TUGAS AKHIR – KI141502

Studi Kinerja Multipath AODV dengan Menggunakan Network Simulator 2 (NS-2)

BIMA BAHTERADI PUTRA
NRP 5110100105

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

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

FINAL PROJECT – KI141502

AODV Multipath Performance Study Using Network Simulator 2 (NS-2)

BIMA BAHTERADI PUTRA
NRP 51101001005

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

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

LEMBAR PENGESAHAN

Studi Kinerja Multipath AODV dengan Menggunakan Network Simulator 2 (NS-2)

TUGAS AKHIR

Diajukan Untuk 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 :
BIMA BAHTERADI PUTRA
NRP : 5110 100 105

Disetujui oleh Dosen Pembimbing Tugas Akhir :

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

SURABAYA
JUNI 2016

Studi Kinerja Multipath AODV dengan Menggunakan Network Simulator 2 (NS-2)

Nama Mahasiswa	**: Bima Bahteradi Putra**
NRP	**: 5110100105**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**

ABSTRAK

Mobile Ad hoc Network (MANET) merupakan suatu inovasi dalam dunia teknologi yang memungkinkan terjadinya komunikasi jaringan tanpa ketersediaan infrastruktur jaringan yang tetap. Ada beberapa macam protokol routing pada MANET salah satunya adalah AOMDV. AOMDV merupakan pengembangan dari AODV. Perbedaan antara AODV dengan AOMDV adalah jumlah rute yang ditemukan dalam setiap proses pencaria rute. Dalam Tugas Akhir ini, dilakukan penelitian terhadap kinerja AOMDV menggunakan Network Simulator 2 (NS-2).

Uji coba dilakukan dengan membuat pola traffic koneksi dan pola pergerakan node yang kemudian disimulasikan dengan menggunakan script tcl protokol routing AOMDV. Proses tersebut akan menghasilkan file otuput berupa trace file. Trace file hasil dari simulasi akan dianalisis untuk menghitung Packet Delivery Ratio (PDR), Routing Overhead (RO), dan End-to-End Delay.

Dari hasil uji coba yang dilakukan, menunjukkan nilai PDR dari AOMDV mengalami penurunan berdasarkan perubahan kecepatan node. Untuk nilai RO dan End-to-End Delay mengalami kenaikan berdasarkan perubahan kecepatan node. Kecepatan node sangat mempengaruhi kinerja AOMDV. Kecepatan node yang tinggi, menyebabkan performa kinerja AOMDV menjadi buruk.

Kata kunci: MANET, AODV, AOMDV, NS-2.

AODV Multipath Performance Study Using Network Simulator 2 (NS-2)

Student's Name	**: Bima Bahteradi Putra**
Student's ID	**: 5110100105**
Department	**: Teknik Informatika FTIF-ITS**
Advisor	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**

ABSTRACT

Mobile Ad hoc Network (MANET) is an innovation on the technological world that enabled a communication within a a network without a sound infrastructure. There are multiple routing protocol usable with MANET, one of them is AOMDV. AOMDV in itself is an upgrade of the existing AODV routing protocol, the main difference between the two is that the number of route found within every path discovery. In this final paper, using Network Simulator 2, the effectiveness with AOMDV routing protocol will be tested.

The test will be done using a connection traffic patterns and node movement pattern in which they will be simulated using a tcl routing protocol AOMDV script. This process will generate an output file in the form of a trace file. This particular trace file will be analyze to count for the PDR (Packet Delivery Ratio), RO (Routing Overhead), and End-to-End Delay.

The result is that the value of the PDR is declining based on the difference of the node's speed. For the RO and End-to-End Delay the value is increasing based on the same node's speed. The speed of the node is highly affecting the effectiveness of AOMDV routing protocol. A higher speed means a worse performance for the routing protocol.

Keywords: MANET, AODV, AOMDV, NS-2.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

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

Perkembangan teknologi informasi dan komunikasi dewasa ini berkembang dengan pesat. Perkembangan ini ditunjukkan dengan terciptanya berbagai macam teknologi yang membantu meningkatkan produktivitas manusia. Salah satu teknologi yang memudahkan manusia untuk saling berkomunikasi adalah *Mobile Ad hoc Network* (MANET). MANET memungkinkan terjadinya komunikasi jaringan tanpa bergantung pada ketersedian infrastruktur jaringan yang tetap. Jaringan MANET merupakan jaringan *wireless* yang terdiri dari kumpulan *node-node* yang bergerak yang memungkinkan untuk melakukan komunikasi secara langsung.

Dalam teknologi MANET tidak bisa dilepaskan dengan proses pengiriman paket data. Proses pencarian rute untuk mengirimkan data dari sumber ke tujuan disebut dengan *routing*. Dalam proses *routing*, data yang berasal dari *node* sumber dikirimkan ke *node-node* lain hingga mencapai *node* tujuan. Terdapat beberapa metode *routing* pada jaringan MANET salah satunya *Ad hoc On-Demand Distance Vector* (AODV). Protokol *routing* AODV bersifat reaktif yang artinya hanya melakukan proses pembentukan rute pada saat dibutuhkan. Setiap kali pencarian rute, AODV hanya menemukan satu pilihan rute atau *unipath*. Apabila terjadi kegagalan rute, maka AODV akan mengulang proses pencarian rute. Untuk mengatasi hal tersebut, dibutuhkan protokol *routing* yang memiliki beberapa pilihan rute atau *multipath* dalam setiap pencarian rute. Oleh karena itu,

protokol *routing* AODV dikembangkan menjadi protokol *Ad hoc On-Demand Multipath Distance Vector* (AOMDV).

Dalam Tugas Akhir ini, akan dilakukan studi kinerja terhadap protokol *routing* AOMDV pada topologi jaringan MANET berdasarkan *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay*.

1.2. Rumusan Masalah

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

1. Bagaimana melakukan simulasi dan mengintegrasikan protokol *routing* AOMDV pada jaringan MANET dengan menggunakan NS-2?
2. Bagaimana kinerja protokol *routing* AOMDV berdasarkan parameter berupa *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO) dan *End-to-End Delay*?

1.3. Batasan Masalah

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

1. Protokol *routing* yang digunakan adalah AOMDV.
2. Skenario uji coba dijalankan pada topologi jaringan MANET.
3. Simulasi pengujian protokol *routing* menggunakan NS-2.
4. Analisis kinerja didasarkan pada *Packet Delivery Ratio* (PDR) , *Routing Overhead* (RO), dan *End-to-End Delay*.

1.4. Tujuan dan Manfaat

Tujuan dari Tugas Akhir ini adalah menganalisa kinerja protokol *routing* AOMDV pada jaringan MANET dengan menggunakan NS-2.

1.5. Metodologi

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

1. Penyusunan proposal Tugas Akhir

   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Pada proposal

`

tersebut dijelaskan secara garis besar tentang alur pembuatan sistem.

2. Studi literatur

   Pada tahap ini dilakukan studi literatur mengenai *tools* dan metode yang digunakan. Literatur yang dipelajari dan digunakan meliputi buku referensi, artikel, jurnal dan dokumentasi dari internet.

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 *node-node* pada jaringan MANET yang dibuat menggunakan *traffic-scenario generator* dan *node-movement 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 *node-node* pada jaringan MANET yang dibuat.

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.

`

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) merupakan sebuah jaringan yang terbentuk dari beberapa *node* yang bergerak bebas dan dinamis. MANET memungkinkan terjadinya komunikasi jaringan tanpa bergantung pada ketersedian infrastruktur jaringan yang tetap. Setiap *node* dalam jaringan MANET dapat bertindak sebagai *host* dan *router*. Setiap *node* dapat saling melakukan komunikasi antara yang satu dengan yang lainnya tanpa adanya *access point*.

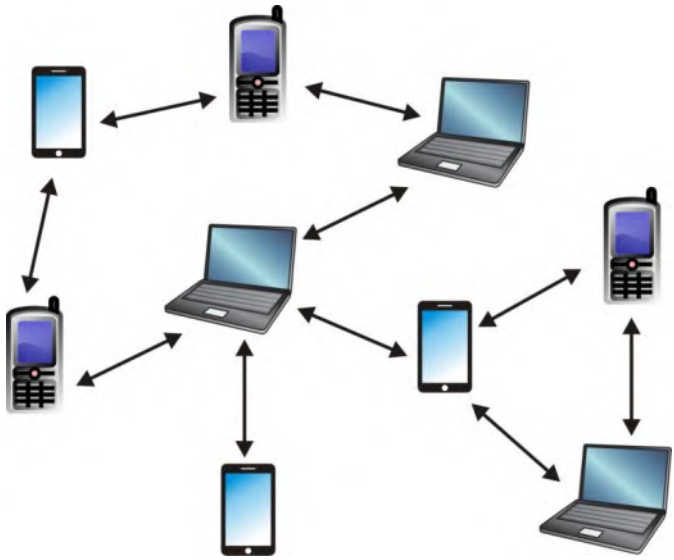
Perangkat di jaringan MANET harus mampu mendeteksi keberadaan perangkat lain dan melakukan pengaturan yang diperlukan untuk melakukan komunikasi dan berbagi data. Pada MANET memungkinkan perangkat untuk mempertahankan koneksi ke jaringan serta dengan mudah menambahkan dan menghapus perangkat pada jaringan. Karena pergerakan *node* yang dinamis, topologi jaringan dapat berubah dengan cepat dan tak terduga dari waktu ke waktu. Jaringan MANET bersifat desentralisasi, di mana organisasi jaringan dan pengiriman pesan harus dijalankan oleh *node* sendiri. [1]

MANET memiliki beberapa karakteristik yaitu :

1. Topologi yang dinamis : *Node* pada MANET dapat bergerak secara bebas dan berpindah-pindah kemana saja. Topologi jaringan yang bisanya *multihop* dapat berubah secara acak dan tidak terpola.
2. Keterbatasan *bandwidth* : Link pada jaringan nirkabel memiliki kapasitas rendah daripada jaringan kabel. Selain itu,

throughput komunikasi nirkabel seringkali lebih rendah dari tingkat transmisi maksimum radio. Hal ini dapat menyebabkan terjadinya *congestion* (kemacetan).

3. Keterbatasan energi : Karena bersifat *mobile*, semua node pada MANET dapat dipastikan sangat mengandalkan baterai sebagai sumber energi. Sehingga diperlukan desain sistem untuk optimasi energi.
4. Keterbatasan Keamanan : Jaringan nirkabel pada umumnya sangat rentan terhadap ancaman keamanan. Beberapa ancaman seperti *eavesdroping*, *spoofing* dan *denial of service* harus lebih diperhatikan dengan cermat. [2]



Gambar 2.1 Jaringan MANET

2.2. Ad Hoc On-Demand Multipath Distance Vector (AOMDV)

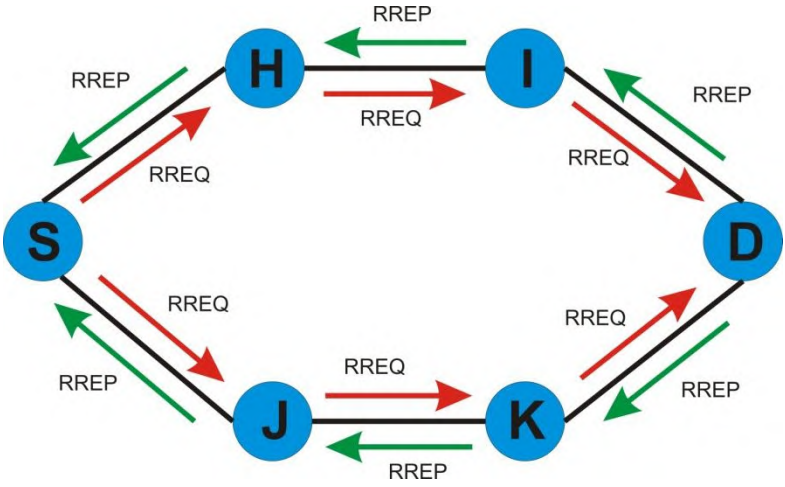
Protokol *routing* Ad Hoc On-Demand Multipath Distance Vector (AOMDV) [3] memiliki banyak persamaan karakteristik dengan Protokol *routing* AODV. AOMDV memiliki konsep bebasis vektor dan menggunakan pendekatan *hop by hop*.

AOMDV juga memiliki dua fitur utama yang mirip dengan AODV yaitu *route discovery* dan *route maintenance*. Perbedaan utama antara AODV dan AOMDV adalah jumlah rute yang ditemukan di setiap pencarian rute atau *route discovery.*

2.2.1. *Route Discovery*

Ketika *source node* memerlukan rute untuk melakukan komunikasi dengan *destination node*, maka *source node* akan mengirimkan paket *route request* (RREQ) secara *broadcast* ke *node-node* tetangga di dalam jaringan dan menunggu *route reply* (RREP). Berbeda dengan AODV, AOMDV menerima semua salinan paket RREQ untuk membuat *reverse path*.

Ketika *intermediate node* menerima paket RREQ, *node* ini akan mengecek apakah ada satu atau lebih *forward path* ke *destination* yang valid. Jika ada, *node* ini akan membuat paket RREP dan mengirim kembali ke *source node* melalui *reverse path.* Jika tidak ada, maka *intermediate node* akan meneruskan paket RREQ hingga ke *destination node,* kemudian *destination node* akan membalas dengan paket RREP.



Gambar 2.2 *Route discovery* pada AOMDV

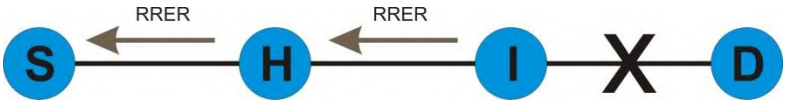
Proses *route discovery* pada AOMDV ditunjukkan Pada Gambar 2.2. Ketika *node* S (*source node*) hendak melakukan komunikasi terhadap *node* D (*destination node*), pertama kali *node* S mengirimkan paket RREQ secara *broadcast* ke *node-node* tetangganya yaitu ke *node* H dan node J. Apabila *node-node* tersebut dalam hal ini *node* H dan *node* J bukan merupakan *destination node*, maka *node* tersebut akan meneruskan paket RREQ secara *broadcast* ke *node* tetangganya tetapi tidak ke *source node*. Selain itu, *node* H dan *node* J akan melakukan *set up reverse path*. Proses tersebut berulang hingga paket RREQ tersebut diterima oleh *destination node* yatiu *node* D. Selanjutnya, *node* D akan mengiirmkan paket RREP sebagai balasan dari paket RREQ yang diterimanya. *Node-node* penerima paket RREP akan melakukan *set-up forward path*.

Pada AODV, apabila *destination node* menerima paket RREQ ganda, maka paket RREQ yang terakhir diterima *destination node* akan di-*drop*. Berbeda dengan AODV, AOMDV menerima semua salian paket RREQ dan akan digunakan untuk membentuk rute cadangan dari *source node* ke *destination node*. Dengan demikian, AOMDV akan memiliki lebih dari satu *path* (*multipath*) sehingga apabila terjadi kerusakan rute atau rute pengiriman putus saat proses pengiriman, *source node* tidak perlu lagi melakukan proses *route discovery* seperti pada AODV karena *source node* akan menggunakan rute cadangan yang telah dibentuk saat *route discovery*.

2.2.2. *Route Maintenance*

Route maintenance pada AOMDV adalah pengembangan sederhana yang ada pada *route maintenance* AODV. AOMDV juga menggunakan peket *route error* (RERR) untuk mengirimkan pesan *error*. Sebuah *node* akan mengirimkan paket RERR apabaila *path* ke *destination node* rusak. AOMDV memiliki beberapa rute untuk menjaga agar komunikasi tetap berlangsung yaitu ketika sebuah *node* menemui *link* yang rusak maka *node* tersebut segera memilih rute cadangan.

`



Gambar 2.3 Paket *RRER* pada AOMDV

Proses pengiriman paket RRER pada AOMDV ditunjukkan pada Gambar 2.3. Pada saat terjadi kerusakan *link* yang menghubungkan antara *node* I dengan *node* D, maka *node* I akan mengirimkan paket RRER ke *node* H. Seterusnya, *node* H akan mengirimkan paket RRER ke *node* S. Setelah *node* S menerima paket RRER, proses komunikasi akan dilakukan kembali dengan menggunakan rute cadangan apabila masih tersedia. Apabila tidak tersedia rute cadangan, maka *node* S akan melakukan proses *route discovery*.

2.3. VirtualBox

VirtualBox adalah aplikasi virtualisasi *cross-platform* yang bersifat *open source*. Disebut aplikasi *cross-platform* karena digunakan untuk meng-*install* sistem operasi di dalam sistem operasi utama pada sebuah perangkat komputer. Dengan menggunakan VirtualBox, sebuah perangkat komputer dapat menjalankan beberapa sistem operasi dalam waktu yang sama. Misalnya, pengguna dapat menjalankan Windows dan Linux di MAC, menjalankan Windows Server 2008 di server Linux, menjalankan Linux di komputer dengan sistem operasi Windows, dan sebagainya.

VirtualBox didesain untuk para profesional dan pengembang di bidang Teknologi Informasi. VirtualBox dapat berjalan pada sistem operasi Windows, Mac OS X, Linux dan Oracle Solaris yang sangat ideal diaplikasikan untuk pengujian, pengembangan, dan simulasi berbagai sistem operasi pada satu mesin. [4]

Pada Tugas Akhir ini, VirtualBox digunakan untuk menjalankan sistem operasi Linux yang berisi *software* NS-2 dan juga sebagai penghubung antara sistem operasi Windows dan Linux.

2.4. Network Simulator 2 (NS-2)

Network Simulator 2 atau biasa disebut NS-2 adalah *software* simulasi yang bersifat *open source* yang dirancang khusus untuk penelitian dalam jaringan komunikasi komputer. NS-2 dikembangkan mengguanakan bahasa pemrograman C ++ dan OTcl. Simulasi kabel maupun nirkabel dan protokol (misal algoritma *routing*, TCP, UDP) dapat disimulasikan dengan menggunakan NS-2.

NS-2 merupakan *software* simulasi jaringan yang paling banyak digunakan dalam penelitian jaringan komputer. Dengan bahasa *script* yang sederhana, sangat memudahkan peneliti untuk melakukan konfigurasi jaringan dan mengamati hasil simulasi dari NS-2. NS-2 telah mendapatkan apresiasi dari kalangan industri, akademisi, dan pemerintah sejak diperkenalkan awal tahun 1989. Berawal dari pengembangan simulator jaringan nyata oleh University of California Berkeley yang merupakan cikal bakal lahirnya Network Simulator. Pada tahun 1995 Defense Advanced Research Projects Agency (DARPA) membantu pengembangan Network Simulator melalui proyek Virtual Internetwork Terstbed (VINT). Sampai saat ini, para peneliti dan pengembang terus bekerja untuk menjadikan NS-2 lebih baik. [5] Pada Tugas Akhir ini digunakan NS-2 versi 2.35 untuk simulasi protokol *routing* AOMDV.

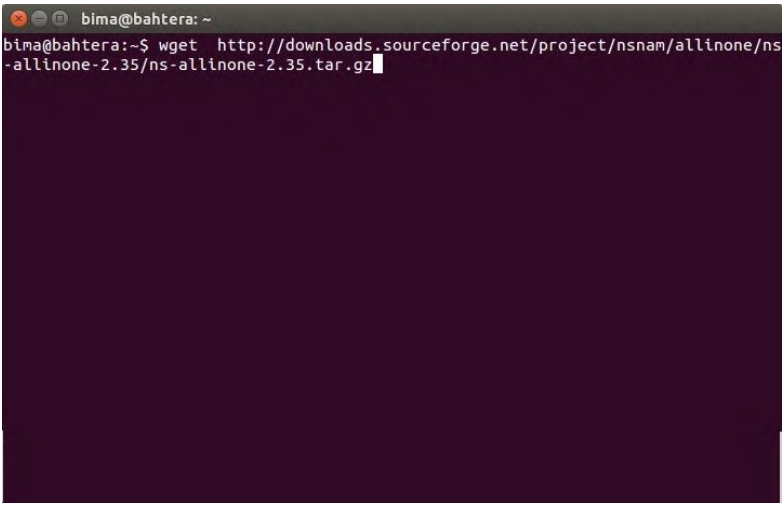
Untuk menggunakan NS-2, terlebih dahulu dilakukan proses instalasi. Proses instalasi NS-2 dimulai dengan melakukan instalasi modul-modul dependensi dari NS-2 yaitu build-essential, autoconf, automake , libx11-dev, libxmu-dev dan gcc-4.4 seperti pada Gambar 2.4.

```
$ sudo apt-get install build-essential autoconf
automake libx11-dev libxmu-dev gcc-4.4
```

Gambar 2.4 Baris perintah untuk instalasi dependensi NS-2

Setelah semua dependensi lengkap, dilakukan unduh modul seperti pada Gambar 2.5 dan dilakukan proses ektraksi file NS-2 seperti yang telah diunduh pada Gambar 2.6.

`



Gambar 2.5 Baris perintah untuk mengunduh NS-2

```
$ tar zxvf ns-allinone-2.35.tar.gz
```

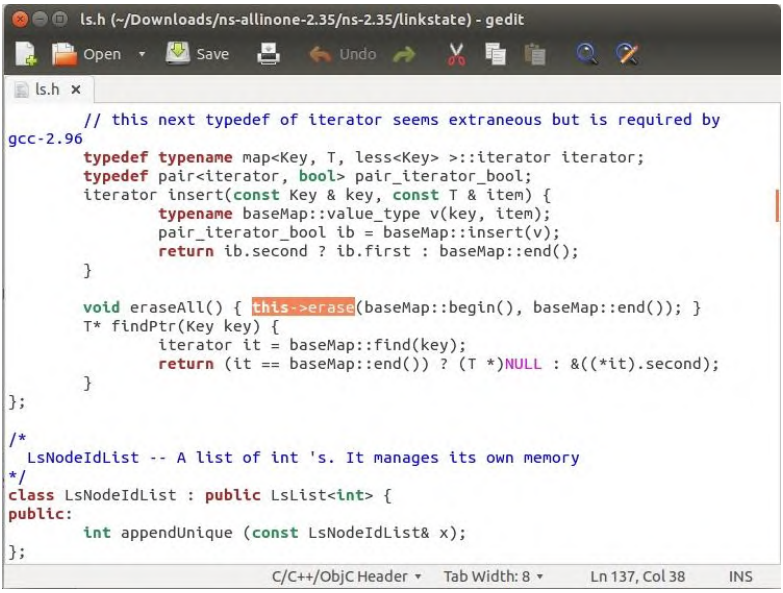
Gambar 2.6 Baris perintah ekstraksi file NS-2

Kemudian dilakukan proses pengubahan *script* pada ls.h yang terdapat pada *folder* /ns-allinone-2.35/ns-2.35/linkstate/ls.h yang ditunjukkan pada Gambar 2.7.

```
$ cd /ns-allinone-2.35/ns-2.35/linkstate/
$ gedit ls.h
```

Gambar 2.7 Baris perintah pengubahan ls.h

Pada *line* ke 137, *erase* diubah menjadi *this->erase* karena jika tidak diubah, akan terjadi kegagalan saat proses instalasi NS-2. Kegagalan instalasi bisa terjadi pada beberapa sistem operasi seperti Fedora, Ubuntu atau Linux Mint. Proses tersebut yang dapat dilihat pada Gambar 2.8.



Gambar 2.8 Proses pengubahan *line of code* pada ls.h

Setelah itu, dilakukan proses instalasi dari NS-2 seperti pada Gambar 2.9.

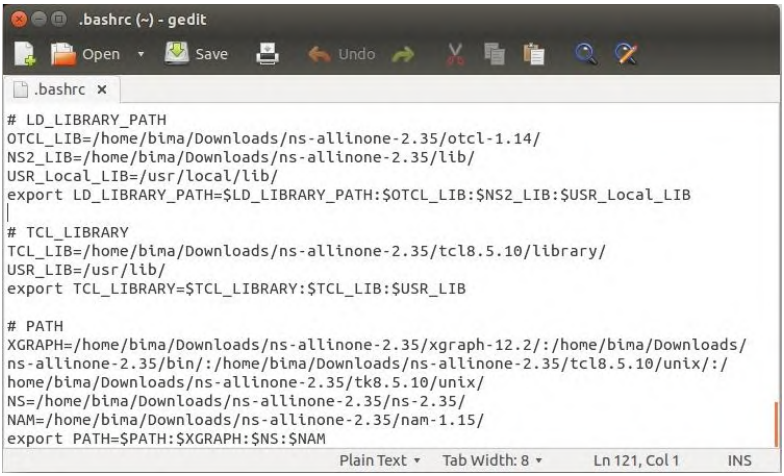
```
$ sudo ./install
```

Gambar 2.9 Baris perintah untuk instalasi NS-2

Setelah proses instalasi selesai, langkah selanjutnya yaitu melakukan pengaturan *path* pada pada file ~/.bashrc agar NS-2 dapat dijalankan dengan baris perintah seperti pada Gambar 2.10. Kemudian menambahkan baris seperti yang ditunjukkan pada Gambar 2.11. Baris tersebut diletakkan di baris terakhir *file* ~/.bashrc.

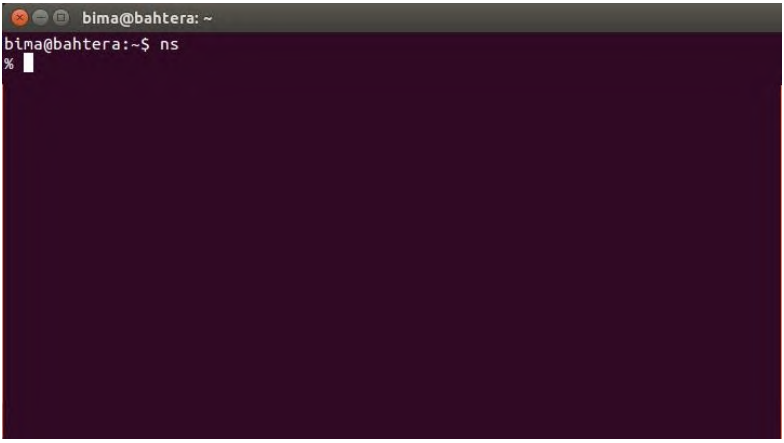
```
$ gedit ~/.bashrc
```

Gambar 2.10 Baris perintah pengaturan ~/.bashrc



Gambar 2.11 Pengaturan ~/.bashrc

Sampai langkah ini, proses instalasi NS-2 sudah selesai dilakukan. Untuk melakukan pengecekan apakah NS-2 telah ter-*install* dengan baik, maka dapat dilakukan dengan mengetikkan baris perintah ‘ns’ pada terminal dan apabila muncul tanda ‘%’ pada terminal berarti NS-2 telah ter-*install* dengan baik seperti pada Gambar 2.12.



Gambar 2.12 Baris perintah pengecekan NS-2

2.5. *Traffic-Scenario Generator*

Traffic-scenario generator [6] merupakan *script* yang digunakan untuk membuat pola *traffic* koneksi antara *node* secara acak. *Script* tersebut digunakan pula untuk menentukan jenis *traffic* koneksi yang akan dibuat yaitu CBR atau TCP. *Traffic-Scenario Generator Script* ini terletak di direktori "~ns/indep-utils/cmu-scen-gen" dengan nama *file* cbrgen.tcl. Untuk menjalankan *script* tersebut, diperlukan baris perintah seperti pada Gambar 2.13.

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

Gambar 2.13 Format baris perintah membuat *traffic* koneksi

Contoh baris perintah untuk menjalankan cbrgen.tcl seperti pada gambar 2.14.

```
ns cbrgen.tcl -type cbr -nn 50 -seed 1.0 -mc 1
-rate 1.0 > cbr-50-test
```

Gambar 2.14 Contoh baris perintah membuat *traffic* koneksi

Parameter yang digunakan pada baris perintah pada Gambar 2.13 dijelaskan pada Tabel 2.1.

Tabel 2.1 Parameter *traffic* koneksi

No.	Parameter	Keterangan
1	-type	Jenis *traffic* yang digunakan yaitu TCP atau CBR
2	-nn	Jumlah *node*
3	-s	Jumlah *seed*
4	-mc	Jumlah koneksi
5	-rate	Jumlah paket per detik

2.6. *Node-Movement Generator*

Node-movement generator [6] merupakan *tool* digunakan untuk menghasilkan pergerakan *node-node* secara acak dalam jaringan nirkabel. Selain itu, *Node-movement generator* berguna

untuk mendefinisikan pergerkaan node dengan kecepatan gerak yang spesifik menuju suatu lokasi acak atau lokasi spesifik yang berada dalam kawasan yang telah ditentukan. Dengan *tool* ini, Sebuah *node* dapat diatur untuk berhenti sementara waktu ketika tiba di lokasi pergerakan dan kemudian *node* bergerak menuju lokasi berikutnya. *Tool* tersebut terletak di direktori "~ns/indep-utils/cmu-scen-gen/setdest" dengan nama *file* setdest. Untuk menjalan setdest, diperlukan baris perintah seperti pada Gambar 2.15.

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

Gambar 2.15 Format baris perintah membuat *node-movement*

Contoh baris perintah untuk menjalankan setdest seperti pada gambar 2.16.

```
./setdest -n 50 -p 2 -M 5 -t 500 -x 800 -y
800 > scen-50-1
```

Gambar 2.16 Contoh baris perintah membuat *node-movement*

Parameter yang digunakan pada baris perintah pada Gambar 2.15 dijelaskan pada Tabel 2.2.

Tabel 2.2 Parameter *node-movement*

No.	Parameter	Keterangan
1	-n	Jumlah *node*
2	-p	Durasi ketika sebuah *node* tetap diam setelah tiba di lokasi pergerakan. Jika nilai ini diatur menjadi 0, maka *node* tidak akan berhenti ketika tiba di lokasi pergerakan dan akan terus bergerak
3	-M	Kecepatan maksimal sebuah *node*. *Node* akan bergerak pada kecepatan acak dalam rentang [0, *maxspeed*]

No.	Parameter	Keterangan
4	-t	Waktu simulasi
5	-x	Panjang maksimal area simulasi
6	-y	Lebar maksimal area simulasi

2.7. NS-2 *Trace File*

NS-2 *Trace File* adalah *file* berekstensi .tr yang merupakan *output* hasil *running* NS-2. *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 dari sebuah protokol *routing.* Dari *Trace File* inilah yang nantinya akan dihitung *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay*. Contoh *trace* pengiriman data paket pada NS-2 dapat dilihat pada Gambar 2.17.

```
s 23.432869857 _1_ AGT  --- 21 cbr 512 [0 0 0 0] --
----- [1:0 2:0 32 0] [21] 0 2
```

Gambar 2.17 *Trace* pengiriman paket data

Huruf "s" di kolom pertama menandakan pengiriman paket (*send*), kolom kedua berisi waktu pengiriman paket pada detik ke 23, 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 512.

Untuk penerimaan data paket, hampir sama dengan pengiriman data paket, yang membedakan adalah kolom pertama menggunakan huruf "r" yang menandakan bahwa paket diterima (*recieve*) dan format selanjutnya sama dengan pengiriman paket. Contoh *trace* penerimaan data paket pada NS-2 dapat dilihat pada Gambar 2.18.

`

```
r 17.975729166 _1_ RTR  --- 16 cbr 512 [0 0 0 0] --
----- [1:0 2:0 32 0] [16] 0 2
```

Gambar 2.18 *Trace* penerimaan paket data

Contoh format untuk paket protokol *routing* AOMDV pada *trace file* seperti pada Gambar 2.19.

```
r 23.991580771 _38_ RTR  --- 0 AOMDV 44 [0 ffffffff
1b 800] ------- [27:255 -1:255 1 0] [0x1 0 [27 2]
4.000000] (HELLO) [0 0]
```

Gambar 2.19 *Trace* pengiriman paket *routing* AOMDV

2.8. Awk

Sering kali banyak tugas yang akan dikerjakan oleh seseorang ketika berhadapan dengan data berupa teks. Proses penyaringan data antara data yang akan digunakan dengan data yang akan dibuang, sering kali membutuhkan proses yang cukup kompleks. Pekerjaan seperti ini, dapat dengan mudah diselesaikan dengan menggunakan awk. Awk adalah sebuah bahasa pemrograman yang digunakan untuk mengelola data berupa teks. Nama awk berasal dari inisial nama pengembangnya yaitu Alfred V. Aho, Peter J. Weinberger, dan Brian W. Kernighan.

Beberapa kegunaan awk adalah sebagai berikut:

- Mengelola database
- Membuat laporan
- Memvalidasi Data
- Menghasilkan indeks dan menampilkan dokumen
- Percobaan membuat algoritma yang digunakan untuk mengubah bahasa komputer ke bahasa lainnya.

Selain itu, awk menyediakan fasilitas yang memudahkan untuk:

- Mengekstrak data untuk diproses
- Mengurutkan data
- Menampilkan komunikasi jaringan yang sederhana. [7]

Fungsi dasar awk adalah untuk mencari *file* per baris (atau unit teks lain) yang berisi pola tertentu. Ketika suatu baris sesuai dengan pola, awk melakukan aksi yang khusus pada baris

tersebut. Awk tetap memproses baris input sedemikian hingga mencapai akhir baris input.

Program pada awk berbeda dari program di kebanyakan bahasa lain, karena program awk bersifat "*data-driven"*; yang mana kamu diminta untuk mendeskripsikan data yang dikehendaki untuk bekerja dan kemudian apa yang akan dilakukan saat data tersebut ditemukan. Kebanyakan bahasa lainnya bersifat "*procedural"*; kamu harus mendeskripsikannya secara detail setiap langkah program yang harus dijalankan. Ketika bekerja dengan bahasa prosedural, biasanya sangat sulit untuk mendeskripsikan data yang hendak diproses oleh program. Oleh karena itu, program awk sering kali terasa lebih mudah untuk ditulis dan dibaca. [8]

Pada Tugas Akhir ini, awk digunakan untuk membuat *script* untuk menghitung *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay* dari hasil *trace* NS-2.

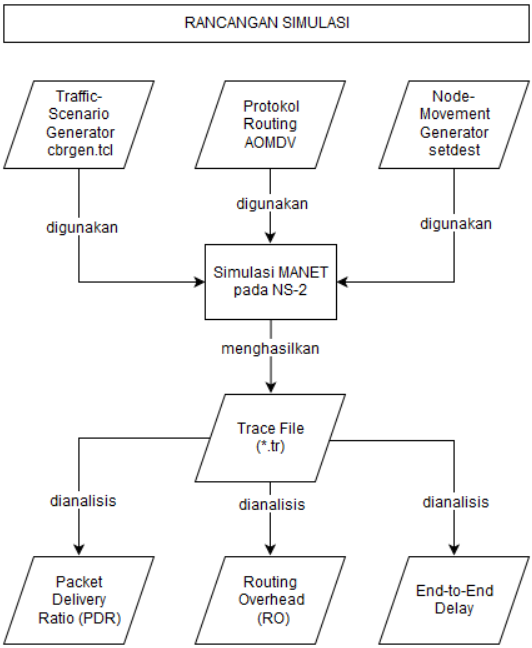
`

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 kinerja terhadap protokol *routing* AOMDV pada jaringan MANET. Ilustrasi rancangan simulasi dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram rancangan simulasi

Dalam pengujian ini, digunakan 50 *node* untuk simulasi protokol *routing* AOMDV. Untuk kecepatan maksimal pergerakan *node* dibuat bervariasi yaitu 5 m/s, 10 m/s, dan 15 m/s. Hasil dari proses uji coba ini yaitu berupa *trace file* yang nantinya akan digunakan untuk menghitung *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO) dan *End-to-End Delay*.

3.2. Perancangan *Shared Folder* antara Sistem Operasi Windows dan VirtualBox

Untuk memudahkan pertukaran data lintas sistem operasi antara sistem operasi Windows dengan sistem operasi Linux pada lingkungan VirtualBox, maka pada perlu dirancang *folder sharing* sebagai penghubung antara sistem operasi Windows dengan sistem operasi Linux pada lingkungan VirtualBox.

3.3. Perancangan Skenario

Perancangan skenario uji coba mobilitas diawali dengan membuat pola *traffic* koneksi antara *node* secara acak menggunakan *Traffic-Scenario Generator*. Kemudian membuat pola pergerakan *node-node* secara acak menggunakan *Node-Movement Generator*. Pada Tugas Akhir ini, pergerakan *node-node* menggunakan 3 variasi kecepatan maksimal yaitu 5 m/s, 10 m/s, dan 15 m/s.

3.3.1. Menentukan Pola *Traffic* Koneksi

Pola *Traffic* koneksi dibuat dengan menjalankan *script* cbgren.tcl yang nantinya akan menghasilkan *output* berupa *file*. *File* tersebut digunakan untuk mengatur pola *traffic* koneksi secara acak antara *node* dalam simulasi. Parameter yang digunakan untuk membuat pola *traffic* koneksi dapat dilihat pada Tabel 3.1.

Tabel 3.1 Parameter pola *traffic* koneksi

No.	Parameter	Spesifikasi
1	Jenis *traffic* (-type)	CBR
2	Jumlah *node* (-nn)	50
3	Jumlah *seed* (-s)	1
4	Jumlah koneksi (-mc)	1

`

No.	Parameter	Spesifikasi
5	Jumlah paket per detik (-rate)	1

3.3.2. Menentukan Pola Pergerakan *Node*

Pola Pergerakan *Node* dibuat dengan menggunkan *Node-Movement Generator* yang nantinya akan menghasilkan *output* berupa *file*. *File* tersebut digunakan untuk mengatur *node-node* agar dapat bergerak secara acak dalam simulasi. Parameter yang digunakan untuk membuat pola pergerakan *node* dapat dilihat pada Tabel 3.2.

Tabel 3.2 Parameter pola pergerakan *node*

No.	Parameter	Spesifikasi
1	Jumlah node (-n)	50
2	Waktu jeda (-p)	2 s
3	Kecepatan maksimal *node* (-M)	- 5 m/s - 10 m/s - 15 m/s
4	Waktu simulasi (-t)	500 s
5	Panjang maksimal area simulasi (-x)	800 m
6	Lebar maksimal area simulasi (-y)	800 m

3.4. Perancangan Simulasi pada NS-2

Pada perancangan kode NS-2 dengan konfigurasi MANET, dilakukan penggabungan skenario pola *traffic* koneksi dan pola pergerakan *node* dengan skrip TCL yang diberikan parameter-parameter untuk membangun percobaan simulasi MANET pada NS-2. Berikut parameter simulasi perancangan sistem MANET yang dapat digunakan dapat dilihat pada Tabel 3.3.

Tabel 3.3 Parameter simulasi

No.	Parameter	Spesifikasi
1	Network simulator	NS-2 versi 2.35
2	Protokol *routing*	AOMDV
3	Waktu simulasi	500 s

No.	Parameter	Spesifikasi
4	Area simulasi	800 m x 800 m
5	Jumlah *node*	50
6	Radius transmisi	100 m
7	*Source / destination*	Statik (*node* 1 / *node* 2)
8	Protokol MAC	IEEE 802.11
9	Model propagasi	Two-ray ground propagation model
10	Tipe antena	OmniAntenna
11	Tipe Interface Queue	Droptail/PriQueue
12	*Mobility model*	*Random Waypoint*
13	Tipe kanal	*Wireless channel*
14	Tipe *trace*	*Old Wireless Format Trace*

3.5. Perancangan Metrik Analisis

Metrik yang akan dianalisis pada Tugas Akhir ini adalah *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay*. Penjelasannya sebagai berikut:

3.5.1. *Packet Delivery Ratio* (PDR)

PDR dihitung dari perbandingan antara paket yang dikirim dengan paket yang diterima. PDR dihitung dengan menggunakan persamaan (3.1), dimana *received* adalah banyaknya paket data yang diterima dan *sent* adalah banyaknya paket data yang dikirimkan.

$$PDR = \frac{\sum received}{\sum sent} \times 100 \quad (3.1)$$

3.5.2. *Routing Overhead* (RO)

Routing Overhead (RO) 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 RO pada *trace file* ditandai dengan paket yang bertipe *send* (s) / *forward* (f) dan terdapat *header* paket dari protokol AOMDV.

`

3.5.3. *End-to-End Delay*

End-to-End Delay dihitung dari rata-rata delay antara waktu paket diterima dan waktu paket dikirim. *End-to-End Delay* dihitung dengan menggunakan persamaan (3.2), 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.

$$End\ to\ End\ Delay = \frac{\sum_{i \leq sent}^{i=0} t_{received[i]} - t_{sent[i]}}{sent} \quad (3.2)$$

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 Ubuntu 14.04 32-bit untuk lingkungan NS-2,
- VirtualBox untuk pengembangan aplikasi NS-2 dan juga digunakan untuk analisis, dan
- Network Simulator 2 versi 2.35.

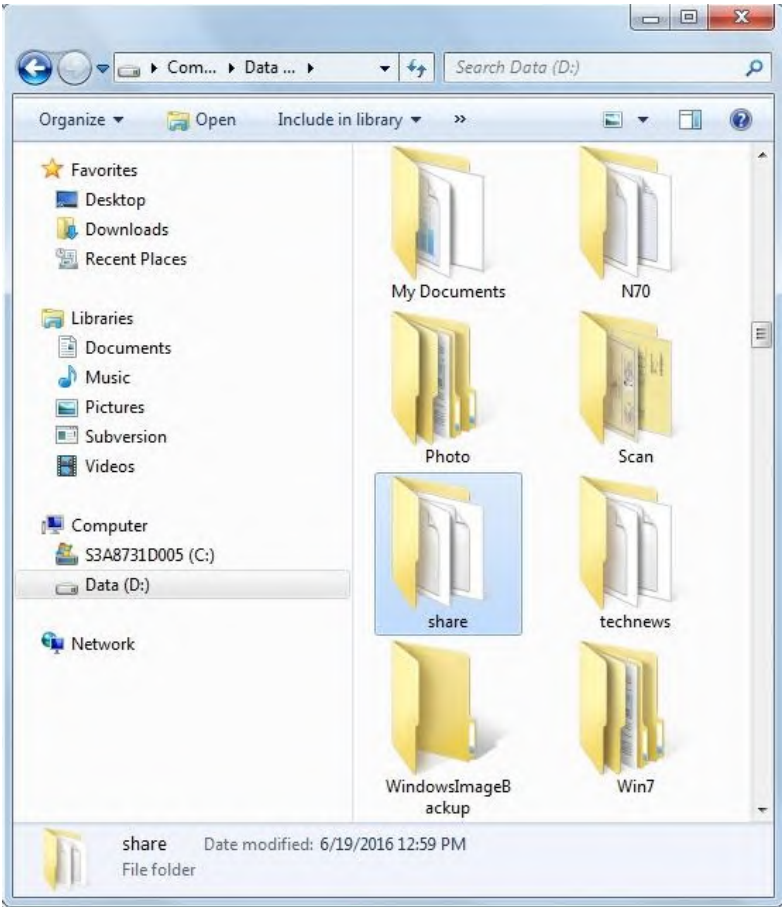
4.1.2. Lingkungan Perangkat Keras

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

- *Processor* Intel(R) Core(TM) i3 CPU @ 2.27GHz,
- Media penyimpanan sebesar 300GB, dan
- RAM sebesar 2 GB DDR3.

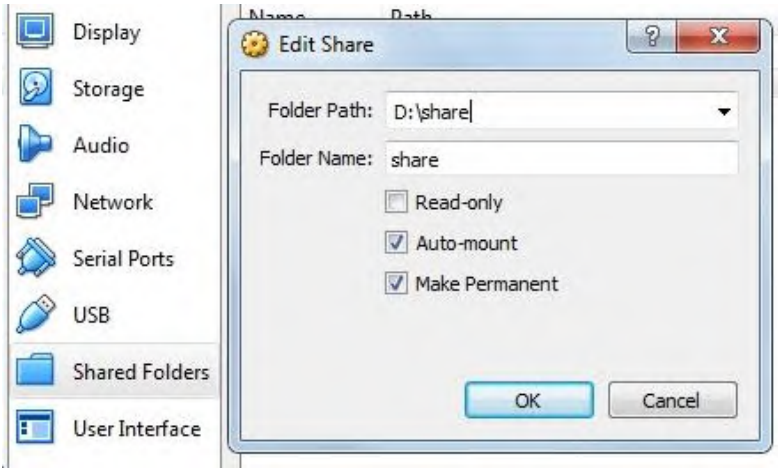
4.2. Implementasi *Shared Folder* antara Sistem Operasi Windows dan VirtualBox

Implementasi pembuatan *shared folder* antara sistem operasi Windows dan sistem operasi Ubuntu pada VirtualBox dimulai dengan membuat *folder* bernama share pada salah satu direktori di sistem operasi Windows misalnya di direktori C: atau D: seperti pada Gambar 4.1.



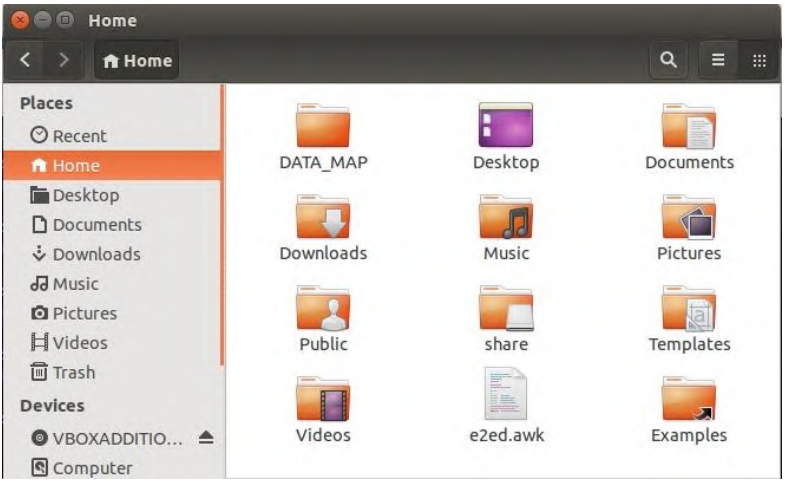
Gambar 4.1 Pembuatan *folder* share pada sistem operasi Windows

Kemudian tunjuk *folder* tersebut pada konfigurasi *shared folder* di VirtualBox. Dalam proses ini sebagai contoh untuk *folder path* yang ditunjuk adalah D:/share, kemudian dicentang pada pilihan *Auto-mount* dan *Make Permanent* seperti pada Gambar 4.2.

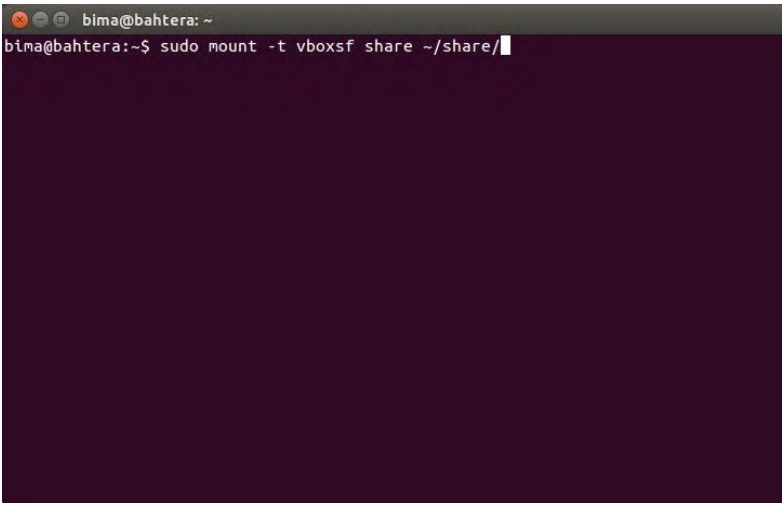


Gambar 4.2 Konfigurasi *shared folder* pada VirtualBox

Kemudian membuat *folder* share pada direktori home di sistem operasi Linux seperti pada Gambar 4.3. Setelah itu dilakukan *mount* pada *shared folder* dengan perintah seperti pada Gambar 4.4.



Gambar 4.3 Pembuatan *folder* share pada sistem operasi Linux



Gambar 4.4 Baris perintah untuk melakukan *mount shared folder*

4.3. Implementasi Pola *Traffic* Koneksi

Untuk menghasilkan pola *traffic* koneksi, digunakan *script* cbrgen.tcl. Baris perintah *script* cbrgen.tcl dapat dilihat pada Gambar 4.5.

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

Gambar 4.5 Baris perintah untuk membuat *traffic* koneksi

Setelah baris perintah pada Gambar 4.5 dijalankan, maka akan menghasilkan *output file*. Isi dari *output file* hasil dari *generate script* cbrgen.tcl ditunjukkan pada Gambar 4.6.

```
#
# nodes: 50, max conn: 1, send rate: 1.0, seed: 1.0
#
#
# 1 connecting to 2 at time 2.5568388786897245
#
```

```
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(2) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 1.0
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 2.5568388786897245 "$cbr_(0) start"
#
#Total sources/connections: 1/1
#
```

Gambar 4.6 Isi *file traffic* koneksi

4.4. Implementasi Pola Pergerakan *Node*

Untuk menghasilkan pola pergerakan *node*, digunakan *Node-Movement Generator* yang bernama setdest. Pada Tugas Akhir ini, pergerakan *node-node* menggunakan tiga variasi kecepatan maksimal yaitu 5 m/s, 10 m/s, dan 15 m/s, dengan masing-masing dilakukan *generate* sebanyak 10 kali. Diawali dengan men-*generate* setdest untuk kecepatan maksimal *node* 5 m/s seperti pada Gambar 4.7, kemudian untuk kecepatan maksimal *node* 10 m/s seperti pada Gambar 4.8, dan yang terakhir untuk kecepatan maksimal *node* 15 m/s seperti Gambar 4.9.

```
./setdest –n 50 –p 2 –M 5 –t 500 –x 800 –y 800 >
scen50-5-1
```

Gambar 4.7 Baris perintah membuat pergerakan *node* dengan kecepatan maksimal 5 m/s

```
./setdest –n 50 –p 2 –M 10 –t 500 –x 800 –y 800 >
scen50-10-1
```

Gambar 4.8 Baris perintah membuat pergerakan *node* dengan kecepatan maksimal 10 m/s

```
./setdest -n 50 -p 2 -M 15 -t 500 -x 800 -y 800 >
scen50-15-1
```

Gambar 4.9 Baris perintah membuat pergerakan *node* dengan kecepatan maksimal 15 m/s

Setelah perintah setdest dijalankan, maka akan menghasilkan *file output*. Gambar 4.10 menunjukkan potongan kode yang terdapat di dalam *file output* sebagai perintah untuk menentukan posisi awal *node* pada koordinat X, Y, dan Z.

```
#
# nodes: 50, pause: 2.00, max speed: 5.00, max x:
800.00, max y: 800.00
#
$node_(0) set X_ 744.660715872755
$node_(0) set Y_ 278.690136123343
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 187.838953144500
$node_(1) set Y_ 563.172177118206
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 485.040583482115
$node_(2) set Y_ 327.219246155956
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 700.069465119030
$node_(3) set Y_ 699.714247682263
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 14.335704171524
$node_(4) set Y_ 545.276282498955
$node_(4) set Z_ 0.000000000000
$node_(5) set X_ 18.504273670001
$node_(5) set Y_ 268.311611704718
$node_(5) set Z_ 0.000000000000
$node_(6) set X_ 741.423826063088
$node_(6) set Y_ 170.475555271589
$node_(6) set Z_ 0.000000000000
$node_(7) set X_ 313.063614516806
$node_(7) set Y_ 478.127382381469
$node_(7) set Z_ 0.000000000000
$node_(8) set X_ 466.999200995973
$node_(8) set Y_ 257.935973641156
```

Gambar 4.10 Posisi awal *node*

```
$ns_ at 2.000000000000 "$node_(0) setdest
621.578610243358 698.659979487135 4.546762783017"
$ns_ at 2.000000000000 "$node_(1) setdest
375.868120179644 737.324967925342 2.569446721916"
$ns_ at 2.000000000000 "$node_(2) setdest
457.424604134708 623.400326275096 2.374029658688"
$ns_ at 2.000000000000 "$node_(3) setdest
704.280531241174 223.084742948792 1.483334069730"
$ns_ at 2.000000000000 "$node_(4) setdest
302.474417843548 302.706101667804 4.180816148783"
$ns_ at 2.000000000000 "$node_(5) setdest
416.953666770450 215.629079204716 0.812711950635"
$ns_ at 2.000000000000 "$node_(6) setdest
538.904622032345 238.455772027094 3.038381399117"
$ns_ at 2.000000000000 "$node_(7) setdest
298.344791972944 792.772313974947 2.747128683003"
$ns_ at 2.000000000000 "$node_(8) setdest
56.554309409879 672.615845665358 0.271355616032"
$ns_ at 2.000000000000 "$node_(9) setdest
369.287237713986 0.214583995900 4.004380333449"
```

Gambar 4.11 Pergerakan *node*

Pada Gambar 4.11 menunjukkan potongan kode sebagai perintah untuk mendefinisikan pergerakan *node*. Sebagai contoh *node* 0 pada waktu 2.0 detik mulai bergerak ke arah tujuan (621, 698) dengan kecepatan 4.54 m/s.

4.5. Implementasi Simulasi pada NS-2

Simulasi dilakukan dengan memanggil *script* protokol *routing* AOMDV. Dalam Tugas Akhir ini, *script* protokol *routing* AOMDV bernama AOMDV.tcl. Pada Gambar 4.12 menunjukkan potongan *script* konfigurasi awal parameter-parameter yang diberikan untuk menjalankan MANET pada NS-2. Baris pertama merupakan konfigurasi tipe *channel* yang digunakan yaitu *Wireless Channel*. Baris kedua merupakan tipe propagasi yang digunakan yaitu model TwoRayGround. Tipe Mac yang digunakan adalah Mac 802.11. Pada *script* tersebut juga dilakukan konfigurasi tipe *queue* dari *interface*, tipe *link layer*, tipe *antenna* dan jumlah maksimum *packet* pada *interface queue.* Selain itu juga dijelaskan pada baris berikutnya yaitu protokol

routing yang digunakan yaitu AOMDV, koordinat x serta koordinat y sebesar 800, nama file tr yaitu aomdv.tr, dan terakhir adalah jumlah node yang digunakan yaitu 50 *node.*

```
set val(chan)           Channel/WirelessChannel
set val(prop)           Propagation/TwoRayGround
set val(netif)          Phy/WirelessPhy
set val(mac)            Mac/802_11
set val(ifq)            Queue/DropTail/PriQueue
set val(ll)             LL
set val(ant)            Antenna/OmniAntenna
set val(ifqlen)         100
set val(rp)             AOMDV
set val(x)              800
set val(y)              800
set val(seed)           0.0
set val(tr)             aomdv.tr
set val(nn)             50
```

Gambar 4.12 Konfigurasi awal parameter NS-2

Pada Gambar 4.13 merupakan pengaturan dari *transmission range* yang digunakan pada simulasi. Nilai yang diubah ialah RXThresh_ (*Receiver Sensitivity Threshold*). Nilai 1.42681e-08 pada variabel tersebut menunjukkan bahwa *range* atau jangkauan yang dicapai ialah sejauh 100 meter.

```
Phy/WirelessPhy set RXThresh_ 1.42681e-08
```

Gambar 4.13 Konfigurasi *Transmission Range* pada NS-2

Sebelum menjalankan *script* AOMDV.tcl, terlebih dahulu melakukan pengeditan pada *script* AOMDV.tcl untuk memanggil *file* skenario *traffic* koneksi dan *file* skenario *node movement* agar simulasi dapat berjalan sesuai dengan parameter-parameter yang telah dirancang. Pada Gambar 4.14 merupakan potongan kode untuk memanggil *file* skenario *traffic* koneksi. Sedangkan pada Gambar 4.15 merupakan potongan kode untuk memanggil *file* skenario *node movement*.

```
set val(cp)      "cbr-50-test"
```

Gambar 4.14 Potongan kode untuk memanggil *file* skenario *traffic* koneksi

```
set val(sc)      "skenario/Scen50-5/Scen50-5-1"
```

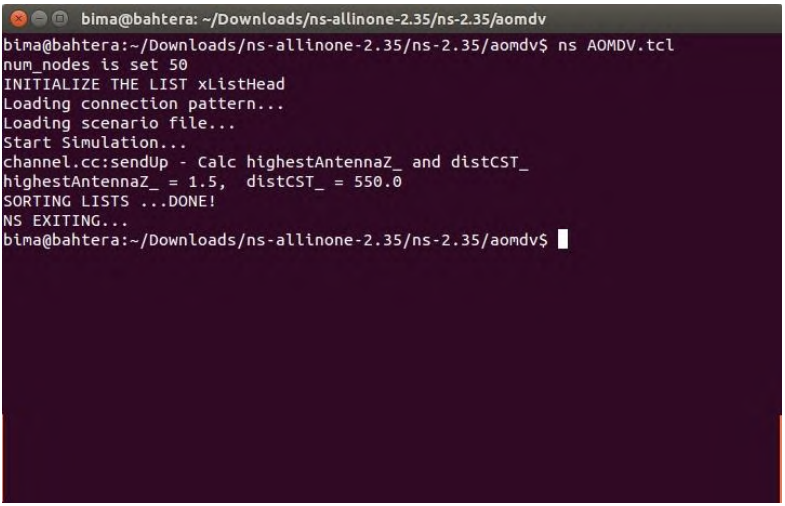
Gambar 4.15 Potongan kode untuk memanggil *file* skenario *node movement*

Untuk menjalankan *script* AOMDV.tcl menggunakan perintah seperti yang ditunjukkan pada Gambar 4.16.

```
ns AOMDV.tcl
```

Gambar 4.16 Baris perintah untuk menjalankan *script* AOMDV.tcl

Akhir dari eksekusi *script* AOMDV.tcl ditunjukkan pada Gambar 4.17.



Gambar 4.17 Hasil eksekusi *script* AOMDV.tcl

Hasil yang didapat setelah menjalankan script AOMDV.tcl yaitu berupa *trace file* berbentuk file dengan ekstensi .tr. *File* .tr inilah yang akan dianalisis parameter-parameternya berupa

Packet Delivery Ratio (PDR), *Routing Overhead* (RO) dan *End-to-End Delay*.

4.6. Implementasi Metrik Analisis

Hasil menjalankan skenario MANET dalam NS-2 dalam bentuk *Trace File* berekstensi .tr dianalisis dengan tiga metrik yaitu *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay* . Implementasi dari tiap metrik menggunakan bahasa pemrograman AWK dan dijelaskan seperti berikut.

4.6.1. *Packet Delivery Ratio* (PDR)

Proses perhitungan PDR dilakukan dengan menghitung jumlah paket data terkirim yang dilakukan oleh *node* 1 dan jumlah paket data yang diterima oleh *node* 2 pada sebuah *trace file*. Paket terkirim didapatkan dengan mencari baris yang memenuhi kondisi yaitu kolom pertama berisi huruf "s" yang berarti *send packet*, kolom ke-3 berisi angka "1" yang artinya *node* 1 melakukan pengiriman paket, kolom ke-4 dan kolom ke-7 berisi huruf "AGT" dan "cbr" yang menandakan pengiriman paket yang dilakukan adalah pengiriman paket data.

Untuk paket yang diterima didapatkan dengan mencari baris yang memenuhi kondisi yaitu kolom pertama berisi huruf "r" yang menandakan *received packet*, kolom ke-3 berisi angka "2" yang artinya node 2 menerima paket data, kolom ke-4 dan kolom ke-7 berisi huruf "AGT" dan "cbr" yang menandakan paket yang diterima adalah paket data. Perhitungan dilakukan sampai baris terakhir *trace file*, dan hasilnya adalah hasil hitung nilai PDR simulasi skenario.

Psedeucode PDR ditunjukkan pada Gambar 4.18 dan implementasinya dapat dilihat pada Lampiran.

```
ALGORITMA PDR AOMDV(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 == "_1_" and $4 == "AGT" and
       $7 == "cbr" )
       sent + 1;

if ($1 == "r" and $3 == "_2_" and $4 == "AGT" and
       $7 == "cbr")
       recv +1;
END (
pdr ← ( recv / sent ) * 100
print sent
print recv
print pdr)
```

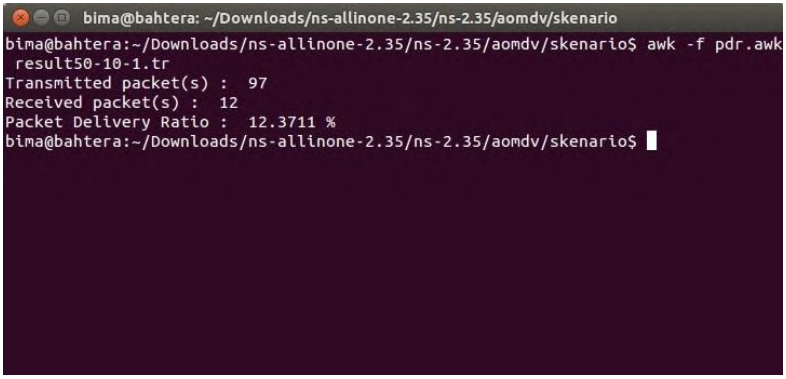
Gambar 4.18 *Pseudeucode* PDR

Contoh baris perintah untuk analisis PDR ditunjukkan pada Gambar 4.19.

```
awk -f pdr.awk result50-10-1.tr
```

Gambar 4.19 Baris perintah menjalankan *script* pdr.awk

Setelah *script* pada Gambar 4.19 dijalankan, akan menghasilkan sebuah *output* seperti yang ditunjukkan pada Gambar 4.20.



Gambar 4.20 Hasil *running script* pdr.awk

4.6.2. *Routing Overhead* (RO)

Baris yang mengandung *Routing Overhead* pada *trace file* ditandai dengan paket yang bertipe *send* (s) / *forward* (f) dan terdapat *header* paket dari protokol AOMDV. Implementasi perhitungan metrik *Routing* O*verhead* AOMDV dihitung dengan menggunakan beberapa kondisi-kondisi yang harus terpenuhi pada *trace file*. Kondisi-kondisi yang harus terpenuhi yatu pada kolom pertama *trace file* diawali dengan huruf "s" yang berarti *send packet* atau huruf "f" yang berarti *forward packet*, kolom ke-4 berisi huruf "RTR" yang berarti paket *routing* dan kolom ke-7 berisi huruf "AOMDV" yang berarti paket *routing* AOMDV. seperti pada Gambar 4.21. Implementasinya dapat dilihat pada Lampiran.

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

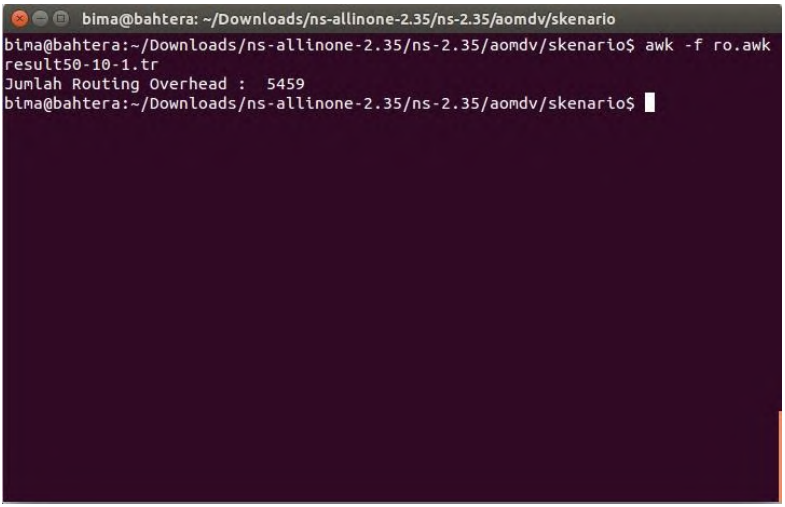
Gambar 4.21 *Pseudeucode Routing Overhead* AOMDV

Trace file dianalisa dengan menggunakan script *Routing Overhead* yang telah dibuat. Untuk baris perintah analisis *Routing Overhead* dapat dilihat pada Gambar 4.22.

```
awk -f ro.awk result50-10-1.tr
```

Gambar 4.22 Baris perintah menjalankan *script* ro.awk

Setelah *script* pada Gambar 4.22 dijalankan, akan menghasilkan sebuah *output* seperti yang ditunjukkan pada Gambar 4.23.



Gambar 4.23 Hasil *running script* ro.awk

4.6.3. *End-to-End Delay* (E2D)

Perhitungan *End-to-End Delay* dilakukan dengan menghitung selisih waktu paket data terkirim yang dilakukan oleh *node* 1 dan waktu paket data diterima oleh *node* 2 di dalam sebuah *trace file*. Pencatatan waktu paket terkirim didapatkan dengan mencari baris yang memenuhi kondisi yaitu kolom pertama berisi huruf "s" yang menandakan *send packet*, kolom ke-3 berisi angka "1" yang artinya *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 paket diterima didapatkan dengan mencari baris yang memenuhi kondisi yaitu kolom pertama berisi huruf "r" yang menandakan *received packet*, kolom ke-3 berisi angka "2" yang artinya *node* yang menerima paket adalah *node* 2, kolom ke-4 dan kolom ke-7 mengandung huruf "AGT" dan "cbr" yang menunjukkan paket yang diterima adalah paket data. Perhitungan dilakukan sampai baris terakhir *trace file*, dan dilakukan perhitungan nilai *End-to-End Delay* dengan

menghitung selisih *delay* paket mulai dari pengiriman sampai paket diterima pada simulasi skenario. *Psedeucode End-to-End Delay* ditunjukkan pada Gamabr 4.24 dan implementasinya dapat dilihat pada Lampiran.

```
ALGORITMA End-to-End Delay AOMDV(trace file)
//Input: trace file simulasi skenario
//Output: jumlah packet receieved, total delay, dan
//        End-to-End Delay

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
```

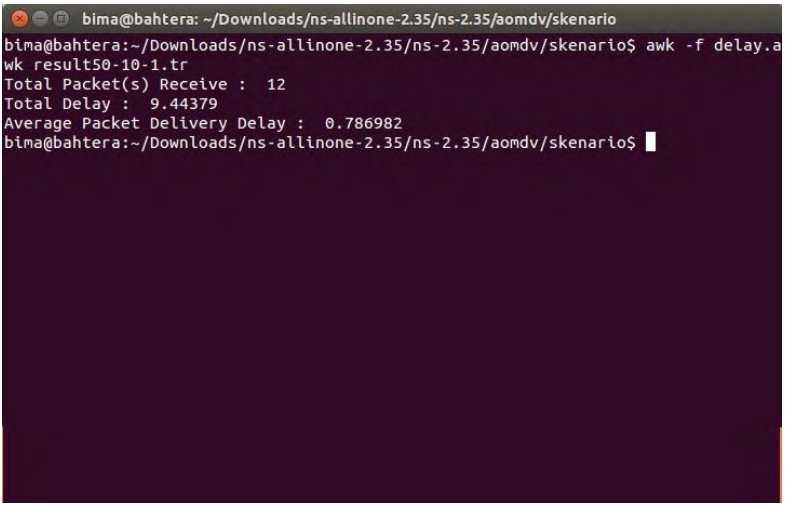
Gambar 4.24 *Pseudeucode* End-to-End Delay

Contoh baris perintah untuk analisis *End-to-End Delay* dapat dilihat pada Gambar 4.25.

```
awk -f delay.awk result50-10-1.tr
```

Gambar 4.25 Baris perintah menjalankan *script* delay.awk

Setelah *script* pada Gambar 4.24 dijalankan, akan menghasilkan sebuah *output* seperti yang ditunjukkan pada Gambar 4.25.



Gambar 4.26 Hasil *running script* delay.awk

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 dengan sistem operasi Windows yang telah terpasang perangkat lunak VirtualBox sehingga terdapat Linux dengan NS-2 terpasang di dalamnya. Untuk Spesifikasi dari laptop yang digunakan untuk uji coba ditunjukkan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Komputer yang Digunakan

Komponen	Spesifikasi
CPU	*Processor* Intel(R) Core(TM) i3 CPU @ 2.27GHz
Sistem Operasi	Windows 7 Home Basic 32-bit
Memori	2 GB DDR3
Penyimpanan	300 GB

Untuk konfigurasi dari VirtualBox yang digunakan untuk uji coba ditunjukkan pada Tabel 5.2.

Tabel 5.2 Konfigurasi VirtualBox

Komponen	Konfigurasi
General	
Name	Ubuntu 14.04
Operating Siystem	Ubuntu (32-bit)
System	
Base Memory	768 MB
Processor	1
Execution Cap	100%
Acceleration	VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

Display	
Video Memory	12 MB
Storage	
Controller	IDE
IDE Primary Master	[Optical Drive] VboxGuestAdditions.iso (55.46 MB)
IDE Secondary Master	[Optical Drive] Empty
Controller	SATA
SATA Port 0	Ubuntu 14.04.vdi (Normal, 8.00 GB)
Audio	
Host Driver	Windows DirectSound
Controller	ICH AC97
Shared Folders	
Shared Folders	1

5.2. Kriteria Pengujian

Pengujian pada protokol *routing* AOMDV menggunakan kriteria-kriteria yang ditunjukkan pada Tabel 5.3.

Tabel 5.3 Kriteria Pengujian

Kriteria	Spesifikasi
Skenario	MANET (*Random Waypoint*)
Jumlah *Node*	50
Jumlah Percobaan	10 kali dengan 3 mobilitas berbeda
Posisi Awal *Node*	Acak
Pergerakan	Acak
Protokol *Routing*	AOMDV

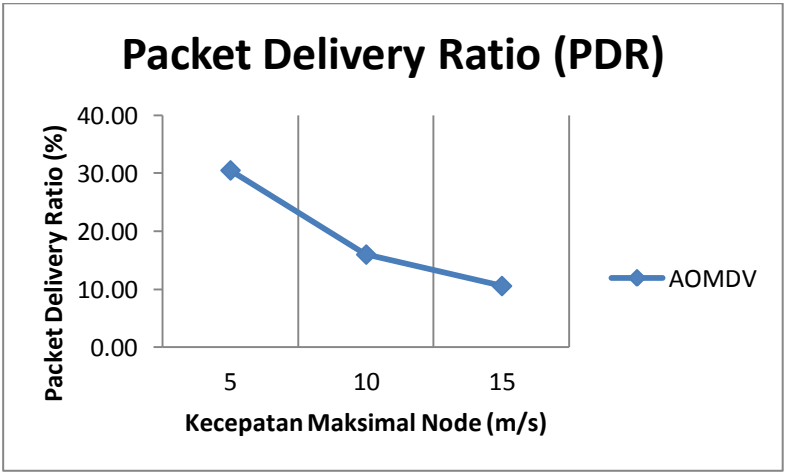
5.3. Analisis *Packet Delivery Ratio* (PDR)

Trace file hasil menjalankan program AOMDV.tcl, dianalisis nilai PDR melalui *script* pdr.awk. Hasil tiap

perhitungan PDR ditabulasikan dan dihitung nilai rata-rata seperti yang ditunjukkan pada Tabel 5.4.

Tabel 5.4 *Packet Delivery Ratio* (PDR) AOMDV

Kecepatan Maksimal *Node* (m/s)	PDR (%)
5	30,53
10	16,01
15	10,59



Gambar 5.1 Grafik *Packet Delivery Ratio* (PDR) AOMDV

Gambar 5.1 menunjukkan grafik performa PDR AOMDV pada jaringan MANET. Terlihat bahwa PDR yang dihasilkan semakin menurun dengan bertambahnya kecepatan maksimal *node*. Ketika kecepatan maksimal *node* 5 m/s, PDR yang dihasilkan yaitu 30,53%. Perubahan nilai PDR terjadi ketika kecepatan maksimal *node* dinaikkan menjadi 10 m/s. Nilai PDR menurun menjadi 16,01%. Begitu juga pada saat kecepatan *node* dinaikkan lagi menjadi 15 m/s. Nilai PDR semakin menurun menjadi 10,59%.

Pada jaringan MANET, *node-node* bergerak sangat dinamis. Semakin cepat pergerakan *node*, kemungkinan

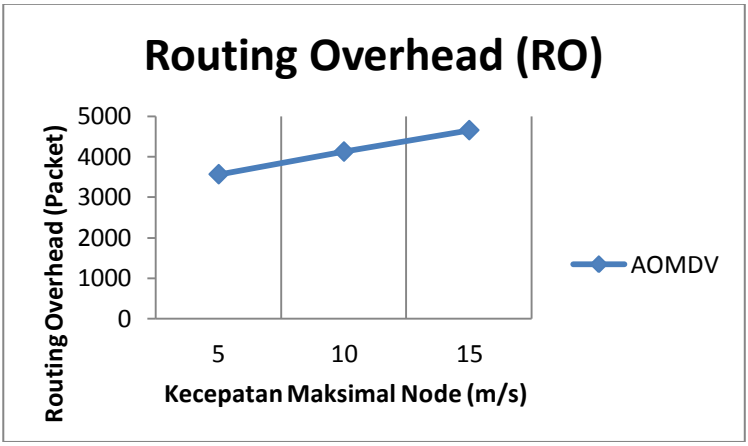
terjadinya banyak rute yang putus saat pengiriman data ataupun kegagalan *routing* yang dilakukan oleh AOMDV semakin besar, sehingga menyebabkan paket data yang dikirim tidak sampai ke tujuan. Hal ini telah dibuktikan dengan nilai PDR yang dihasilkan semakin menurun dengan bertambahnya kecepatan maksimal *node*.

5.4. Analisis Routing Overhead (RO)

Trace file hasil menjalankan program AOMDV.tcl, dianalisis nilai RO menggunakan *script* ro.awk. Hasil tiap perhitungan RO ditabulasikan dan dihitung nilai rata-rata seperti yang ditunjukkan pada Tabel 5.5.

Tabel 5.5 *Routing Overhead* (RO) AOMDV

Kecepatan Maksimal *Node* (m/s)	*Routing Overhead* (RO) (Jumlah Kontrol Paket)
5	3566,2
10	4125,5
15	4654,3



Gambar 5.2 Grafik *Routing Overhead* (RO) AOMDV

Gambar 5.2 menunjukkan grafik hasil pengujian *Routing Overhead* (RO) pada AOMDV. Nilai RO menunjukkan peningkatan berdasarkan kecepatan maksimal *node* dalam simulasi. Pada saat kecepatan *node* 5 m/s, nilai RO yang dihasilkan yaitu 3566,2. Namun pada saat kecepatan maksimal node dinaikkan menjadi 10 m/s terlihat bahwa nilai RO mengalami peningkatan menjadi 4125,5. Peningkatan nilai RO juga terjadi ketika kecepatan maksimal *node* dinaikkan menjadi 15 m/s. Pada saat kecepatan maksimal *node* 15 m/s, nilai RO naik menjadi 4654,3.

Kenaikan kecepatan maksimal *node* mengakibatkan kenaikan pada nilai RO. Hal ini terjadi karena ketika *node* bergerak cepat, pengiriman paket *routing* berjenis *send* maupun *forward* yang dihasilkan lebih banyak. Dengan semakin banyaknya paket *routing send* dan *forward* yang dihasilkan, pengiriman paket data yang dilakukan memiliki kemungkinan yang lebih besar untuk sampai ke *node* tujuan.

5.5. Analisis *End-to-End Delay*

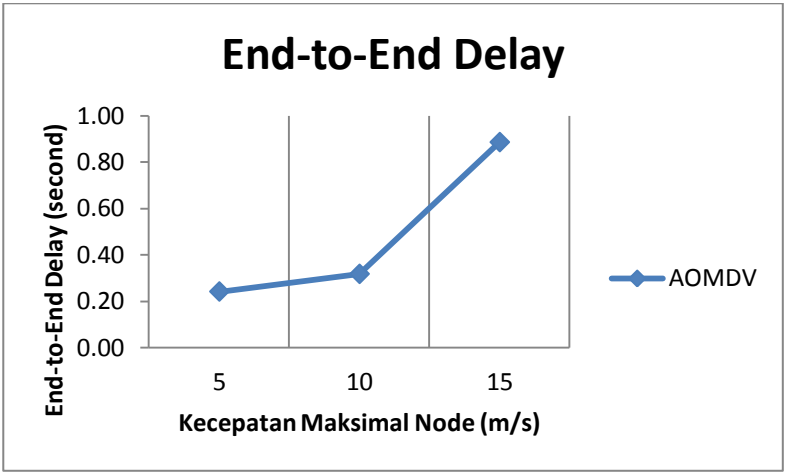
Trace file hasil menjalankan program AOMDV.tcl, dianalisis nilai *End-to-End Delay* melalui *script* delay.awk. Hasil tiap perhitungan *End-to-End Delay* ditabulasikan dan dihitung nilai rata-rata seperti yang ditunjukkan pada Tabel 5.6.

Tabel 5.6 *End-to-End Delay* AOMDV

Kecepatan Maksimal *Node* (m/s)	*End-to-End Delay* (*second*)
5	0,24
10	0,32
15	0,89

Performa *End-to-End Delay* pada AOMDV menunjukkan peningkatan berdasarkan kecepatan maksimal *node* dalam simulasi seperti yang ditunjukkan pada pada Gambar 5.3. Nilai *end-to-end delay* menunjukkan 0,24 *second* pada saat kecepatan *node* 5 m/s. Ketika kecepatan *node* dinaikkan menjadi 10 m/s, terlihat bahwa nilai *end-to-end delay* mengalami peningkatan

menjadi 0,32 *second*. Peningkatan nilai *end-to-end delay* juga terlihat ketika kecepatan *node* dinaikkan menjadi 15 m/s yaitu 0,89 *second*.



Gambar 5.3 Grafik *End-to-End Delay* AOMDV

Peningkatan nilai *end-to-end delay* disebabkan karena pergerakan *node* yang cepat dan dinamis. Pergerakan *node* yang cepat bisa mengakibatkan *node* bergerak menjauh dari jangkauan *transmission range*. Hal ini memungkinkan terjadinya rute putus saat pengiriman paket data sehingga pengiriman paket dimasukkan ke dalam antrian dan menunggu rute baru terbetuk sebelum pengiriman paket data dilanjutkan kembali.

LAMPIRAN

```
#
# nodes: 50, pause: 2.00, max speed: 5.00, max
x: 800.00, max y: 800.00
#
$node_(0) set X_ 744.660715872755
$node_(0) set Y_ 278.690136123343
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 187.838953144500
$node_(1) set Y_ 563.172177118206
$node_(1) set Z_ 0.000000000000
$node_(2) set X_ 485.040583482115
$node_(2) set Y_ 327.219246155956
$node_(2) set Z_ 0.000000000000
$node_(3) set X_ 700.069465119030
$node_(3) set Y_ 699.714247682263
$node_(3) set Z_ 0.000000000000
$node_(4) set X_ 14.335704171524
$node_(4) set Y_ 545.276282498955
$node_(4) set Z_ 0.000000000000
$node_(5) set X_ 18.504273670001
$node_(5) set Y_ 268.311611704718
$node_(5) set Z_ 0.000000000000
$node_(6) set X_ 741.423826063088
$node_(6) set Y_ 170.475555271589
$node_(6) set Z_ 0.000000000000
$node_(7) set X_ 313.063614516806
$node_(7) set Y_ 478.127382381469
$node_(7) set Z_ 0.000000000000
$node_(8) set X_ 466.999200995973
$node_(8) set Y_ 257.935973641156
$node_(8) set Z_ 0.000000000000
$node_(9) set X_ 248.085195664717
$node_(9) set Y_ 707.280392738600
$node_(9) set Z_ 0.000000000000
$node_(10) set X_ 770.806870685851
$node_(10) set Y_ 595.938663546751
$node_(10) set Z_ 0.000000000000
$node_(11) set X_ 341.735136853743
$node_(11) set Y_ 446.778279386899
$node_(11) set Z_ 0.000000000000
$node_(12) set X_ 705.775560904601
$node_(12) set Y_ 357.922292698137
```

```
43  $node_(12) set Z_ 0.000000000000
44  $node_(13) set X_ 530.829673449642
45  $node_(13) set Y_ 181.805066255725
46  $node_(13) set Z_ 0.000000000000
47  $node_(14) set X_ 630.746683849793
48  $node_(14) set Y_ 595.277780810785
49  $node_(14) set Z_ 0.000000000000
50  $node_(15) set X_ 492.283862828054
51  $node_(15) set Y_ 736.648072214518
52  $node_(15) set Z_ 0.000000000000
53  $node_(16) set X_ 33.425136970456
54  $node_(16) set Y_ 479.588885734437
55  $node_(16) set Z_ 0.000000000000
56  $node_(17) set X_ 688.011231996663
57  $node_(17) set Y_ 301.810950314784
58  $node_(17) set Z_ 0.000000000000
59  $node_(18) set X_ 278.528757843650
60  $node_(18) set Y_ 776.303940795180
61  $node_(18) set Z_ 0.000000000000
62  $node_(19) set X_ 211.429916130710
63  $node_(19) set Y_ 131.086351365308
64  $node_(19) set Z_ 0.000000000000
65  $node_(20) set X_ 467.598338718632
66  $node_(20) set Y_ 342.472784415432
67  $node_(20) set Z_ 0.000000000000
68  $node_(21) set X_ 456.509833911910
69  $node_(21) set Y_ 601.548583321764
70  $node_(21) set Z_ 0.000000000000
71  $node_(22) set X_ 375.094452784314
72  $node_(22) set Y_ 464.537213887959
73  $node_(22) set Z_ 0.000000000000
74  $node_(23) set X_ 131.426949644649
75  $node_(23) set Y_ 51.573186257673
76  $node_(23) set Z_ 0.000000000000
77  $node_(24) set X_ 446.138826756942
78  $node_(24) set Y_ 667.528129659093
79  $node_(24) set Z_ 0.000000000000
80  $node_(25) set X_ 521.406063170280
81  $node_(25) set Y_ 114.348220775004
82  $node_(25) set Z_ 0.000000000000
83  $node_(26) set X_ 408.448298684612
84  $node_(26) set Y_ 546.188603529527
85  $node_(26) set Z_ 0.000000000000
86  $node_(27) set X_ 657.662497459399
```

`

```
87  $node_(27) set Y_ 485.785144251611
88  $node_(27) set Z_ 0.000000000000
89  $node_(28) set X_ 472.400844995719
90  $node_(28) set Y_ 680.414537695754
91  $node_(28) set Z_ 0.000000000000
92  $node_(29) set X_ 205.351387596011
93  $node_(29) set Y_ 77.822610034398
94  $node_(29) set Z_ 0.000000000000
95  $node_(30) set X_ 143.391165376027
96  $node_(30) set Y_ 575.921476025057
97  $node_(30) set Z_ 0.000000000000
98  ...
99  $god_ set-dist 0 1 4
100 $god_ set-dist 0 2 2
101 $god_ set-dist 0 3 2
102 $god_ set-dist 0 4 5
103 $god_ set-dist 0 5 4
104 $god_ set-dist 0 6 1
105 $god_ set-dist 0 7 3
106 $god_ set-dist 0 8 2
107 $god_ set-dist 0 9 3
108 $god_ set-dist 0 10 2
109 $god_ set-dist 0 11 3
110 $god_ set-dist 0 12 1
111 $god_ set-dist 0 13 1
112 $god_ set-dist 0 14 2
113 $god_ set-dist 0 15 3
114 $god_ set-dist 0 16 4
115 $god_ set-dist 0 17 1
116 $god_ set-dist 0 18 3
117 $god_ set-dist 0 19 3
118 $god_ set-dist 0 20 2
119 $god_ set-dist 0 21 2
120 $god_ set-dist 0 22 3
121 $god_ set-dist 0 23 4
122 $god_ set-dist 0 24 3
123 $god_ set-dist 0 25 2
124 $god_ set-dist 0 26 3
125 $god_ set-dist 0 27 1
126 $god_ set-dist 0 28 3
127 $god_ set-dist 0 29 3
128 $god_ set-dist 0 30 4
129 $god_ set-dist 0 31 2
130 $god_ set-dist 0 32 4
```

```
131 $god_ set-dist 0 33 4
132 $god_ set-dist 0 34 4
133 $god_ set-dist 0 35 3
134 $god_ set-dist 0 36 1
135 $god_ set-dist 0 37 1
136 $god_ set-dist 0 38 4
137 $god_ set-dist 0 39 3
138 $god_ set-dist 0 40 4
139 $god_ set-dist 0 41 1
140 $god_ set-dist 0 42 2
141 $god_ set-dist 0 43 2
142 $god_ set-dist 0 44 4
143 $god_ set-dist 0 45 2
144 $god_ set-dist 0 46 2
145 $god_ set-dist 0 47 2
146 $god_ set-dist 0 48 3
147 $god_ set-dist 0 49 4
148 ...
149 $ns_ at 2.000000000000 "$node_(0) setdest
    621.578610243358 698.659979487135
    4.546762783017"
150 $ns_ at 2.000000000000 "$node_(1) setdest
    375.868120179644 737.324967925342
    2.569446721916"
151 $ns_ at 2.000000000000 "$node_(2) setdest
    457.424604134708 623.400326275096
    2.374029658688"
152 $ns_ at 2.000000000000 "$node_(3) setdest
    704.280531241174 223.084742948792
    1.483334069730"
153 $ns_ at 2.000000000000 "$node_(4) setdest
    302.474417843548 302.706101667804
    4.180816148783"
154 $ns_ at 2.000000000000 "$node_(5) setdest
    416.953666770450 215.629079204716
    0.812711950635"
155 $ns_ at 2.000000000000 "$node_(6) setdest
    538.904622032345 238.455772027094
    3.038381399117"
156 $ns_ at 2.000000000000 "$node_(7) setdest
    298.344791972944 792.772313974947
    2.747128683003"
157 $ns_ at 2.000000000000 "$node_(8) setdest
    56.554309409879 672.615845665358
```

```
     0.271355616032"
158  $ns_ at 2.000000000000 "$node_(9) setdest
     369.287237713986 0.214583995900
     4.004380333449"
159  $ns_ at 2.000000000000 "$node_(10) setdest
     491.751969578771 796.943672970935
     1.423683890869"
160  $ns_ at 2.000000000000 "$node_(11) setdest
     183.520714140569 454.653424995000
     2.789255044229"
161  $ns_ at 2.000000000000 "$node_(12) setdest
     570.511713872300 470.157771602463
     3.274314971887"
162  $ns_ at 2.000000000000 "$node_(13) setdest
     633.849911727193 95.576011734039
     0.667407725430"
163  $ns_ at 2.000000000000 "$node_(14) setdest
     553.124245593753 474.318933919617
     0.998231463049"
164  $ns_ at 2.000000000000 "$node_(15) setdest
     578.584084553991 578.243289579312
     0.585124540773"
165  $ns_ at 2.000000000000 "$node_(16) setdest
     113.922795210020 23.851208240039
     2.340269154351"
166  $ns_ at 2.000000000000 "$node_(17) setdest
     443.530918530755 388.602649241069
     0.373810262594"
167  $ns_ at 2.000000000000 "$node_(18) setdest
     180.406456981912 700.831931683477
     2.763142397100"
168  $ns_ at 2.000000000000 "$node_(19) setdest
     721.713493884636 490.411902313511
     2.428247269726"
169  $ns_ at 2.000000000000 "$node_(20) setdest
     169.091434583771 287.356116755417
     2.768288112898"
170  ...
171  $ns_ at 2.217410878673 "$god_ set-dist 8 31 2"
172  $ns_ at 2.217410878673 "$god_ set-dist 17 31
     1"
173  $ns_ at 2.217410878673 "$god_ set-dist 31 36
     2"
174  $ns_ at 2.502044977887 "$god_ set-dist 17 43
```

```
     1"
175  $ns_ at 2.502044977887 "$god_ set-dist 31 43
     2"
176  $ns_ at 2.503699022468 "$god_ set-dist 23 42
     4"
177  $ns_ at 2.503699022468 "$god_ set-dist 35 42
     3"
178  $ns_ at 2.503699022468 "$god_ set-dist 38 42
     4"
179  $ns_ at 2.503699022468 "$god_ set-dist 40 42
     4"
180  $ns_ at 2.503699022468 "$god_ set-dist 42 44
     4"
181  $ns_ at 2.503699022468 "$god_ set-dist 42 46
     2"
182  $ns_ at 2.568161791845 "$god_ set-dist 7 36 3"
     $ns_ at 2.568161791845 "$god_ set-dist 7 45 2"
183  $ns_ at 2.568161791845 "$god_ set-dist 9 36 4"
184  $ns_ at 2.568161791845 "$god_ set-dist 9 45 3"
185  $ns_ at 2.614842151213 "$god_ set-dist 0 18 4"
186  $ns_ at 2.614842151213 "$god_ set-dist 13 18
187  4"
188  $ns_ at 2.614842151213 "$god_ set-dist 17 18
     4"
189  $ns_ at 2.614842151213 "$god_ set-dist 18 21
     2"
190  $ns_ at 2.614842151213 "$god_ set-dist 18 27
     3"
191  $ns_ at 2.614842151213 "$god_ set-dist 18 42
     5"
192  $ns_ at 2.614842151213 "$god_ set-dist 18 43
     3"
193  $ns_ at 2.614842151213 "$god_ set-dist 18 46
     4"
194  $ns_ at 2.614842151213 "$god_ set-dist 18 47
     4"
195  $ns_ at 2.737803430273 "$god_ set-dist 5 46 2"
196  $ns_ at 2.737803430273 "$god_ set-dist 19 46
     1"
197  $ns_ at 2.930407764356 "$god_ set-dist 37 41
     1"
198  $ns_ at 2.930407764356 "$god_ set-dist 41 42
     2"
199  $ns_ at 3.220282908443 "$god_ set-dist 0 31 1"
```

`

```
200 $ns_ at 3.238122526250 "$god_ set-dist 3 43 2"
201 $ns_ at 3.238122526250 "$god_ set-dist 10 43
    2"
202 $ns_ at 3.238122526250 "$god_ set-dist 27 43
    1"
203 $ns_ at 3.463269254426 "$god_ set-dist 0 4 4"
    ...
204 #
205 # Destination Unreachables: 0
206 #
207 # Route Changes: 7752
208 #
209 # Link Changes: 2232
210 #
211 # Node | Route Changes | Link Changes
212 #    0 |           387 |           57
213 #    1 |           304 |           61
214 #    2 |           402 |           97
215 #    3 |           363 |           88
216 #    4 |           234 |          101
217 #    5 |           325 |           62
218 #    6 |           279 |          114
219 #    7 |           362 |           60
220 #    8 |           199 |           85
221 #    9 |           362 |          123
222 #   10 |           323 |           52
223 #   11 |           264 |           93
224 #   12 |           313 |          127
225 #   13 |           253 |           54
226 #   14 |           251 |           65
227 #   15 |           317 |          106
228 #   16 |           396 |           71
229 #   17 |           227 |           66
230 #   18 |           338 |          139
231 #   19 |           344 |          149
232 #   20 |           251 |          136
233 #   21 |           355 |           58
234 #   22 |           299 |          112
235 #   23 |           272 |           44
236 #   24 |           322 |           87
237 #   25 |           254 |           46
238 #   26 |           356 |           87
239 #   27 |           376 |           76
240 #   28 |           330 |          100
```

```
241 |  #    29 |             343 |             109
242 |  #    30 |             429 |             120
243 |  #    31 |             234 |             102
244 |  #    32 |             284 |             113
245 |  #    33 |             300 |             128
246 |  #    34 |             335 |              96
247 |  #    35 |             310 |             108
248 |  #    36 |             298 |              79
249 |  #    37 |             259 |              49
250 |  #    38 |             271 |              69
251 |  #    39 |             222 |              70
252 |  #    40 |             371 |             135
253 |  #    41 |             301 |             126
254 |  #    42 |             307 |              28
255 |  #    43 |             355 |             108
256 |  #    44 |             322 |              84
257 |  #    45 |             178 |              90
258 |  #    46 |             416 |             130
259 |  #    47 |             280 |              61
260 |  #    48 |             278 |              48
261 |  #    49 |             353 |              95
262 |  #
```

Gambar 8.1 Contoh skenario *node-movement* dari setdest

```
1   | set val(chan)
2   | Channel/WirelessChannel
3   | set val(prop)
4   | Propagation/TwoRayGround
5   | set val(netif)          Phy/WirelessPhy
6   | set val(mac)            Mac/802_11
7   | set val(ifq)
8   | Queue/DropTail/PriQueue
9   | set val(ll)             LL
10  | set val(ant)            Antenna/OmniAntenna
11  | set val(ifqlen)         100
12  | set val(rp)             AOMDV
13  | set val(x)              800
14  | set val(y)              800
15  | set val(seed)           0.0
16  | set val(tr)             aomdv.tr
17  | set val(nn)             50
18  |
19  | set val(cp)             "skenario/Scen50-
```

`

```
5/Scen50-5-1"
set val(sc)             "cbr-50-test"

set val(stop)           100.0

set ns_               [new Simulator]

#
# open traces
#

set tracefd  [open aomdv.tr w]
$ns_ trace-all $tracefd

Phy/WirelessPhy set RXThresh_ 1.42681e-08

set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]

set chan_1_ [new $val(chan)]

$ns_ node-config -adhocRouting $val(rp) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
                 -propType $val(prop) \
                 -phyType  $val(netif)  \
                 -channel $chan_1_ \
                 -topoInstance $topo \
                 -agentTrace ON \
                 -routerTrace ON \
                 -macTrace OFF

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

puts "Loading connection pattern..."
source $val(cp)
```

```
64
65  puts "Loading scenario file..."
66  source $val(sc)
67
68  for {set i 0} {$i < $val(nn) } {incr i} {
69      $ns_ initial_node_pos $node_($i) 20
70  }
71
72  for {set i 0} {$i < $val(nn) } {incr i} {
73      $ns_ at $val(stop).000000001 "$node_($i)
74  reset";
75  }
76
77  $ns_ at $val(stop).000000001 "puts \"NS
78  EXITING...\"; $ns_ halt"
79
80  proc stop {} {
81      global ns_ tracefd
82      $ns_ flush-trace
83      close $tracefd
84  }
85
86  puts "Start Simulation..."
87  $ns_ run
```

Gambar 8.2 *Script* AOMDV.tcl

```
1   BEGIN {
2   sent = 0;
3   recv = 0;
4   pdr = 0;
5   }
6   {
7   # count packet send
8   if ($1 == "s" && $3 == "_1_" && $4 == "AGT" &&
9   $7 == "cbr")
10  {
11          sent++;
12  }
13  # count packet receive
14  if ($1 == "r" && $3 == "_2_" && $4 == "AGT" &&
15  $7 == "cbr")
16  {
17          recv++;
18  }
```

`

```
}
END {
pdr = (recv/sent) * 100;
print ("Transmitted packet(s) : ", sent);
print ("Received packet(s) : ", recv);
print ("Packet Delivery Ratio : ", pdr, "%");
}
```

Gambar 8.3 Implementasi *Packet Delivery Ratio* (PDR)

```
BEGIN {
rt_paket = 0;
}
{
if (($1 == "s" || $1 == "f") && $4 == "RTR" &&
$7 == "AOMDV")
{
       rt_paket++;
}
}
END {
print ("Jumlah Routing Overhead : ",
rt_paket);
}
```

Gambar 8.4 Implementasi *Routing Overhead* (RO)

```
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;
}
{
# count packet send
if ($1 == "s" && $3 == "_1_" && $4 == "AGT" &&
$7 == "cbr")
{
```

```
20          pkt_sent[$6] = $2;
21    }
22    # count packet receive
23    if ($1 == "r" && $3 == "_2_" && $4 == "AGT" &&
24    $7 == "cbr" &&  recv_id != $6 )
25    {
26          recv++;
27          recv_id = $6;
28          pkt_recv[$6] = $2;
29    }
30    }
31    END {
32    for (i in pkt_recv){
33          delay += pkt_recv[i] - pkt_sent[i];
34    }
35    avg_delay = delay / recv;
36    print ("Total Packet(s) Receive : ",recv);
37    print ("Total Delay : ",delay);
38    print ("Average Packet Delivery Delay :
39    ",avg_delay);
40    }
```

Gambar 8.5 Implementasi *End-to-End Delay*

```
1     M 2.00000 0 (189.51, 621.26, 0.00), (402.04,
      163.29), 2.23
2     M 2.00000 1 (273.81, 97.67, 0.00), (324.39,
      197.86), 2.71
3     M 2.00000 2 (268.18, 446.83, 0.00), (139.50,
      456.28), 2.93
4     M 2.00000 3 (744.78, 345.05, 0.00), (333.41,
      749.55), 0.20
5     M 2.00000 4 (738.26, 329.69, 0.00), (529.55,
      433.88), 2.18
6     M 2.00000 5 (784.11, 314.65, 0.00), (316.55,
      696.24), 1.94
7     M 2.00000 6 (713.94, 119.17, 0.00), (510.02,
      698.48), 4.98
8     M 2.00000 7 (68.05, 666.14, 0.00), (600.45,
      565.79), 1.27
9     M 2.00000 8 (208.97, 376.48, 0.00), (659.36,
      218.36), 2.87
10    M 2.00000 9 (33.40, 286.84, 0.00), (599.45,
      485.56), 1.98
```

`

```
11  M 2.00000 10 (47.35, 538.14, 0.00), (761.87,
    736.62), 2.04
12  M 2.00000 11 (301.51, 545.18, 0.00), (12.97,
    767.31), 4.48
13  M 2.00000 12 (136.40, 566.59, 0.00), (29.65,
    699.04), 2.27
14  M 2.00000 13 (528.89, 695.04, 0.00), (229.49,
    605.97), 1.62
15  M 2.00000 14 (746.40, 663.90, 0.00), (30.78,
    129.43), 1.93
16  M 2.00000 15 (191.43, 765.70, 0.00), (265.49,
    256.73), 2.46
17  M 2.00000 16 (617.11, 54.64, 0.00), (149.48,
    688.15), 3.65
18  M 2.00000 17 (726.46, 72.56, 0.00), (425.29,
    300.97), 2.87
19  M 2.00000 18 (423.28, 179.96, 0.00), (692.84,
    180.99), 2.96
20  M 2.00000 19 (649.47, 797.03, 0.00), (250.16,
    440.84), 1.44
21  M 2.00000 20 (549.37, 461.03, 0.00), (627.26,
    726.15), 0.47
22  M 2.00000 21 (780.38, 688.23, 0.00), (104.50,
    399.03), 1.70
23  M 2.00000 22 (405.12, 797.82, 0.00), (489.35,
    361.90), 1.68
24  M 2.00000 23 (38.32, 285.65, 0.00), (712.15,
    542.77), 3.83
25  M 2.00000 24 (209.42, 780.70, 0.00), (95.80,
    537.80), 0.77
26  M 2.00000 25 (78.22, 783.03, 0.00), (24.85,
    793.47), 2.31
27  M 2.00000 26 (180.66, 520.02, 0.00), (178.20,
    47.72), 4.13
28  M 2.00000 27 (280.51, 167.65, 0.00), (2.60,
    114.11), 4.21
29  M 2.00000 28 (782.56, 317.64, 0.00), (50.75,
    14.67), 0.09
30  M 2.00000 29 (527.47, 736.47, 0.00), (334.69,
    644.79), 0.96
31  M 2.00000 30 (734.70, 9.64, 0.00), (352.60,
    662.26), 1.29
32  ...
33  s 2.556838879 _1_ AGT  --- 0 cbr 512 [0 0 0 0]
```

```
      ------- [1:0 2:0 32 0] [0] 0 3
34    r 2.556838879 _1_ RTR  --- 0 cbr 512 [0 0 0 0]
      ------- [1:0 2:0 32 0] [0] 0 3
35    s 2.556838879 _1_ RTR  --- 0 AOMDV 52 [0 0 0
      0] ------- [1:255 -1:255 30 0] [0x2 0 1 [2 0]
      [1 4]] (REQUEST)
36    r 2.558319106 _27_ RTR  --- 0 AOMDV 52 [0
      ffffffff 1 800] ------- [1:255 -1:255 30 0]
      [0x2 0 1 [2 0] [1 4]] (REQUEST)
37    s 2.567128902 _27_ RTR  --- 0 AOMDV 52 [0
      ffffffff 1 800] ------- [27:255 -1:255 29 0]
      [0x2 1 1 [2 0] [1 4]] (REQUEST)
38    r 2.568649130 _1_ RTR  --- 0 AOMDV 52 [0
      ffffffff 1b 800] ------- [27:255 -1:255 29 0]
      [0x2 1 1 [2 0] [1 4]] (REQUEST)
39    s 2.901169360 _27_ RTR  --- 0 AOMDV 44 [0 0 0
      0] ------- [27:255 -1:255 1 0] [0x1 0 [27 2]
      4.000000] (HELLO) [0 0]
40    r 2.902385584 _1_ RTR  --- 0 AOMDV 44 [0
      ffffffff 1b 800] ------- [27:255 -1:255 1 0]
      [0x1 0 [27 2] 4.000000] (HELLO) [0 0]
41    s 3.123323931 _1_ AGT  --- 1 cbr 512 [0 0 0 0]
      ------- [1:0 2:0 32 0] [1] 0 3
42    r 3.123323931 _1_ RTR  --- 1 cbr 512 [0 0 0 0]
      ------- [1:0 2:0 32 0] [1] 0 3
43    s 3.379665362 _1_ RTR  --- 0 AOMDV 44 [0 0 0
      0] ------- [1:255 -1:255 1 0] [0x1 0 [1 4]
      4.000000] (HELLO) [0 0]
44    r 3.380921580 _27_ RTR  --- 0 AOMDV 44 [0
      ffffffff 1 800] ------- [1:255 -1:255 1 0]
      [0x1 0 [1 4] 4.000000] (HELLO) [0 0]
45    s 3.758338051 _27_ RTR  --- 0 AOMDV 44 [0 0 0
      0] ------- [27:255 -1:255 1 0] [0x1 0 [27 2]
      4.000000] (HELLO) [0 0]
46    r 3.759474265 _1_ RTR  --- 0 AOMDV 44 [0
      ffffffff 1b 800] ------- [27:255 -1:255 1 0]
      [0x1 0 [27 2] 4.000000] (HELLO) [0 0]
47    s 4.000000000 _1_ RTR  --- 0 AOMDV 52 [0 0 0
      0] ------- [1:255 -1:255 30 0] [0x2 0 2 [2 0]
      [1 6]] (REQUEST)
48    r 4.001240212 _27_ RTR  --- 0 AOMDV 52 [0
      ffffffff 1 800] ------- [1:255 -1:255 30 0]
      [0x2 0 2 [2 0] [1 6]] (REQUEST)
49    s 4.005970474 _27_ RTR  --- 0 AOMDV 52 [0
```

`

```
     ffffffff 1 800] ------- [27:255 -1:255 29 0]
     [0x2 1 2 [2 0] [1 6]] (REQUEST)
50   r 4.007490686 _1_ RTR  --- 0 AOMDV 52 [0
     ffffffff 1b 800] ------- [27:255 -1:255 29 0]
     [0x2 1 2 [2 0] [1 6]] (REQUEST)
51   s 4.283639667 _1_ RTR  --- 0 AOMDV 44 [0 0 0
     0] ------- [1:255 -1:255 1 0] [0x1 0 [1 6]
     4.000000] (HELLO) [0 0]
52   r 4.284895877 _27_ RTR  --- 0 AOMDV 44 [0
     ffffffff 1 800] ------- [1:255 -1:255 1 0]
     [0x1 0 [1 6] 4.000000] (HELLO) [0 0]
53   s 4.416954558 _1_ AGT  --- 2 cbr 512 [0 0 0 0]
     ------- [1:0 2:0 32 0] [2] 0 3
54   r 4.416954558 _1_ RTR  --- 2 cbr 512 [0 0 0 0]
     ------- [1:0 2:0 32 0] [2] 0 3
55   s 4.573541088 _27_ RTR  --- 0 AOMDV 44 [0 0 0
     0] ------- [27:255 -1:255 1 0] [0x1 0 [27 2]
     4.000000] (HELLO) [0 0]
56   r 4.574717295 _1_ RTR  --- 0 AOMDV 44 [0
     ffffffff 1b 800] ------- [27:255 -1:255 1 0]
     [0x1 0 [27 2] 4.000000] (HELLO) [0 0]
57   s 4.998540437 _1_ AGT  --- 3 cbr 512 [0 0 0 0]
     ------- [1:0 2:0 32 0] [3] 0 3
58   r 4.998540437 _1_ RTR  --- 3 cbr 512 [0 0 0 0]
     ------- [1:0 2:0 32 0] [3] 0 3
59   s 5.000000000 _1_ RTR  --- 0 AOMDV 52 [0 0 0
     0] ------- [1:255 -1:255 30 0] [0x2 0 3 [2 0]
     [1 8]] (REQUEST)
60   r 5.001500204 _27_ RTR  --- 0 AOMDV 52 [0
     ffffffff 1 800] ------- [1:255 -1:255 30 0]
     [0x2 0 3 [2 0] [1 8]] (REQUEST)
61   s 5.002821333 _27_ RTR  --- 0 AOMDV 52 [0
     ffffffff 1 800] ------- [27:255 -1:255 29 0]
     [0x2 1 3 [2 0] [1 8]] (REQUEST)
62   r 5.004281537 _1_ RTR  --- 0 AOMDV 52 [0
     ffffffff 1b 800] ------- [27:255 -1:255 29 0]
     [0x2 1 3 [2 0] [1 8]] (REQUEST)
63   s 5.256784052 _1_ RTR  --- 0 AOMDV 44 [0 0 0
     0] ------- [1:255 -1:255 1 0] [0x1 0 [1 8]
     4.000000] (HELLO) [0 0]
64   r 5.257740254 _27_ RTR  --- 0 AOMDV 44 [0
     ffffffff 1 800] ------- [1:255 -1:255 1 0]
     [0x1 0 [1 8] 4.000000] (HELLO) [0 0]
65   s 5.623812020 _27_ RTR  --- 0 AOMDV 44 [0 0 0
```

```
      0] ------- [27:255 -1:255 1 0] [0x1 0 [27 2]
      4.000000] (HELLO) [0 0]
66    r 5.624928219 _1_ RTR  --- 0 AOMDV 44 [0
      ffffffff 1b 800] ------- [27:255 -1:255 1 0]
      [0x1 0 [27 2] 4.000000] (HELLO) [0 0]
67    s 6.000000000 _1_ RTR  --- 0 AOMDV 52 [0 0 0
      0] ------- [1:255 -1:255 30 0] [0x2 0 4 [2 0]
      [1 10]] (REQUEST)
68    r 6.001040197 _27_ RTR  --- 0 AOMDV 52 [0
      ffffffff 1 800] ------- [1:255 -1:255 30 0]
      [0x2 0 4 [2 0] [1 10]] (REQUEST)
69    s 6.005005921 _27_ RTR  --- 0 AOMDV 52 [0
      ffffffff 1 800] ------- [27:255 -1:255 29 0]
      [0x2 1 4 [2 0] [1 10]] (REQUEST)
70    r 6.006146118 _1_ RTR  --- 0 AOMDV 52 [0
      ffffffff 1b 800] ------- [27:255 -1:255 29 0]
      [0x2 1 4 [2 0] [1 10]] (REQUEST)
71    s 6.295710993 _1_ AGT  --- 4 cbr 512 [0 0 0 0]
      ------- [1:0 2:0 32 0] [4] 0 3
72    r 6.295710993 _1_ RTR  --- 4 cbr 512 [0 0 0 0]
      ------- [1:0 2:0 32 0] [4] 0 3
73    ...
```

Gambar 8.6 Contoh *trace file* dari *script* AOMDV.tcl

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 kinerja protokol *routing* AOMDV pada jaringan MANET adalah sebagai berikut :
   - Dengan penambahan jumlah kecepatan node, performa *Packet Delivery Ratio* yang dihasilkan mengalami penurunan. Penurunan sebesar 14.5% dari kecepatan maksimal *node* 5 m/s ke 10 m/s dan penurunan sebesar 5.42% dari kecepatan maksimal node 10 m/s ke 15 m/s.
   - *Routing Overhead* yang dihasilkan mengalami peningkatan berdasarkan penambahan jumlah kecepatan *node* dengan kenaikan rata-rata 544.05.
   - Performa *End-to-End Delay* yang dihasilkan juga mengalami peningkatan berdasarkan penambahan jumlah kecepatan *node*. Kenaikan sebesar 0.08 s dari kecepatan maksimal *node* 5 m/s ke 10 m/s dan kenaikan sebesar 0.57 s dari kecepatan maksimal node 10 m/s ke 15 m/s.
2. Kecepatan *node* sangat mempengaruhi performa kinerja protokol *routing* AOMDV. *Node* yang bergerak dengan cepat menyebabkan perubahan topologi jaringan dengan cepat dan tak terduga dari waktu ke waktu sehingga performa kinerja AOMDV menjadi buruk.

6.2. Saran

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

1. Perlu dikembangkan penelitian pada protokol dari kelas lain seperti *position based* dan proaktif.
2. Perlu dikembangkan penelitian menggunakan sekenario yang lebih riil seperti menggunakan *Simulation of Urban Mobility* (SUMO) atau simulator lainnya.

`

DAFTAR PUSTAKA

[1] A. O. Bang dan P. L. Ramteke, "MANET : History, Challenges And Applications," *International Journal of Application or Innovation in Engineering & Management*, vol. II, no. 9, 2013.

[2] S. Corson dan J. Macker, "Mobile Ad hoc Networking (MANET) : Routing Protocol Performance Issues and Evaluation Considerations," *Network Working Group,* 1999.

[3] M. K. Marina dan S. R. Das, "Ad hoc on-demand multipath distance vector routing," *WIRELESS COMMUNICATIONS AND MOBILE COMPUTING,* 2006.

[4] "VirtualBox," Oracle, [Online]. Available: https://www.virtualbox.org/manual/ch01.html. [Diakses 26 Juni 2016].

[5] T. Issariyakul dan E. Hossain, Introduction to Network Simulator NS2, New York: Springer, 2012.

[6] "Information Sciences Institute," [Online]. Available: http://www.isi.edu/nsnam/ns/tutorial/nsscript7.html. [Diakses 26 Juni 2016].

[7] "GNU Operating System," [Online]. Available: http://www.gnu.org/software/gawk/manual/gawk.html. [Diakses 26 Juni 2016].

[8] "Bengkel Ubuntu," 8 November 2014. [Online]. Available: http://bengkelubuntu.org/teks/awk/Praktikum%2001%20-%20Berkenalan%20dengan%20AWK.pdf. [Diakses 26 Juni 2016].

BIODATA PENULIS

Bima Bahteradi Putra, biasa dipanggil Bima, dilahirkan di Trenggalek pada tanggal 23 Januari 1992. Penulis adalah anak kedua dari tiga bersaudara. Penulis menempuh TK Dharma Wanita 1 Karangan (1996-1998), SD Negeri 3 Karangan (1998-2004), SMP Negeri 1 Karangan (2004-2007), SMA Negeri 1 Trenggalek (2007-2010). Pada tahun 2010, penulis memulai pendidikan S1 Jurusan Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember Surabaya yang terdaftar dengan NRP 5110100105. Di jurusan Teknik Informatika, penulis mengambil bidang minat Arsitektur dan Jaringan Komputer (AJK). Penulis dapat dihubungi melalui alamat *e-mail* bimabahtera@gmail.com.