TUGAS AKHIR – IF184802

MODIFIKASI PEMILIHAN RUTE PADA *AD-HOC ON-DEMAND DISTANCE VECTOR* (AODV) BERDASARKAN TINGKAT ENERGY DAN JARAK DI LINGKUNGAN VANETS

GALUH AAN RAMADHAN
NRP 05111540000026

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

Dosen Pembimbing II
Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc.

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR – IF184802

MODIFIKASI PEMILIHAN RUTE PADA *AD-HOC ON-DEMAND DISTANCE VECTOR* (AODV) BERDASARKAN TINGKAT ENERGY DAN JARAK DI LINGKUNGAN VANETS

GALUH AAN RAMADHAN
NRP 05111540000026

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

Dosen Pembimbing II
Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc.

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – IF184802

MODIFICATION OF ROUTE SELECTION ON AD-HOC ON-DEMAND DISTANCE VECTOR (AODV) BASED ON ENERGY LEVEL AND DISTANCE IN VANETS ENVIRONMENT

GALUH AAN RAMADHAN
NRP 05111540000026

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

Second Advisor
Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc.

Department of Informatics Engineering
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology
Surabaya 2019

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

MODIFIKASI PEMILIHAN RUTE PADA *AD-HOC ON-DEMAND DISTANCE VECTOR* (AODV) BERDASARKAN TINGKAT ENERGY DAN JARAK DI LINGKUNGAN VANETS

TUGAS AKHIR

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

Oleh:
GALUH AAN RAMADHAN
NRP: 05111540000026

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr.Eng. Radityo Anggoro, S.Kom., M.Sc. .....
(NIP. 198410162008121002) (Pembimbing 1)

2. Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc. .....
(NIP. 197110302002121001) (Pembimbing 2)

SURABAYA
JANUARI, 2020

(Halaman ini sengaja dikosongkan)

MODIFIKASI PEMILIHAN RUTE PADA *AD-HOC ON-DEMAND DISTANCE VECTOR* (AODV) BERDASARKAN TINGKAT ENERGY DAN JARAK DI LINGKUNGAN VANETS

Nama Mahasiswa	**: Galuh Aan Ramadhan**
NRP	**: 05111540000026**
Departemen	**: Informatika FTIK-ITS**
Dosen Pembimbing 1	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**
Dosen Pembimbing 2	**: Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc.**

Abstrak

Dalam jaringan ad hoc, komunikasi antar perangkat nirkabel dapat dilakukan walupun dalam posisi yang sangat dinamis. Sehingga dibutuhkan sebuah jaringan yang dapat mengatur lalu lintas perangkat tersebut untuk menjaga kualitas hubungan antar perangkatnya. Lalu terwujudlah sebuah jaringan yang disebut *Vehicular Ad Hoc Networks* (VANETs).

VANETs memiliki karakteristik dengan mobilitas yang sangat tinggi dan terbatas pada pola pergerakannya. Ada banyak *routing protocol* yang dapat diimplementasikan pada VANETs, salah satunya adalah *Ad hoc On demand Distance Vector* (AODV).

AODV merupakan *reactive routing protocol*, sebuah protokol yang hanya akan membuat rute ketika *node* sumber membutuhkannya. AODV memiliki fase yang bernama *route discovery* yang digunakan untuk meminta dan meneruskan informasi rute yang terdiri dari proses pengiriman *Route Request* (RREQ) dan *Route Reply* (RREP).

Pada Tugas Akhir ini akan dilakukan modifikasi pemilihan rute untuk pengiriman data berdasarkan energi dan jarak pada routing protocol AODV. Metode ini akan memlihih rute berdasarkan node

yang memiliki tingkat energy paling besar dan memliki jarak yang paling minimum.

Hal ini dilakukan agar dapat meningkatkan kualitas pengiriman paket AODV untuk memilih rute yang paling optimal dengan parameter energy dan jarak. Dari hasil uji coba, AODV yang dimodifikasi pada skenario grid berhasil meningkatkan nilai Packet Delivery Ratio (PDR) hingga 62,11%, penurunan rata-rata End-to-End Delay (E2E) sebesar 60,28%, dan penurunan nilai Routing Overhead (RO) hingga 6,34% sedangkan pada skenario real berhasil meningkatkan nilai Packet Delivery Ratio (PDR) hingga 41,14%, penurunan rata-rata End-to-End Delay (E2E) sebesar 53,63%, dan penurunan nilai Routing Overhead (RO) hingga 7,13%.

Kata kunci: VANETs, AODV, NS2, SUMO, Energi, Jarak

MODIFICATION OF ROUTE SELECTION ON AD-HOC ON-DEMAND DISTANCE VECTOR (AODV) BASED ON ENERGY LEVEL AND DISTANCE IN VANETS ENVIRONMENT

Student's Name	**: Galuh Aan Ramadhan**
Student's ID	**: 05111540000026**
Department	**: Informatics – FTIK ITS**
First Advisor	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**
Second Advisor	**: Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc.**

Abstract

In ad hoc network, wireless inter-device communication can be carried out in the dynamic condition, therefore, network traffic controller device is needed to protect the quality connection between the devices. Then a network called Vehicular Ad Hoc Networks (VANETs) is finally formed.

VANETs have superior mobility and limited on its movement pattern. There are so many routing protocols which can be implemented on VANETs, such as Ad Hoc On demand Distance Vector (AODV).

AODV is a reactive routing protocol which makes a route when node source needs it. AODV has a phase called route discovery that is used to claim and transmit the route which consists of Route Request (RREQ) transmission process and Route Reply (RREP).

In this Final Project, route selection modification will be done for data transmission based on energy and AODV routing protocol distance. This method will choose the route based on node by its highest energy level and shortest distance.

This study is carried out to improve AODV transmission quality to choose the most optimal route by its parameters, energy

and distance. Based on experiment, the modified AODV on grid scenario increased the Packet Delivery Ratio (PDR) value to 62.11%, decreased the End-to-End Delay (E2E) average at 60,28% and decreased the Routing Overhead (RO) value to 6.34%. While on real scenario, the modified AODV increased the Packet Delivery Ration (PDR) value to 41.14%, decreased the End-to-End Delay (E2E) average at 53,63% and decreased the Routing Overhead (RO) value to 7.13%.

Keyword: VANETs, AODV, NS2, SUMO, Energy, Distance

KATA PENGANTAR

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul **"Modifikasi Pemilihan Rute pada *Ad-hoc On-Demand Distance Vector* (AODV) berdasarkan rasio Tingkat Energy dan Jarak di Lingkungan VANETS".**

Penulis berharap semoga apa yang tertulis di dalam buku Tugas Akhir ini dapat berkah dan bermanfaat untuk pengembangan ilmu pengetahuan saat ini.

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

1. Allah SWT atas semua rahmat yang diberikan sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Bapak Suyono dan Ibu Sutri Ningsih selaku kedua orangtua penulis atas segala dukungan berupa doa yang tiada henti-hentinya sehingga penulis dapat mengerjakan Tugas Akhir ini.
3. Septia Intan Wulani selaku kakak penulis atas segala dukungan yang telah diberikan sehingga penulis tetap semangat dalam mengerjakan Tugas Akhir ini.
4. Bapak Dr. Eng. Radityo Anggoro, S.Kom., M.Sc., dan Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc. selaku dosen pembimbing, atas arahan dan bantuannya dalam pengerjaan Tugas Akhir ini.
5. Sahabat SMA Ahmad Ilham Ali Yafi dan Imaniar Fitri selalu mendampingi penulis dalam menyelesaikan Tugas Akhir ini
6. Sahabat Kuliah Fauzan Abid Ramadhan, Neny Lukitasari, Dwi Irsalina, M. NurFaizal, Vicky Mahfudy dan teman – teman angkatan 2015 yang selama ini sudah menemani penulis selama masa perkuliahan di ITS.

7. Teman-teman AJK yang sudah membantu selama penulis mengerjakan Tugas Akhir ini.
8. Dan kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa masih terdapat banyak kekurangan, kesalahan, maupun kelalaian yang telah penulis lakukan. Oleh karena itu, saran dan kritik yang membangun sangat dibutuhkan untuk penyempurnaan Tugas Akhir ini.

Surabaya, Desember 2019

Galuh Aan Ramadhan

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi dan informasi membuat perangkat dan teknologi komunikasi berkembang dengan cepat yang mana mengakibatkan adanya desain dan implementasi berbagai macam jaringan di berbagai jenis lingkungan. Salah satu jaringan yang bertumbuh pesat tersebut ialah *Vehicular Ad-Hoc Network* (VANETs). *Vehicular Ad-Hoc Network* (VANETs) merupakan turunan dari MANETs (*Mobile Ad Hoc Network*). VANETs menggunakan jaringan wireless berbasis Ad Hoc yang diterapkan pada kendaraan bergerak. VANET menjadi salah satu area riset yang berkembang di beberapa tahun terakhir guna mendukung ITS (*Intelligent Transportation System*). Untuk berkomunikasi di lingkungan VANET, kendaraan berinteraksi antara mereka sendiri dengan kendaraan lain yang disebut dengan V2V (*Vehicle to Vehicle*) communication atau kendaraan juga dapat berkomunikasi dengan infrastruktur seperti RSU (*Road Side Unit*) yang disebut dengan V2I (*Vehicle to Infrastructure*) communication atau V2R (*Vehicle to Roadside*) communication.

Dengan meningkatnya lalu lintas kendaraan di daerah perkotaan, diperlukan efisiensi penggunaan sumber daya yang tersedia untuk meminimalisir beban kerja dan konsumsi energi. Untuk memperkecil jumlah konsumsi energi tersebut, maka diperlukan sebuah perancangan optimalisasi energi agar energi yang dibutuhkan semakin efisien [1].

Pada Tugas Akhir ini diusulkan suatu optimasi penggunaan energi pada protokol AODV untuk menganalisis performa berdasarkan tingkat Energi dan jarak. Hasil akhir yang diharapkan adalah mengetahui perbandingan kinerja antara AODV dan AODV yang telah dimodifikasi yang diukur berdasarkan

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

1.2 Rumusan Masalah

Berikut beberapa hal yang menjadi rumusan masalah pada Tugas Akhir ini:

1. Bagaimana cara menentukan rute berdasarkan *Energy/Distance* untuk pengiriman data?
2. Bagaimana dampak modifikasi AODV terhadap performa yang diukur berdasarkan *Packet Delivery Ratio (PDR), End-to-End Delay, dan Routing Overhead*?

1.3 Batasan Permasalahan

Batasan masalah pada Tugas Akhir ini adalah sebagai berikut:

1. Jaringan yang digunakan adalah *Vehicular Ad hoc Networks* (VANETs).
2. Routing protocol yang digunakan yaitu AODV.
3. Proses pengujian menggunakan *network simulator* NS-2.
4. Area simulasi dibuat dengan *Simulation of Urban Mobility* (SUMO).

1.4 Tujuan

Untuk menganalisa performa modifikasi AODV berdasarkan *Energy/Distance*.

1.5 Manfaat

Mengetahui dampak modifikasi AODV terhadap performa yang diukur berdasarkan *Packet Delivery Ratio (PDR), End-to-End Delay, dan Routing Overhead*.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat dari hasil pembuatan tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Sub bab metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terakhir terdapat pula sub bab jadwal kegiatan yang menjabarkan jadwal pengerjaan tugas akhir.

1.6.2 Studi Literatur

Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam melakukan analisis yaitu mengenai *network simulator* NS-2, *Vehicular Ad hoc Networks* (VANETs), dan *reactive routing protocol* AODV.

1.6.3 Analisis dan Desain Sistem

Pada tahap ini dilakukan analisis dari hasil percobaan modifikasi *routing protocol* AODV yang dibuat. Data yang dianalisis berasal dari perhitungan *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), dan *End-to-End Delay* paket dari node ke node lainnya.

1.6.4 Implementasi Sistem

Implementasi merupakan tahap untuk membangun metode-metode yang sudah diajukan pada proposal Tugas Akhir. Pada tahap ini dilakukan implementasi menggunakan NS-2 sebagai *simulator*, Bahasa C/C++ sebagai bahasa pemrograman, dan SUMO sebagai *tools* untuk uji coba dan mengimplentasikan desain sistem yang sudah dirancang.

1.6.5 Pengujian dan Evaluasi

Pengujian dilakukan dengan VANETs *simulator generator* dan *traffic generator* untuk membuat simulasi keadaan topologi untuk diujikan. Kemudian simulasi yang dibuat pada VANETs *simulator generator* dan *traffic generator* tersebut dijalankan pada *network simulator* NS-2 dan akan menghasilkan *trace file.* Dari *trace file* tersebut akan dihitung *packet delivery ratio, End-to-End Delay, dan Routing Overhead* pengiriman paket untuk menguji performa *routing protocol* adaptif yang dibuat. Ketiga parameter ini akan dianalisis perbandingannya untuk beberapa konfigurasi tertentu.

1.6.6 Penyusunan Buku

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi sistem yang telah dibuat.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

1. Bab I. Pendahuluan

Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori atau penjelasan dari metode, algoritma, *library*, dan *tools* yang digunakan dalam penyusunan Tugas Akhir ini. Bab ini berisi tentang penjelasan singkat mengenai VANETs, AODV, NS2, OpenStreetMap, Java OpenStreetMap (JOSM), SUMO, dan AWK.
3. Bab III. Perancangan
   Bab ini berisi pembahasan mengenai perancangan skenario mobilitas *grid* dan *real*, perancangan simulasi pada NS2, perancangan modifikasi AODV, serta perancangan metrik analisis (*Packet Delivery Ratio*, *End-to-End Delay*, *dan Routing Overhead*).
4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi yang berbentuk kode sumber dari proses modifikasi protokol AODV, pembuatan simulasi pada NS2, SUMO, dan perhitungan metrik analisis.
5. Bab V. Uji Coba dan Evaluasi
   Bab ini berisikan hasil uji coba dan evaluasi dari implementasi yang telah dilakukan untuk menyelesaikan masalah yang dibahas pada Tugas Akhir.
6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.
7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam Tugas Akhir.
8. Lampiran
   Dalam lampiran terdapat tabel-tabel data hasil uji coba dan kode sumber program secara keseluruhan.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar atau penjelasan dari metode dan *tools* yang digunakan dalam Tugas Akhir.

2.1 *Vehicular Ad hoc Networks* (VANETs)

Vehicular Ad Hoc Networks (VANETs) adalah teknologi baru yang memadukan jaringan ad hoc, *wireless* LAN (WLAN), dan teknologi seluler untuk menciptakan komunikasi antar kendaraan yang cerdas dan meningkatkan keselamatan dan efisiensi lalu lintas jalan. VANET dibedakan dari jenis jaringan ad hoc lainnya berdasarkan arsitektur jaringannya yang hibrida, karakteristik gerakan *node*, dan skenario aplikasinya yang baru. VANET menciptakan banyak tantangan yang unik tentang penelitian teknologi jaringan dan desain *routing protocol* yang efisien untuk VANET merupakan hal yang krusial [2].

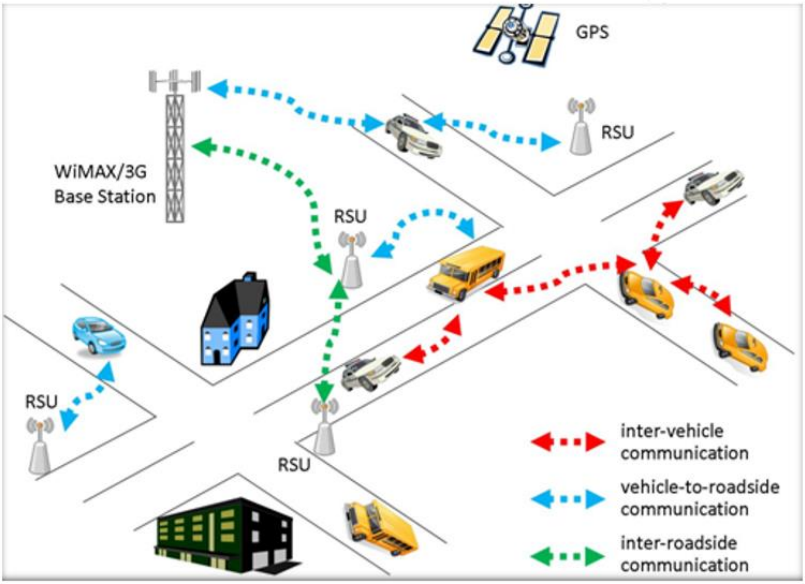
VANET terdiri dari komunikasi *vehicle-to-vehicle* dan *vehicle-to-infrastructure* berdasarkan teknologi jaringan nirkabel area lokal. Beberapa aplikasi (contoh: peringatan tabrakan dan informasi lalu lintas area lokal untuk pengemudi), sumber daya (spektrum berlisensi, sumber daya yang dapat diisi ulang), dan lingkungan (contoh: pola arus lalu lintas kendaraan, permasalahan privasi) menjadikan VANET sebagai area yang unik dari komunikasi nirkabel [3].

Dengan komunikasi *vehicle-to-vehicle* dan *vehicle-to-roadside*, kecelakaan dapat dicegah, contohnya dengan tidak masuk ke area kemacetan, sehingga dapat menambah efisiensi lalu lintas yaitu dengan mengambil rute alternatif.

Dalam VANET, mobilitas kendaraan dibatasi oleh jalan dan bangunan. Kecepatan kendaraan dimulai dari nol hingga batas kecepatan dari kendaraan atau aturan lalu lintas yang ada dijalan

tersebut. Jaringan VANET dipengaruhi oleh ukuran dan kecepatan kendaraan, lokasi geografis antar kendaraan, dan frekuensi komunikasi yang biasanya jarang terjadi karena kanal komunikasi yang tidak reliable. Dari masalah yang ditimbulkan oleh berbagai karakteristik tersebut, periset mengajukan berbagai isu riset di bidang routing, diseminasi data, data sharing, dan isu sekuritas. Protokol yang digunakan untuk VANET saat ini adalah protokol-protokol yang didesain untuk jaringan Mobile Ad Hoc Network (MANET). Protokol-protokol tersebut tidak dapat menyelesaikan permasalah dari karakteristik unik VANET sehingga tidak cocok untuk komunikasi vehicle-to-vehicle (V2V) dalam VANET [4].

Ilustrasi VANET dapat dilihat pada Gambar 2.1.



Gambar 2.1 Ilustrasi VANET [5]

Dalam Tugas Akhir ini, penulis akan mengimplementasikan *routing protocol* AODV yang dimodifikasi dan menguji performa protokol tersebut pada lingkungan VANETs.

2.2 *Ad-hoc On-Demand Distance Vector* (AODV)

Ad hoc On-Demand Distance Vector (AODV) merupakan algoritma untuk pengoperasian jaringan *ad hoc*. Setiap host seluler beroperasi sebagai router khusus dan rute dapat diperoleh sesuai perminataan (*on demand)*. Algoritma ini sangat cocok untuk jaringan yang dinamis, seperti yang dibutuhkan pengguna untuk memanfaatkan jaringan *ad hoc*. AODV menyediakan rute *loop-free* bahkan pada saat memperbaiki koneksi yang rusak [6].

Proses pencarian *path* diinisiasikan ketika *node* sumber ingin berkomunikasi dengan *node* lainnya yang belum ada informasinya di dalam tabel. Setiap *node* menyimpan 2 penghitung, *node sequence number* dan *broadcast_id. Node* sumber menginisiasi pencarian *path* dengan mem-*broadcast* paket *route request* (RREQ) kepada tetangganya [6]. Struktur paket RREQ dapat dilihat pada Tabel 2.1.

Tabel 2.1 Struktur Paket RREQ

source_addr	*source_sequence_#*	*broadcast_id*
dest_addr	*dest_sequence_#*	*hop_cnt*

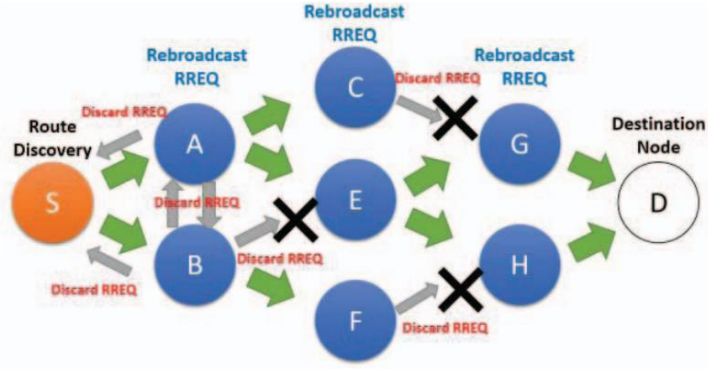
Setiap tetangga yang menerima paket RREQ harus mengirim *route reply* (RREP) kepada *node* sumber, atau mem-*broadcast* ulang paket RREQ ke tetangganya. Sebuah *node* bisa menerima paket broadcast yang sama dari tetangga-tetangganya. Ketika *node* menerima paket RREQ, jika dia sudah menerima paket RREQ dengan *broadcast_id* dan *source address* yang sama, *node* tersebut akan mengedrop paket RREQ yang baru dan tidak mem-*broadcast* ulang paket tersebut [6].

AODV mengadopsi karakteristik *on-demand* pada DSR. Namun, AODV memiliki mekanisme yang sangat berbeda untuk menjaga informasi *routing*. AODV menggunakan tabel *routing* tradisional, satu entri per tujuan. Berbeda dengan DSR yang dapat mempertahankan beberapa *cache* entri rute untuk setiap tujuan. AODV bergantung pada entri tabel *routing* untuk menyebarkan kembali RREP sumber dan untuk merutekan paket data ke tujuan [7].

Pada setiap *node* yang menggunakan protokol AODV pasti memiliki sebuah *routing table* dengan *field* sebagai berikut:

- *Destination Address:* berisi alamat dari *node* tujuan.
- *Destination Sequence Number*: *sequence number* dari jalur komunikasi.
- *Next Hop*: alamat *node* yang akan meneruskan paket data.
- *Hop Count*: jumlah *hop* yang harus dilakukan agar paket dapat mencapai *node* tujuan.
- *Lifetime*: waktu dalam milidetik yang diperlukan *node* untuk menerima RREP.
- *Routing Flags*: status jalur. Terdapat tiga tipe status, yaitu *up* (valid), *down* (tidak valid) atau sedang diperbaiki.

Ilustrasi proses pencarian rute pada AODV dapat dilihat pada Gambar 2.2.



Gambar 2.2 Proses pencarian rute pada AODV [8]

Ilustrasi pada Gambar 2.2 menunjukkan bagaimana *source node*, yaitu *node* S mencari rute menuju *destination node*, yaitu *node* D. *Node* S akan membuat paket RREQ dan melakukan *broadcast* kepada semua *node* tetangganya (*neighbor node*). Jika *destination sequence number* yang terdapat pada paket RREQ sama atau lebih kecil dari yang ada pada *routing table* dan rute menuju *node* tujuan belum ditemukan, maka paket tersebut tidak akan dilanjutkan (*drop*). Jika *destination sequence number* pada RREQ lebih besar dibandingkan dengan yang terdapat pada *routing table*, maka *entry* pada *routing table* akan diperbarui dan paket tersebut akan diteruskan oleh *neighbor node* sekaligus membuat *reverse path* menuju *source node.* Paket RREQ akan diteruskan hingga mencapai *node* D. Kemudian, jika rute menuju *node* D sudah terbentuk di dalam *routing table* dan memiliki *routing flags* "*up*", maka *node* F akan mengirimkan paket RREP melalui rute tersebut menuju *node*.

Pada Tugas Akhir ini, penulis menggunakan *routing protocol* AODV yang akan diimplementasikan pada lingkungan VANETs dengan beberapa skenario

2.3 *Simulation of Urban Mobility* (SUMO)

Simulation of Urban Mobility (SUMO) [9] merupakan sebuah aplikasi simulasi mikroskopis *multi-modal traffic* yang bersifat *open source*. SUMO mensimulasikan bagaimana permintaan lalu lintas dari kendaraan yang bergerak melalui jaringan yang diberikan. Simulasi ini memungkinkan untuk mengatasi manajemen lalu lintas. Setiap kendaraan dimodelkan secara eksplisit, memiliki rute sendiri, dan bergerak secara individu. Simulasinya bersifat deterministik namun disediakan beberapa pengaturan untuk pengacakan skenario lalu lintas. Penulis menggunakan beberapa fitur dari aplikasi ini untuk membuat peta skenario *grid* dan mengkonversi output menjadi format yang dapat digunakan pada *network simulator* NS-2.

SUMO terdiri dari beberapa *tools* yang dapat membantu pembuatan simulasi lalu lintas pada tahap-tahap yang berbeda. Berikut penjelasan fungsi *tools* yang digunakan dalam pembuatan Tugas Akhir ini:

- netgenerate
  netgenerate merupakan *tool* yang berfungsi untuk membuat peta berbentuk seperti *grid*, *spider*, dan bahkan *random network*. Sebelum proses netgenerate, pengguna dapat menentukan kecepatan maksimum jalan dan membuat *traffic light* pada peta. Hasil dari netgenerate ini berupa *file* dengan ekstensi .net.xml. Pada Tugas Akhir ini netgenerate digunakan untuk membuat peta skenario *grid*.
- netconvert
  netconvert merupakan program CLI yang berfungsi untuk melakukan konversi dari peta seperti OpenStreetMap menjadi format *native* SUMO. Pada Tugas Akhir ini penulis menggunakan netconvert untuk mengonversi peta dari OpenStreetMap.
- randomTrips.py
  Tool dalam SUMO untuk membuat rute acak yang akan dilalui oleh kendaraan dalam simulasi.
- duarouter
  Tool dalam SUMO untuk melakukan perhitungan rute berdasarkan definisi yang diberikan dan memperbaiki kerusakan rute.
- sumo
  Program yang melakukan simulasi lalu lintas berdasarkan data-data yang didapatkan dari netgenerate (skenario *grid*) atau netconvert dari randomTrips.py. Hasil simulasi dapat di-*export* ke sebuah *file* untuk dikonversi menjadi format lain.
- sumo-gui
  GUI untuk melihat simulasi yang dilakukan oleh SUMO secara grafis.
- traceExporter.py

Tool yang bertujuan untuk mengonversi *output* dari sumo menjadi format yang dapat digunakan pada *simulator* lain. Pada Tugas Akhir ini penulis menggunakan traceExporter.py untuk mengonversi data menjadi format .tcl yang dapat digunakan pada NS-2

Pada Tugas Akhir ini, penulis menggunakan SUMO untuk menghasilkan skenario VANETs, peta area simulasi, dan pergerakan *node* sehingga menyerupai keadaan lalu lintas yang sebenarnya. Untuk setiap skenario VANETs yang dibuat menggunakan SUMO, akan dihasilkan pergerakan *node* yang acak sehingga setiap skenario memiliki pergerakan yang berbeda.

2.4 OpenStreetMap (OSM)

OpenStreetMap (OSM) [10] adalah sebuah proyek yang dirancang untuk membuat sebuah peta dunia yang dapat diubah oleh siapapun secara bebas. Dua faktor pendukung dalam perkembangan OSM adalah ketersediaan informasi peta sebagian besar daerah di dunia dan munculnya alat navigasi portabel yang terjangkau. OSM dianggap sebagai pedoman dalam informasi geografis yang diberikan secara sukarela. Hingga saat ini OSM telah memiliki lebih dari dua juta pengguna yang terdaftar yang dapat mengambil data menggunakan survey, alat GPS, *aerial photography*, dan sumber lainnya. Data-data pada OSM berlisensi *Open Database License* sehingga dapat digunakan bebas oleh semua orang.

Pada Tugas Akhir ini, penulis menggunakan data yang tersedia pada OSM untuk membuat skenario lalu lintas berdasarkan peta daerah di Surabaya. Peta yang diambil lalu digunakan untuk simulasi skenario *rea*l VANETs.

2.5 *Java OpenStreetMap Editor* (JOSM)

JOSM [11] merupakan aplikasi berbasis Java yang digunakan untuk menyunting data yang dihasilkan dari OpenStreetMap (OSM). Aplikasi ini dapat diunduh di halaman josm.openstreetmap.de dan tidak membutuhkan koneksi internet pada proses pemakaiannya.

Pada Tugas Akhir ini, penulis menggunakan aplikasi ini untuk menyunting dan merapikan peta yang diunduh dari OpenStreetMap yaitu dengan menghilangkan dan menyambungkan jalan yang ada. Penyuntingan juga dilakukan dengan menghilangkan gedung – gedung yang ada di peta.

2.6 AWK

AWK merupakan *Domain Specific Language* (DSL) yang dirancang untuk *text processing* dan digunakan sebagai alat ekstraksi data [12]. AWK bersifat *data-driven* yang berisi perintah-perintah yang akan dijalankan pada data tekstual secara langsung pada file atau digunakan sebagai bagian dari *pipeline*.

Pada Tugas Akhir ini AWK digunakan untuk memproses data dari hasil simulasi NS-2 dan menganalisis *packet delivery ratio, End-to-End Delay, dan Routing Overhead.*

2.7 *Network Simulator-2* (NS-2)

NS-2 [13] adalah sebuah *discrete event simulator* yang dirancang untuk membantu penelitian jaringan komputer. Dalam proses simulasinya, NS-2 menggunakan bahasa pemrograman C++ dan OTcl. Bahasa C++ digunakan untuk implementasi bagian-bagian jaringan yang disimulasikan. OTcl digunakan untuk menulis skenario simulasi jaringan.

Pada Tugas Akhir ini, NS-2 digunakan untuk melakukan simulasi lingkungan VANETs menggunakan protokol AODV yang sudah dimodifikasi. *Trace file* yang dihasilkan oleh NS-2 digunakan

untuk mengukur peforma *routing* protokol AODV yang sudah dimodifikasi.

2.7.1 Instalasi NS-2

Sebelum melakukan instalasi NS-2, hal pertama yang harus dilakukan adalah meng-*install* dependensi yang dibutuhkan. Salah satu dependensi dari NS-2 adalah GCC versi 4.9. Gambar 2.5 menunjukkan dependensi NS-2 beserta GCC 4.9 dan cara meng-*install*-nya pada distribusi Debian dan turunannya.

```
sudo apt-get install build-essential autoconf
→ automake libxmu-dev
```

Gambar 2.3 Perintah untuk menginstall dependency NS-2

Setelah semua dependensi ter-*install*, selanjutnya unduh *source code* NS-2. Jika proses unduh sudah selesai, lakukan ekstraksi file tarball seperti yang ditunjukkan pada Gambar 2.4.

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

Gambar 2.4 Perintah untuk mengunduh dan mengekstrak NS-2

Lakukan navigasi menuju folder "linkstate" yang terdapat pada ns-allinone-2.35/ns-2.35/linkstate. Kemudian buka file yang bernama "ls.h" dan pergi ke baris 137 pada kode tersebut. Setelah itu ubah kata "erase" menjadi "this->erase". Potongan kode dari perubahan yang dilakukan pada file tersebut dapat dilihat pada Gambar 2.7.

```
1.  void eraseAll(){this-
    >erase(baseMap::begin(), baseMap::end()); }
```

Gambar 2.5 Baris kode yang diubah pada file ls.h

Dalam Tugas Akhir ini penulis akan menggunakan aggregate initialization sehingga harus menambahkan beberapa opsi CFLAGS dalam Makefile NS. Opsi CFLAGS yang ditambahkan adalah "-std=c++0x -Wno-literal-suffix -Wno-narrowing". Perubahan yang dilakukan dapat dilihat pada Gambar 2.5. Perubahan terakhir yang harus dilakukan adalah mengubah setiap baris yang menggunakan fungsi "hash" menjadi "::hash" pada source code "mdart_adp.cc" di dalam direktori ns-allinone-2.35/ns-2.35/mdart. Perubahan ini dilakukan agar fungsi built-in std::hash dapat dibedakan dengan fungsi hash yang digunakan pada source code "mdart_adp.cc" ketika proses compile.

Setelah semua tahap diatas selesai, jalankan skrip instalasi NS-2 dengan memasukkan perintah ns-allinone-2.35/./install pada terminal dan tunggu hingga proses instalasi selesai.

2.7.2 Skrip OTcl

OTcl [14] merupakan ekstensi object oriented dari bahasa pemrograman Tcl. Pada NS-2, skenario dan pengaturan lingkungan simulasinya menggunakan bahasa OTcl sebagai bahasa scriptingnya. Kelas-kelas yang terdapat pada OTcl terhubung dengan kode C++. Sehingga pembuatan skenario simulasi tidak perlu menggunakan bahasa C++. Gambar 2.6 merupakan contoh kode OTcl pada skenario simulasi NS-2 untuk melakukan pengaturan lingkungan simulasi.

```
1.  set val(chan) Channel/WirelessChannel;
2.  set val(prop) Propagation/TwoRayGround;
3.  set val(netif) Phy/WirelessPhy;
4.  set val(mac) Mac/802_11;
5.  set val(ifq) Queue/DropTail/PriQueue;
6.  set val(ll) LL;
7.  set val(ant) Antenna/OmniAntenna;
8.  set val(ifqlen) 1000;
9.
10. set ns_ [new Simulator]
11.
12. set topo [new Topography]
13. set channel_ [new $val(chan)]
14.     $ns_ node-config
15.     -adhocRouting $val(rp) -llType $val(ll)
16.     -macType $val(mac) -ifqType $val(ifq)
17.     -ifqLen $val(ifqlen) -antType $val(ant)
18.     -propType $val(prop) -phyType $val(netif)
19.     -channel $channel_ -agentTrace ON
20.     -routerTrace ON -macTrace OFF
21.     -movementTrace OFF -topoInstance $topo
```

Gambar 2.6 Potongan kode pengaturan lingkungan simulasi VANETs

Baris 1 hingga baris 8 digunakan untuk mengatur lingkungan simulasi. Kemudian pada baris 10, objek simulator yang akan menjalankan simulasi VANET diinisiasi. Baris 12 menginisiasi koneksi wireless yang akan digunakan oleh setiap node. Pada baris 13, semua variabel pengaturan pada node dimasukkan. Seluruh node akan menggunakan pengaturan yang sama ketika simulasi berjalan.

2.7.3 *Trace File*

Trace file berisikan informasi detail pengiriman paket data. *Trace file* didapatkan dari hasil simulasi yang dilakukan NS-2. *Trace file* digunakan untuk menganalisis performa *routing protocol* yang disimulasikan. Detail penjelasan *trace file* ditunjukkan pada Tabel 2.2.

Tabel 2.2 Detail Penjelasan Trace File AODV

Kolom ke-	Penjelasan	Isi
1	*Event*	s: *sent* r: *received* f: *forwarded* D: *dropped*
2	*Time*	Waktu terjadinya *event*
3	ID *Node*	_x: dari 0 hingga banyak *node* pada topologi
4	*Layer*	AGT: *application* RTR: *routing* LL: *link layer* IFQ: *packet queue* MAC: *MAC* PHY: *physical*
5	*Flag*	--- : Tidak ada
6	*Sequence Number*	Nomor paket
7	*Packet Type*	AODV paket *routing* AODV cbr: berkas paket CBR (*Constant Bit Rate*) RTS: *Request To Send* yang dihasilkan MAC 802.11 CTS: *Clear To Send* yang dihasilkan MAC 802.11 ACK: MAC ACK ARP: Paket *link layer address resolution protocol*
8	Ukuran	Ukuran paket pada *layer* saat itu
9	Detail MAC	[a b c d] a: perkiraan waktu paket b: alamat penerima c: alamat penerima d: IP header
10	*Flag*	-------- : Tidak ada

11	*Detail* IP *source, destination,* dan *nexthop*	[a:b c:d e f] a: IP *source node* b: *port source node* c: IP *destination node* (jika -1 berarti *broadcast*) d: *port destination node* e: IP *header* ttl f: IP *nexthop* (jika 0 berarti *node* 0 atau *broadcast*)

```
r 0.000916039 _69_ RTR --- 0 AODV 44 [0 ffffffff 17 800] –
→ ----[23:255 -1:255 1 0] [0x1 1 [23 2] 4.000000] (HELLO)
```

(a) **Contoh paket HELLO**

```
s 50.000000000 _148_ RTR --- 0 AODV 48 [0 0 0 0] -------
→ [148:255 -1:255 30 0] [0x2 1 1 [149 0] [148 72]]
(REQUEST)
```

(b) **Contoh paket RREQ**

```
s 50.021614475 _149_ RTR --- 0 AODV 44 [0 0 0 0] -------
→ [149:255 148:255 30 3] [0x4 1 [149 68] 10.000000] (REPLY)
```

(c) **Contoh paket RREP**

```
s 62.000000000 _125_ RTR --- 0 AODV 32 [0 0 0 0] -------
→ [125:255 -1:255 1 0] [0x8 1 [148 0] 0.000000] (ERROR)
```

(d) **Contoh paket RRER**

```
r 62.042373176 _149_ AGT --- 12 cbr 532 [13a 95 3f 800] ---
→ ---[148:0 149:0 25 149] [12] 6 0
```

(e) **Contoh paket data**

Gambar 2.7 Contoh Pola Paket Trace File NS-2

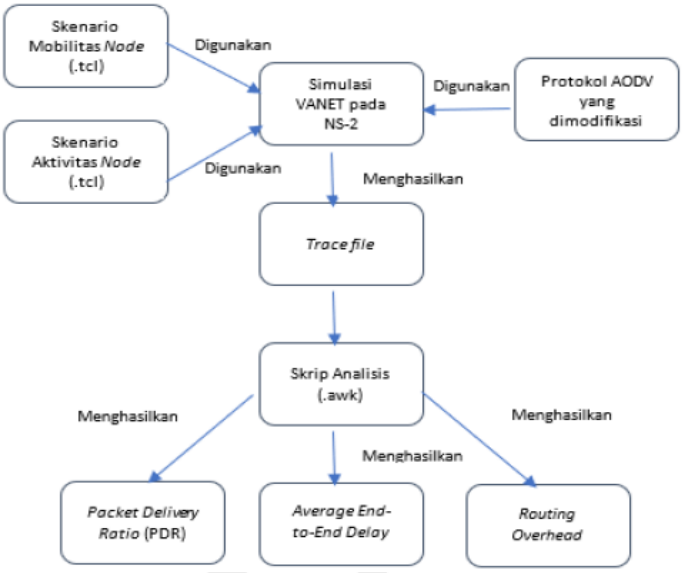
(Halaman ini sengaja dikosongkan)

BAB III
PERANCANGAN

Perancangan merupakan bagian yang sangat penting dari implementasi sistem secara teknis sehingga bab ini akan menjelaskan perancangan sistem yang dibuat dalam Tugas Akhir. Bagian yang dijelaskan pada bab ini berawal dari deskripsi umum sistem hingga perancangan skenario, alur, dan implementasinya.

3.1 Deskripsi Umum

Pada Tugas Akhir ini akan diimplementasikan *routing protocol* AODV dengan memodifikasi pada bagian proses *route discovery* yang dijalankan pada simulator NS-2. Diagram dari rancangan simulasi dari AODV asli dan AODV modifikasi dapat dilihat pada pada Gambar 3.1.



Gambar 3.1 Diagram Alur Rancangan Simulasi

Modifikasi akan diawali dengan menggunakan fungsi HELLO *messages* yang terdapat pada AODV untuk mengetahui informasi Energy dan lokasi setiap node. Setelah mendapatkan informasi tersebut, maka dilakukan penghitungan nilai ED (*Energy/Distance*) Setelah didapatkan nilai ED, maka destination node akan mengecek nilai ED yang terbesar, setelah didapatkan nilai ED terbesar, maka destination node akan mengirim RRER ke souce node yang berisi informasi tersebut yang selanjutnya digunakan untuk pengiriman data.

Dalam Tugas Akhir ini, terdapat 2 jenis skenario yang digunakan sebagai perbandingan pengukuran, yaitu peta berbentuk grid dan peta riil dari lingkungan lalu lintas kota Banyuwangi. Skenario grid dibuat dengan bantuan aplikasi SUMO. Skenario riil yang didasarkan pada peta lalu lintas kota Banyuwangi diambil dari OpenStreetMap dan dirapikan menggunakan aplikasi JOSM. Setelah file peta sudah terbentuk, dilakukan simulasi lalu lintas dengan SUMO. Hasil simulasi SUMO digunakan untuk simulasi protokol AODV modifikasi pada NS-2. Kemudian hasil simulasi NS-2 dianalisis dengan menggunakan skrip AWK untuk menghitung metrik analisis berupa *packet delivery ratio, average end-to-end delay,* dan *routing overhead.* Daftar istilah yang sering digunakan pada buku Tugas Akhir ini dapat dilihat pada Tabel 3.1.

Tabel 3.1 Daftar Istilah

No.	Istilah	Penjelasan
1	AODV	Singkatan dari *Ad hoc On-demand Distance Vector.* Protokol yang digunakan pada Tugas Akhir ini.
2	EE-AODV	*Energy Efficient Ad hoc On-demand Distance Vector Predicting Node Trend.* Protokol AODV yang sudah dimodifikasi.

No.	Istilah	Penjelasan
3	PDR	*Packet Delivery Ratio*. Salah satu metrik analisis yang diukur. Berupa jumlah pengiriman paket yang terkirim.
4	E2E	*Average End-to-End Delay*. Jeda waktu yang diukur saat paket terkirim.
5	RO	*Routing Overhead*. Jumlah *control packet* yang terkirim
6	RREQ	*Route Request*. Paket *request* pada AODV yang dikirim untuk mendapatkan rute
7	RREP	*Route Reply*. Paket *reply* pada AODV yang dikirim ke *node* sumber melalui rute yang sudah terbuat.
8	*Euclidean distance*	Persamaan matematika untuk menghitung jarak antara dua titik.
9	Nilai E/D	Rasio perbandingan *Energy/Distance*

3.2 Perancangan Skenario Mobilitas

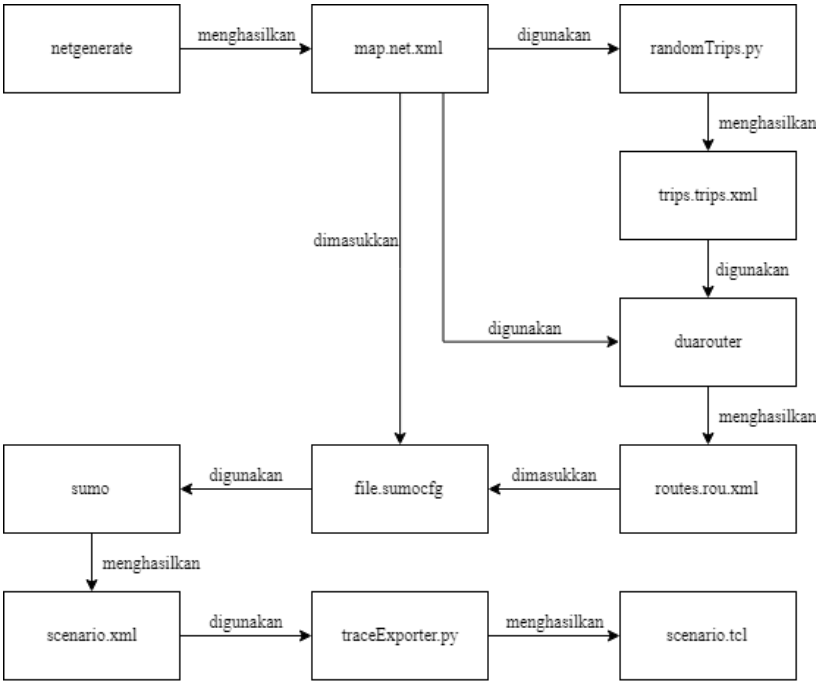
Perancangan skenario mobilitas dimulai dengan membuat area simulasi, pergerakan *node*, dan implementasi pergerakan *node*. Dalam Tugas Akhir ini, terdapat dua macam area simulasi yang akan digunakan yaitu peta *grid* dan *real*. Peta *grid* yang dimaksud adalah bentuk jalan berpetak – petak sebagai contoh jalan berpotongan yang sederhana. Peta *grid* digunakan sebagai simulasi awal VANETs karena lebih stabil. Peta *grid* didapatkan dengan menentukan panjang dan jumlah petak area menggunakan SUMO. Sedangkan yang dimaksud peta *real* adalah peta asli / nyata yang digunakan sebagai area simulasi. Peta *real* didapatkan dengan mengambil daerah yang diinginkan sebagai area simulasi menggunakan OpenStreetMap. Pada Tugas Akhir ini, peta *real* yang diambil penulis adalah salah satu area di kota Banyuwangi.

3.2.1 Perancangan Skenario *Grid*

Perancangan peta *grid* diawali dengan menentukan luas area peta *grid* yang dibutuhkan. Luas area tersebut bisa didapatkan dengan menentukan jumlah titik persimpangan yang diinginkan, secara *default* peta *grid* berbentuk persegi empat, sehingga dari jumlah persimpangan tersebut dapat diketahui berapa banyak petak persegi yang dibutuhkan. Dengan mengetahui jumlah petak yang dibutuhkan, dapat ditentukan panjang tiap petak sehingga mendapatkan luas area yang dibutuhkan yaitu 1000 m x 1000 m. Dengan 4 titik persimpangan, maka akan didapatkan 9 petak dan panjang tiap peta adalah 300m.

Peta *grid* yang telah ditentukan luasnya tersebut kemudian dibuat dengan menggunakan *tools* SUMO yaitu netgenerate. Panjang jalan dan jumlah persimpangan yang sudah ditentukan dimasukkan sebagai argumen untuk netgenerate. Selain titik persimpangan dan panjang tiap petak *grid*, dibutuhkan juga pengaturan kecepatan kendaraan menggunakan *tools* tersebut. Peta *grid* yang dihasilkan oleh netgenerate akan memiliki ekstensi .net.xml. Peta *grid* ini kemudian digunakan untuk membuat pergerakan *node* dengan *tools* SUMO yaitu menggunakan *tools* randomTrips lalu diproses oleh duarouter untuk memperbaiki masalah konektivitas rute (jika ada).

Skenario mobilitas *grid* dihasilkan dengan menggabungkan *file* peta *grid* dan *file* pergerakan *node* yang telah dibuat. Penggabungan tersebut menghasilkan *file* dengan ekstensi .xml. Selanjutnya, untuk dapat diterapkan pada NS-2, *file* skenario mobilitas *grid* yang berekstensi .xml dikonversi ke dalam bentuk *file* .tcl. Konversi ini dilakukan menggunakan *tool* traceExporter. Alur pembuatan peta *grid* dapat dilihat pada Gambar 3.2.



Gambar 3.2 Alur perancangan skenario Grid

3.2.2 Perancangan Skenario *Real*

Perancangan skenario *real* diawali dengan memilih daerah yang akan dijadikan model simulasi. Pada Tugas Akhir ini, digunakan peta dari OpenStreetMap untuk mengambil peta area yang ditentukan. Setelah memilih area, dilakukan pengunduhan dengan menggunakan fitur *export* yang telah disediakan oleh OpenStreetMap. Peta hasil *export* tersebut memiliki ekstensi .osm. Selanjutnya potongan peta tersebut dirapikan menggunakan JOSM dengan menghapus jalan yang terputus dari potongan peta tersebut sehingga menjadi daerah yang tertutup.

Setelah mendapatkan peta area yang akan dijadikan simulasi, peta tersebut dikonversi ke dalam bentuk *file* dengan ekstensi .net.xml menggunakan *tools* SUMO yaitu netconvert. Tahap berikutnya memiliki tahapan yang sama seperti merancang peta skenario *grid*, yaitu membuat pergerakan *node* menggunakan randomTrips dan duarouter. Kemudian dilakukan penggabungan *file* peta *real* yang sudah dikonversi ke dalam *file* dengan ekstensi .net.xml dan *file* pergerakan *node* yang sudah dibuat sebelumnya. Hasil dari penggabungan tersebut merupakan *file* skenario berkekstensi .xml. *File* yang dihasilkan tersebut dikonversi ke dalam bentuk *file* dengan ekstensi .tcl agar dapat diterapkan pada NS-2. Alur pembuatan peta *real* dapat dilihat pada Gambar 3.3.

Gambar 3.3 Alur perancangan skenario real

3.3 Perancangan Modifikasi *Routing Protocol* AODV

Proses *route* discovery pada *routing protocol* AODV yang telah dimodifikasi diawali dengan mendapatkan nilai Energy dan lokasi dari masing-masing *node* dengan cara source *membroadcast* RREQ ke node tetangga. Setelah mendapatkan informasi tersebut, maka akan dihitung nilai E/D(Energy/Distance). Perhitungan jarakdihitung menggunakan persamaan *Euclidean Distance*,

sedangkan perhitungan energy dipilih berdasarkan node yang memiliki nilai energy paling besar saat itu. Setelah diketahui nilai E/D terbesar, maka destination akan mengirim RRER yang berisi informasi nilai E/D tersebut ke source node.

3.3.1 Perancangan Perhitungan Nilai Energy dan Jarak

Pada modifikasi AODV terdapat nilai MRE (Maxinmum Residual Energy). Nilai *Maximum Residual Energy* (MRE) dapat dihitung dengan menyimpan nilai sisa energi sebuah node di field MRE pada paket RREQ dan akan dibandingkan dengan sisa energi node selanjutnya, jika terdapat energi yang lebih besar maka nilai MRE akan di-update dengan nilai sisa energi node tersebut. Selain itu, *Sum Residual Energy* (SRE) dapat dihitung dengan menambahkan seluruh nilai sisa energi dalam sebuah rute. Adapun perhitungan jarak menggunakan rumus *Euclidean Distance* untuk menghitung jarak antar 2 node. Nilai MRE, SRE dan perhitungan Jarak dapat dilihat pada persamaan 3.1, 3.2, dan 3.3.

$$\mathrm{MRE} = \max_{\forall ni \in rj}(r(ni)) \tag{3.1}$$

$$\mathrm{SRE} = \Sigma(r(ni)) \tag{3.2}$$

$$\mathrm{Distance} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{3.3}$$

Keterangan:

r = Rute
n = *Node* dalam rute

q_i : Titik koordinat tujuan
p_i : Titik koordinat awal

3.3.2 Perancangan Perhitungan Nilai ED

Pada AODV modifikasi, ketika paket RREQ sampai ke node tujuan, nilai ED (*Energy/Distance*) akan dihitung. Nilai ED dapat dihitung dengan cara nilai SRE *(Sum Residual Energy)* dibagi dengan *Distance*. Rumus perhitungan nilai D dapat dilihat pada Persamaan 3.4

$$\mathrm{ED} = \frac{\mathrm{SRE}}{\mathrm{Distance}} \tag{3.4}$$

Keterangan:

3.3.3 Perancangan Pemilihan Rute

Setelah paket RREQ sampai ke node tujuan maka akan diperiksa apakah paket RREQ merupakan paket baru, jika paket RREQ merupakan paket baru maka nilai ED akan dihitung dan disimpan di routing table.

Jika paket menerima paket RREQ lagi dan waiting time belum expired maka akan dibandingkan nilai ED paket RREQ yang baru sampai dengan nilai ED yang sudah disimpan di routing table, apabila nilai ED dari paket RREQ yang baru lebih besar maka nilai ED di routing table akan di-update dengan nilai ED dari paket RREQ yang baru. Sebaliknya, jika nilai ED dari paket RREQ yang baru lebih kecil. maka paket RREQ yang baru akan dibuang (discard). Hal ini akan terus dilakukan hingga waiting time habis (expired). Jika waiting

time habis, maka node tujuan akan mengirimkan paket RREP kembali ke node sumber sesuai dengan rute yang terdapat di routing table atau yang memiliki nilai ED terbesar. Pseudocode pemilihan rute RREP dapat dilihat pada Gambar 3.4

```
if (rtr_size > 1)
      new_ED = Calculate_ ED(…)
      ED = calculate_ED(…)

If (new_ED > ED)
      Rt_update
Else if
      drop
```

Gambar 3.4 Pseudocode Pemilihan Rute

3.3.4 *Packet Delivery Ratio* (PDR)

Packet delivery ratio merupakan perbandingan dari jumlah paket data yang dikirim dengan paket data yang diterima. PDR dapat menunjukkan keberhasilan paket yang dikirimkan. Semakin tinggi PDR artinya semakin baik pengiriman paket yang dilakukan. Rumus untuk menghitung PDR dapat dilihat pada persamaan 3.7.

$$\text{PDR} = \frac{\text{received}}{\text{sent}} \text{x } 100\ \% \tag{3.5}$$

dimana,
received : Banyak paket data yang diterima

sent : Banyak paket data yang dikirimkan

3.3.5 *Average End-to-End Delay* (E2E)

Average End-to-End Delay merupakan perhitungan rata-rata waktu ketika paket data diterima dan waktu ketika paket dikirimkan dalam satuan detik. *Delay* tiap paket didapat dari rentang waktu antara *node* asal saat mengirimkan paket dan *node* tujuan menerima paket. Semua paket, termasuk delay yang diakibatkan oleh paket routing juga diperhitungkan. Delay tiap paket tersebut semua dijumlahkan dan dibagi dengan jumlah paket yang berhasil diterima. Berikut adalah persamaan untuk menghitung E2E:

$$\mathrm{E2E} = \frac{\sum_{i=1}^{n} \mathrm{RecvTime} - \mathrm{SentTime}}{n} \tag{3.6}$$

dimana,

RecvTime : Waktu *node* asal mengirimkan paket
SentTime : Waktu *node* tujuan menerima paket
n : Jumlah paket yang berhasil diterima

3.3.6 *Routing Overhead* (RO)

Routing Overhead adalah jumlah paket kontrol *routing* yang ditransmisikan per data paket ke *node* tujuan selama simulasi terjadi. *Routing Overhead* didapatkan dengan menjumlahkan semua paket kontrol *routing* yang ditransmisikan, baik itu paket *route request* (RREQ), *route reply* (RREP), maupun *route error* (RERR). Perhitungan *Routing Overhead* dapat dilihat dengan persamaan 3.9.

$$\mathrm{RO} = \sum_{i=1}^{n} \mathrm{packet\ sent}$$

(3.7

dimana,

packet sent	: Paket RREQ, RREP, dan RRER yang dikirimkan
n	: Jumlah paket yang dikirimkan

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi berupa kode sumber untuk membangun program.

4.1 Implementasi Skenario Mobilitas

Implementasi skenario mobilitas VANETs dibagi menjadi dua, yaitu skenario *grid* yang menggunakan peta jalan berpetak dan skenario *real* yang menggunakan peta hasil pengambilan suatu area di kota Surabaya.

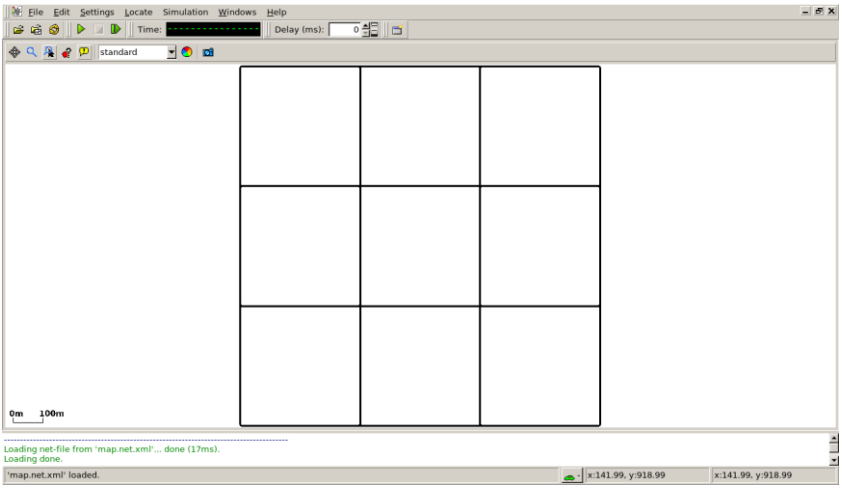
4.1.1 Skenario *Grid*

Dalam mengimplementasikan skenario *grid*, SUMO menyediakan *tools* untuk membuat peta *grid* yaitu netgenerate. Pada Tugas Akhir ini, penulis membuat peta *grid* dengan luas 1000 m x 1000 m yang terdiri dari titik persimpangan antara jalan vertikal dan jalan horisontal sebanyak 4 titik x 4 titik. Dengan jumlah titik persimpangan sebanyak 4 titik tersebut, makat terbentuk 9 buah petak. Sehingga untuk mencapai luas area sebesar 1000 m x 1000 m dibutuhkan luas per petak sebesar 300 m x 300 m. Berikut perintah netgenerate untuk membuat peta tersebut dengan kecepatan *default* kendaraan sebesar 40 km/h atau setara dengan 11 m/s dapat dilihat pada Gambar 4.1.

```
netgenerate --grid --grid.number=4 --
grid.length=300 --default.speed=11 --
tls.guess=1 --output-file=map.net.xml
```

Gambar 4.1 Perintah netgenerate

Setelah itu akan didapat *file* peta berekstensi .xml. Gambar hasil peta yang telah dibuat dengan netgenerate dapat dilihat pada Gambar 4.2.



Gambar 4.2 Hasil Generate Peta Grid

Setelah peta terbentuk, maka dilakukan pembuatan *node* dan pergerakan *node* dengan menentukan titik awal dan titik akhir setiap *node* secara random menggunakan *tools* randomTrips yang terdapat di SUMO. Perintah penggunaan *tools* randomTrips untuk membuat *node* sebanyak *n node* dengan pergerakannya dapat dilihat pada Gambar 4.3.

```
python $SUMO_HOME/tools/randomTrips.py -n
map.net.xml -e 98 -l --trip-
attributes="departLane=\"best\"
departSpeed=\"max\"
departPos=\"random_free\"" -o trip.trips.xml
```

Gambar 4.3 Perintah randomTrips

Selanjutnya dibuatkan rute yang digunakan kendaraan untuk mencapai tujuan dari *file* hasil sebelumnya menggunakan *tools* duarouter. Perintah penggunaan *tools* duarouter dapat dilihat pada Gambar 4.4.

```
duarouter -n map.net.xml -t trip.trips.xml -
o route.rou.xml --ignore-errors --repair
```

Gambar 4.4 Perintah duarouter

Ketika menggunakan *tools* duarouter, SUMO memastikan bahwa jalur untuk *node-node* yang di*generate* tidak akan melenceng dari jalur peta yang sudah di*generate* menggunakan *tools* randomTrips. Selanjutnya untuk menjadikan peta dan pergerakan *node* yang telah di*generate* menjadi sebuah skenario dalam bentuk *file* berekstensi .xml, dibutuhkan sebuah *file* skrip dengan ekstensi .sumocfg untuk menggabungkan *file* peta dan rute pergerakan *node*. Isi dari *file* skrip .sumocfg dapat dilihat pada Gambar 4.5.

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <configuration xmlns:xsi="http://www.w3.org/2001/XMLSc
    hema-
    instance" xsi:noNamespaceSchemaLocation="http://sumo.d
    lr.de/xsd/sumoConfiguration.xsd">
3.     <input>
4.        <net-file value="map.net.xml"/>
5.        <route-files value="routes.rou.xml"/>
6.     </input>
7.     <time>
8.        <begin value="0"/>
9.        <end value="200"/>
10.    </time>
11. </configuration>
```

Gambar 4.5 File Skrip .sumocfg

File .sumocfg disimpan dalam direktori yang sama dengan *file* peta dan *file* rute pergerakan *node*. Untuk percobaan sebelum dikonversi, *file* .sumocfg dapat dibuka dengan menggunakan *tools* sumo-gui. Kemudian buat *file* skenario dalam bentuk *file* .xml dari sebuah *file* skrip berekstensi .sumocfg menggunakan *tools* SUMO. Perintah untuk menggunakan *tools* SUMO dapat dilihat pada Gambar 4.6.

```
sumo -c file.sumocfg --fcd-output
scenario.xml
```

Gambar 4.6 Perintah SUMO untuk membuat skenario .xml

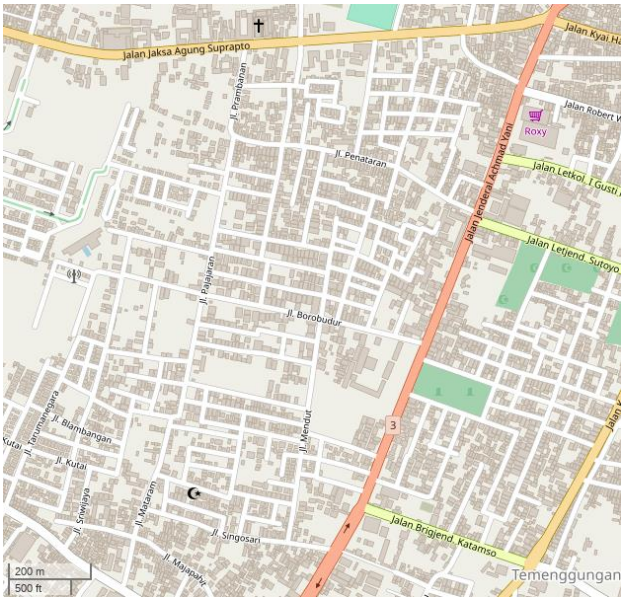
File skenario berekstensi .xml selanjutnya dikonversi ke dalam bentuk *file* berekstensi .tcl agar dapat disimulasikan menggunakan NS-2. *Tools* yang digunakn untuk melakukan konversi ini adalah traceExporter. Perintah untuk menggunakan traceExporter dapat dilihat pada Gambar 4.7.

```
python $SUMO_HOME/tools/traceExporter.py --
fcd-input=scenario.xml --ns2mobility-
output=scenario.tcl
```

Gambar 4.7 Perintah traceExporter

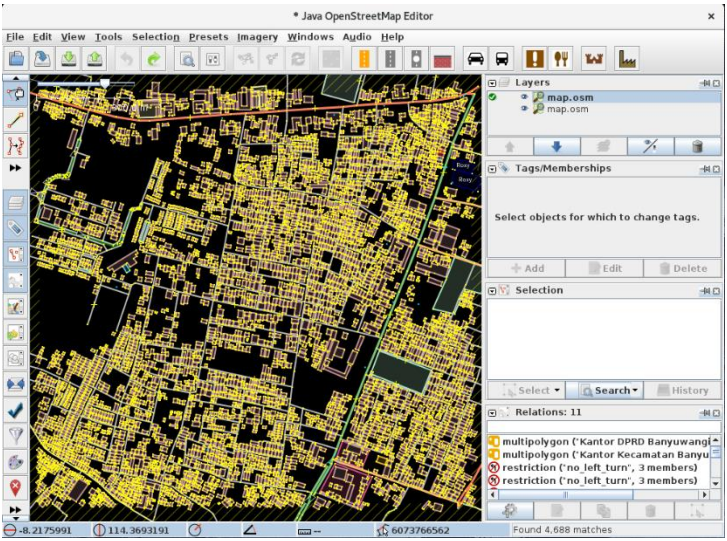
4.1.2 Skenario *Real*

Dalam mengimplementasikan skenario *real*, langkah pertama adalah menentukan area yang akan dijadikan area simulasi. Pada Tugas Akhir ini penulis mengambil area jalan sekitar Jl. Jenderal Ahmad Yani, Banyuwangi. Setelah menentukan area simulasi, ekspor data peta dari OpenStreetMap seperti yang ditunjukkan pada Gambar 4.8.

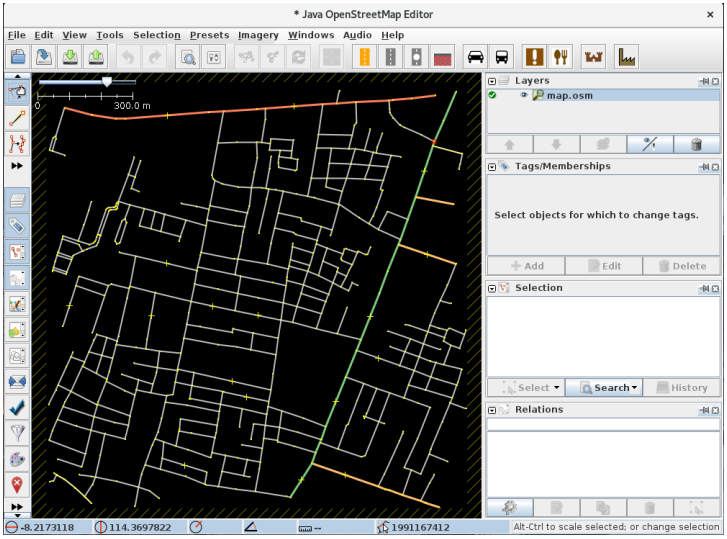


Gambar 4.8 Ekspor Peta dari OpenStreetMap

File hasil ekspor dari OpenStreetMap tersebut adalah *file* peta dengan ekstensi .osm. Kemudian peta tersebut disunting melalui program JOSM. Tujuannya adalah untuk menghapus jalan dan objek-objek yang tidag digunakan untuk simulasi. Kemudian peta yang sudah disunting disimpan dengan ekstensi .osm. Cuplikan dari proses penyuntingan pada JOSM dapat dilihat pada Gambar 4.9 dan 4.10.



Gambar 4.9 Cuplikan Peta Sebelum Proses Penyuntingan pada JOSM



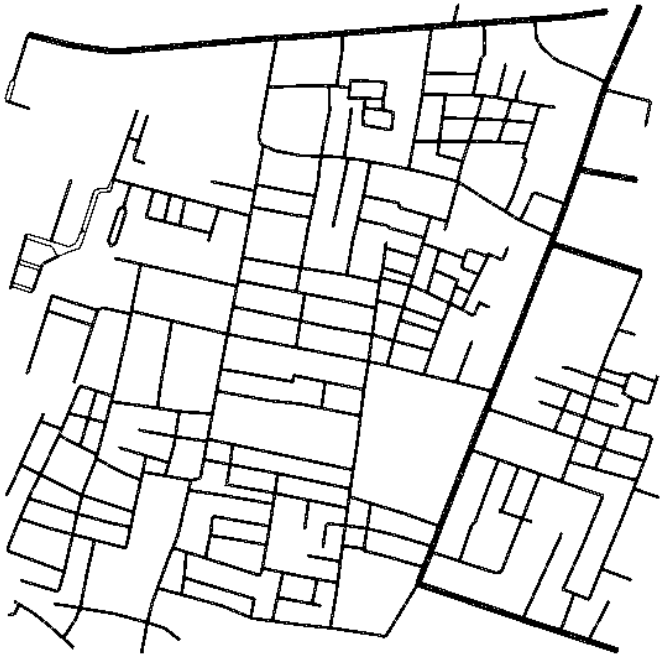
Gambar 4.10 Cuplikan Peta Setelah Proses Penyuntingan pada JOSM

Setelah proses penyuntingan peta selesai, dilakukan konversi *file* .osm tersebut menjadi peta dalam bentuk *file* berekstensi .xml menggunakan *tools* netconvert dari SUMO. Perintah untuk menggunakan netconvert dapat dilihat pada Gambar 4.11.

```
netconvert --osm-files map.osm --output-
file map.net.xml
```

Gambar 4.11 Perintah netconvert

Hasil konversi peta dari *file* berekstensi .osm menjadi *file* berekstensi .xml dapat dilihat menggunakan *tools* sumo-gui seperti yang ditunjukkan pada Gambar 4.12.



Gambar 4.12 Hasil Konversi Peta Real

Langkah selanjutnya sama dengan ketika membuat skenario mobilitas *grid*, yaitu membuat *node* asal dan *node* tujuan menggunakan *tool* randomTrips. Lalu membuat rute *node* untuk sampai ke tujuan menggunakan *tool* duarouter. Kemudian membuat *file* skenario berekstensi .xml menggunakan *tool* SUMO dengan bantuan *file* skrip berekstensi .sumocfg. Selanjutnya dilakukan konversi *file* skenario berekstensi .tcl untuk dapat disimulasikan pada NS-2 menggunakan *tool* traceExporter. Perintah untuk menggunakan *tools* tersebut sama dengan ketika membuat skenario *grid* di atas.

4.2 Implementasi Modifikasi pada *Routing Protocol* AODV untuk Pemilihan Rute

Pada Tugas Akhir ini dilakukan modifikasi pada *routing protocol* AODV untuk pemilihan rute secara optimal berdasarkan Tingkat Energy dan jarak *(Energy/Distance)*. Sehingga pada Tugas Akhir ini bisa dilihat performa pada *routing protocol* AODV dapat meningkat.

Implementasi modifikasi *routing protocol* ini dibagi menjadi tiga bagian yaitu:

- Implementasi Penghitungan Energy dan Jarak
- Implementasi Penghitungan Nilai ED
- Implementasi Pemilihan Rute

Kode implementasi dari *routing protocol* AODV pada NS-2 versi 2.35 berada pada direktori ns-2.35/aodv. Di dalam direktori tersebut terdapat beberapa *file* diantaranya seperti aodv.h, aodv.cc, dan sebagainya. Pada Tugas Akhir ini penulis memodifikasi *file* aodv.cc untuk dapat memilih tingkat energy dan menghitung jarak dan pemilihan rute. Pada bagian ini penulis akan menjelaskan langkah-langkah dalam mengimplemetnasikan modifikasi *routing protocol* AODV.

4.2.1 Implementasi Penghitungan Energy dan Jarak

Langkah awal yang dilakukan untuk menghitung nilai Minimum Residual Energy (MRE) dan Sum of Residual Energy (SRE) seperti yang telah dirancang pada subbab 3.3.1 . Nilai MRE akan dihitung dengan cara membandingkan nilai energi di node dengan nilai MRE di field, jika nilai MRE lebih besar, nilai di *field* RREQ akan di-*update.* Sedangkan, untuk menghitung nilai SRE adalah dengan menambahkan energi di node dengan jumlah energi di *field* SRE pada RREQ. Perhitungan jarak antar 2 node menggunakan persamaan Euclidean Distance disimpan pada field *minD.* Sedangkan, untuk menghitung nilai jarak adalah dengan menambahkan jumlah minD di *field* Distance pada RREQ.Perhitungan MRE, SRE dan minD termasuk dalan fungsi AODV::recvRequest pada kode sumber aodv.cc yang terdapat pada folder ns2.3.5/aodv. Untuk potongan kode tersebut bisa dilihat pada Gambar 4.13

```
Node* thisnode =
Node::get_node_by_address(index);
double nowenergy = thisnode->energy_model()-
>energy();
MobileNode *mn;
mn = (MobileNode *)
(Node::get_node_by_address(rq->rq_dst));
MobileNode *mn1,*mn2;

#define DISTANCE(x0,y0,z0,x1,y1,z1) (sqrt(
pow(x0-x1,2) + pow(y0-y1,2) + pow(z0-z1,2)))

double myX, myY, myZ, chX,chY,chZ,currD,
minD;
```

```
/*Mendapatkan kordinat node*/
  mn1 =
(MobileNode*)Node::get_node_by_address(index);
  myX = mn1->X();
  myY = mn1->Y();
  myZ = mn1->Z();

  mn2 =
(MobileNode*)Node::get_node_by_address(rq-
>rq_dst);
  chX = mn2->X();
  chY = mn2->Y();
  chZ = mn2->Z();


  /*Menghitung jarak*/
  minD = DISTANCE(myX,myY,myZ,chX,chY,chZ);


            FILE * pFile;
            pFile = fopen (
"energy_distance_table" , "a" );
            fprintf (pFile, " I recieve the rqt
!!! node # (recv) %d is in permit table has
energy = %f distance between node_i and Node_j is
%f at time = %f
\n",index,nowenergy,minD,CURRENT_TIME);
            fclose (pFile);
```

Gambar 4.13 Potongan Kode Perhitungan Energy dan Jarak

4.2.2 Implementasi Penghitungan Nilai ED

Pada tahap selanjutnya dilakukan perhitungan Nilai ED *(Energy/Distance)*. Perhitungan ini dilakukan pada fungsi AODV::calculate_ed yang terletak pada kode sumber aodv.cc yang terletak pada ns2.35/aodv. Nilai energy (SRE) disimpan dalam variable *nowenergy*, dan jarak disimpan dalam variable *Distance*. Potongan kode untuk Penghitungan Nilai ED dapat dilihat pada Gambar 4.14

```
double AODV::calculate_ed(double nowenergy,
double distance){
double fed;

if (distance > 0)
      fed= (nowenergy/distance);
else
      return 0;
}
```

Gambar 4.14 Potongan Kode Penghitungan Nilai ED

4.2.3 Implementasi Pemilihan Rute

Pada tahap selanjutnya setelah penghitungan nilai D, fungsi AODV::recvReqToRep. Fungsi ini untuk menyimpan nilai ED di dalam routing table, lalu fungsi akan menunggu *time expired*, jika belum expired maka node tujuan akan menunggu paket RREQ baru yang masuk dengan nilai ED yang baru. Apabila nilai ED yang baru lebih tinggi maka routing table akan di-update dengan nilai ED yang baru, jika tidak memenuhi maka paket RREQ di-discard. Setelah itu, maka node tujuan akan mengirimkan RREP sesuai dengan rute dengan nilai ED yang terdapat di *routing table*. Potongan kode untuk pemilihan rute dapat dilihat pada Gambar 4.15

```
void AODV::recvReqToRep(){
if (new_ED  > ED){
      rt_update(rt,
arr_rtr[rtr_id.back()].seqno,
arr_rtr[rtr_id.back()].src, (CURRENT_TIME +
REV_ROUTE_LIFE), ED);
      #ifdef DEBUG
        FILE *fp;
        fp = fopen("debug.txt", "a");
        fprintf(fp, "\n%f %s function: update
      rt, new ED > old ED\n", CURRENT_TIME,
      __FUNCTION__);
        fclose(fp);
      #endif
}
      else {
      rtr_id.erase(rtr_id.begin() + index);
}
```

Gambar 4.15 Potongan Kode Pemilihan Rute

4.3 Implementasi Simulasi pada NS-2

Implementasi simulasi VANETs diawali dengan pengaturan lingkungan simulasi pada sebuah *file* OTcl. *File* ini berisikan konfigurasi setiap *node* dan langkah-langkah yang dilakukan selama simulasi. Potongan konfigurasi lingkungan simulasi dapat dilihat pada Gambar 4.29.

```
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) 1000;
set val(nn) 100;
set val(seed) 1.0;
set val(rp) AODV;
set val(stop) 300;
set val(activityfile) "cbr.tcl";
set val(mobilityfile) "scenario.tcl";

# Initialize ns
set ns_ [new Simulator]
set tracefd [open trace.tr w]
$ns_ trace-all $tracefd

# Set up topography object
set topo [new Topography]
$topo load_flatgrid 1000 1000
set god_ [create-god $val(nn)]
set channel_ [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 $channel_ -agentTrace ON
    -routerTrace ON -macTrace OFF
    -movementTrace OFF -topoInstance $topo
```

Gambar 4.16 Potongan Kode Konfigurasi Simulasi NS-2

Pada konfigurasi dilakukan pemanggilan terhadap *activity file* yang berisikan konfigurasi *node* asal, *node* tujuan, dan pengiriman paket, serta *mobility file* yang berisi pergerakan *node* yang telah di*generate* oleh SUMO. Penjelasan dari pengaturan *node* dapat dilihat

pada Tabel 4.1. Kode implementasi selengkapnya dapat dilihat pada lampiran A.1 Kode Skenario NS-2.

Tabel 4.1 Penjelasan Parameter Pengaturan Node

Parameter	Value	Penjelasan
adhocRouting	AODV	Menggunakan *Routing Protocol* AODV
llType	LL	Menggunakan *link layer* standar
mactType	Mac/802_11	Menggunakan tipe MAC 802.11 karena komunikasi data bersifat wireless
ifqType	Queue/DropTail/PriQueue	Menggunakan priority queue sebagai antrian paket dan paket yang dihapus saat antrian penuh adalah paket yang paling baru
ifqLen	50	Jumlah maksimal paket pada antrian
antType	Antenna/OmniAntenna	Jenis antena yang digunakan adalah omni antenna
propType	Propagation/TwoRayGround	Tipe propagasi sinyal wireless adalah two-ray ground
phyType	Phy/WirelessPhy	Komunikasi menggunakan media nirkabel
channel	Channel/WirelessChannel	Kanal komunikasi yang digunakan
topoInstance	$topo	Topologi yang digunakan daat menjalankan skenario
agentTrace	ON	Pengaktifan pencatatan aktifitas dari agen routing protocol
routerTrace	ON	Pengaktifan pencatatan pada aktifitas routing protocol

macTrace	OFF	Matikan pencatatan MAC layer pada trace file
movementTrace	OFF	Matikan pencatatan pergerakan *node*

Konfigurasi untuk *activity file* bisa dilakukan dengan membuat *file* berekstensi .tcl untuk menyimpan konfigurasi tersebut. Potongan konfigurasi *file traffic* dapat dilihat pada Gambar 4.30.

```
1.  set udp_(0) [new Agent/UDP]
2.  $ns_ attach-agent $node_(98) $udp_(0)
3.  set null_(0) [new Agent/Null]
4.  $ns_ attach-agent $node_(99) $null_(0)
5.  set cbr_(0) [new Application/Traffic/CBR]
6.  $cbr_(0) set packetSize_ 512
7.  $cbr_(0) set interval_ 1
8.  $cbr_(0) set random_ 1
9.  $cbr_(0) set maxpkts_ 10000
10. $cbr_(0) attach-agent $udp_(0)
11. $ns_ connect $udp_(0) $null_(0)
12. $ns_ at 2.5568388786897245 "$cbr_(0) start"
```

Gambar 4.17 Implementasi Simulasi Activity File

Pada konfigurasi tersebut, ditentukan *node* sumber dan *node* tujuan pengiriman paket. Pengiriman dimulai pada detik ke- 2.55. Implementasi konfigurasi *activity file* untuk simulasi pada NS-2 dapat dilihat pada lampiran A.2 Kode Konfigurasi *Traffic*.

Skenario simulasi dijalankan dengan perintah pada Gambar 4.31. Setelah simulasi selesai akan menghasilkan *output* berupa trace file hasil simulasi yang akan digunakan untuk analisis. Isi dari trace file tersebut adalah catatan seluruh event dari setiap paket yang tersebar di dalam lingkungan simulasi.

```
ns scenario.tcl
```

Gambar 4.18 Perintah Untuk Menjalankan Simulasi NS-2

4.4 Implementasi Metrik Analisis

Simulasi yang telah dijalankan oleh NS-2 menghasilkan sebuah *trace file* yang berisikan data mengenai apa saja yang terjadi selama simulasi. *Trace file* disimpan dalam bentuk *file* berekstensi .tr. Data *trace file* tersebut digunakan untuk bahan analisis performa *routing protocol* dengan diukur dalam beberapa metrik parameter. Pada Tugas Akhir ini, metrik yang akan dianalisis adalah PDR, E2E, dan RO.

4.4.1 Implementasi *Packet Delivery Ratio* (PDR)

PDR didapatkan dengan cara menghitung setiap baris terjadinya *event* pengiriman dan penerimaan paket data yang dikirim melalui agen pada *trace file*. Skrip menyaring setiap baris yang mengandung *string* AGT karena kata kunci tersebut menunjukan *event* yang berhubungan dengan paket komunikasi data. Penghitungan dilakukan dengan menjumlahkan paket yang dikirimkan dan paket yang diterima dengan menggunakan karakter pada kolom pertama sebagai *filter*. Kolom pertama menunjukan event yang terjadi dari sebuah paket. Setelah itu nilai PDR dihitung dengan cara persamaan 3.7. Pseudocode untuk menghitung PDR dapat dilihat pada Gambar 4.32. Skrip awk untuk menghitung PDR dapat dilihat pada lampiran A.7 Kode Skrip AWK *Packet Delivery Ratio*.

```
for i = 1 to number of rows
      if in a row contains "s" and in AGT then
            sent increment
      if in a row contains "r" and in AGT then
            received increment
      end if
end for
PDR <- received / sent
```

Gambar 4.19 Pseudocode untuk Menghitung PDR

Contoh perintah untuk memanggil skrip AWK untuk menghitung PDR pada trace file dan contoh keluarannya dapat dilihat pada Gambar 4.33 dan 4.34.

```
awk -f pdr.awk tracefile.tr
```

Gambar 4.20 Perintah Untuk Menjalankan Skrip AWK Perhitungan PDR

```
Sent : 300  Recv : 217  Ratio :  0.7233
```

Gambar 4.21 Contoh Keluaran dari Skrip Perhitungan PDR

4.4.2 Implementasi *Average End-to-End Delay (E2E)*

Dalam perhitungan E2E yang perlu diperhatikan adalah waktu dari sebuah *event* yang tercatat pada kolom ke-2 dengan *filter event* pada kolom ke-4 yaitu layer AGT dan *event* pada kolom pertama untuk membedakan paket yang dikirim atau diterima. Setelah seluruh baris yang memenuhi didapatkan, akan dihitung *delay* dari paket dengan mengurangi waktu dari paket diterima dengan waktu dari paket dikirim dengan memiliki *id* paket yang sama.

Setelah mendapatkan *delay* paket, langkah selanjutnya adalah dengan mencari rata-rata dari *delay* tersebut dengan menjumlahkan semua *delay* paket dan membaginya dengan jumlah paket. Formula perhitungan RO dapat dilihat pada persamaan 3.8. *Pseudocode* untuk menghitung rata-rata E2E dapat dilihat pada Gambar 4.35. Skrip awk untuk menghitung E2E dapat dilihat pada lampiran A.8 Kode Skrip AWK Rata-Rata *End-to-End Delay*.

```
for i = 1 to number of rows
     counter increment
     if layer is AGT and event is "s" then
      start_time[packet_id] <- time
     if layer is AGT and event is "r" then
      end_time[packet_id] <- time
     end if
     delay[packet_id] <- end_time[packet_id] -
start_time[packet_id]
     sum_delay += delay[packet_id]
end for
E2E <- sum delay / counter
```

Gambar 4.22 Pseudocode untuk Perhitungan Rata-Rata E2E

Contoh perintah untuk memanggil skrip AWK untuk menghitung E2E pada trace file dan contoh keluarannya dapat dilihat pada Gambar 4.36 dan 4.37.

```
awk -f e2e.awk tracefile.tr
```

Gambar 4.23 Perintah Untuk Menjalankan Skrip AWK Perhitungan E2E

```
E2E (ms) : 87.4493
```

Gambar 4.24 Contoh Keluaran dari Skrip Perhitungan E2E

4.4.3 Implementasi *Routing Overhead* (RO)

Untuk menghitung RO yang perlu dilakukan adalah menjumlahkan setiap paket dengan *filter event sent* pada kolom pertama dan *event layer* RTR pada kolom ke-4. Formula perhitungan RO dapat dilihat pada persamaan 3.9. *Pseudocode* untuk menghitung RO dapat dilihat pada Gambar 4.38. Skrip AWK untuk menghitung RO dapat dilihat pada lampiran A.9 Kode Skrip AWK *Routing Overhead*.

```
for i = 1 to number of rows
     if in a row contains “s” and in RTR then
        ro increment
     end if
end for
```

Gambar 4.25 Pseudocode untuk Perhitungan Routing Overhead25

Contoh perintah untuk memanggil skrip AWK untuk menghitung RO pada trace file dan contoh keluarannya dapat dilihat pada Gambar 4.39 dan 4.40.

```
awk -f ro.awk tracefile.tr
```

Gambar 4.26 Perintah Untuk Menjalankan Skrip AWK Perhitungan RO

```
Overhead : 29830
```

Gambar 4.27 Contoh Keluaran dari Skrip Perhitungan RO

(Halaman ini sengaja dikosongkan)

BAB V
UJICOBA DAN EVALUASI

Pada bab ini akan dilakukan tahap ujicoba dan evaluasi sesuai dengan rancangan dan implementasi. Dari hasil yang didapatkan setelah melakukan uji coba, akan dilakukan evaluasi sehingga dapat ditarik kesimpulan pada bab selanjutnya.

5.1 Lingkungan Uji Coba

Uji coba dilakukan pada perangkat dengan spesifikasi seperti yang tertera pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat yang Digunakan

Komponen	**Spesifikasi**
CPU	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz (8 CPUs) , ~2.8GHz
Sistem Operasi	Debian 9 LTS
Linux Kernel	Linux kernel 4.9
Memori	8.0 GB

Adapun versi perangkat lunak yang digunakan dalam Tugas Akhir ini adalah sebagai berikut:

- SUMO versi 0.25.0 untuk pembuatan skenario mobilitas VANETs.
- JOSM versi 10301 untuk penyuntingan peta OpenStreetMap.
- NS-2 versi 2.35 untuk simulasi skenario VANETs.

Parameter lingkungan uji coba yang digunakan pada NS-2 dapat diihat pada Tabel 5.2.

Pengujian dilakukan dengan menjalankan skenario yang disimulasikan pada NS-2. Dari simulasi tersebut dihasilkan sebuah *trace file* dengan ekstensi .tr yang akan dianalisis dengan bantuan skrip awk untuk mendapatkan PDR, E2E, dan RO.

Tabel 5.2 Lingkungan Uji Coba

No.	Parameter	Spesifikasi
1	Network simulator	NS-2.35
2	Routing protocol	AODV dan EE-AODV
3	Waktu simulasi	200 detik
4	Area simulasi	1000 m x 1000 m (grid) 1400 m x 1400 m (real)
5	Jumlah *Node*	50,10,150,200
6	Radius transmisi	400m
7	Agen	Constant Bit Rate (CBR)
8	Energi awal	100 Joule
9	Ukuran paket	512 Bytes
10	*Transmission Energy*	1,8 Watt
11	*Reception Energy*	1,2 Watt
12	Protokol MAC	IEEE 802.11p

5.2 Hasil Uji Coba

Hasil uji coba dari skenario grid dan skenario riil dapat dilihat sebagai berikut:

5.2.1 Hasil Uji Coba Skenario *Grid*

Pengujian pada skenario *grid* digunakan untuk melihat perbandingan *packet delivery ratio* (PDR), rata-rata *end-to-end delay* (E2E), *routing overhead*, dan *forwarded route request* antara *routing protocol* AODV asli dan *routing protocol* AODV yang telah dimodifikasi (EE-AODV).

Pengambilan data uji PDR, rata-rata *end-to-end delay*, *routing overhead*, dan *forwarded route request* dengan luas area 1000 m x 1000 m dan *node* sebanyak 50 untuk lingkungan yang jarang, 100, 150 untuk lingkungan yang sedang, dan 200 untuk lingkungan yang padat. Untuk uji coba setiap jumlah *node* dilakukan sebanyak 10 kali, karena dari setiap uji coba setiap node akan menghasilkan hasil yang

berbeda-beda, lalu akan di rata-rata untuk mengetahui performa dari AODV yang telah di modifikasi.
Hasil pengambilan data *packet delivery ratio* pada skenario *grid* 50, 100, 150, dan 200 *node* dengan menggunakan AODV asli dan AODV yang telah dimodifikasi (EE-AODV) dapat dilihat pada tabel 5.3 hingga 5.5.

Tabel 5.3 Hasil Rata – Rata PDR Skenario GRID

Jumlah *Node*	AODV Asli	AODV Modifikasi	Perbedaan (%)
50	48,35	78,90	30,55
100	48,58	80,48	31,9
150	53,29	77,54	24,25
200	49,73	75,14	25,41

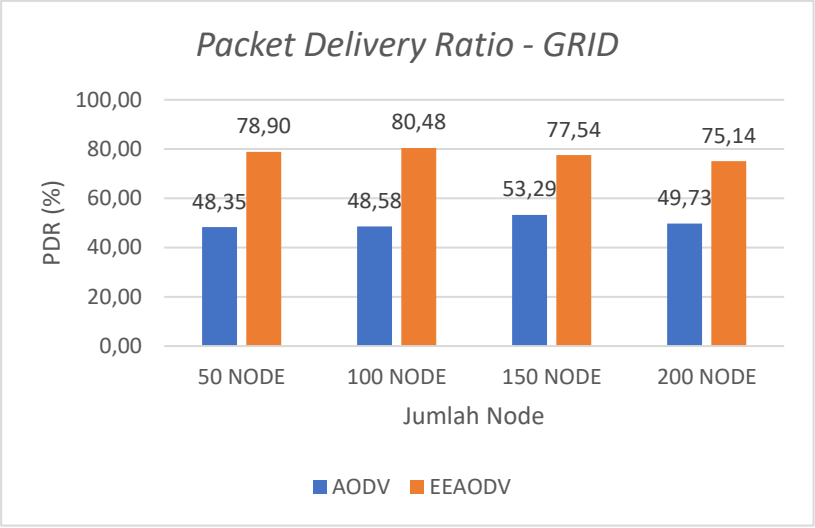
Tabel 5.4 Hasil Rata – Rata E2E Skenario GRID

Jumlah *Node*	AODV Asli	AODV Modifikasi	Perbedaan (ms)
50	1093,16	337,24	755,93
100	1279,97	535,7	744,27
150	1304,33	520,33	784
200	1161,76	512,92	648,84

Tabel 5.5 Hasil Rata – Rata RO Skenario GRID

Jumlah *Node*	AODV Asli	AODV Modifikasi	Perbedaan
50	19,89	17,71	2,19
100	37,09	34,64	2,45
150	53,51	51,47	2,04
200	70,75	68,15	2,6

Dari data di atas, dibuat grafik yang merepresentasikan hasil perhitungan PDR, E2E, dan RO yang ditunjukkan pada Gambar 5.1, Gambar 5.2 dan Gambar 5.3.

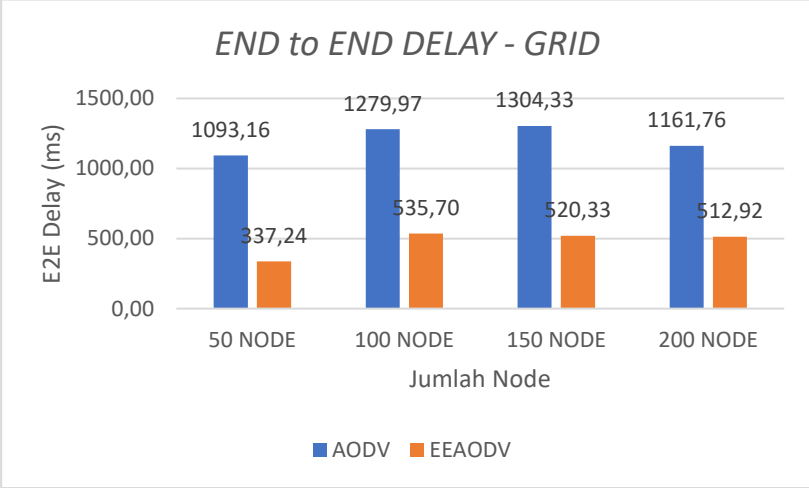


Gambar 5.1 Grafik Rata-Rata *Packet Delivery Ratio* pada Skenario *Grid*

Berdasarkan grafik pada Gambar 5.1, dapat dilihat bahwa *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami kenaikan disetiap percobaan jumlah *node*. Pada uji coba 50 *node*, menghasilkan perbedaan selisih PDR sebesar 30,55, dimana terjadi kenaikan sebesar 63,2% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal PDR tersebut. Pada uji coba 100 *node*, menghasilkan perbedaan selisih PDR sebesar 31,9 dimana terjadi kenaikan sebesar 72,02% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal PDR tersebut. Pada uji coba 150 *node*, menghasilkan perbedaan selisih PDR sebesar 24,25 dimana terjadi kenaikan sebesar 44,61% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal PDR tersebut. Pada uji coba 200 *node*, menghasilkan perbedaan selisih PDR sebesar 25,41 dimana terjadi kenaikan sebesar 58,71% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal PDR tersebut.

Jika keempat uji coba tersebut dibandingkan, dapat dilihat bahwa pada *routing* protocol AODV yang dimodifikasi menghasilkan PDR yang lebih baik. Nilai rata-rata kenaikan PDR pada skenario *grid* adalah sebesar 62.11%.

Dari hasil diatas, grafik PDR yang fluktuatif dapat disebabkan karena adanya perbedaan bentuk rute dari masing-masing skenario yang berbeda dan perbedaan letak dan kecepatan masing-masing node yang *digenerate* secara acak (*random)* oleh SUMO.



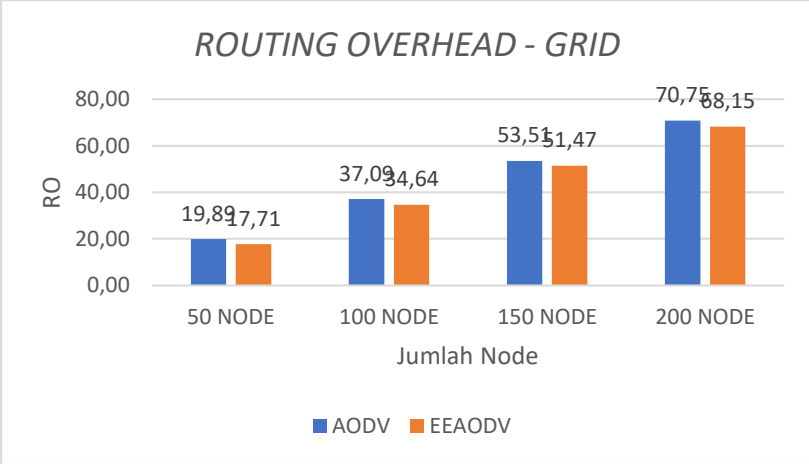
Gambar 5.2 Grafik *E2E Delay* pada Skenario Grid

Berdasarkan grafik pada Gambar 5.2, dapat dilihat bahwa rata-rata E2E antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami kenaikan disetiap percobaan jumlah *node*. Pada uji coba 50 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 755,93 ms , dimana *routing protocol* AODV yang dimodifikasi lebih unggul. Pada uji coba 100 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 744,27 ms , dimana *routing protocol* AODV yang dimodifikasi lebih unggul. Pada uji coba 150 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 784 ms , dimana

routing protocol AODV yang dimodifikasi lebih unggul. Pada uji coba 200 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 648,84 ms , dimana *routing protocol* AODV yang telah dimodifikasi lebih unggul.

Jika keempat uji coba tersebut dibandingkan, dapat dilihat bahwa pada *routing* protocol AODV yang dimodifikasi menghasilkan *end-to-end delay* yang lebih baik. Nilai rata-rata kenaikan *end-to-end delay* pada skenario *grid* adalah sebesar 60,28%.

Dari hasil diatas, grafik *End to End Delay* yang fluktuatif dapat disebabkan karena adanya perbedaan bentuk rute dari masing-masing skenario yang berbeda dan perbedaan letak dan kecepatan masing-masing node yang *digenerate* secara acak (*random)* oleh SUMO.



Gambar 5.3 Grafik *Routing Overhead* Skenario *Grid*

Berdasarkan grafik pada gambar 5.3, dapat dilihat bahwa *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami penurunan disetiap percobaan jumlah *node*. Pada uji coba 50 *node*, menghasilkan perbedaan selisih RO sebesar 2,19 dimana terjadi penurunan sebesar 10,99% dan *routing protocol* AODV yang

dimodifikasi lebih unggul dalam hal RO tersebut. Pada uji coba 100 *node*, menghasilkan perbedaan selisih RO sebesar 2,45 dimana terjadi penurunan sebesar 6,6% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal RO tersebut. Pada uji coba 150 *node*, menghasilkan perbedaan selisih RO sebesar 2,04 dimana terjadi penurunan sebesar 3,82% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal RO tersebut. Pada uji coba 200 *node*, menghasilkan perbedaan selisih RO sebesar 2,6 dimana terjadi penurunan sebesar 3,67% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal RO tersebut.

Jika keempat uji coba tersebut dibandingkan, dapat dilihat bahwa pada AODV yang telah dimodifikasi menghasilkan RO yang lebih bagus atau dalam hal ini lebih rendah daripada AODV asli. Nilai rata-rata penurunan RO pada AODV yang dimodifikasi adalah sebesar 6,34%.

Dari hasil diatas, grafik *Routing Overhead* yang konstan naik disebabkan karena setiap pertambahan node, makan akan semakin banyak pula packet routing yang dikirim yang terdiri dari *Route Request (RREQ), Route Reply (RREP),* dan *Route Error (RRER).*

5.2.2 Hasil Uji Coba Skenario *Real*

Sama seperti pengujian pada skenario *grid*, pengujian pada skenario riil digunakan untuk melihat perbandingan PDR, E2E, dan RO antara AODV asli dengan AODV Modifikasi dengan berbagai macam kombinasi nilai parameter faktor pergerakan node yang sama dengan yang dilakukan pada skenario *grid*.

Sama seperti pengujian pada skenario *grid*, pengujian PDR, E2E, dan RO pada skenario *real* dilakukan dengan skenario mobilitas *random* pada peta *grid* dengan luas area 1400 m x 1400 m dan *node* sebanyak 50 *node*, 100 *node*, 150 *node* dan 200 *node* dilakukan pada kecepatan yang bervariasi dengan batas maksimal 40 km/jam atau hampir setara dengan 11m/s. Untuk uji coba setiap jumlah *node*

dilakukan sebanyak 10 kali, karena dari setiap uji coba setiap node akan menghasilkan hasil yang berbeda-beda, lalu akan di rata-rata untuk mengetahui performa dari AODV yang telah di modifikasi. Hasil analisis dapat dilihat pada tabel 5.6 hingga 5.8.

Tabel 5.6 Hasil Rata – Rata PDR Skenario Real

Jumlah *Node*	AODV Asli	AODV Modifikasi	Perbedaan
50	53,13	71,49	18,36
100	53,76	80,48	26,72
150	55,08	77,54	22,46
200	48,2	75,14	26,94

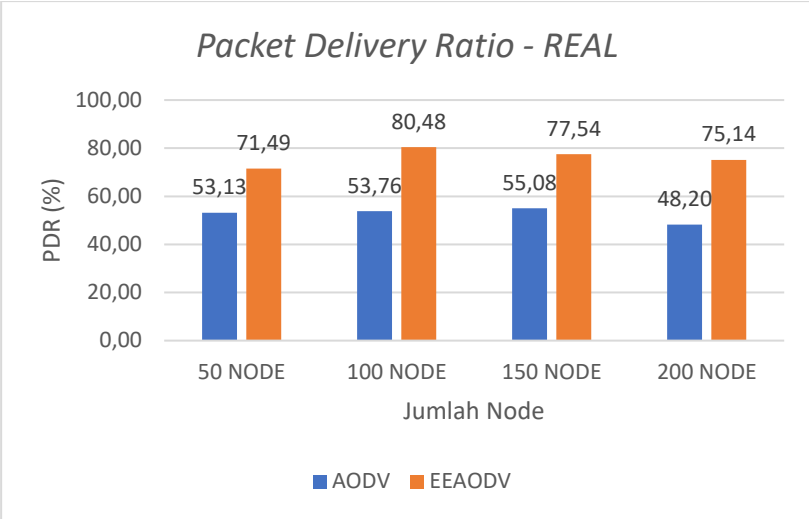
Tabel 5.7 Hasil Rata – Rata E2E Skenario Real

Jumlah *Node*	AODV Asli	AODV Modifikasi	Perbedaan (ms)
50	1110,99	841,2	269,79
100	1067,73	396,5	671,23
150	971,87	514,51	457,36
200	1384,48	400,22	984,26

Tabel 5.8 Hasil Rata – Rata RO Skenario Real

Jumlah *Node*	AODV Asli	AODV Modifikasi	Perbedaan
50	20,72	18,31	2,41
100	37,6	34,83	2,77
150	54,18	51,62	2,57
200	71,36	68,13	3,23

Dari data di atas, dibuat grafik yang merepresentasikan hasil perhitungan PDR, E2E, dan RO yang ditunjukkan pada Gambar 5.4, Gambar 5.5 dan Gambar 5.6



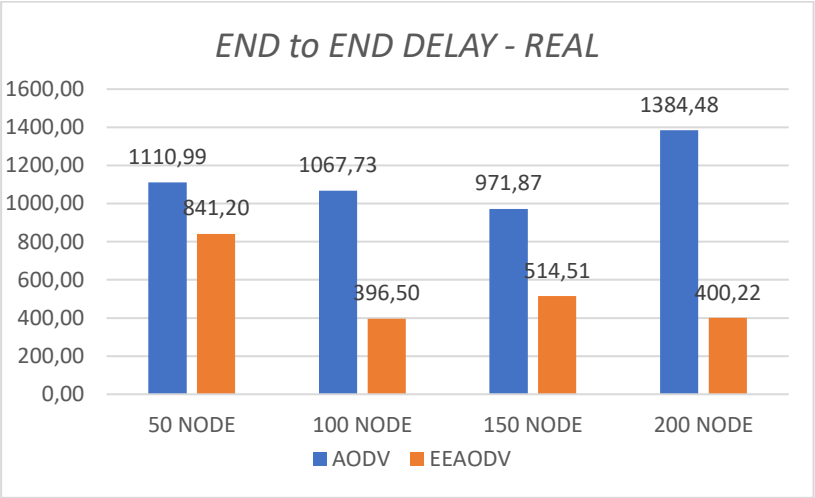
Gambar 5.4 Grafik *Packet Delivery Ratio* Skenario *Real*

Berdasarkan grafik pada Gambar 5.4, dapat dilihat bahwa *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami kenaikan. Pada uji coba 50 *node*, menghasilkan perbedaan selisih PDR sebesar 18,36 dimana terjadi kenaikan sebesar 34,55% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal PDR tersebut. Pada uji coba 100 *node*, menghasilkan perbedaan selisih PDR sebesar 26,72 dimana terjadi kenaikan sebesar 39,08% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal PDR tersebut. Pada uji coba 150 *node*, menghasilkan perbedaan selisih PDR sebesar 22,46 dimana terjadi kenaikan sebesar 42,83% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal PDR tersebut. Pada uji coba 200 *node*, menghasilkan perbedaan selisih PDR sebesar 26,49 dimana terjadi kenaikan sebesar 50,16% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal PDR tersebut.

Jika keempat uji coba tersebut dibandingkan, dapat dilihat bahwa pada *routing* protocol AODV yang dimodifikasi menghasilkan PDR

yang lebih baik. Nilai rata-rata kenaikan PDR pada skenario *real* adalah sebesar 41,14%.

Dari hasil diatas, grafik PDR yang fluktuatif dapat disebabkan karena adanya perbedaan bentuk rute dari masing-masing skenario yang berbeda dan perbedaan letak dan kecepatan masing-masing node yang *digenerate* secara acak (*random)* oleh SUMO.



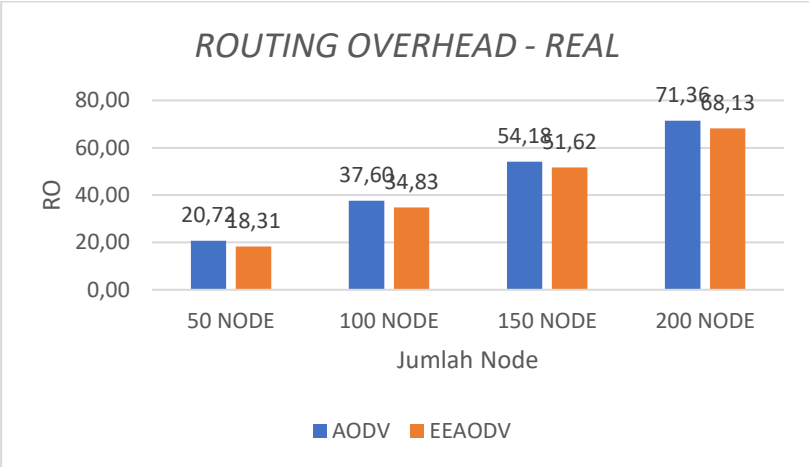
Gambar 5.5 Grafik *End-to-End Average Delay* Skenario *Real*

Berdasarkan grafik pada Gambar 5.5, dapat dilihat bahwa rata-rata E2E antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami kenaikan disetiap percobaan jumlah *node*. Pada uji coba 50 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 269,79 ms, dimana *routing protocol* AODV yang dimodifikasi lebih unggul. Pada uji coba 100 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 671,23 ms, dimana *routing protocol* AODV yang dimodifikasi lebih unggul. Pada uji coba 150 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 457,36 ms, dimana *routing protocol* AODV yang dimodifikasi lebih unggul. Pada uji

coba 200 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 984,26 ms, dimana *routing protocol* AODV yang telah dimodifikasi lebih unggul.

Jika keempat uji coba tersebut dibandingkan, dapat dilihat bahwa pada *routing* protocol AODV yang dimodifikasi menghasilkan *end-to-end delay* yang lebih baik. Nilai rata-rata kenaikan *end-to-end delay* pada skenario *grid* adalah sebesar 53,63%.

Dari hasil diatas, grafik *End to End Delay* yang fluktuatif dapat disebabkan karena adanya perbedaan bentuk rute dari masing-masing skenario yang berbeda dan perbedaan letak dan kecepatan masing-masing node yang *digenerate* secara acak (*random)* oleh SUMO.



Gambar 5.6 Grafik *Routing Overhead* Skenario *Real*

Berdasarkan grafik pada gambar 5.6, dapat dilihat bahwa *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami penurunan disetiap percobaan jumlah *node*. Pada uji coba 50 *node*, menghasilkan perbedaan selisih RO sebesar 2,41 dimana terjadi penurunan sebesar 11,65% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal RO tersebut. Pada uji coba 100

node, menghasilkan perbedaan selisih RO sebesar 2,77 dimana terjadi penurunan sebesar 7,37% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal RO tersebut. Pada uji coba 150 *node*, menghasilkan perbedaan selisih RO sebesar 2,57 dimana terjadi penurunan sebesar 4,74% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal RO tersebut. Pada uji coba 200 *node*, menghasilkan perbedaan selisih RO sebesar 2,82 dimana terjadi penurunan sebesar 4,53% dan *routing protocol* AODV yang dimodifikasi lebih unggul dalam hal RO tersebut.

Jika keempat uji coba tersebut dibandingkan, dapat dilihat bahwa pada AODV yang telah dimodifikasi menghasilkan RO yang lebih bagus atau dalam hal ini lebih rendah daripada AODV asli. Nilai rata-rata penurunan RO pada AODV yang dimodifikasi adalah sebesar 7,13%.

Dari hasil diatas, grafik *Routing Overhead* yang konstan naik disebabkan karena setiap pertambahan node, maka akan semakin banyak pula packet routing yang dikirim yang terdiri dari *Route Request (RREQ), Route Reply (RREP),* dan *Route Error (RRER).*

BAB VI
KESIMPULAN DAN SARAN

Pada Bab ini akan diberikan kesimpulan yang diperoleh dari Tugas Akhir yang telah dikerjakan dan saran tentang pengembangan dari Tugas Akhir ini yang dapat dilakukan di masa yang akan datang.

6.1 Kesimpulan

Kesimpulan yang diperoleh pada uji coba dan evaluasi Tugas Akhir ini adalah sebagai berikut:

1. AODV yang dimodifikasi (EE-AODV) dengan penerapan pemilihan rute berdasarkan *Energy/Distance* telah berhasil meningkatkan performa pada *routing protocol* AODV.
2. Dampak penerapan modifikasi terhadap performa protokol EE-AODV pada skenario grid adalah kenaikan rata-rata *Packet Delivery Ratio* (PDR) sebesar 62,11%, penurunan rata-rata *End-to-End Delay* (E2E) sebesar 60,28% dan penurunan rata-rata *Routing Overhead* (RO) sebesar 6,34%.
3. Dampak penerapan modifikasi terhadap performa protocol EE-AODV pada skenario real adalah kenaikan rata-rata *Packet Delivery Ratio* (PDR) sebesar 41,14%, penurunan rata-rata *End-to-End Delay* (E2E) sebesar 53,63% dan penurunan rata-rata *Routing Overhead* (RO) sebesar 7,13%.

6.2 Saran

Saran yang dapat diberikan dari hasil uji coba dan evaluasi adalah sebagai berikut:

1. Untuk mendapatkan hasil uji coba yang lebih baik, dapat dilakukan uji coba yang lebih banyak.
2. Kedepannya bisa mengimplementasikan dengan menambahkan aspek lain yang dijadikan untuk acuan pemilihan rute seperti kecepatan, arah, dan lain-lain.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

[1] S. David dan R. Navaneethakrishnan, "ENERGY CONSUMPTION AND LOAD BALANCING COMPARED WITH VANET AND MANET," *International Journal of Pure and Applied Mathematics,* vol. 116, pp. 257-265, 2017.

[2] F. Li dan Wang Yu, "Routing in Vehicular Ad Hoc Networks: A Survey," *IEEE Vehicular Technology Magazine,* vol. 2, no. 2, pp. 12-22, 2007.

[3] H. Hartenstein dan L. Laberteaux, "A Tutorial Survey on Vehicular Ad Hoc Network," *IEEE Communications Magazine,* vol. 46, no. 6, pp. 164-171, 2008.

[4] Xi Yu, Huaqun Guo dan Wai-Choong Wong, "A Reliable Routing Protocol for VANET Communications," pp. 1748-1753, July 2011.

[5] "NS3 Code. VANET Project," [Online]. Available: http://ns3-code.com/ ns3-vanet-projects/. [Diakses 10 Januari 2019].

[6] C. Perkins dan E. Royer, "Ad-hoc on-demand distance vector routing," dalam *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, USA, USA, 1999.

[7] C. Perkins, E. Royer, S. Das dan M. Marina, "Performance comparison of two on-demand routing protocols for ad hoc networks," *IEEE Personal Communications,* vol. 8, no. 1, pp. 16-28, 2001.

[8] M. Yoshimachi dan Y. Manabe, "A New AODV Route Discovery Protocol to Achieve Fair Routing for Mobile Ad Hoc Networks," dalam *2016 6th International Conference on Information Communication and Management*, 2016.

[9] "SUMO," [Online]. Available: http://www.sumo.dlr.de. [Diakses 22 November 2018].

[10] "OpenStreetMap," [Online]. Available: https://www.openstreetmap.org. [Diakses 23 November 2018].

[11] "JOSM," [Online]. Available: https://josm.openstreetmap.de/. [Diakses 23 November 2018].

[12] "AWK," [Online]. Available: http://tldp.org/LDP/abs/html/awk.html. [Diakses 25 Januari 2019].

[13] S. McCanne dan S. Floyd, "The Network Simulator- NS-2," [Online]. Available: https://www.isi.edu/nsnam/ns/. [Diakses 24 November 2018].

[14] David Wetherall, "OTcl," 1997. [Online]. Available: http://otcl-tclcl. sourceforge.net/otcl/. [Diakses 20 Januari 2019].

[15] Jaspreet Singh, Kartik Sharma, "Energy Efficient AODV routing protocol For Mobile Ad-hoc," *International Journal Of Engineering And Computer Science,* p. 4, 2015.

[16] Dr. Annapurna P Patil, Varsha Chandan B, Aparna S, Greeshma R, Akshatha H P, "An Improved Energy Efficient AODV Routing Protocol for MANETs," *2014 Eleventh International Conference on Wireless and Optical Communications Networks (WOCN),* p. 5, 2014.

LAMPIRAN

A.1 Kode Skenario NS-2

```
set val(chan)           Channel/WirelessChannel
set val(prop)           Propagation/TwoRayGround
set val(netif)          Phy/WirelessPhy
set val(mac)            Mac/802_11
set val(ifq)            Queue/DropTail/PriQueue
set val(ll)             LL
set val(ant)            Antenna/OmniAntenna
set val(ifqlen)         50
set val(nn)             100
set val(rp)             AODV
set val(x)              1000
set val(y)              1000
set val(stop)       200
set val(traffic)       "cbr.tcl"
set val(mobility)       "scenario.tcl"
set opt(energymodel)    EnergyModel
set opt(initialenergy)  100


set ns        [new Simulator]
$ns use-newtrace
set tracefd       [open output-$val(nn).tr w]
set namtrace      [open output-$val(nn).nam w]


#******throughput******
set f0 [open throughput-$val(nn).tr w]
# *** Packet Loss Trace ***
set f3 [open lost-$val(nn).tr w]
# *** Packet Delay Trace ***
set f2 [open delay-$val(nn).tr w]
# set up topography object
set ppdr [open pdr-$val(nn).tr w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
```

```
37.
38. # set up topography object
39. set topo       [new Topography]
40.
41. $topo load_flatgrid $val(x) $val(y)
42. set god_ [create-god $val(nn)]
43.
44. #
45. #  Create nn mobilenodes [$val(nn)] and attach them to
    the channel.
46. #
47.
48. # configure the nodes
49.
50. if { $val(rp) == "AODV" } {
51. set val(prop)  Propagation/twoRayGround
52. }
53.
54. $ns node-config -energyModel $opt(energymodel) \
55.          -initialEnergy $opt(initialenergy) \
56.                  -rxPower 1.2 \
57.                  -txPower 1.8 \
58.
59.         $ns node-config -adhocRouting $val(rp) \
60.              -llType $val(ll) \
61.              -macType $val(mac) \
62.              -ifqType $val(ifq) \
63.              -ifqLen $val(ifqlen) \
64.              -antType $val(ant) \
65.              -propType $val(prop) \
66.              -phyType $val(netif) \
67.              -channelType $val(chan) \
68.              -topoInstance $topo \
69.                          -agentTrace ON \
70.              -routerTrace ON \
71.              -macTrace OFF \
72.              -movementTrace ON
73.
74. source $val(mobility)
75. source $val(traffic)
76.
77.
78. # Define node initial position in nam
```

```
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "$node_($i) reset";
}
proc energy {} {
global val
exec awk -f energy_trace.awk output-
$val(nn).tr >energy-$val(nn).tr
exec awk -f stats.awk output-$val(nn).tr
}

# ending nam and the simulation
#$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop).00000001 "energy"
$ns at $val(stop).00000002 "stop"
$ns at $val(stop).00000002 "puts \"end simulation\" ;
$ns halt"
proc stop {} {
           global ns tracefd namtrace f0 f1 f2 f3 val
    ppdr
           $ns flush-trace
           close $tracefd
           close $namtrace
           exec ./nam output-$val(nn).nam &
           exit 0

       }

       $ns run
```

A.2 Kode Konfigurasi *Traffic*

```
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(0)
set sink0 [new Agent/LossMonitor]
$ns_ attach-agent $node_(9) $sink0
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 1
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $sink0
$ns_ at 2.5568388786897245 "$cbr_(0) start"
$node_(1) color red
$ns_ at 2.5568388786897245 "$node_(1) color red"
$ns_ at 2.5568388786897245 "$node_(1) label source1"
$node_(9) color red
$ns_ at 2.5568388786897245 "$node_(9) color red"
$ns_ at 2.5568388786897245 "$node_(9) label destinatio
n1"
```

A.3 Kode Fungsi recvRequest()

```
669.        void
670.        AODV::recvRequest(Packet *p) {
671.        struct hdr_ip *ih = HDR_IP(p);
672.        struct hdr_aodv_request *rq = HDR_AODV_REQUEST
    (p);
673.        aodv_rt_entry *rt;
674.          /*
675.           * Drop if:
676.           *      - I'm the source
677.           *      - I recently heard this request.
678.           */
679.        //=============================================
    ==============================
680.        Node* thisnode = Node::get_node_by_address(ind
    ex);
681.        double nowenergy = thisnode->energy_model()-
    >energy();
682.        MobileNode *mn;
683.        mn = (MobileNode *) (Node::get_node_by_address
    (rq->rq_dst));
684.        MobileNode *mn1,*mn2;
685.
686.          double myX, myY, myZ, chX,chY,chZ,currD, min
    D;
687.
688.          /*Get My Coordinates*/
689.          mn1 = (MobileNode*)Node::get_node_by_address
    (index);
690.          myX = mn1->X();
691.          myY = mn1->Y();
692.          myZ = mn1->Z();
693.
694.          mn2 = (MobileNode*)Node::get_node_by_address
    (rq->rq_dst);
695.          chX = mn2->X();
696.          chY = mn2->Y();
697.          chZ = mn2->Z();
698.
699.
```

```
700.          /*Initially set the current distance to my d
    istance*/
701.          minD = DISTANCE(myX,myY,myZ,chX,chY,chZ);
702.
703.
704.                    FILE * pFile;
705.                    pFile = fopen ( "energy_distance_t
    able" , "a" );
706.                    fprintf (pFile, " I recieve the rq
    t !!! node # (recv) %d is in permit table has  energy
    = %f distance between node_i and Node_j is %f at time
    = %f  \n",index,nowenergy,minD,CURRENT_TIME);
707.                    fclose (pFile);
708.
709.
710.
711.
712.        bool isInPermit = false;
713.        for (int i =0; i<5; i++) {   //□□o□□
714.        if( rq->rq_permit[i] == index )
715.        isInPermit = true;
716.        }
717.
718.        if(isInPermit == false&&rq-
    >rq_dst != index ) {
719.            Packet::free(p);
720.
721.            return;
722.        }
723.
724.
725.
726.
727.          if(rq->rq_src == index) {
728.        #ifdef DEBUG
729.            fprintf(stderr, "%s: got my own REQUEST\n"
    , __FUNCTION__);
730.        #endif // DEBUG
731.            Packet::free(p);
732.            return;
733.          }
734.
```

```
735.         if (id_lookup(rq->rq_src, rq-
   >rq_bcast_id)) {
736.
737.       #ifdef DEBUG
738.           fprintf(stderr, "%s: discarding request\n",
    __FUNCTION__);
739.       #endif // DEBUG
740.
741.           Packet::free(p);
742.           return;
743.         }
744.
745.         /*
746.          * Cache the broadcast ID
747.          */
748.         id_insert(rq->rq_src, rq->rq_bcast_id);
749.
750.
751.
752.
753.          * route is in the route table.
754.          */
755.         aodv_rt_entry *rt0; // rt0 is the reverse rou
   te
756.
757.           rt0 = rtable.rt_lookup(rq->rq_src);
758.           if(rt0 == 0) { /* if not in the route table
    */
759.           // create an entry for the reverse route.
760.             rt0 = rtable.rt_add(rq->rq_src);
761.           }
762.
763.           rt0->rt_expire = max(rt0-
   >rt_expire, (CURRENT_TIME + REV_ROUTE_LIFE));
764.
765.           if ( (rq->rq_src_seqno > rt0-
   >rt_seqno ) ||
766.                ((rq->rq_src_seqno == rt0-
   >rt_seqno) &&
767.                 (rq->rq_hop_count < rt0->rt_hops)) ) {
768.           // If we have a fresher seq no. or lesser #
   hops for the
```

```
769.          // same seq no., update the rt entry. Else
    don't bother.
770.       rt_update(rt0, rq->rq_src_seqno, rq-
    >rq_hop_count, ih->saddr(),
771.                         max(rt0-
    >rt_expire, (CURRENT_TIME + REV_ROUTE_LIFE)) );
772.            if (rt0->rt_req_timeout > 0.0) {
773.            // Reset the soft state and
774.            // Set expiry time to CURRENT_TIME + ACTI
    VE_ROUTE_TIMEOUT
775.            // This is because route is used in the f
    orward direction,
776.            // but only sources get benefited by this
     change
777.              rt0->rt_req_cnt = 0;
778.              rt0->rt_req_timeout = 0.0;
779.              rt0->rt_req_last_ttl = rq-
    >rq_hop_count;
780.              rt0-
    >rt_expire = CURRENT_TIME + ACTIVE_ROUTE_TIMEOUT;
781.            }
782.
783.            /* Find out whether any buffered packet c
    an benefit from the
784.             * reverse route.
785.             * May need some change in the following
    code - Mahesh 09/11/99
786.             */
787.            assert (rt0->rt_flags == RTF_UP);
788.            Packet *buffered_pkt;
789.            while ((buffered_pkt = rqueue.deque(rt0-
    >rt_dst))) {
790.              if (rt0 && (rt0-
    >rt_flags == RTF_UP)) {
791.           assert(rt0->rt_hops != INFINITY2);
792.                forward(rt0, buffered_pkt, NO_DELAY);

793.              }
794.            }
795.          }
796.          // End for putting reverse route in rt tabl
    e
797.
```

```
798.
799.         /*
800.          * We have taken care of the reverse route st
    uff.
801.          * Now see whether we can send a route reply.

802.          */
803.
804.         rt = rtable.rt_lookup(rq->rq_dst);
805.
806.         // First check if I am the destination ..
807.
808.         if(rq->rq_dst == index) {
809.
810.        #ifdef DEBUG
811.           fprintf(stderr, "%d - %s: destination sendi
    ng reply\n",
812.                               index, __FUNCTION__);
813.        #endif // DEBUG
814.
815.
816.           // Just to be safe, I use the max. Somebody
     may have
817.           // incremented the dst seqno.
818.           seqno = max(seqno, rq->rq_dst_seqno)+1;
819.           if (seqno%2) seqno++;
820.
821.           sendReply(rq-
    >rq_src,           // IP Destination
822.                     1,                    // Hop Count
823.                     index,          // Dest IP Addres
824.                     seqno,       // Dest Sequence Num
825.                     MY_ROUTE_TIMEOUT,     // Lifetim
826.                     rq-
    >rq_timestamp);    // timestamp
827.
828.           Packet::free(p);
829.         }
830.
831.
832.
833.         else if (rt && (rt-
    >rt_hops != INFINITY2) &&
```

```
834.                (rt->rt_seqno >= rq-
    >rq_dst_seqno) ) {
835.
836.            //assert (rt->rt_flags == RTF_UP);
837.            assert(rq->rq_dst == rt->rt_dst);
838.
839.            sendReply(rq->rq_src,
840.                      rt->rt_hops + 1,
841.                      rq->rq_dst,
842.                      rt->rt_seqno,
843.                  (u_int32_t) (rt-
    >rt_expire - CURRENT_TIME),
844.                  //             rt-
    >rt_expire - CURRENT_TIME,
845.                      rq->rq_timestamp);
846.            // Insert nexthops to RREQ source and RREQ
    destination in the
847.            // precursor lists of destination and sourc
    e respectively
848.            rt->pc_insert(rt0-
    >rt_nexthop); // nexthop to RREQ source
849.            rt0->pc_insert(rt-
    >rt_nexthop); // nexthop to RREQ destination
850.
851.        #ifdef RREQ_GRAT_RREP
852.
853.            sendReply(rq->rq_dst,
854.                      rq->rq_hop_count,
855.                      rq->rq_src,
856.                      rq->rq_src_seqno,
857.                  (u_int32_t) (rt-
    >rt_expire - CURRENT_TIME),
858.                  //             rt-
    >rt_expire - CURRENT_TIME,
859.                      rq->rq_timestamp);
860.        #endif
861.
862.
863.             Packet::free(p);
864.          }
865.          /*
866.           * Can't reply. So forward the  Route Request
```

```
867.          */
868.         else {
869.           ih->saddr() = index;
870.           ih->daddr() = IP_BROADCAST;
871.           rq->rq_hop_count += 1;
872.           // Maximum sequence number seen en route
873.           if (rt) rq->rq_dst_seqno = max(rt-
    >rt_seqno, rq->rq_dst_seqno);
874.
875.
876.           forward((aodv_rt_entry*) 0, p, DELAY);
877.         }
878.
879.       }
```

A.4 Kode Fungsi sendRequest()

```
1166.       void
1167.       AODV::sendRequest(nsaddr_t dst) {
1168.       // Allocate a RREQ packet
1169.       Packet *p = Packet::alloc();
1170.       struct hdr_cmn *ch = HDR_CMN(p);
1171.       struct hdr_ip *ih = HDR_IP(p);
1172.       struct hdr_aodv_request *rq = HDR_AODV_REQUEST
    (p);
1173.       aodv_rt_entry *rt = rtable.rt_lookup(dst);
1174.
1175.
1176.       rq->rq_energy_threshold = ENERGY_THRESHOLD;
1177.
1178.       if ( nb_highEnergy(1) )
1179.       rq->rq_permit[0] = nb_highEnergy(1)-
    >nb_addr;
1180.       if ( nb_highEnergy(2) )
1181.       rq->rq_permit[1] = nb_highEnergy(2)-
    >nb_addr;
1182.       if ( nb_highEnergy(3) )
1183.       rq->rq_permit[2] = nb_highEnergy(3)-
    >nb_addr;
1184.
1185.       if ( nb_highEnergy(4) )
1186.       rq->rq_permit[3] = nb_highEnergy(4)-
    >nb_addr;
1187.       if ( nb_highEnergy(5) )
1188.
1189.
1190.        assert(rt);
1191.
1192.
1193.        if (rt->rt_flags == RTF_UP) {
1194.          assert(rt->rt_hops != INFINITY2);
1195.          Packet::free((Packet *)p);
1196.          return;
1197.        }
1198.
1199.        if (rt->rt_req_timeout > CURRENT_TIME) {
```

```
1200.           Packet::free((Packet *)p);
1201.           return;
1202.         }
1203.
1204.         if (rt->rt_req_cnt > RREQ_RETRIES) {
1205.           rt-
    >rt_req_timeout = CURRENT_TIME + MAX_RREQ_TIMEOUT;
1206.           rt->rt_req_cnt = 0;
1207.         Packet *buf_pkt;
1208.           while ((buf_pkt = rqueue.deque(rt-
    >rt_dst))) {
1209.               drop(buf_pkt, DROP_RTR_NO_ROUTE);
1210.           }
1211.           Packet::free((Packet *)p);
1212.           return;
1213.         }
1214.
1215.       #ifdef DEBUG
1216.           fprintf(stderr, "(%2d) - %2d sending Route
    Request, dst: %d\n",
1217.                               ++route_request, index, rt
    ->rt_dst);
1218.       #endif // DEBUG
1219.
1220.
1221.         rt->rt_req_last_ttl = max(rt-
    >rt_req_last_ttl,rt->rt_last_hop_count);
1222.
1223.         if (0 == rt->rt_req_last_ttl) {
1224.         // first time query broadcast
1225.           ih->ttl_ = TTL_START;
1226.         }
1227.         else {
1228.         // Expanding ring search.
1229.           if (rt->rt_req_last_ttl < TTL_THRESHOLD)
1230.             ih->ttl_ = rt-
    >rt_req_last_ttl + TTL_INCREMENT;
1231.           else {
1232.           // network-wide broadcast
1233.             ih->ttl_ = NETWORK_DIAMETER;
1234.             rt->rt_req_cnt += 1;
1235.           }
1236.         }
```

```
1237.
1238.       // remember the TTL used  for the next time
1239.       rt->rt_req_last_ttl = ih->ttl_;
1240.
1241.
1242.       rt->rt_req_timeout = 2.0 * (double) ih-
    >ttl_ * PerHopTime(rt);
1243.       if (rt->rt_req_cnt > 0)
1244.         rt->rt_req_timeout *= rt->rt_req_cnt;
1245.       rt->rt_req_timeout += CURRENT_TIME;
1246.
1247.
1248.       if (rt-
    >rt_req_timeout > CURRENT_TIME + MAX_RREQ_TIMEOUT)
1249.         rt-
    >rt_req_timeout = CURRENT_TIME + MAX_RREQ_TIMEOUT;
1250.       rt->rt_expire = 0;
1251.
1252.     #ifdef DEBUG
1253.       fprintf(stderr, "(%2d) - %2d sending Route Re
    quest, dst: %d, tout %f ms\n",
1254.                    ++route_request,
1255.               index, rt->rt_dst,
1256.               rt->rt_req_timeout - CURRENT_TIME);
1257.     #endif  // DEBUG
1258.
1259.
1260.       // Fill out the RREQ packet
1261.       // ch->uid() = 0;
1262.       ch->ptype() = PT_AODV;
1263.       ch->size() = IP_HDR_LEN + rq->size();
1264.       ch->iface() = -2;
1265.       ch->error() = 0;
1266.       ch->addr_type() = NS_AF_NONE;
1267.       ch-
    >prev_hop_ = index;          // AODV hack
1268.
1269.       ih->saddr() = index;
1270.       ih->daddr() = IP_BROADCAST;
1271.       ih->sport() = RT_PORT;
1272.       ih->dport() = RT_PORT;
1273.
1274.       // Fill up some more fields.
```

```
1275.         rq->rq_type = AODVTYPE_RREQ;
1276.         rq->rq_hop_count = 1;
1277.         rq->rq_bcast_id = bid++;
1278.         rq->rq_dst = dst;
1279.         rq->rq_dst_seqno = (rt ? rt->rt_seqno : 0);
1280.         rq->rq_src = index;
1281.         seqno += 2;
1282.         assert ((seqno%2) == 0);
1283.         rq->rq_src_seqno = seqno;
1284.         rq->rq_timestamp = CURRENT_TIME;
1285.
1286.
1287.
1288.
1289.         Scheduler::instance().schedule(target_, p, 0.
      );
1290.
1291.       }
```

A.5 Kode Fungsi sendHello()

```
1394.       void
1395.       AODV::sendHello() {
1396.       Packet *p = Packet::alloc();
1397.       struct hdr_cmn *ch = HDR_CMN(p);
1398.       struct hdr_ip *ih = HDR_IP(p);
1399.       struct hdr_aodv_reply *rh = HDR_AODV_REPLY(p);

1400.
1401.       #ifdef DEBUG
1402.       fprintf(stderr, "sending Hello from %d at %.2f
    \n", index, Scheduler::instance().clock());
1403.       #endif // DEBUG
1404.       //MobileNode *thisnode;
1405.        rh->rp_type = AODVTYPE_HELLO;
1406.        //rh->rp_flags = 0x00;
1407.        rh->rp_hop_count = 1;
1408.        rh->rp_dst = index;
1409.        rh->rp_dst_seqno = seqno;
1410.        rh-
    >rp_lifetime = (1 + ALLOWED_HELLO_LOSS) * HELLO_INTERV
    AL;
1411.
1412.
1413.        Node* thisnode = Node::get_node_by_address(in
    dex);
1414.        double nowenergy = thisnode->energy_model()-
    >energy();
1415.        rh->rp_energy = nowenergy;
1416.       MobileNode *mn1,*mn2;
1417.
1418.         double myX, myY, myZ, chX,chY,chZ,currD, min
    D;
1419.
1420.         /*Get My Coordinates*/
1421.         mn1 = (MobileNode*)Node::get_node_by_address
    (index);
1422.         myX = mn1->X();
1423.         myY = mn1->Y();
1424.         myZ = mn1->Z();
```

```
1425.
1426.         mn2 = (MobileNode*)Node::get_node_by_address
    (ih->daddr());
1427.         chX = mn2->X();
1428.         chY = mn2->Y();
1429.         chZ = mn2->Z();
1430.
1431.         minD = DISTANCE(myX,myY,myZ,chX,chY,chZ);
1432.
1433.
1434.        // ch->uid() = 0;
1435.        ch->ptype() = PT_AODV;
1436.        ch->size() = IP_HDR_LEN + rh->size();
1437.        ch->iface() = -2;
1438.        ch->error() = 0;
1439.        ch->addr_type() = NS_AF_NONE;
1440.        ch-
    >prev_hop_ = index;           // AODV hack
1441.
1442.        ih->saddr() = index;
1443.        ih->daddr() = IP_BROADCAST;
1444.        ih->sport() = RT_PORT;
1445.        ih->dport() = RT_PORT;
1446.        ih->ttl_ = 1;
1447.
1448.        Scheduler::instance().schedule(target_, p, 0.
    0);
1449.      }
```

A.6 Kode Fungsi recvHello()

```
1454.       void
1455.       AODV::recvHello(Packet *p) {
1456.       //struct hdr_ip *ih = HDR_IP(p);
1457.       struct hdr_aodv_reply *rp = HDR_AODV_REPLY(p);

1458.       AODV_Neighbor *nb;
1459.
1460.
1461.        nb = nb_lookup(rp->rp_dst);
1462.        if(nb == 0) {
1463.          nb_insert(rp->rp_dst,rp-
    >rp_energy+0.0001*Random::uniform());
1464.        }
1465.        else {
1466.
1467.
1468.          nb->nb_energy = rp->rp_energy;
1469.
1470.          nb->nb_expire = CURRENT_TIME +
1471.                          (1.5 * ALLOWED_HELLO_LOSS *
     HELLO_INTERVAL);
1472.        }
1473.
1474.        Packet::free(p);
1475.       }
```

A.7 Kode Skrip AWK *Packet Delivery Ratio*

```
BEGIN {
        sendLine = 0;
        recvLine = 0;
        fowardLine = 0;
}

$0 ~/^s.* AGT/ {
        sendLine ++ ;
}

$0 ~/^r.* AGT/ {
        recvLine ++ ;
}

$0 ~/^f.* RTR/ {
        fowardLine ++ ;
}

END {
        printf "cbr s:%d r:%d, r/s Ratio:%.4f, f:%d \n",
sendLine, recvLine, (recvLine/sendLine),fowardLine;
}
```

A.8 Kode Skrip AWK Rata-Rata *End-to-End Delay*

```
BEGIN{
    sum_delay = 0;
    count = 0;
}
{
    if ($2 >= 101) {

        if($4 == "AGT" && $1 == "s" && seqno < $6) {
          seqno = $6;
        }

        if($4 == "AGT" && $1 == "s") {
            start_time[$6] = $2;
        }

        else if(($7 == "cbr") && ($1 == "r")) {
            end_time[$6] = $2;
        }

        else if($1 == "D" && $7 == "cbr") {
            end_time[$6] = -1;
        }
    }
}
END {

    for(i=0; i<=seqno; i++) {
        if(end_time[i] > 0) {
            delay[i] = end_time[i] - start_time[i];
            count++;
        }
        else {
            delay[i] = -1;
        }
    }
```

```
36.    for(i=0; i<=seqno; i++) {
37.        if(delay[i] > 0) {
38.         n_to_n_delay = n_to_n_delay + delay[i];
39.        }
40.    }
41. n_to_n_delay = n_to_n_delay/count;
42. printf "End-to-
    End Delay \t= " n_to_n_delay * 1000 " ms \n";
```

A.9 Kode Skrip AWK *Routing Overhead*

```
BEGIN {
rt_pkts = 0;
}
{
        if (($1 == "s" || $1 == "f") && ($4 == "RTR") &&
($7 == "AODV")) {

                rt_pkts++;
}
    }
}
END {
    printf "Routing Packets \t= %d \n", rt_pkts;
}
```

BIODATA PENULIS

Galuh Aan Ramadhan, lahir di Surabaya, 14 Januari 1997. Penulis adalah putra dari pasangan Sujono. Penulis merupakan anak kedua dari dua bersaudara. Penulis menempuh pendidikan sekolah dasar di SDN Wadung Asri pada tahun 2003 lalu melanjutkan pendidikan sekolah menengah pertama di SMPN 35 Surabaya pada tahun 2009 kemudian dilanjutkan dengan menempuh pendidikan menengah atas di SMA Negeri 15 Surabaya pada tahun 2012. Selanjutnya penulis melanjutkan pendidikan sarjana di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya mulai tahun 2015. Penulis juga aktif dalam kegiatan kepanitian seperti SCHEMATICS 2016 dan SCHEMATICS 2017 divisi Kewirausahaan.Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Arsitektur dan Jaringan Komputer (AJK). Penulis pernah melakukan kerja praktik di PT. Bulog Juli – Agustus 2018 dan membuat aplikasi berbasis Web Pemasaran dan Informasi Divisi Komersial BULOG Divre Jawa Timur. Penulis dapat dihubungi melalui nomor *handphone*: 081350005141 atau *email*: aan.ramadhan77@gmail.com.