model propagasi *free space* ketika jaraknya bertambah. Namun, Model ini tidak memberikan hasil yang baik untuk jarak yang terlalu dekat dikarenakan osilasi yang disebabkan oleh konstruktif dan destruktif yang merupakan kombinasi dari model ini.

2.5. Propagasi *Free space*

Model Propagasi *Free space* digunakan untuk memperkirakan kekuatan sinyal yang diterima ketika *transmitter* dan *receiver* tidak memiliki penghalang antara mereka (*clear*, *unobstructued*, *line-of-sight*). Contoh komunikasi yang menggunakan propagasi ini adalah komunikasi satelit dan *microwave line-of sight radio*. Dengan penggunaan propagasi ini pada skala yang besar, propagasi ini memprediksikan bahwa kekuatan *power* yang diterima menurun sejalan dengan kenaikan *power* pada jarak antara *transmitter* dan *receiver* (*t-r separation*). *Powerfree space* diterima oleh antena *receiver* yang dipisahkan dari antena *transmitter* disimbolkan dengan *d*, yang diberikan oleh rumus berikut:

$$P_r(d) = P_t G_t G_r \lambda^2 / ((4\pi)^2 d^2 L)$$

dimana, P_t adalah *power* yang ditransmisikan, $P_r(d)$ adalah *power* yang diterima dimana merupakan fungsi dari *t-r separation*, Gt adalah tegangan antena pada transmiter, d adalah jarak *t-r separation* dalam meter, *L* adalah *loss factor* sistem yang tidak berhubungan dengan propagasi *(L* 1), dan λ adalah panjang gelombang dalam meter. Nilai P_t dan P_r harus memiliki satuan yang sama sedangkan Gt dan Grtidak memiliki satuan. Nilai *loss*es *L* (*L* 1) biasanya dikarenakan adanya atenuasi pada jalur transmisi, *filter loss*es dan antena *loss*es dalam sistem komunikasi. Nilai *L* = 1 mengindikasikan tidak adanya *loss* dalam sistem perangkat keras. Model *free space* biasanya merepresentasikan daerah komunikasi yang berada disekitar /

`

lingkaran *transmitter*. Jika *receiver* berada di lingkaran tersebut, semua paket akan diterima. Selain itu, paket tersebut akan tidak mungkin diterima.

2.6. Network Simulator 2 (NS-2)

Network Simulator 2 (NS-2) merupakan alat simulasi jaringan yang bersifat *open source* yang banyak digunakan dalam mempelajari struktur dinamik dari jaringan komunikasi. Simulator ini ditargetkan pada penelitian jaringan dan memberikan dukungan yang baik untuk simulasi *routing*, protokol *multicast* dan protokol IP, seperti UDP, TCP, RTP, jaringan nirkabel dan jaringan satelit. Beberapa keuntungan menggunakan *network simulator* sebagai perangkat lunak simulasi yaitu *network simulator* dilengkapi dengan *tools* validasi, pembuatan simulasi dengan menggunakan *network simulator* jauh lebih mudah daripada menggunakan *software developer* seperti Delphi atau C++, *network simulator* bersifat *open source* di bawah GPL (Gnu Public License) dan dapat digunakan pada sistem operasi Windows dan sistem operasi Linux [11].

Pada Tugas Akhir ini digunakan NS-2 versi 2.26 sebagai aplikasi simulasi jaringan skenario jaringan MANET yang menambahkan propagasi *Two Ray Ground* dan *Free Space* menggunakan protokol MAODV. NS-2 dijalankan pada sistem operasi Linux dan proses instalasinya akan disajikan pada bagian Lampiran.

2.7. Virtual Box

Oracle VM VirtualBox adalah sebuah sebuah perangkat lunak (software) virtualisasi yang dapat digunakan untuk mengeksekusi sistem operasi tambahan di dalam sebuah sistem operasi utama, atau istilah kerennya adalah menjalankan 2 sistem operasi secara bersamaan. Misalkan seseorang mempunyai sistem operasi windows yang terinstal di komputernya, kemudian orang ini juga dapat menjalankan sistem operasi lain seperti linux dalam waktu yang bersamaan.

VirtualBox pertama kali dikembangkan oleh perusahaan Innotek GmbH yang berada di Jerman. Perusahaan ini diakusisi oleh Sun Microsystem dan menjadi milik Oracle saat pengakusisian oleh Sun Microsystem.

Pada Tugas Akhir ini Virtual Box digunakan untuk menjadi media virtual untuk menjalankan system operasi Red Hat 32-bit.

2.8. AWK

Awk adalah sebuah pemrograman seperti pada *shell* atau C yang memiliki karakteristik yaitu sebagai *tools* yang cocok filter / manipulasi. Awk adalah penggabungan dari nama lengkap sang author, yaitu : Alfred V. Aho, Peter J. Weinberger dan Brian W. Kernighan. Awk atau juga disebut Gawk (GNU awk), yaitu bahasa pemrograman umum dan *utility* standard POSIX 1003.2. Jika kecepatan merupakan hal yang penting, awk adalah bahasa yang sangat sesuai. Awk sangat baik untuk manipulasi *file* teks. Secara umum bahasa pemrograman awk dapat digunakan untuk mengelola database sederhana, membuat laporan, memvalidasi data, menghasilkan indeks dan menampilkan dokumen, membuat algoritma yang digunakan untuk mengubah bahasa komputer ke bahasa lainnya. Dengan kata lain awk menyediakan fasilitas yang dapat memudahkan untuk memecah bagian data untuk proses selanjutnya, mengurutkan data dan menampilkan komunikasi jaringan yang sederhana.

Fungsi dasar awk adalah untuk mencari *file* per baris (atau unit teks lain) yang berisi pola tertentu. Ketika suatu baris sesuai dengan pola, awk melakukan aksi yang khusus pada baris tersebut. awk tetap memproses baris *input* sedemikian hingga mencapai akhir baris *input*. Program pada awk berbeda dari program di kebanyakan bahasa lain, karena program awk bersifat "data-driven" yang mana diperlukan pendeskripsian data yang dikehendaki untuk bekerja dan kemudian apa yang akan dilakukan saat data tersebut ditemukan. Kebanyakan bahasa lainnya bersifat "procedural" maka dari itu diharuskan mendeskripsikannya secara detail setiap langkah program yang

`

harus dijalankan. Ketika bekerja dengan bahasa prosedural, biasanya sangat sulit untuk mendeskripsikan data yang hendak diproses oleh program. Oleh karena itu, program awk sering kali terasa lebih mudah untuk ditulis dan dibaca [10].

Pada Tugas Akhir ini AWK digunakan untuk membuat *script* untuk menghitung *Packet delivery ratio* (PDR) dan *End-to-end delay* (E2D) dari hasil trace NS-2.

2.9. Perl

Perl singkatan dari *Practical Extraction and Report Language* merupakan bahasa pemrograman yang banyak digunakan untuk pemrosesan data *file* ASCII pada sistem UNIX. Bahasa pmrograman ini dibuat oleh Larry Wall dengan tujuan untuk memudahkan tugas-tugas administrasi sistem UNIX. Namun Perl saat ini telah berevolusi menjadi bahasa pemrograman dan merupakan salah satu alat yang dapat digunakan untuk pemrograman web.

Pada Tugas Akhir ini perl digunakan tidak jauh beda dengan penggunaan AWK pada Tugas Akhir ini. Perl bergungsi untuk mendapatkan hasil *Routing overhead* (RO) dari hasil *Trace File* NS-2.

2.10. NS-2 *Trace File*

NS-2 *Trace File* merupakan *output* hasil *running* NS-2 yang berekstensi (.tr). *Trace File* berisi *log* tentang semua jenis pengiriman dan penerimaan paket yang terjadi selama simulasi. Di dalam *Trace File* tercatat berbagai jenis paket sesuai jenis protokol *routing* yang digunakan. Tiap baris *log* ini dianalisis untuk mendapatkan performa protokol. Contoh pengiriman data paket pada NS-2 *Trace File* dapat dilihat pada Gambar 2.2

```
s 26.000000000 _1_ AGT  --- 618 cbr 64 [0 0 0
0] ------- [1:0 0:0 32 0] [26] 0 0
```

Gambar 2.2 *Trace* Pengiriman Paket Data

Huruf "s" di kolom pertama menandakan pengiriman paket (*send*), kolom kedua berisi waktu pengiriman paket pada detik ke 26, kolom ketiga merupakan *node* tempat *event* terjadi yaitu pada *node* 1, kolom ke empat berisi AGT yang menandakan pengiriman paket data, kolom ke lima merupakan tempat terjadinya *event* spesial semisal *collision*, kolom ke 6 merupakan id unik paket, kolom ke tujuh berisi tipe paket yang dikirimkan yaitu cbr, kolom ke delapan merupakan ukuran paket dalam *byte* yaitu 64.

Untuk penerimaan data paket seperti berikut tidak jauh beda dengan pengiriman data paket, pembedanya yaitu kolom pertama dengan huruf inisial "r" yang menandakan paket diterima dan format selanjutnya sama persis dengan paket pengiriman. Contoh penerimaan data paket pada NS-2 *Trace File* dapat dilihat pada Gambar 2.3 *Trace* Penerimaan Paket DataGambar 2.3

```
r 26.011627928 _0_ AGT  --- 618 cbr 84 [13a 0
16 800] ------- [1:0 0:0 27 0] [26] 5 0
```

Gambar 2.3 *Trace* Penerimaan Paket Data

Contoh format untuk paket *routing* MAODV pada *trace file* seperti pada Gambar 2.4

```
r 30.334151244 _12_ RTR  --- 741 AODV 64 [0
ffffffff 18 800] ------- [24:255 -1:255 32 0]
[1 17 [HELLO 24 0 17]]
```

Gambar 2.4 *Trace* Pengiriman Paket Data AODV

2.11. *Generator File Node-Movement* dan *Traffic-Connection*

2.11.1. *File Mobility Generator (Node-Movement)*

Tools yang disebut 'setdest' dikembangkan oleh CMU (Carnegie Mellon University) untuk menghasilkan *random movement* dari *node* dalam jaringan nirkabel. Mendefinisikan *node movement* dengan kecepatan gerak yang spesifik menuju lokasi acak atau lokasi spesifik yang berada dalam kawasan yang

`

telah ditentukan. Ketika *node* tiba ke lokasi pergerakan, *node* tersebut bisa di atur untuk berhenti untuk sementara waktu. Setelah itu, *node* terus bergerak menuju lokasi berikutnya. Lokasi 'setdest' adalah di *directory:* '~ns/indep-utils/cmu-scen-gen/setdest/'

Pengguna harus menjalankan program 'setdest' sebelum menjalankan program simulasi. Beberapa perintah 'setdest' ditunjukkan pada Gambar 2.4 dan keterangannnya ditunjukkan pada Tabel 2.1.

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

Gambar 2.5 Format *Command Line* 'setdest'

Tabel 2.1 Keterangan pada *Command Line* 'setdest'

Parameter	Keterangan
-v *version*	Versi simulator yang digunakan
-n *num*	Jumlah *node* dalam skenario
-p *pausetime*	Durasi ketika sebuah *node* tetap diam setelah tiba dilokasi pergerakan. Jika nilai 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 bergerakan 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

Perintah 'setdest' menghasilkan *file* output yang berisi jumlah *node* dan mobilitas yang akan digunakan dalam *file* Tcl selama simulasi. *File* output, selain mengandung skrip pergerakan, juga mengandung beberapa statistik lain tentang perubahan *link* dan rute.

Untuk membuat skenario *node*-movement yang terdiri dari 50 *node*, bergerak dengan kecepatan maksimum 20.0 m / detik dengan jeda rata-rata antar gerakan sebesar 2 detik, simulasi akan berhenti setelah 200 detik dengan batas topologi yang diartikan sebagai 500 X 500 meter2, baris perintah akan terlihat seperti seperti pada Gambar 2.5

```
./setdest -v 1 -n 50 -p 2.0 -M 20.0 -t 200 -x
500 -
y 500 > scen-20-test
```

Gambar 2.6 Contoh *Command Line 'setdest'*

File output ditulis ke "stdout" secara default. Di sini output disimpan ke dalam *file* "scen-20-test". *File* dimulai dengan posisi awal *node* dan berlanjut menetapkan *node*-movement.

```
$ns_ at 2.000000000000 "$node_(0) setdest
90.441179033457 44.896095544010
1.373556960010
```

Gambar 2.7 Hasil *Output file* 'scen-20-test'

Command line pada Gambar 2.6 dari 'scen-20-test' mendefinisikan bahwa *node* (0) pada waktu 2.0 detik mulai bergerak ke arah tujuan (90,44, 44,89) pada kecepatan 1.37m / detik. *Command line* ini dapat digunakan untuk mengubah arah dan kecepatan gerak dari *mobile node*. Arahan untuk General Operations *Direct*or l (GOD) yang ada juga di *file node*movement.

Objek "GOD" digunakan untuk menyimpan informasi global tentang keadaan dari lingkungan jaringan dan *node* di sekitarnya. Namun isi dari *file* 'GOD' tidak boleh diketahui oleh setiap bagian dalam simulasi. Dalam simulasi di sini objek "GOD" hanya digunakan untuk menyimpan sebuah *array* dari jumlah *hop* terpendek yang diperlukan untuk mencapai dari satu *node* ke *node* yang lain. Objek "GOD" tidak menghitung jumlah

`

hop yang diperlukan *on-the-fly* selama simulasi berjalan, karena memerlukan waktu yang cukup. Namun "GOD" menghitung *hop* di akhir simulasi. Informasi yang dimuat ke dalam objek "GOD" dari pola pergerakan *file* terdapat pada baris perintah di Gambar 2.7.

```
$ns_ at 899.642 "$god_ set-dist 23 46 2"
```

Gambar 2.8 *Command Line 'GOD'* pada *Output file* "scen-20-test'

Ini berarti bahwa jarak terpendek antara *node* 23 dan *node* 46 berubah menjadi 2 *hop* di waktu 899,642 detik. Program 'setdest' menghasilkan *file node-movement* menggunakan algoritma *random way point*. Perintah-perintah yang termasuk dalam program utama untuk memuat *file-file* ini dalam objek "GOD"

2.11.2. *File Traffic-Connection*

Random traffic connection pada TCP dan CBR bisa diseting antara mobilitas *node* menggunakan skrip *traffic* skenario generator. Untuk menghasilkan alur traffic yang acak, dapat digunakan skrip Tcl yang disebut "cbrgen. Skrip ini membantu untuk menghasilkan *traffic load* atau beban trafik. Beban dapat berupa TCP atau CBR. Skrip ini disimpan di dalam *file* 'CMU-scen-gen ' yang terletak di direktori ~ns/indep-utils/cmu-scen-gen.

Program "cbrgen.tcl" digunakan sesuai dengan perintah pada Gambar 2.8 dan dengan keterangan yang ditunjukkan pada Tabel 5.7.

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

Gambar 2.9 Format *Command Line* untuk cbrgen.tcl

Tabel 2.2 Keterangan untuk *Command Line file* cbrgn.tcl

Parameter	Keterangan
-type cbr|tcp	Jenis *traffic* yang digunakan TCP atau CBR
-nn *node*s	Jumlah total *node*
-s *seed*	*Random Seed*
-mc connections	Jumlah koneksi
-rate rate	Jumlah paket per detik. Pada CBR panjang paket adalah tetap sebesar 512 *bytes* selama simulasi

Pada CBR, data rate dapat dihitung sebagai berikut:

*Data Rate (bits/second) = 512 bytes*8 bits/bytes * rate (packets/second defined in "cbrgen")*

Command line pada Gambar 2.9 membuat sebuah *file* koneksi CBR antara 50 *node*, yang memiliki maksimal 20 koneksi, dengan nilai *seed* 1,0 dan jumlah paket per detik sebanyak 4.0.

```
ns cbrgen.tcl -type cbr -nn 50 -seed 1.0 -mc
20 -
rate 4.0 > cbr-20-test
```

Gambar 2.10 Contoh *Command Line* untuk cbrgen.tcl

Dari *file* cbr-20-test (ke mana output dari generator akan diarahkan) sehingga menghasilkan salah satu koneksi cbr yang terlihat seperti pada Gambar 2.10.

`

```
#
# 2 connecting to 3 at time
82.557023746220864
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(3) $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)
$ns_ at 82.557023746220864 "$cbr_(0) start"
```

Gambar 2.11 koneksi CBR pada *Output* cbr-20-test

Dengan demikian koneksi UDP adalah *setup* antara *node* 2 dan 3. Jumlah sumber UDP (dipilih antara *node* 0-50) dan jumlah *setup* koneksi diindikasikan sebagai 20 koneksi di akhir *file* cbr- 20-test [10].

(Halaman ini sengaja dikosongkan)

`

BAB III
PERANCANGAN

Pada bab ini akan dijelaskan mengenai dasar perancangan perangkat lunak yang akan dibuat dalam tugas akhir ini. Berawal dari deskripsi umum sistem hingga perancangan skenario, alur dan implementasinya. Sehingga bab ini secara khusus akan menjelaskan perancangan sistem yang dibuat dalam Tugas Akhir.

3.1. Deskripsi Umum

Pada Tugas Akhir ini akan dilakukan analisis tentang performa protocol MAODV pada *Mobile Ad hoc Network* (MANET) dengan penambahan propagasi *Two Ray Groud* and propagasi *Free space.* Rancangan simulasi yang dibuat dapat dilihat pada Gambar 3.1.

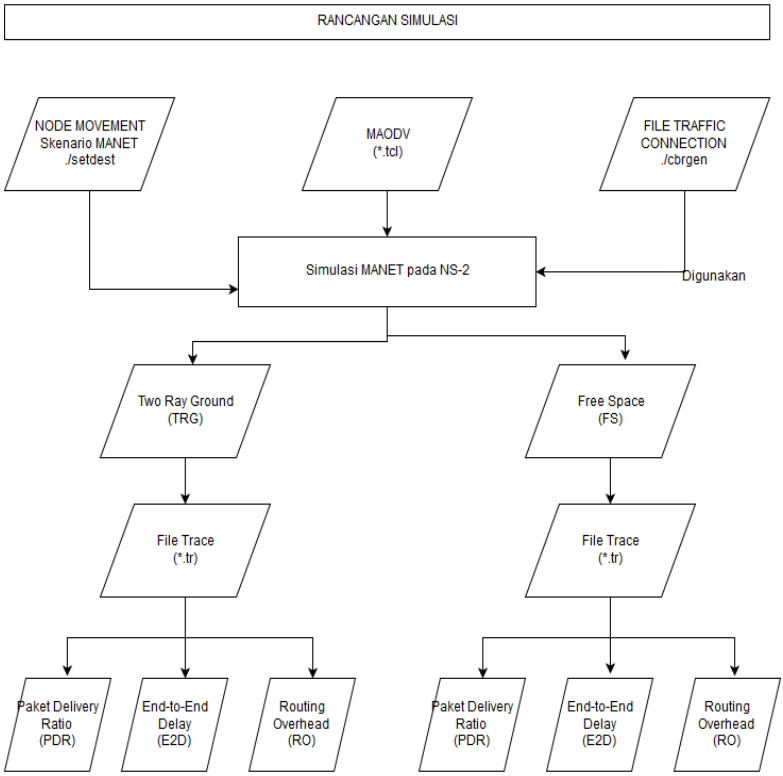
Dalam peneitian ini, terdapat 2 jenis propagasi yang digunakan sebagai pembandingan pengukuran performa dari protocol MAODV dengan menggunakan *scenario* dan trafik pada jaringan MANET dengan kata lain, kita menggunakan sebuah *scenario* dan trafik yang acak dengan menentukan parameter masing-masing nya.

Untuk membentuk sebuah *file traffic-connection* dibutuhkan sebuah cbrgen.tcl dengan meng-*generate* tipe sebuah jaringan, *Random seed* yang dibutuhkan, Jumlah koneksi dan *Rate* alias jumlah paket per detik serta pada CBR panjang paket adalah tetap yaitu sebesar 512 *bytes* selama simulasi.

Sedangkan untuk membuat sebuah *scenario* pada MANET ini adalah dengan perintah ‘setdest’ yang akan menghasilkan *file output* yang berisi jumlah *node* dan mobilitas yang akan digunakan dalam *file* TCl selama simulasi ini berlangsung. *File output*, selain mengandung skrip pergerakan, juga mengandung beberapa statistik lain tetang perubahan *link* dan rute.

Pada setiap skenario, jumlah kecepatan *node* dibuat bervariasi yaitu 5, 10, 15 dan 20 m/s dan juga dengan penambahan luas area yaitu 500m x 500m, 1000m x 1000m,

1500m x 1500m dan 2000m x 2000m. 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* (E2D), dan *Routing overhead* (RO). Dari hasil metrik tersebut dianalisis performa MAODV berdasarkan kedua propagasi yang ditambahkan ke protokol yang sama.



Gambar 3.1 Tahapan Rancang Simulasi

3.2. Perancangan Skenario

Perancangan skenario uji coba mobilitas MANET diawali dengan melakukan pembuatan skenario pada *mobility generation*

yang bersifat *Random Way Point* kemudian membuat koneksi dengan menggunakan *file traffic-connection* yang sudah ada pada NS-2. Pada Tugas Akhir ini pembuatan skenario untuk melihat pergerakan *node* dibedakan berdasarkan 4 kecepatan maksimum yaitu 5, 10, 15 dan 20 m/s. Sedangkan untuk koneksinya hanya digunakan 2 *node* untuk menentukan *node* pengirim dan *node* penerima paket dan juga dengan penambahan luas area. Penjelasan untuk perancangan skenario pada *mobility generator* dan pembuatan koneksi antar *node* adalah sebagai berikut:

3.2.1. Skenario *Mobility Generation*

Skenario *mobility generation* dibuat dengan membuat *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.

Table 3.1 Parameter Skenario *Mobility Generation* Berdasarkan Kecepatan

No.	Parameter	Spesifikasi
1	Jumlah *Node*	50
2	Waktu Simulasi	500 detik
3	Luas Area Simulasi	800 m x 800 m
3	Kecepatan Maksimal	5, 10, 15, 20 m/s
5	Sumber *Traffic*	CBR
6	Waktu Jeda (dalam detik)	0
7	Ukuran Paket	256 *byte*
8	*Rate* Paket	1 paket per detik
9	Jumlah Maksimal Koneksi	1
10	Model mobilitas yang digunakan	*Random Way Point*

Table 3.2 Parameter Skenario *Mobility Generation* Berdasarkan Luas Area

No.	Parameter	Spesifikasi
1	Jumlah *Node*	50
2	Waktu Simulasi	500 detik
3	Luas Area Simulasi	500mx500m, 1000mx1000m, 1500mx1500m, 2000mx2000m
3	Kecepatan Maksimal	5 m/s
5	Sumber *Traffic*	CBR
6	Waktu Jeda (dalam detik)	0
7	Ukuran Paket	256 *byte*
8	*Rate* Paket	1 paket per detik
9	Jumlah Maksimal Koneksi	1
10	Model mobilitas yang digunakan	*Random Way Point*

3.2.2. *Traffic-Connection Generation*

Traffic-Connection dibuat dengan menjalankan program cbrgen.tcl yang telah disediakan oleh NS-2 nya sendiri yang nanti akan menghasilkan sebuah *output* yang *file* .tct dan digunakan sebagai koneksi untuk menghubungkan antar *node* yang ada pada skenario selama simulasi ini.

Table 3.3 Parameter *Traffic-Connection Generation*

No.	Parameter	Spesifikasi
1	-type cbr|tcp	CBR
2	-nn *node*s	10
3	-s *seed*	1.0

`

4	-mc connections	1
5	-rate rate	1

3.3. Perancangan Simulasi pada NS-2

Pada perancangan kode NS-2 dengan konfigurasi MANET, dilakukan penggabungan skenario mobilitas dengan skrip TCL yang diberikan parameter-parameter untuk membangun percobaan simulasi MANET pada NS-2. Berikut parameter simulasi perancangan sistem MANET yang digunakan dapat dilihat pada Table 3.4

Table 3.4 Parameter simulasi

No.	Parameter	Spesifikasi
1	*Network Simulator*	NS-2, 2.26
2	*Routing Protocol*	MAODV
3	Waktu simulasi	500 detik
3	Jumlah *Node*	50
5	Area simulasi	800 m x 800 m
6	Kecepatan Paket	5, 10, 15, 20 m/s
7	Radius transmisi	250 m
8	Tipe koneksi	UDP
9	Tipe data	*Constant Bit Rate* (CBR)
10	*Source / Destination*	Statik (*Node* 0 / *Node* 40 - *Node* 49)
11	Kecepatan generasi paket	1 paket per detik
12	Ukuran paket data	256 *bytes* / 512 *bit per second*
13	Protokol MAC	IEEE 802.11
14	Mode propagasi	- *Two Ray Ground* - *Free space*
15	Tipe Antena	Omni Antenna

16	Tipe Interface Queue	Droptail/PriQueue
17	Tipe kanal	*Wireless Channel*
18	Tipe trace	*Old Wireless Format Trace*
19	Tipe Peta	MANET *(random way point)*

3.4. Perancangan Metrik Analisis

Metrik yang akan dianalisis pada Tugas Akhir ini adalah *Packet delivery ratio* (PDR), *End-to-end delay* (E2D), dan *Routing overhead* (RO). Penjelasannya sebagai berikut:

3.4.1. *Packet delivery ratio* (PDR)

PDR dihitung dari perbandingan antara paket yang dikirim dengan paket yang diterima. PDR dihitung dengan menggunakan persamaan dibawah, dimana *received* adalah banyaknya paket data yang diterima dan *sent* adalah banyaknya paket data yang dikirimkan.

$$PDR = \frac{received}{sent} \times 100$$

3.4.2. *End-to-end Delay* (E2D)

E2D dihitung dari rata-rata *delay* antara waktu paket diterima dan waktu paket dikirim. E2D dihitung dengan menggunakan persamaan dibawah, dimana $t_{received[i]}$ adalah waktu penerimaan paket dengan urutan / id ke-*i*, $t_{sent[i]}$ adalah waktu pengiriman paket dengan urutan / id ke-*i*, dan *sent* adalah banyaknya paket data yang dikirimkan.

$$E2D = \frac{\sum_{i \leq sent}^{i=0} t_{received[i]} - t_{sent[i]}}{sent}$$

3.4.3. *Routing overhead* (RO)

Routing overhead adalah jumlah paket kontrol *routing* yang ditransimisikan per data paket yang terkirim ke tujuan selama simulasi terjadi. RO dihitung berdasarkan paket *routing* yang

`

ditransmisikan. Baris yang mengandung *routing overhead* pada *trace file* ditandai dengan paket yang bertipe *send* (s) / *forward* (f) dan terdapat *header* paket dari protokol MAODV.

`

BAB IV
IMPLEMENTASI

Bab ini merupakan bahasan mengenai implementasi dari perancangan sistem yang telah dijabarkan pada bab-bab sebelumnya.

4.1. Lingkungan Pembangunan Perangkat Lunak

Pembangunan perangkat lunak dilakukan pada lingkungan pengembangan sebagai berikut:

4.1.1. Lingkungan Perangkat Lunak

Adapun perangkat lunak yang digunakan untuk pengembangan sistem adalah sebagai berikut:

- Sistem Operasi Centos 6.2 32 bit untuk lingkungan NS-2.26
- Windows 7 SP2 64 bit
- Microsoft Office 2010
- Virtual Box

4.1.2. Lingkungan Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk implementasi perangkat lunak Tugas Akhir adalah sebagai berikut:

- *Processor* Intel(R) Core(TM) i3-3110M CPU @ 2.40GHz,
- Media penyimpanan sebesar 500GB, dan
- RAM sebesar 4 GB DDR3.

4.2. Implementasi Skenario dan *Traffic*

Implementasi skenario mobilitas MANET dijadikan 2 bagian yaitu adalah *scenario mobility generation* dan *traffic-connection generation.*

4.2.1. Skenario *Mobility Generation*

Dalam implementasi skenario pada *mobility generation* menggunakan aplikasi bawaan dari NS-2 nya sendiri dan telah menyediakan semua fasilitasnya yang bernama ‘setdest’. Ini merupakan alat untuk meng-*generate* semua kebutuhan dalam sebuah *scenario* dalam percobaan nantinya. Fungsi utama dari sebuah *scenario* adalah untuk megetahui efek dari sebuah mobilitas ataupun sebuah pergerakan, simulasi dilakukan dengan sebuah pola yang berbeda berdasarkan kecepatannya. Nantinya akan menghasilkan sebuah *file node movement* dengan mengguakan algoritama *Random Way Point.* Format perintah pada Gambar 4.1 yang digunakan untuk menghasilkan *node* acak sebagai berikut:

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

Gambar 4.1 Format *Command Line* ‘setdest’

Pada simulasi ini kita akan menentukan 4 kecepatan berbeda sebagai berikut: pada skenario A sebesar 5 m/s, skenario B sebesar 10 m/s, skenario C sebesar 15 m/s dan skenario D sebesar 20 m/s. kita dapat membuat ini dengan langsung ke direktori ‘setdest’ ini di dalam *folder* NS-2 masing-masing. Karena disini memakai ns-26 makanya langsung aja ke direktori “~ NS-26/indep-utils/CMUscen-gen/setdest/” untuk menjalankan semuanya kita bisa masukkan seperti dibawah ini:

```
./setdest –v 1 –n 50 –p 0 –s 5 –t 500 –x 800 –y 800
> scen-800x800-50-0-1-1
./setdest –v 1 –n 50 –p 0 –s 10 –t 500 –x 800 –y
800 > scen-800x800-50-0-1-11
./setdest –v 1 –n 50 –p 0 –s 15 –t 500 –x 800 –y
800 > scen-800x800-50-0-1-21
```

Gambar 4.2 Implementasi pada ‘setdest’ dengan Kecepatan Berbeda

Pada Gambar 4.2 dapat dilihat implementasi perintah pada ‘setdest’ dengan berbagai macam kecepatan maksimal yang diberikan secara berbeda-beda, setiap kecepatan dikasih 10 skenario berbeda-beda juga untuk mendapatkan hasil yang valid tentunya jadi akan ada 40 skenario pada simulasi ini. Dibawah akan diberikan potongan-potongan dari hasil seperti di Gambar 4.3 dibawah ini:

```
#
# nodes: 50, pause: 0.00, max speed: 5.00  max x =
800.00, max y: 800.00
#
$node_(0) set X_ 673.853281736719
$node_(0) set Y_ 615.485067856706
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 82.318585068515
$node_(1) set Y_ 254.657151030537
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 441.802724053843
$node_(2) set Y_ 222.832186694512
$node_(2) set Z_ 0.000000000000
```

Gambar 4.3 Posisi *node* dalam X,Y dan Z

```
$ns_ at 0.000000000000 "$node_(0) setdest
58.753936602925 359.026424465118 2.566225140769"
$ns_ at 0.000000000000 "$node_(1) setdest
370.651067117095 21.823643459780 0.592811521679"
$ns_ at 0.000000000000 "$node_(2) setdest
656.742068334099 528.607118770080 4.878384590778"
$ns_ at 0.000000000000 "$node_(3) setdest
556.376122247016 160.724870821175 4.514217913141"
$ns_ at 0.000000000000 "$node_(4) setdest
379.278882707005 214.287099934117 2.782944243833"
```

Gambar 4.4 Perpindahan *node*

4.2.2. *Traffic-Connection Generation*

Skenario mobilitas MANET pada penelitian ini dibagi menjadi dua bagian sesuai dengan propagasinya yaitu skenario MANET untuk propagasi *Two Ray Ground* dan *Free space*

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

Gambar 4.5 Format *Command Line* untuk cbrgen.tcl

Pada Gambar 4.7 merupakan perintah untuk membentuk sebuah koneksi menggunakan file cbrgen.tcl dan hasilnya dapat dilihat pada Gambar 4.8 yang merupakan pengiriman dari *node* 0 ke *node* 40 smapai dengan *node* 49 yang total 10 *node.*

```
#
# First Multicast group
# 1 sender: node 0
# receiver(s): nodes 40 through 49
#
set udp_(0) [new Agent/UDP]
$udp_(0) set dst_addr_ 0xE000000
$ns_ attach-agent $node_(0) $udp_(0)
#
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 256
$cbr_(0) set interval_ 0.50
$cbr_(0) set random_ 1
# send enough packets to keep simulation nearly
busy: 2 packets
# a second, starting at 30, stopping at 899: 2*870
= 1740
$cbr_(0) set maxpkts_ 1740
$cbr_(0) attach-agent $udp_(0)
$cbr_(0) set dst_ 0xE000000
$ns_ at 30.0 "$cbr_(0) start"
```

Gambar 4.6 *Traffic* yang akan digunakan pada simulasi

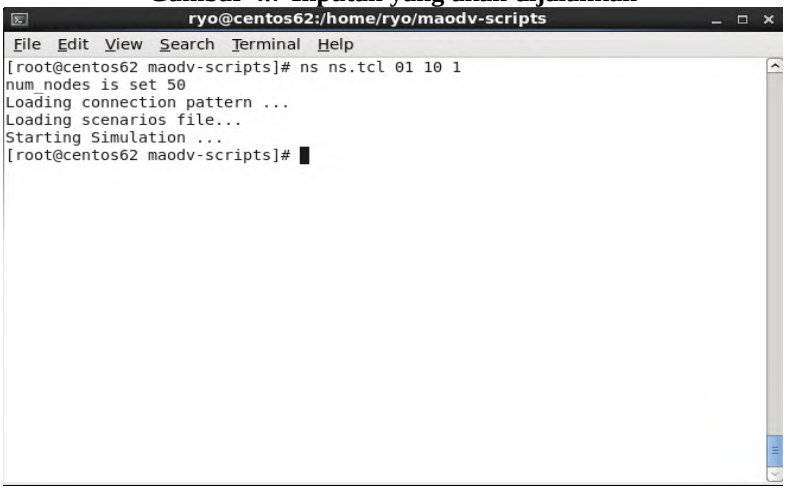
4.3. Implementasi Simulasi pada NS-2

Untuk mensimulasikan MANET dalam lingkungan NS-2, skrip tcl yang telah ada *pacth routing protocol* MAODV. Untuk skenario *mobility generation* dan *traffic-connection generation* yang disimpan pada *file* teks diberikan parameter-parameter yang sesuai dengan perancangan agar dapat dijalankan pada NS-2. Parameter-parameter tersebut dibuat menggukan Bahasa Tcl/Otcl.

Pada gambar 4.9 merupakan sebuah masukan sebuah inputan pada simulasi ini yang disana merupakan perpaduan antara perintah menjalankan ns.tcl dengan 3 inputan dasar adalah banyaknya pengirim, banyak penerima paket dan nomor dari skenario yang akan dipakai. Pada gambar

```
if {$argc != 3} {
        error "Usages: ns ns.tcl <no_of_senders>
<no_receivers> <scenario>"
}
```

Gambar 4.7 Inputan yang akan dijalankan



Gambar 4.8 Perintah untuk Menjalan Simulasi

Terlihat pada Gambar 4. 10 detail dari sebuah perintah menjalankan simulasi yang berarti “ns” merupakan perintah untuk menjalankan sebuah simulasi yang berbasis *network simulator* , “ns.tcl” merupakan *file* yang berisi kodingan dari MAODV yang akan digunakan dalam simulasi ini, “01” merupakan jumlah dari pengirim yaitu 1 pengirim saja sedangkan *node* yang akan pengirim itu ada di *file traffic* dari pengirim 01 dan periman 10, sedangkan untuk “10” sudah jelas adalah jumlah penerima paket dari *node* 1 tersebut dan yang paling kanan adalah nomor dari skenario yang telah dijelaskan diawal.

```
set opt(stop)       500
set nodes    50
set mobility 1
set scenario [lindex $argv 2]
set pausetime 0
set traffic  cbr
set senders  [lindex $argv 0]
set receivers   [lindex $argv 1]

set ns_ [new Simulator]
set topo [new Topography]
$topo load_flatgrid 800 800
```

Gambar 4.9 Parameter Awal dalam Simulasi

Pada gambar 4.11 telah dijelaskan bawah waktu pada simulasi ini adalah 500 detik dengan menggunakan 50 *nodes*. Pada simulasi ini juga menggunakan *mobility* dengan 1 dengan *pause time* sama dengan 0 yang berarti tidak ada jeda dalam pengiriman paket ini. Jenis *traffic* yang akan menghantarkan simulasi ini adalah dengan tipe cbr. Dan pada baris terkahir dari Gambar 4.11 tersebut menunjukkan luas dari area simulasi ini adalah 800 meter kali 800 meter.

```
$ns_ node-config -adhocRouting AODV \
                    -llType LL \
                    -macType Mac/802_11 \
                    -ifqLen 50 \
                    -ifqType
Queue/DropTail/PriQueue \
                    -antType Antenna/OmniAntenna \
                    -propType
Propagation/TwoRayGround \
                    -phyType Phy/WirelessPhy \
                    -channel [new
Channel/WirelessChannel] \
                    -topoInstance $topo \
                    -agentTrace ON \
                    -routerTrace ON \
                    -macTrace OFF \
                    -movementTrace OFF
```

Gambar 4.10 Konfigurasi *Trace File*

Skrip yang ditunjukkan pada Gambar 4.12 merupakan skrip unutk pengaturan varibel global yang dibawali dengan pembuatan simulator baru.

Berhungan pada tugas akhir kali ini melakukan perbandingan antara dua propagasi yaitu propagasi *Two Ray Ground* dan *free space.* Jadi untuk menjalankan propagasi *free space* karena di Gambar 4.12 menunjukkan sebuah parameter untuk *Two Ray Ground*. Jadi kalo ingin mengganti jadi sebuah propagasi *free space* seperti di Gambar 4.13.

```
$ns_ node-config -adhocRouting AODV \
                    -llType LL \
                    -macType Mac/802_11 \
                    -ifqLen 50 \
                    -ifqType
Queue/DropTail/PriQueue \
                    -antType Antenna/OmniAntenna \
                    -propType Propagation/FreeSpace
\
                    -phyType Phy/WirelessPhy \
                    -channel [new
Channel/WirelessChannel] \
                    -topoInstance $topo \
                    -agentTrace ON \
                    -routerTrace ON \
                    -macTrace OFF \
                    -movementTrace OFF
```

Gambar 4.11 Parameter untuk *Free space*

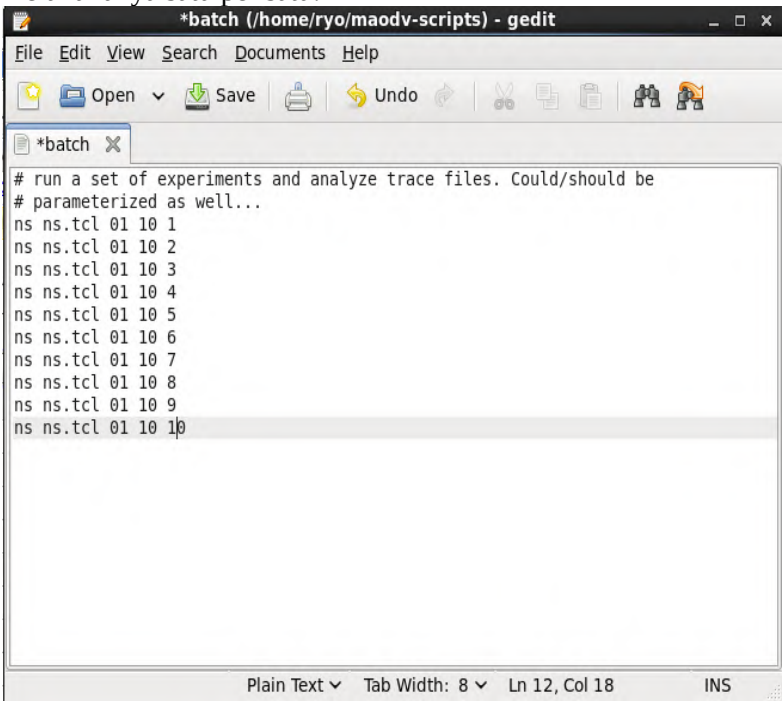
```
puts "Loading connection pattern ..."
source "traffic/$traffic-$senders-$receivers"

puts "Loading scenarios file..."
source "scenarios/scen-800x800-$nodes-$pausetime-
$mobility-$scenario"
```

Gambar 4.12 Pemanggilan Skenario dan Trafik

Pada gambar 4.15 menjelaskan asal sebuah skenario yang diformat dengan “scen” ditambah luas simulasi jumlah *node* serta *pause time,mobility* dan menjadi pembeda dari semua skenario adalah nomor dari skenario tersebut dapat juga kita lihat di Gambar 4.10 untuk perintah menjalankan sebuah simulasinya.

Untuk mempermuda kita dalam menjalankan sebuah simulasi ini yang dari awal kita memberi masing-masing kecepatan dengan 10 skenario yang berbeda sebagai contoh untuk menjalan sebuah simulasi dengan kecepatan 5 m/s. Maka kita membuah sebuah rangkuman sebuah simulasi dengan sebuah *file* yang bernama *batch* bisa dilihat di Gambar 4.16, disana dengan sekali eksekusi bisa menghasilkan 10 *trace file* tanpa harus melakukanya satu per satu.



Gambar 4.13 *File batch*

Untuk mendapatkan hasil sebuah *trace file* dapat melihat Gambar 4.16 . *Set tracefd* merupakan pengaturan untuk direktori *trace file* dan pada *set tracefd* dapat dilakukan pengaturan untuk menghasilkan *trace file* sesuai engan keinginan pengguna. Terdapat dua jenis format *trace file* yang disediakan yaitu *old trace format* dan *new format* namun pada Tugas Akhir ini digunakan yang jenis *old format.*

```
set tracefd [open ./trace-$pausetime-
$mobility-$scenario-$senders-$receivers w]
$ns_ trace-all $tracefd
```

Gambar 4.14 Output

4.4. Implementasi Metrik Analisis

Hasil menjalankan skenario MANET dalam NS-2 dalam bentuk *Trace File* berekstensi .tr dianalisis dengan 3 (tiga) metrik yaitu *Packet delivery ratio* (PDR), *End-to-end delay* (E2D), dan *Routing overhead* (RO). Implementasi dari tiap metrik menggunakan bahasa pemrograman perl dan AWK serta dijelaskan seperti berikut.

4.4.1. *Packet delivery ratio* (PDR)

Proses Proses perhitungan PDR dapat dilakukan dengan menghitung jumlah paket data terkirim yang dilakukan oleh *node* 0 karena diawal kita sudah mengatur pengiriman oleh *node* 0, sedangkan penerimanya adalah dari *node* 40 sampai dengan *node* 49 totalnya adalah sepuluh *node* penerima pada sebuah *trace file.*Penambahan jumlah paket terkirim dilakukan apabila pada bari *trace* yang bersangkutan mengandung semua kondisi dimana kolom pertama mengandung huruf “s” yang menandakan *send packet,* kolom ke-3 menujukkan bahwa *node* yang melakukan pengiriman adalah *node* 0, kolom ke-4 mengandung kata “AGT” dan ke-7 mengandung “cbr”.

Sedangkan untuk paket penerimannya ditandai oleh kolom pertama yang mengandung huruf “r” dengan kata lain adalah

receive packet yaitu semua yang bersangkutan dengan huruf "s" adalah semua *node* penerima paket tersebut. Sedangkan untuk kolom ke-3 nya adalah pemberian pilihan antara *node* 40 sampai dengan *node* 49 karena penerima yang diambil hanyalah dalam jangkauan tersebut saja, kolom ke-4 berbeda dengan sebelumnya kali ini kita melirik kata "RTR" yang menandakan sebuah *routing*. Dan untuk kolom ke-7 sama dengan sebelumnya adalah "cbr". Perihtungan dilakukan sampai baris terakhir *trace file* dan hasilnya adalah hasil hitung nilai PDR simulasi skenario.

Psedeucode PDR ditunjukkan pada Gambar 4.15 dan implementasinya dapat dilihat pada Lampiran.

```
ALGORITMA PDR(trace file)
//Input: trace file simulasi skenario
//Ouput: jumlah packet sent, packet received, dan
//        PDR
BEGIN (
sent ← 0
recv ← 0
recv_id ← 0
pdr ← 0)
(if ($1 == "s" and $3 == "_0_" and $4 == "AGT" and
       $7 == "cbr" )
       sent + 1;

if ($1 == "r" and $4 == "RTR" and $7 == "cbr" and
(($3=="_40_") or ($3=="_41_") or ($3=="_42_") or
($3=="_43_") or ($3=="_44_") or ($3=="_45_") or
($3=="_46_") or ($3=="_47_") or ($3=="_48_") or
($3=="_49_")))
       recv +1;
END (
pdr ← ( recv / sent ) * 100
print sent
print recv
print pdr)
```

Gambar 4.15 Pseudeucode PDR

Contoh perintah untuk analisis PDR dari *trace file* skenario pertama untuk *Two Ray Ground* dengan protokol MAODV dan jumlah *node* 50 seperti pada Gambar 4.16.

```
./analyze.perl trace-0-1-1-01-10 hasilTRG-1-01-10
```

Gambar 4.16 Perintah Menjalankan Skrip analyze.perl

Contoh *output* dari menjalankan skrip tersebut ditunjukkan pada Gambar 4.17 akan menghasilkan *file* yang bernama hasilTRG-1-01-10 dan isi *file* tersebut sebagai berikut:

```
The number of receivers is : 10
The number of sent-out packets is: 935
The number of received packets is: 9207
Ratio is 0.984705882352941
Latency is 0.0209839328706416
```

Gambar 4.17 Hasil *Running* dari *file* analyze.perl

4.4.2. *End-to-end delay* (E2D)

Perhitungan E2D dilakukan dengan menghitung selisih waktu paket data terkirim yang dilakukan oleh *node* 1 dan waktu paket data diterima oleh *node* 0 di dalam satu *trace file*. Pencatatan waktu paket terkirim pada kolom ke-2 dilakukan apabila pada baris *trace* yang bersangkutan mengandung semua kondisi yaitu kolom pertama mengandung huruf "s" yang menandakan *send packet*, kolom ke-3 menunjukkan *node* yang melakukan pengiriman adalah *node* 1, kolom ke-4 dan kolom ke-7 mengandung huruf "AGT" dan "cbr" yang menunjukkan pengiriman paket yang dilakukan adalah pengiriman paket data. Perhitungan waktu dan pencatatan ID serta jumlah paket diterima dilakukan apabila baris *trace* yang bersangkutan mengandung kondisi dimana yaitu kolom pertama mengandung huruf "r" yang menandakan *received packet*, kolom ke-3 menunjukkan *node* yang menerima paket adalah *node* 0, kolom ke-4 dan kolom ke-7 mengandung huruf "AGT" dan "cbr" yang menunjukkan penerimaan paket yang adalah penerimaan paket data. Perhitungan dilakukan sampai baris terakhir *trace file*, dan

dilakukan perhitungan nilai E2D dengan menghitung selisih *delay* paket mulai dari pengiriman sampai paket diterima pada simulasi skenario. *Psedeucode* E2D ditunjukkan pada Gambar 4.18 dan implementasinya dapat dilihat pada Lampiran.

```
ALGORITMA E2D(trace file)
//Input: trace file simulasi skenario
//Ouput: jumlah packet receieved, total delay, dan
//        E2D

BEGIN (
for i in pkt_id
pkt_id[i] ← 0
for i in pkt_sent
       pkt_sent[i] ← 0
for i in pkt_recv
       pkt_recv[i] ← 0
delay = avg_delay ← 0
recv ← 0
recv_id ← 0)

(if ($1 == "s" and $3 == "_1_" and $4 == "AGT" and
     $7 == "cbr)
     pkt_sent[$6] ← $2

if ($1 == "r" and $3 == "_0_" and $4 == "AGT" and
    $7 == "cbr" and  recv_id != $6 )
              recv + 1
              recv_id ← $6
              pkt_recv[$6] ← $2;
END (
for i in pkt_recv
       delay += pkt_recv[i] - pkt_sent[i]
avg_delay ← delay / recv;
print  recv
print delay
print avg_delay
```

Gambar 4.18 Pseudeucode E2D

Contoh perintah untuk analisis E2D dari *trace file* skenario grid 1 dengan protokol MAODV dan jumlah *node* 50 untuk uji

cobanya sama seperti yang ada digambar 4.2 dan hasil sama dengan di gambar 4.3

4.4.3. *Routing overhead* (RO)

Implementasi perhitungan metrik *Routing overhead* MAODV dihitung apabila kondisi-kondisi yang ada terpenuhi yatu pada kolom pertama diawali dengan huruf "s" yang berarti *send packet* atau huruf "f" yang berarti *forward packet*, kolom ke-4 mengandung huruf "RTR" yang berarti paket *routing* dan kolom ke-7 mengandung "AODV" yang berarti paket *routing* MAODV. seperti pada Gambar 4.19. Implementasinya dapat dilihat pada Lampiran.

```
ALGORITMA RO-MAODV(trace file)
//Input: trace file simulasi skenario
//Ouput: jumlah routing overhead protokol OLSR
BEGIN (
rt_pkts ← 0)
(if (($1=="s" || $1=="f") && $4 == "RTR" &&
       $7 == "AODV")
       rt_pkts + 1)
END (
print rt_pkts)
```

Gambar 4.19 Pseudeucode RO

Contoh perintah untuk analisis RO dari *trace file* skenario *grid* 1 dengan protokol MAODV dan jumlah *node* 50 seperti Gambar 4.20.

```
awk -f ro.awk grid50-1.tr
```

Gambar 4.20 Perintah untuk Menjalankan RO

Contoh *output* dari menjalankan *script* diatas seperti pada Gambar 4.21.

```
total no of routing packets       10139
```

Gambar 4.21 Hasil *running* RO

(Halaman ini sengaja dikosongkan)

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dibahas mengenai pengujian dari skenario NS-2 yang telah dibuat. Pengujian fungsionalitas akan dibagi ke dalam beberapa skenario pengujian.

5.1. Lingkungan Platform

Uji coba dilakukan pada laptop yang telah terpasang dua buah sistem operasi yaitu Windows dan Linux. Spesifikasi komputer yang digunakan ditunjukkan pada Table 5.1 Spesifikasi Komputer yang Digunakan

Table 5.1 Spesifikasi Komputer yang Digunakan

Komponen	Spesifikasi
CPU	Intel(R) Core(TM) i3-3110 CPU @ 2.4GHz
Sistem Operasi	Centos 6.2 32 bit
Memori	4 GB DDR3
Media Penyimpanan	500 GB

5.2. Kriteria Pengujian

Pengujian pada *mobility generator* MOVE dan C4R menggunakan kriteria. Pada Gambar 5.2 menunjukkan kriteria-kriteria yang ditentukan didalam skenario.

Table 5.2 Kriteria Pengujian Berdasarkan Kecepatan

Kriteria	**Spesifikasi**
Skenario	MANET
Jumlah *Node*	50
Jumlah Percobaan	10 kali dengan 4 mobilitas berbeda
Jarak *Node* 1 dan *Node* 0	2 variasi, jarak dekat dan jarak jauh
Posisi Awal *Node*	Acak

Pergerakan	Acak
Protokol Routing	MAODV
Pengiriman Paket Data	500 detik

Table 5.3 Kriteria Pengujian Berdasarkan Luas Area

Kriteria	**Spesifikasi**
Skenario	MANET
Jumlah *Node*	50
Jumlah Percobaan	10 kali dengan 4 Luas Area berbeda
Jarak *Node* 1 dan *Node* 0	2 variasi, jarak dekat dan jarak jauh
Posisi Awal *Node*	Acak
Pergerakan	Acak
Protokol Routing	MAODV
Pengiriman Paket Data	500 detik

Table 5.4 Konfigurasi Virtual Box

Komponen	Konfigurasi
General	
Name	maodv
Operating System	Red Hat (32-bit)
System	
Base Memory	1874 MB
Boot Order	Floppy, Optical, Hard Disk
Acceleration	PAE/NS, KVM Paravirtualization
Display	
Video Memory	12 MB
Remote Desktop Server	Disabled
Video Capture	Disabled
Storage	
Controller	IDE
IDE Secondary Master	[Optical Driver]

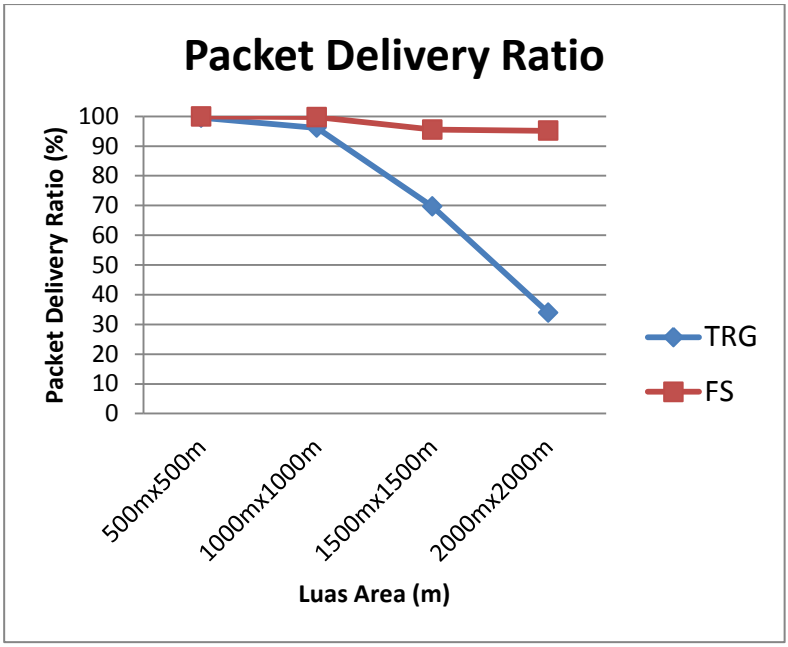
	VBoxGuestAdditions.iso (55.46 MB)
Controller	SATA
SATA Port 0	Maodv.vhd (Normal, 20.00 GB)
Audio	
Host Driver	Windows *Direct*Sound
Controller	ICH AC97
Network	
Adapter 1	Intel PRO/1000 MT Desktop (Bridged Adapter, Qualcomm Atheros AR948WB-EG Wireless Network Adapter)
USB	
USB Controller	OHCI
Device Filters	0 (0 active)
Shared Folders	
None	
Description	
None	

5.3. Analisis *Packet delivery ratio* (PDR)

Trace file hasil menjalankan program skenario *grid* dan riil dianalisis nilai PDR melalui skrip pdr.awk. Hasil tiap perhitungan PDR skenario ditabulasikan dan dirata-ratakan menjadi tabel berikut.

Table 5.5 *Packet delivery ratio* Skenario Grid Berdasarkan Luas Area

Luas Area	PDR (%)	
	Two Ray Ground	*Free space*
500m x 500m	99.49	99.94
1000m x 1000m	96.11	99.74
1500m x 1500m	69.62	95.53
2000m x 2000m	33.93	95.15



Gambar 5.1 Grafik *Packet delivery ratio* Berdasarkan Luas Area

Pada Table 5.5 terlihat nilai yang didapatkan dari hasil pencarian *Packet Delivery Ratio* dari pengujian dengan menggunakan luas area yang berbeda yaitu dengan menggunakan 4 luas area yang berbeda.

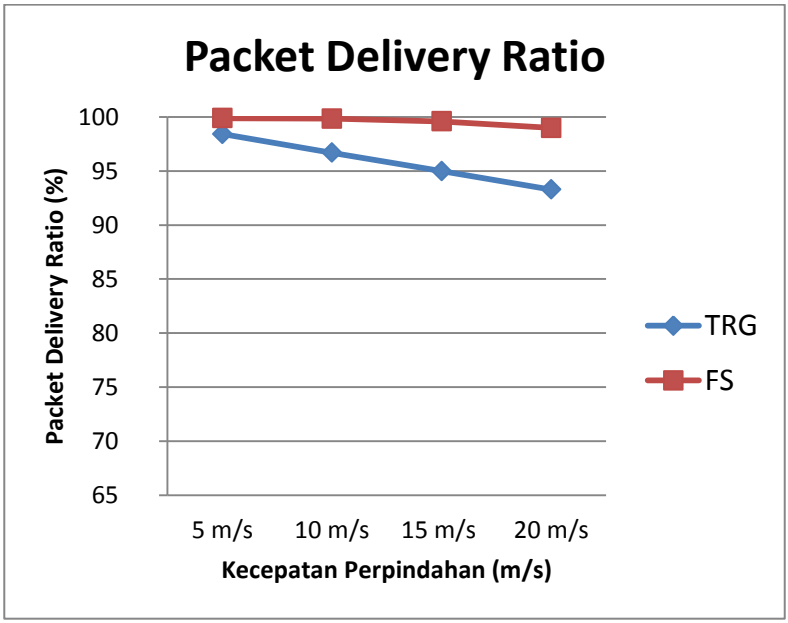
Untuk *Two Ray Ground* mendapatkan hasil pada 500m x 500m sebesar 99.49% , sedangkan pada 1000m x 1000m dengan hasil 96.11%, pada luas 1500m x 1500m dengan nilai 69,62% dan pada luas area dengan 2000m x 2000m mendapatkan nilai 33.93%.

Sedangkan pada propagasi *Free Space* pada luas 500m x 500m mendapatkan nilai 99,94%, pada 1000m x 1000m mendapatkan nilai 99.74%, pada 1500m x 1500m memperoleh nilai 95.53% dan terakhir pada 2000m x 2000m mendapatkan nilai 95.15%.

Dari semua pengujian dengan perbedaan luas area seperti diatas untuk propagasi *Two Ray Ground* mendapatkan penurunan yang sangat tajam disaat penambahan dari 1000m x 1000m ke 1500m x 1500m dari 96.11% ke 69.62%. dan untuk propagasi *Free Space* hanya mengalami pengurangan yang sangat kecil dan cenderung untuk stabil.

Table 5.6 *Packet delivery ratio* Skenario *Grid* Berdasarkan Kecepatan

Kecepatan Pengiriman	PDR (%)	
	Two Ray Ground	*Free space*
5	98.41	99.87
10	96.67	99.82
15	94.97	99.58
20	93.27	98.97



Gambar 5.2 Grafik *Packet delivery ratio* Berdasarkan Kecepatan

Gambar 5.1 menunjukkan grafik performa PDR pada protokol MAODV dengan menggunakan propagasi yang berbeda yaitu progasi *Two Ray Ground* dan *Free space* pada jaringan MANET. Terlihat sekali bahwa PDR yang dihasilkan semakin menurun oleh propagasi *Two Ray Ground* maupun dengan propagasi *Free space* seiringin dengan bertambah kecepatan. Untuk keseluruhan kecepatan disana juga terlihat propagasi *Free space* mendapatkan nilai yang lebih tinggi disemua kecepatannya dibandingkan pada propagasi *Two Ray Ground.*

Dari tabel 5.4 dihasilkan PDR dari *Two Ray Ground* dengan kecepatan 5 m/s dihasilkan 98,41% , pada kecepatan 10 m/s didapatkan 96,67%, pada kecepatan 15 m/s didapatkan PDR 94,97% dan pada kecepatan 20 m/s didapatkan PDR sebesar 93,27%.

Untuk hasil dari propagasi *Free space* untuk kecepatan 5 m/s didapat PDR sebesar 99,87%, pada kecepatan 10 m/s sebesar 99,82%, pada kecepatan 15 m/s diperoleh PDR sebesar 99,58 dan pada kecepatan 20 m/s didapat hasil sebesar 98,97%.

Nilai PDR yang dihasilkan oleh propagasi *Two Ray Ground* memiliki nilai yang cenderung menurun lebih tajam dibandingkan dengan propagasi *Free space.* Nilai PDR yang dihasilkan pada kecepatan *node* 5 m/s memiliki nilai lebih baik daripada kecepatan 10-20 m/s bisa disebut dengan semakin tinggi kecepatan yang dikasih kesebuah *node* makan akan turun nilai yang didapatkan.

Terlihat dari nilai PDR yang dihasilkan oleh propagasi *Free space* lebih stabil dibandingkan dengan propagasi *Two Ray Ground* dengan memiliki jarak nilai yang sangat sedikit. Dan untuk performanya tidak jauh beda dengan yang dihasilkan oleh propagasi *Two Ray Ground* yang dimana nilai dari kecepatan 5 m/s lebih dari baik dari kecepatan 10-20 m/s, juga sama dengan sebelumnya adalah dimana semakin kecepatan akan mengurangi nilai dari PDR tersebut.

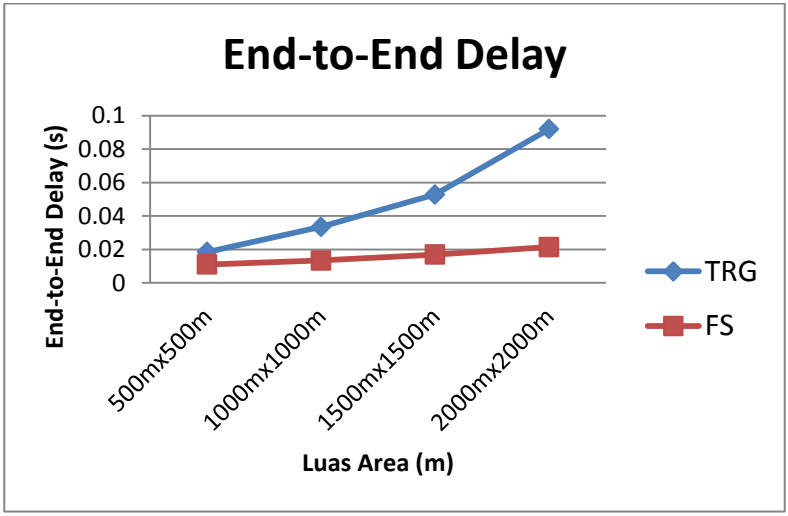
`

5.4. Analisis *End-to-end delay* (E2D)

Trace file hasil menjalankan program skenario *grid* dan riil dianalisis nilai *End-to-end delay* melalui *script delay*.awk. Hasil tiap perhitungan E2D skenario ditabulasikan dan dirata-ratakan menjadi Table 5.8

Table 5.7 End-to-end delay Skenario Grid Berdasarkan Luas Area

Luas Area	E2D (s)	
	Two Ray Ground	*Free space*
500m x 500m	0.01848	0.01099
1000m x 1000m	0.03346	0.0134
1500m x 1500m	0.05277	0.01691
2000m x 2000m	0.09198	0.02145



Gambar 5.3 Grafik *End-to-end delay* Berdasarkan Luas Area

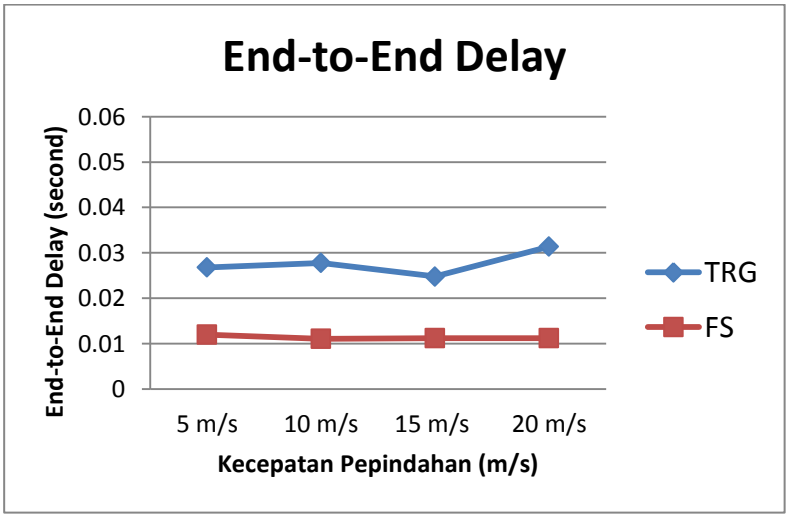
Pada Table 5.7 dapat melihat hasil dari sebuah pengujian dengan menggunakan *End-to-end Delay* pada propagasi *Two Ray Ground* dan propagasi *Free Space* dengan parameter luas area yang berbeda dan terdapat 4 luas area yang digunakan dalam simulasi ini.

Untuk hasil dari pengujian dengan propagasi *Two Ray Ground* pada 500m x 500m mendapatkan *delay* sebesar 0.01848s, pada 1000m x 1000m sebesar 0.03346s, pada 1500m x 1500m mendapatkan nilai *delay* sebesar 0.05277s dan terakhir pada luas area 2000m x 2000m mendapatkan sebesar 0.09198.

Sedangkan pada pengujian dengan propagasi *Free Space* untuk luas area 500m x 500m mendapatkan *delay* sebesar 0.01099, pada 1000m x 1000m 0.0134, pada 1500m x 1500m sebesar 0,01691 dan pada luas area 2000m x 2000m mendapatkan *delay* sebesar 0.02145.

Table 5.8 *End-to-end delay* Skenario *Grid* Berdasarkan Kecepatan

Kecepatan Pengiriman	E2D (second)	
	Two Ray Ground	*Free space*
5	0.026	0.01199
10	0.027	0.01107
15	0.024	0.01121
20	0.031	0.01120



Gambar 5.4 Grafik *End-to-end delay* Berdasarkan Kecepatan

Performa *End-to-end delay* pada propagasi *Two Ray Ground* dan *Free space* memiliki nilai yang sama-sama tidak stabil yang terjadi nilai naik dan turun saat diberi kecepatan yang berbeda-beda.

Adapun hasil yang didapatkan oleh propagasi *Two Ray Ground* pada *End-to-end delay* ini adalah untuk kecepatan 5 m/s didapatkan sebesar 0,026 s, sedangkan pada 10 m/s didapatkan hasil sebesar 0,027 s, pada 15 m/s sebesar 0,024 s dan terakhir pada 20 m/s didapatkan hasil 0,031 s.

Pada percobaan dengan propagasi *Free space* didapatkan hasil pada kecepatan 5 m/s mendapatkan *delay* sebesar 0,01199 s, pada 10 m/s sebesar 0,01107 s, dikecepatan 15 m/s mendapatkan *delay* sebesar 0,01121 dan pada kecepatan 20 m/s mendapatkan hasil 0,01120 s.

Performa E2D pada propagasi *Two Ray Ground* memiliki nilai yang tidak stabil tapi masih memiliki nilai yang lebih dari propagasi *Free space*. Disaat kecepatan awal 5 m/s ke 10 m/s nilai mengalami kenaikin, dari 10 m/s ke 15 m/s nilai turun dengan siknifikan dan lebih rendah dari nilai kecepatan awal 5 m/s sedangkan dari kecepatan 15 m/s ke 20 m/s mengalami kenaikan signifikan dan lebih tinggi dari kenaikan awal pada 5 m/s ke 10 m/s.

Pada performa E2D yang dihasilkan oleh propagasi *Free space* sama memiliki nilai yang tidak stabil tapi lebih memiliki perpindahan nilai yang signifikan dengan bisa disebut nilainya hanya bertambah sedikit dan juga pengurangan tidak terlalu tinggi. Pada kecepatan 5 m/s ke 10 m/s terjadi pengurangan nilai, pada kecepatan 10 m/s sampai dengan 20 m/s terjadi kenaikan nilai dan semuanya lebih tinggi dari kecepatan awal pada 5 m/s.

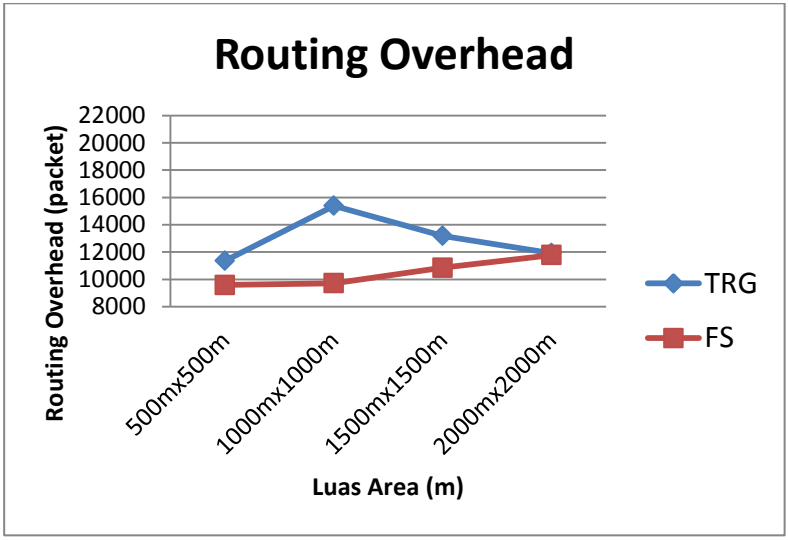
5.5. Analisis *Routing overhead* (RO)

Trace file hasil menjalankan program skenario *grid* dan riil dianalisis nilai *Routing overhead* melalui *script* ro.awk. Hasil tiap

perhitungan RO skenario ditabulasikan dan dirata-ratakan menjadi tabel berikut.

Table 5.9 Routing overhead Skenario Grid Berdasarkan Luas Area

Luas Area	E2D (s)	
	Two Ray Ground	*Free space*
500m x 500m	11365	9586
1000m x 1000m	15388	9716
1500m x 1500m	13187	10853
2000m x 2000m	11917	11780



Gambar 5.5 Grafik *Routing overhead* Berdasarkan Luas Area

Pada Table 5.9 merupakan hasil dari pengujian dari protokol MAODV dengan menggunakan propagasi *Two Ray Ground* dan *Free Space* dengan menggunakan parameter luas area ynag berbeda untuk mendapatkan sebuah nilai *Routing Overhead.*

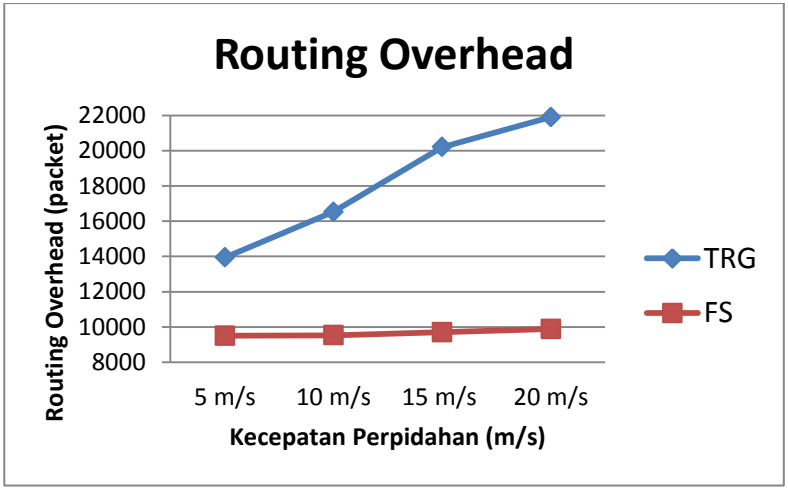
Nilai yang didapat dari *Routing Overhead* pada propagasi *Two Ray Ground* dengan penambahan luas area adalah pada

500m x 500m mendapatkan sebanyak 11365paket, pada 1000m x 1000m mendapatkan 15388paket, pada 1500m x 1500m mendapatkan paket sejumlah 13187paket dan terakhir pada 2000m x 2000m adalah 11917paket.

Sedangkan dengan menggunakan propagasi *Free Space* untuk *Routing Overhead* pada luas area 500m x 500m mendapatkan hasil 9586paket, 1000m x 1000m mendapatkan nilai 9716 paket, pada 1500m x 1500m mendapatkan 10853paket dan pada 2000m x 2000m mendapatkan nilai 11780 paket.

Table 5.10 *Routing overhead* Skenario Grid Berdasarkan Kecepatan

Kecepatan Pengiriman	*ROUTING OVERHEAD*	
	Two Ray Ground	*Free space*
5	13941.4	9504.2
10	16525.8	9525.5
15	20205.3	9698.7
20	21898.5	9882.7



Gambar 5.6 Grafik *Routing overhead* Berdasarkan Kecepatan

Performa *Rouuting Overhead* pada propagasi *Two Ray Ground* dan *Free space* mendapatkan nilai yang cenderung sama-sama memiliki peningkatan disaat diberi penambahan kecepatan. Pada propagasi *Two Ray Ground* lebih me milki nilai kenaikan signifikan sedangakan pada propagasi *Free space* lebih cenderung memiliki perpindahan nilai sedikit.

Adapun hasil *Routing overhead* oleh *Two Ray Ground* pada kecepatan 5 m/s adalah 13941,4 paket, pada kecepatan 10 m/s mendapat 16525,8 paket, pada kecepatan 15 m/s mendapat 20205,3 paket dan dikecepatan 20 m/s terdapat 21898,5 paket.

Untuk *Routing overhead* oleh propagasi *Free space* pada kecepatan 5 m/s mendapatkan 9502,2 paket, pada kecepatan 10 m/s mendapat 9525,5 paket, pada kecepatan 15 m/s diperoleh sebanyak 9698,7 paket dan pada kecepatan 20 m/s mendapat 9882,7 paket.

Nilai *Routing overhead* yang dihasilkan oleh propagasi *Two Ray Ground*, pada kecepatan 10-20 m/s memiliki nilai yang lebih besar dari kecepatan awal 5 m/s jadi setiap pertambahan kecepatan yang diberika mengkasilkan pertambahan nilai dan pada *Two Ray Ground* ini terjadi kenaikan signifikan setiap pertabahan kecepatannya.

Sedangkan pada propagasi *Free space* juga terjadi kenaikan nilai tapi tidak sesignifikan yang terjadi pada propagasi *Two Ray Ground*, disini tetap terjadi kenaikan nilai tapi hanya kenaikan kecil saja. Nilai pada kecepatan 10-20 m/s juga memiliki nilai lebih dari kecepatan awal 5 m/s.

`

LAMPIRAN

Instalasi NS-2

Dokumentasi tentang cara instalasi NS-2 dan ini kita menggunakan salah satu tipe dari ns-2 yaitu ns 2.26 pada centos 6.2

- Pertama kali dilakukan instalasi modul-modul dependensi dari NS-2 yaitu gcc-c++, compat-gcc-34-c++, automake, autoconf, libtool, libX11-devel, libXext-devel, libXau-devel, libXmu-devel, xorg-x11-proto-devel.

```
$ yum install gcc-c++ compat-gcc-34-c++ automake
autoconf libtool libX11-devel libXext-devel libXau-
devel libXmu-devel xorg-x11-proto-devel
```

Gambar 8.1 Perintah untuk instalasi dependensi NS-2

- Setelah semua dependensi lengkap, *file* ns-2.26 dengan nama ns-allinone-2.26.tar.gz yang telah diunduh dapat dipindahkan ke direktori home pada centosnya dan mulailah dengan mengektrak *file* .tar.gz nya
- Setelah melakukan extraksi mulailah langkat patch untuk gcc nya dengan *file* patch yang telah diunduh sebelumnya dengan nama *file* ns-2.26-gcc410.patch

```
$ tar xvf ns-allinone-2.26.tar.gz
$ patch -p0 < ns-2.26-gcc410.patch
```

Gambar 8.2 Gambar Proses ekstrak dan patch

- Kemudian selanjutnya kita mengekpor gcc yang terlah dipasang pada centos tersebut dengan perintah seperti dibawah ini dan jangan lupa untuk masuk kedirektori ns-allinone nya

```
$ cd ns-allinone-2.26/
$ export CC=gcc34 CXX=g++34
```

Gambar 8.3 Gambar Perintah untuk *export* gcc

- Setelah semuanya siap selanjutnya langsung dilakukan pemasangan ns 2.26.

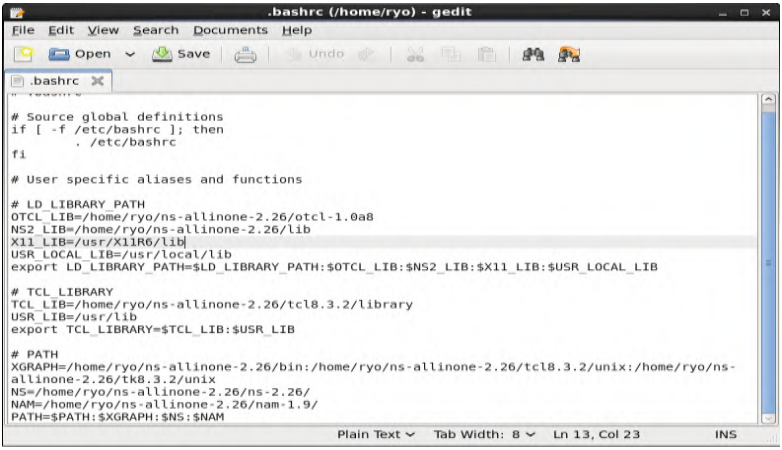
```
$ ./install
```

Gambar 8.4 Gambar Perintah untuk instalasi NS-2.26

- Setelah pemasangan ns-2.26 berjalan dengan lancar tanpa error, langkah selanjutnya adalah menambahkan configurasi pada .bashrc yang terletak dihome direktori dan jangan lupa configurasi sesuai dengan nama direktori serta versi yang ada di *folder* ns-allinone nya masing-masing.

```
$ cd ..
$ gedit .bashrc
```

Gambar 8.5 Gambar Perintah menambahkan configurasi di .bashrc



Gambar 8.6 Gambar Menambakan configurasi di .bashrc

- Setelah memasukkan configurasi di .bashrc ,selanjutnya ada kita harus melakukan validasi terhadap semua apa yang telah dipasang sebelumnya.

```
$ cd ns-allinone-2.26/ns-26/
$ .validate
```

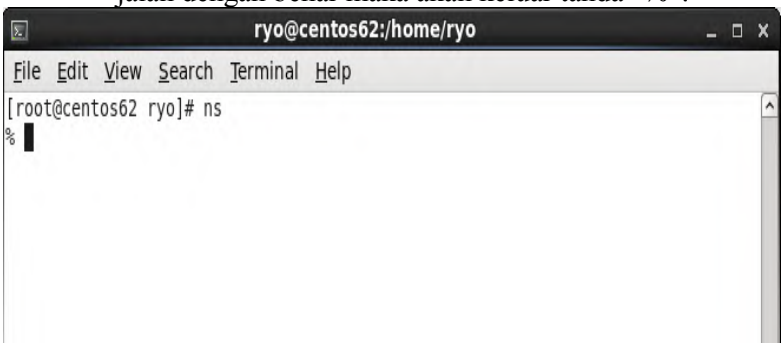
Gambar 8.7 Proses untuk Menvalidasi yang telah diinstall

- Sebelum menjalankan perintah ns, maka kita harus melakukan pemanggilan *file* .bashrc yang telah kita configurasi sebelumnya.

```
$ source /home/ryo/.bashrc
```

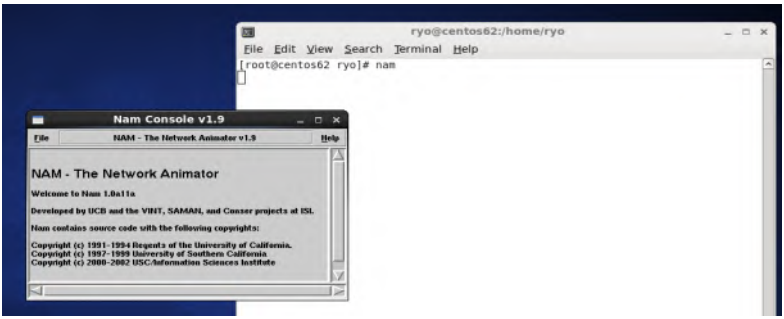
Gambar 8.8 untuk selalu mengarahkan ke Home

- Untuk memastikan jalannya sebuah ns maka kita tinggal melakukan pengujian sebagai berikut ini.jika ns sudah jalan dengan benar maka akan keluar tanda "%".



Gambar 8.9 Uji Coba Menjalankan NS

- Kalo untuk melakukan pengujian jalannya nam tidak jauh seperti melakukannya pada pengujian ns sebelumnya.



Gambar 8.10 Menguji jalannya NAM

```
#
# First Multicast group
# 1 sender: node 0
# receiver(s): nodes 40 through 49
#
set udp_(0) [new Agent/UDP]
$udp_(0) set dst_addr_ 0xE000000
$ns_ attach-agent $node_(0) $udp_(0)
#
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 256
$cbr_(0) set interval_ 0.50
$cbr_(0) set random_ 1
# send enough packets to keep simulation
nearly busy: 2 packets
# a second, starting at 30, stopping at 899:
2*870 = 1740
$cbr_(0) set maxpkts_ 1740
$cbr_(0) attach-agent $udp_(0)
$cbr_(0) set dst_ 0xE000000
$ns_ at 30.0 "$cbr_(0) start"
```

Gambar 8.11 Trafik yang digunakan dalam Tugas Akhir

```
#
# nodes: 50, pause: 0.00, max speed: 5.00  max
x = 800.00, max y: 800.00
```

```
4   #
5   $node_(0) set X_ 673.853281736719
6   $node_(0) set Y_ 615.485067856706
7   $node_(0) set Z_ 0.000000000000
8   $node_(1) set X_ 82.318585068515
9   $node_(1) set Y_ 254.657151030537
10  $node_(1) set Z_ 0.000000000000
11  $node_(2) set X_ 441.802724053843
12  $node_(2) set Y_ 222.832186694512
13  $node_(2) set Z_ 0.000000000000
14  $node_(3) set X_ 111.265399480053
15  $node_(3) set Y_ 273.750410447848
16  $node_(3) set Z_ 0.000000000000
17  $node_(4) set X_ 71.548143886499
18  $node_(4) set Y_ 393.615072283879
19  $node_(4) set Z_ 0.000000000000
20  $node_(5) set X_ 672.659601250942
21  $node_(5) set Y_ 630.490624518611
22  $node_(5) set Z_ 0.000000000000
23  $node_(6) set X_ 55.305674370281
24  $node_(6) set Y_ 699.923768263344
25  $node_(6) set Z_ 0.000000000000
26  $node_(7) set X_ 363.496994694549
27  $node_(7) set Y_ 98.622534729887
28  $node_(7) set Z_ 0.000000000000
29  $node_(8) set X_ 610.053987915448
30  $node_(8) set Y_ 746.912384069938
31  $node_(8) set Z_ 0.000000000000
32  $node_(9) set X_ 479.296398650308
33  $node_(9) set Y_ 795.003648046581
34  $node_(9) set Z_ 0.000000000000
35  $node_(10) set X_ 384.363187651044
36  $node_(10) set Y_ 88.860532474469
37  $node_(10) set Z_ 0.000000000000
38  $node_(11) set X_ 163.085215799912
39  $node_(11) set Y_ 637.332420368945
40  $node_(11) set Z_ 0.000000000000
41  $node_(12) set X_ 368.281274801703
42  $node_(12) set Y_ 543.423568511406
43  $node_(12) set Z_ 0.000000000000
44  $node_(13) set X_ 68.135022877735
45  $node_(13) set Y_ 142.638965712140
46  $node_(13) set Z_ 0.000000000000
47  $node_(14) set X_ 205.903834017924
```

```
48  $node_(14) set Y_ 186.041290568325
49  $node_(14) set Z_ 0.000000000000
50  $node_(15) set X_ 245.571130299660
51  $node_(15) set Y_ 619.992976672607
52  $node_(15) set Z_ 0.000000000000
53  $node_(16) set X_ 176.346202911812
54  $node_(16) set Y_ 391.766547571738
55  $node_(16) set Z_ 0.000000000000
56  $node_(17) set X_ 198.554163169876
57  $node_(17) set Y_ 107.784055177840
58  $node_(17) set Z_ 0.000000000000
59  ...
60  $ns_ at 0.000000000000 "$node_(0) setdest
    58.753936602925 359.026424465118
    2.566225140769"
61  $ns_ at 0.000000000000 "$node_(1) setdest
    370.651067117095 21.823643459780
    0.592811521679"
62  $ns_ at 0.000000000000 "$node_(2) setdest
    656.742068334099 528.607118770080
    4.878384590778"
63  $ns_ at 0.000000000000 "$node_(3) setdest
    556.376122247016 160.724870821175
    4.514217913141"
64  $ns_ at 0.000000000000 "$node_(4) setdest
    379.278882707005 214.287099934117
    2.782944243833"
65  $ns_ at 0.000000000000 "$node_(5) setdest
    610.362758989319 111.143181081345
    4.791317308926"
66  $ns_ at 0.000000000000 "$node_(6) setdest
    449.056853867095 624.061970087907
    2.867511752165"
67  $ns_ at 0.000000000000 "$node_(7) setdest
    349.744270729555 72.449550561026
    1.580528830818"
68  $ns_ at 0.000000000000 "$node_(8) setdest
    220.354485566200 168.148610874757
    0.524549623743"
69  $ns_ at 0.000000000000 "$node_(9) setdest
    785.575371282100 377.310721781252
    0.960211608262"
70  $ns_ at 0.000000000000 "$node_(10) setdest
    761.430545518537 61.079735659199
```

```
      0.431619664835"
71    $ns_ at 0.000000000000 "$node_(11) setdest
      328.303702056215 450.662147099554
      4.633294825611"
72    $ns_ at 0.000000000000 "$node_(12) setdest
      730.403847788461 34.585735945458
      3.061178176833"
73    $ns_ at 0.000000000000 "$node_(13) setdest
      576.122962458426 473.886093163934
      4.473737650676"
74    $ns_ at 0.000000000000 "$node_(14) setdest
      744.861110982278 237.280588353603
      4.814430200821"
75    $ns_ at 0.000000000000 "$node_(15) setdest
      305.200842647295 33.327229258619
      3.093769044733"
76    $ns_ at 0.000000000000 "$node_(16) setdest
      377.285829762801 15.528997785885
      1.184938099810"
77    $ns_ at 0.000000000000 "$node_(17) setdest
      507.056756473101 13.702590123317
      0.988153177438"
78    $ns_ at 0.000000000000 "$node_(18) setdest
      334.242882747790 129.150539930750
      3.359435404130"
79    $ns_ at 0.000000000000 "$node_(19) setdest
      301.898113357557 462.906648098627
      4.013957668772"
80    $ns_ at 0.000000000000 "$node_(20) setdest
      328.728269873066 53.722989925738
      3.897318352396"
81    $ns_ at 0.000000000000 "$node_(21) setdest
      86.272471292101 58.730279332003
      3.873697307596"
      ...
82    $god_ set-dist 0 1 4
83    $god_ set-dist 0 2 2
84    $god_ set-dist 0 3 4
85    $god_ set-dist 0 4 3
86    $god_ set-dist 0 5 1
87    $god_ set-dist 0 6 3
88    $god_ set-dist 0 7 3
89    $god_ set-dist 0 8 1
90    $god_ set-dist 0 9 2
```

```
91  | $god_ set-dist 0 10 3
92  | $god_ set-dist 0 11 3
93  | $god_ set-dist 0 12 2
94  | $god_ set-dist 0 13 4
95  | $god_ set-dist 0 14 3
96  | $god_ set-dist 0 15 2
97  | $god_ set-dist 0 16 3
98  | $god_ set-dist 0 17 4
99  | $god_ set-dist 0 18 1
100 | $god_ set-dist 0 19 2
101 | $god_ set-dist 0 20 1
102 | $god_ set-dist 0 21 2
103 | $god_ set-dist 0 22 3
104 | $god_ set-dist 0 23 1
105 | $god_ set-dist 0 24 1
106 | $god_ set-dist 0 25 2
107 | $god_ set-dist 0 26 3
108 | $god_ set-dist 0 27 2
109 | $god_ set-dist 0 28 3
110 | $god_ set-dist 0 29 2
111 | $god_ set-dist 0 30 1
112 | $god_ set-dist 0 31 4
113 | $god_ set-dist 0 32 2
114 | $god_ set-dist 0 33 2
115 | $god_ set-dist 0 34 3
116 | $god_ set-dist 0 35 2
117 | $god_ set-dist 0 36 2
118 | $god_ set-dist 0 37 2
119 | $god_ set-dist 0 38 2
120 | $god_ set-dist 0 39 3
121 | $god_ set-dist 0 40 3
122 | $god_ set-dist 0 41 4
123 | $god_ set-dist 0 42 1
124 | $god_ set-dist 0 43 4
125 | $god_ set-dist 0 44 3
126 | $god_ set-dist 0 45 3
127 | $god_ set-dist 0 46 3
128 | $god_ set-dist 0 47 4
129 | $god_ set-dist 0 48 1
130 | $god_ set-dist 0 49 1
131 | $god_ set-dist 1 2 2
132 | $god_ set-dist 1 3 1
133 | $god_ set-dist 1 4 1
134 | ...
```

```
135 $ns_ at 0.258766641235 "$god_ set-dist 4 13 1"
136 $ns_ at 0.539856160653 "$god_ set-dist 2 5 2"
137 $ns_ at 0.539856160653 "$god_ set-dist 5 7 3"
138 $ns_ at 0.539856160653 "$god_ set-dist 5 10 3"
139 $ns_ at 0.539856160653 "$god_ set-dist 5 14 3"
140 $ns_ at 0.539856160653 "$god_ set-dist 5 19 2"
141 $ns_ at 0.539856160653 "$god_ set-dist 5 21 2"
142 $ns_ at 0.539856160653 "$god_ set-dist 5 22 3"
143 $ns_ at 0.539856160653 "$god_ set-dist 5 29 2"
144 $ns_ at 0.539856160653 "$god_ set-dist 5 32 2"
145 $ns_ at 0.539856160653 "$god_ set-dist 5 33 2"
146 $ns_ at 0.539856160653 "$god_ set-dist 5 34 3"
147 $ns_ at 0.539856160653 "$god_ set-dist 5 36 2"
148 $ns_ at 0.539856160653 "$god_ set-dist 5 39 3"
149 $ns_ at 0.539856160653 "$god_ set-dist 5 44 3"
150 $ns_ at 0.539856160653 "$god_ set-dist 5 45 3"
151 $ns_ at 0.539856160653 "$god_ set-dist 5 46 3"
152 $ns_ at 0.539856160653 "$god_ set-dist 5 48 1"
153 $ns_ at 1.249700055353 "$god_ set-dist 4 47 1"
154 $ns_ at 1.421657814546 "$god_ set-dist 29 31
    2"
155 $ns_ at 1.421657814546 "$god_ set-dist 31 34
    3"
156 $ns_ at 1.421657814546 "$god_ set-dist 31 39
    2"
157 $ns_ at 1.421657814546 "$god_ set-dist 31 45
    3"
158 $ns_ at 1.421657814546 "$god_ set-dist 31 46
    1"
159 $ns_ at 1.609598327437 "$god_ set-dist 6 28 2"
    $ns_ at 1.609598327437 "$god_ set-dist 15 28
160 1"
    $ns_ at 1.609598327437 "$god_ set-dist 28 40
161 2"
    $ns_ at 1.782266359356 "$god_ set-dist 7 25 2"
162 $ns_ at 1.782266359356 "$god_ set-dist 10 25
    2"
163 $ns_ at 1.782266359356 "$god_ set-dist 13 25
    2"
164 $ns_ at 1.782266359356 "$god_ set-dist 17 25
    2"
165 $ns_ at 1.782266359356 "$god_ set-dist 25 26
    1"
166 $ns_ at 1.782266359356 "$god_ set-dist 25 31
```

```
    2"
167 $ns_ at 1.782266359356 "$god_ set-dist 25 41
    2"
168 $ns_ at 1.782266359356 "$god_ set-dist 25 44
    2"
169 $ns_ at 1.782266359356 "$god_ set-dist 25 46
    2"
170 $ns_ at 1.806508953106 "$god_ set-dist 6 32 3"
    $ns_ at 1.806508953106 "$god_ set-dist 6 39 4"
171 $ns_ at 1.806508953106 "$god_ set-dist 11 32
    2"
172 $ns_ at 1.806508953106 "$god_ set-dist 11 39
    3"
173 $ns_ at 1.806508953106 "$god_ set-dist 15 32
    2"
174 $ns_ at 1.806508953106 "$god_ set-dist 15 39
    3"
175 $ns_ at 1.806508953106 "$god_ set-dist 25 32
    1"
176 $ns_ at 1.806508953106 "$god_ set-dist 25 39
    2"
177 $ns_ at 1.806508953106 "$god_ set-dist 27 32
    2"
178 $ns_ at 1.806508953106 "$god_ set-dist 27 39
    3"
179 ...
180 #
181 # Destination Unreachables: 0
182 #
183 # Route Changes: 7474
184 #
185 # Link Changes: 2375
186 #
187 # Node | Route Changes | Link Changes
188 #    0 |           297 |           83
189 #    1 |           267 |           59
190 #    2 |           265 |           81
191 #    3 |           413 |          142
192 #    4 |           290 |           88
193 #    5 |           267 |           82
194 #    6 |           379 |           66
195 #    7 |           366 |          138
196 #    8 |           357 |           44
197 #    9 |           404 |           41
```

```
198 #    10 |              261 |               63
199 #    11 |              159 |               95
200 #    12 |              362 |              128
201 #    13 |              366 |              179
202 #    14 |              333 |              158
203 #    15 |              264 |              142
204 #    16 |              254 |               80
205 #    17 |              250 |               62
206 #    18 |              213 |              102
207 #    19 |              385 |              130
208 #    20 |              328 |              129
209 #    21 |              327 |               94
210 #    22 |              410 |              143
211 #    23 |              262 |               84
212 #    24 |              407 |               41
213 #    25 |              239 |              118
214 #    26 |              188 |              102
215 #    27 |              317 |               83
216 #    28 |              233 |               66
217 #    29 |              342 |              121
218 #    30 |              348 |               75
219 #    31 |              225 |               94
220 #    32 |              434 |              142
221 #    33 |              240 |              113
222 #    34 |              257 |               91
223 #    35 |              400 |               59
224 #    36 |              184 |               74
225 #    37 |              429 |               51
226 #    38 |              158 |               98
227 #    39 |              299 |              118
228 #    40 |              260 |              100
229 #    41 |              253 |              117
230 #    42 |              196 |              102
231 #    43 |              455 |               60
232 #    44 |              212 |               75
233 #    45 |              301 |               41
234 #    46 |              210 |               69
235 #    47 |              288 |              108
236 #    48 |              339 |              119
237 #    49 |              255 |              100
238 #
239
240
```

Gambar 8.12 Contoh *node movement*

```
1   # TCL script takes three parameters: #senders,
2   #receivers, #scenario
3
4   #
5   # Check for command-line parameters
6   #
7
8   if {$argc != 3} {
9           error "Usages: ns ns.tcl
10  <no_of_senders> <no_receivers> <scenario>"
11  }
12
13  set opt(stop)       500
14  set nodes    50
15  set mobility 1
16  set scenario [lindex $argv 2]
17  set pausetime 0
18  set traffic  cbr
19  set senders  [lindex $argv 0]
20  set receivers   [lindex $argv 1]
21
22  set ns_ [new Simulator]
23  set topo [new Topography]
24  $topo load_flatgrid 800 800
25
26  set tracefd [open ./trace-$pausetime-
27  $mobility-$scenario-$senders-$receivers w]
28  $ns_ trace-all $tracefd
29
30  set god_ [create-god $nodes]
31  $ns_ node-config -adhocRouting AODV \
32                      -llType LL \
33                      -macType Mac/802_11 \
34                      -ifqLen 50 \
35                      -ifqType
36  Queue/DropTail/PriQueue \
37                      -antType
38  Antenna/OmniAntenna \
39                      -propType
40  Propagation/TwoRayGround \
41                      -phyType Phy/WirelessPhy \
42                      -channel [new
43  Channel/WirelessChannel] \
```

```
44                          -topoInstance $topo \
45                          -agentTrace ON \
46                          -routerTrace ON \
47                          -macTrace OFF \
48                          -movementTrace OFF
49
50  for {set i 0} {$i < $nodes} {incr i} {
51          set node_($i) [$ns_ node]
52          $node_($i) random-motion 0;
53  }
54
55  puts "Loading connection pattern ..."
56  source "traffic/$traffic-$senders-$receivers"
57
58  puts "Loading scenarios file..."
59  source "scenarios/scen-800x800-$nodes-
60  $pausetime-$mobility-$scenario"
61
62  for {set i 0} {$i < $nodes} {incr i} {
63          $ns_ at $opt(stop) "$node_($i) reset";
64  }
65
66  $ns_ at $opt(stop) "$ns_ halt"
67
68  puts "Starting Simulation ..."
69  $ns_ run
```

Gambar 8.13 Skrip dari MAODV

```
1
2   #!/usr/bin/perl
3   $totalReceivers = 10;
4   $totalNodes = 50;
5   $totalPid = 20000;
6
7   ($traceFile, $outFile) = @ARGV;
8   $outFile = ">".$outFile;
9   open (MYFILE, $traceFile) || die "cannot open
10  the trace file\n";
11  open (PACKET, $outFile) || die "cannot open
```

```
12  the output file\n";
13
14  $i = 0;
15  while($i < $totalPid){
16    $send[$i] = 0;
17    $sendTime[$i] = 0;
18    $j = 0;
19    while($j<$totalNodes){
20      $recv[$i][$j] = 0;
21      $latency[$i][$j] = 0;
22      $j++;
23    }
24    $i++;
25  }
26
27  while (<MYFILE>)
28  {
29    $current_time = findTime($_);
30    if (m/cbr/ && m/^s/ && m/AGT/) {
31      $id = findPacketId($_);
32      $send[$id] = 1;
33      $sendTime[$id] = $current_time;
34    }
35    elsif(m/cbr/ && m/^r/ && m/RTR/){
36      $id = findPacketId($_);
37      $node = findNodeId($_);
38      if ($recv[$id][$node] == 0){
39        $recv[$id][$node] = 1;
40        $latency[$id][$node] = $current_time -
41  $sendTime[$id];
42      }
43    }
44  }
45
46  print "almost done!\n";
47  close (MYFILE);
48
49  $totalSend = 0;
50  $totalRecv = 0;
51  $totalLatency = 0;
52  $i = 0;
53  while ($i < $totalPid){
54    if ($send[$i] == 1){
55      $totalSend++;
```

```
56      $j = $totalNodes - $totalReceivers;
57      if ($j == 0 ) {$j = 1;}
58      while ($j < $totalNodes){
59        if ($recv[$i][$j] == 1){
60          $totalRecv++;
61          $totalLatency += $latency[$i][$j];
62        }
63        $j++;
64      }
65    }
66    $i++;
67  }
68
69  print PACKET "The number of receivers is : ",
70  $totalReceivers, "\n";
71  print PACKET "The number of sent-out packets
72  is: ", $totalSend, "\n";
73  print PACKET "The number of received packets
74  is: ", $totalRecv, "\n";
75  if ($totalReceivers != $totalNodes) {
76    print PACKET "Ratio is ",
77  $totalRecv/($totalSend*$totalReceivers), "\n";
78  }
79  else {
80    print PACKET "Ratio is ",
81  $totalRecv/($totalSend*($totalReceivers - 1)),
82  "\n";
83  }
84
85  print PACKET "Latency is ",
86  $totalLatency/$totalRecv, "\n";
87
88  close (PACKET);
89
90  sub findPacketId {
91        @fields = split(/ /, $_[0]);
92        $pid = $fields[6];
93        if ($fields[4] != "") {$pid =
94  $fileds[5];}
95        return $pid;
96  }
97
98  sub findTime {
99        @fields = split(/ /, $_[0]);
```

```
100         $theTime = $fields[1];
101         return $theTime;
102 }
103
104 sub findNodeId {
105   $startPo = index($_[0], "_")+1;
106   $endPo = index($_[0], "_", $startPo);
107   $len = $endPo - $startPo;
108   $node = substr($_[0], $startPo, $len);
109   return $node;
110 }
```

Gambar 8.14 Implementasi dari PDR dan E2D dengan Perl

```
1   BEGIN{
2   sent = 0;
3   }
4
5   {
6   if(($1=="s" || $1=="f")&& $4=="RTR" &&
7   $7=="AODV")
8   {
9   sent ++ ;
10  }
11  }
12
13  END{
14   printf " Routing overhead:%d\n",sent;
15  }
16
17
```

Gambar 8.15 Implementasi dari RO dengan AWK

```
1   M 0.00000 0 (673.85, 615.49, 0.00), (58.75,
    359.03), 2.57
2   M 0.00000 1 (82.32, 254.66, 0.00), (370.65,
    21.82), 0.59
3   M 0.00000 2 (441.80, 222.83, 0.00), (656.74,
    528.61), 4.88
4   M 0.00000 3 (111.27, 273.75, 0.00), (556.38,
    160.72), 4.51
```

5	M 0.00000 4 (71.55, 393.62, 0.00), (379.28, 214.29), 2.78
6	M 0.00000 5 (672.66, 630.49, 0.00), (610.36, 111.14), 4.79
7	M 0.00000 6 (55.31, 699.92, 0.00), (449.06, 624.06), 2.87
8	M 0.00000 7 (363.50, 98.62, 0.00), (349.74, 72.45), 1.58
9	M 0.00000 8 (610.05, 746.91, 0.00), (220.35, 168.15), 0.52
10	M 0.00000 9 (479.30, 795.00, 0.00), (785.58, 377.31), 0.96
11	M 0.00000 10 (384.36, 88.86, 0.00), (761.43, 61.08), 0.43
12	M 0.00000 11 (163.09, 637.33, 0.00), (328.30, 450.66), 4.63
13	M 0.00000 12 (368.28, 543.42, 0.00), (730.40, 34.59), 3.06
14	M 0.00000 13 (68.14, 142.64, 0.00), (576.12, 473.89), 4.47
15	M 0.00000 14 (205.90, 186.04, 0.00), (744.86, 237.28), 4.81
16	M 0.00000 15 (245.57, 619.99, 0.00), (305.20, 33.33), 3.09
17	M 0.00000 16 (176.35, 391.77, 0.00), (377.29, 15.53), 1.18
18	M 0.00000 17 (198.55, 107.78, 0.00), (507.06, 13.70), 0.99
19	M 0.00000 18 (640.94, 608.15, 0.00), (334.24, 129.15), 3.36
20	M 0.00000 19 (614.74, 215.92, 0.00), (301.90, 462.91), 4.01
21	M 0.00000 20 (793.46, 569.67, 0.00), (328.73, 53.72), 3.90
22	M 0.00000 21 (775.31, 225.67, 0.00), (86.27, 58.73), 3.87
23	M 0.00000 22 (796.09, 120.37, 0.00), (112.46, 176.14), 4.65
24	M 0.00000 23 (677.05, 470.30, 0.00), (233.05, 19.00), 4.06
25	M 0.00000 24 (585.66, 596.47, 0.00), (776.46, 613.18), 2.72
26	M 0.00000 25 (358.17, 485.98, 0.00), (501.66, 98.49), 4.36

27	M 0.00000 26 (230.01, 261.95, 0.00), (489.24, 227.08), 4.65
28	M 0.00000 27 (333.17, 675.29, 0.00), (116.57, 490.69), 4.60
29	M 0.00000 28 (197.26, 369.62, 0.00), (129.50, 476.36), 0.19
30	M 0.00000 29 (497.82, 196.26, 0.00), (193.38, 578.38), 2.03
31	M 0.00000 30 (799.30, 695.07, 0.00), (46.26, 523.50), 1.92
32	M 0.00000 31 (52.11, 137.60, 0.00), (246.63, 152.81), 2.25
33	M 0.00000 32 (422.83, 229.34, 0.00), (492.93, 716.66), 4.10
34	M 0.00000 33 (625.54, 221.97, 0.00), (569.05, 191.83), 4.43
35	M 0.00000 34 (683.32, 58.46, 0.00), (285.82, 539.56), 1.49
36	M 0.00000 35 (306.66, 470.23, 0.00), (274.48, 697.18), 4.26
37	M 0.00000 36 (587.21, 211.46, 0.00), (650.47, 258.19), 1.38
38	M 0.00000 37 (332.17, 550.77, 0.00), (678.43, 752.26), 0.85
39	M 0.00000 38 (480.62, 380.96, 0.00), (202.55, 556.37), 0.09
40	M 0.00000 39 (521.08, 57.80, 0.00), (315.70, 283.56), 0.84
41	M 0.00000 40 (136.85, 767.49, 0.00), (274.79, 192.74), 2.79
42	M 0.00000 41 (21.57, 174.58, 0.00), (557.34, 429.21), 2.09
43	M 0.00000 42 (468.32, 650.78, 0.00), (498.09, 275.21), 2.37
44	M 0.00000 43 (58.23, 179.17, 0.00), (20.34, 607.57), 1.52
45	M 0.00000 44 (315.01, 47.89, 0.00), (131.52, 548.30), 1.08
46	M 0.00000 45 (693.31, 70.57, 0.00), (165.89, 59.98), 0.53
47	M 0.00000 46 (301.13, 89.92, 0.00), (148.77, 758.47), 0.80
48	M 0.00000 47 (143.56, 147.30, 0.00), (310.34, 529.16), 4.28

```
49  M 0.00000 48 (605.56, 386.63, 0.00), (769.60,
    474.00), 0.99
50  M 0.00000 49 (495.58, 502.35, 0.00), (415.43,
    98.59), 3.36
51  s 0.012618241 _41_ RTR  --- 0 AODV 48 [0 0 0
    0] ------- [41:255 -1:255
    5 0] [0x2 0 2 [234881024 0] [41 4]] (REQUEST)
52  s 0.013273170 _43_ RTR  --- 0 AODV 48 [0 0 0
    0] ------- [43:255 -1:255
    5 0] [0x2 0 2 [234881024 0] [43 4]] (REQUEST)
53  r 0.013718365 _43_ RTR  --- 0 AODV 48 [0
    ffffffff 29 800] -------
    [41:255 -1:255 5 0] [0x2 0 2 [234881024 0] [41
    4]] (REQUEST)
54  r 0.013718401 _31_ RTR  --- 0 AODV 48 [0
    ffffffff 29 800] -------
    [41:255 -1:255 5 0] [0x2 0 2 [234881024 0] [41
    4]] (REQUEST)
55  r 0.013718430 _13_ RTR  --- 0 AODV 48 [0
    ffffffff 29 800] -------
    [41:255 -1:255 5 0] [0x2 0 2 [234881024 0] [41
    4]] (REQUEST)
56  r 0.013718576 _1_ RTR  --- 0 AODV 48 [0
    ffffffff 29 800] -------
    [41:255 -1:255 5 0] [0x2 0 2 [234881024 0] [41
    4]] (REQUEST)
57  r 0.013718658 _47_ RTR  --- 0 AODV 48 [0
    ffffffff 29 800] -------
    [41:255 -1:255 5 0] [0x2 0 2 [234881024 0] [41
    4]] (REQUEST)
58  r 0.013718687 _3_ RTR  --- 0 AODV 48 [0
    ffffffff 29 800] -------
    [41:255 -1:255 5 0] [0x2 0 2 [234881024 0] [41
    4]] (REQUEST)
59  r 0.013718857 _14_ RTR  --- 0 AODV 48 [0
    ffffffff 29 800] -------
    [41:255 -1:255 5 0] [0x2 0 2 [234881024 0] [41
    4]] (REQUEST)
60  r 0.013718872 _17_ RTR  --- 0 AODV 48 [0
    ffffffff 29 800] -------
    [41:255 -1:255 5 0] [0x2 0 2 [234881024 0] [41
    4]] (REQUEST)
61  r 0.013718990 _4_ RTR  --- 0 AODV 48 [0
    ffffffff 29 800] -------
```

```
     [41:255 -1:255 5 0] [0x2 0 2 [234881024 0] [41
     4]] (REQUEST)
62   r 0.013718995 _26_ RTR  --- 0 AODV 48 [0
     ffffffff 29 800] -------
     [41:255 -1:255 5 0] [0x2 0 2 [234881024 0] [41
     4]] (REQUEST)
63   s 0.014256873 _42_ RTR  --- 0 AODV 48 [0 0 0
     0] ------- [42:255 -1:255
     5 0] [0x2 0 2 [234881024 0] [42 4]] (REQUEST)
64   r 0.014668488 _41_ RTR  --- 0 AODV 48 [0
     ffffffff 2b 800] -------
     [43:255 -1:255 5 0] [0x2 0 2 [234881024 0] [43
     4]] (REQUEST)
65   r 0.014668491 _13_ RTR  --- 0 AODV 48 [0
     ffffffff 2b 800] -------
     [43:255 -1:255 5 0] [0x2 0 2 [234881024 0] [43
     4]] (REQUEST)
66   r 0.014668505 _31_ RTR  --- 0 AODV 48 [0
     ffffffff 2b 800] -------
     [43:255 -1:255 5 0] [0x2 0 2 [234881024 0] [43
     4]] (REQUEST)
67   r 0.014668629 _1_ RTR  --- 0 AODV 48 [0
     ffffffff 2b 800] -------
     [43:255 -1:255 5 0] [0x2 0 2 [234881024 0] [43
     4]] (REQUEST)
68   r 0.014668668 _47_ RTR  --- 0 AODV 48 [0
     ffffffff 2b 800] -------
     [43:255 -1:255 5 0] [0x2 0 2 [234881024 0] [43
     4]] (REQUEST)
69   r 0.014668726 _3_ RTR  --- 0 AODV 48 [0
     ffffffff 2b 800] -------
     [43:255 -1:255 5 0] [0x2 0 2 [234881024 0] [43
     4]] (REQUEST)
70   r 0.014668858 _14_ RTR  --- 0 AODV 48 [0
     ffffffff 2b 800] -------
     [43:255 -1:255 5 0] [0x2 0 2 [234881024 0] [43
     4]] (REQUEST)
71   r 0.014668889 _17_ RTR  --- 0 AODV 48 [0
     ffffffff 2b 800] -------
     [43:255 -1:255 5 0] [0x2 0 2 [234881024 0] [43
     4]] (REQUEST)
72   r 0.014669000 _26_ RTR  --- 0 AODV 48 [0
     ffffffff 2b 800] -------
     [43:255 -1:255 5 0] [0x2 0 2 [234881024 0] [43
```

```
     4]] (REQUEST)
73   r 0.014669081 _4_ RTR  --- 0 AODV 48 [0
     ffffffff 2b 800] -------
     [43:255 -1:255 5 0] [0x2 0 2 [234881024 0] [43
     4]] (REQUEST)
74   s 0.014702229 _3_ RTR  --- 0 AODV 48 [0
     ffffffff 2b 800] -------
     [3:255 -1:255 4 0] [0x2 1 2 [234881024 0] [43
     4]] (REQUEST)
75   s 0.014917217 _48_ RTR  --- 0 AODV 48 [0 0 0
     0] ------- [48:255 -1:255
     5 0] [0x2 0 2 [234881024 0] [48 4]] (REQUEST)
76   s 0.014988415 _26_ RTR  --- 0 AODV 48 [0
     ffffffff 2b 800] -------
     [26:255 -1:255 4 0] [0x2 1 2 [234881024 0] [43
     4]] (REQUEST)
77   r 0.015197304 _24_ RTR  --- 0 AODV 48 [0
     ffffffff 2a 800] -------
     [42:255 -1:255 5 0] [0x2 0 2 [234881024 0] [42
     4]] (REQUEST)
78   r 0.015197331 _27_ RTR  --- 0 AODV 48 [0
     ffffffff 2a 800] -------
     [42:255 -1:255 5 0] [0x2 0 2 [234881024 0] [42
     4]] (REQUEST)
79   r 0.015197355 _9_ RTR  --- 0 AODV 48 [0
     ffffffff 2a 800] -------
     [42:255 -1:255 5 0] [0x2 0 2 [234881024 0] [42
     4]] (REQUEST)
80   r 0.015197444 _8_ RTR  --- 0 AODV 48 [0
     ffffffff 2a 800] -------
     [42:255 -1:255 5 0] [0x2 0 2 [234881024 0] [42
     4]] (REQUEST)
81   r 0.015197465 _18_ RTR  --- 0 AODV 48 [0
     ffffffff 2a 800] -------
     [42:255 -1:255 5 0] [0x2 0 2 [234881024 0] [42
     4]] (REQUEST)
82   r 0.015197557 _5_ RTR  --- 0 AODV 48 [0
     ffffffff 2a 800] -------
     [42:255 -1:255 5 0] [0x2 0 2 [234881024 0] [42
     4]] (REQUEST)
83   r 0.015197568 _0_ RTR  --- 0 AODV 48 [0
     ffffffff 2a 800] -------
     [42:255 -1:255 5 0] [0x2 0 2 [234881024 0] [42
     4]] (REQUEST)
```

```
84   s 0.015218875 _4_ RTR  --- 0 AODV 48 [0
     ffffffff 29 800] -------
     [4:255 -1:255 4 0] [0x2 1 2 [234881024 0] [41
     4]] (REQUEST)
85   s 0.015448220 _40_ RTR  --- 0 AODV 48 [0 0 0
     0] ------- [40:255 -1:255
     5 0] [0x2 0 2 [234881024 0] [40 4]] (REQUEST)
86   r 0.016432719 _1_ RTR  --- 0 AODV 48 [0
     ffffffff 3 800] ------- [3:255
     -1:255 4 0] [0x2 1 2 [234881024 0] [43 4]]
     (REQUEST)
87   r 0.016432965 _43_ RTR  --- 0 AODV 48 [0
     ffffffff 3 800] -------
     [3:255 -1:255 4 0] [0x2 1 2 [234881024 0] [43
     4]] (REQUEST)
88   r 0.016433001 _26_ RTR  --- 0 AODV 48 [0
     ffffffff 3 800] -------
     [3:255 -1:255 4 0] [0x2 1 2 [234881024 0] [43
     4]] (REQUEST)
89   r 0.016433024 _4_ RTR  --- 0 AODV 48 [0
     ffffffff 3 800] ------- [3:255
     -1:255 4 0] [0x2 1 2 [234881024 0] [43 4]]
     (REQUEST)
90   r 0.016433033 _28_ RTR  --- 0 AODV 48 [0
     ffffffff 3 800] -------
     [3:255 -1:255 4 0] [0x2 1 2 [234881024 0] [43
     4]] (REQUEST)
91   r 0.016433034 _14_ RTR  --- 0 AODV 48 [0
     ffffffff 3 800] -------
     …
```

Gambar 8.16 Salah satu contoh *Trace File*

BAB VI
PENUTUP

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

6.1. Kesimpulan

Kesimpulan yang dapat diambil dalam Tugas Akhir ini adalah sebagai berikut:

1. Performa keinerja protokol *routing* MAODV pada jaringan MANET adalah sebagai berikut:
    - Dengan penambahan jumlah kecepatan *node* dan luas area, performa *packet delivery ratio* yang dihasilkan pada propagasi *Two Ray Ground* dan *Free space* sama-sama mengalami penurunan, dan propagasi *Free Space* mendapatkan nilai yang lebih baik dikarenakan tidak adanya halangan disaat pengiriman paket jadi *Free Space* mendapatkan jumlah paket yang lebih banyak terkirim.
    - Performa *End-to-end delay* pada penambahan kecepatan pengiriman paket mendapatkan hasil *Free Space* lebih baik dari *Two Ray Ground* serta pada penambahana luas area juga mendapatkan nilai *Free Space* baik dikarenakan *Free Space* melakukan pengiriman langsung tanpa adanya penghalan sehingga dapatlah *delay* yang lebih kecil.
    - *Routing overhead* pada penambahan kecepatan paket dan luas area mendapatkan hasil *Free Space* lebih baik dengan mendapatkan nilai RO lebih kecil dari *Two Ray Ground.* Jadi kesimpulannya *Free Space* melakukan pengiriman paket lebih sedikit dari *Two Ray Ground* karena terjadinya banyak kesalahan

dalam pengiriman sehingga membutuhkan pengiriman berulang kali.

2. Kecepatan *node*, luas area pada simulasi dan juga menggunakan propagasi yang beda menghasilkan sebuah pengaruh terhadap performa protokol MAODV.

6.2. Saran

Dalam pengerjaan Tugas Akhir ini terdapat beberapa saran untuk perbaikan serta pengembangan sistem yang telah dikerjakan sebagai berikut:

1. Perlu dibandingankan dengan sebuah protokol yang mempunyai basis yang berbeda seperti ODMRP dan protokol lainnya.
2. Perlu ditambahan *factor* pembanding lainnya seperti paket drop dan lainnya.
3. Perlu juga dilakukan dengan menggukan skenario yang lebih rill dan mempunyai visualisasi dengan jelas seperti dengan menggunakan *Simulation of Urban Mobility* (SUMO) dan lainnya.

DAFTAR PUSTAKA

[1] Arifin, M. S. Hadi, H. Amran dan N. Putra R, "Analisis Performansi Routing AODV pada Jaringan VANet," PENS-ITS, Surabaya, 2011.

[2] Surateno, "OPTIMASI PENENTUAN ZONA PADA PROTOKOL ROUTING HOPNET DENGAN TEKNIK MIN-SEARCHING," ITS, Surabaya, 2010.

[3] R. Kumawat and V. Somani, "Comparative Analysis of DSDV and OLSR Routing Protocols in MANET at Different Traffic Load," International Journal of Computer Applications (IJCA), Mandsaur, 2011.

[4] "Porsot Inc Network and Programming Learning," [Online]. Available: http://porsot.org/tag/virtualbox/. [Diakses 6 Juni 2015].

[5] OpenStreetMap, "OpenStreetMap Wiki," Media Wiki, 10 Juli 2014. [Online]. Available: https://wiki.openstreetmap.org/wiki. [Accessed 8 Mei 2015].

[6] nsnam, "NS-3," nsnam, 2015. [Online]. Available: https://www.nsnam.org/. [Accessed 17 Februari 2015].

[7] D. Krajzewicz, J. Erdmann, M. Behrisch and L. Bieker, "Recent Development and Applications of SUMO – Simulation of Urban MObility," International Journal on Advances in Systems and Measurements, Berlin, 2012.

[8] J. Bennett, OpenStreetMap, Olton Birmingham: Packt Publishing Ltd., 2010.

[9] Rechnernetze und Telematik University of Freiburg, "CoNe Computer Networks and Telematics," 2011. [Online]. Available: http://archive.cone.informatik.uni-freiburg.de/teaching/vorlesung/manet-s07/exercises/DSDV.ppt. [Accessed 1 April 2015].

[10] Graduate Institute of Communication Engineering,

"Advanced Network Technology," 13 March 2008. [Online]. Available: http://ant.comm.ccu.edu.tw/course/96_Network_Simulation/1_Lectures/UM-OLSR.ppt. [Accessed 1 April April].

[11] I. K. A. Mogi, Penerapan Kinetic Graph Frameworl dalam Protokol AODV untuk Fleet Mobility Model pada VANET, Surabaya: ITS, 2013.

[12] "BengkelUbuntu.org," 8 November 2014. [Online]. Available: http://bengkelubuntu.org/teks/awk/Praktikum%2001%20-%20Berkenalan%20dengan%20AWK.pdf. [Accessed 5 April 2015].

`

BIODATA PENULIS

Wiryo Febdila, biasa dipanggil Wiryo / Ryo, dilahirkan di Panyakalan pada tanggal 14 Februari tahun 1993. Penulis adalah anak pertama dari dua bersaudara. Penulis menempuh pendidikan TK Dharmawanita Koto baru Solok (1997-1998), SD Negeri 22 Koto Baru Solok (1998-2004), SMP Swasta Ar-Risalah Padang (2004-2007), MA Swasta Ar-Risalah Padang (2007-2010). Pada tahun 2010, penulis mengikuti Jalur beasiswa santri berpretasi dari Kementrian Agama (KEMENAG) dan diterima di strata satu Jurusan Teknik Informatika Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya angkatan 2010 yang terdaftar dengan NRP 5110100705. Di Jurusan Teknik Informatika ini, penulis mengambil bidang minat Arsitektur dan Jaringan Komputer (AJK). Selama menempuh kuliah, penulis juga aktif sebagai anggota dari CSSMORA ITS.Serta penulis juga aktif sebagai anggota IMAMI SURABAYA (Ikatan Mahasiswa Minang Surabaya).Penulis juga pernah menjadi anggota IBC (ITS Badminton Club).Serta penulis juga sering mengikuti pelatihan dan seminar dalam bilang wirausahan berbasis kampus ataupun nasional. Penulis dapat dihubungi melalui alamat *e-mail* iryo93@gmail.com.