TUGAS AKHIR – KI141502

MODIFIKASI PEMILIHAN RUTE PADA *AD HOC ON-DEMAND DISTANCE VECTOR* (AODV) BERDASARKAN FAKTOR *THROUGHPUT* DAN *HOP* DI LINGKUNGAN VANETS

FATURRAHMAN MA'RUF
NRP 05111540000027

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

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

Departemen Teknik Informatika
Fakultas Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR – KI141502

MODIFIKASI PEMILIHAN RUTE PADA *AD HOC ON-DEMAND DISTANCE VECTOR* (AODV) BERDASARKAN FAKTOR *THROUGHPUT* DAN *HOP* DI LINGKUNGAN VANETS

FATURRAHMAN MA'RUF
NRP 05111540000027

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

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

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES - KI141502

MODIFICATION OF ROUTE SELECTION IN AD HOC ON-DEMAND DISTANCE VECTOR (AODV) BASED ON THROUGHPUT FACTOR AND HOP IN VANET ENVIRONMENT

FATURRAHMAN MA"RUF
NRP 05111540000027

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

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

Department of Informatics
Faculty of Intelligent Electrical and Informatics Technology
Sepuluh Nopember Institute of Technology
Surabaya 2020

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

MODIFIKASI PEMILIHAN RUTE PADA *AD HOC ON-DEMAND DISTANCE VECTOR* (AODV) BERDASARKAN FAKTOR *THROUGHPUT* DAN *HOP* 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:
FATURRAHMAN MA'RUF
NRP: 05111540000027

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., Ph.D.

.....
(NIP. 198106202005011003) (Pembimbing 2)

SURABAYA
JUNI, 20

(Halaman ini sengaja dikosongkan)

MODIFIKASI PEMILIHAN RUTE PADA *AD HOC ON-DEMAND DISTANCE VECTOR* (AODV) BERDASARKAN FAKTOR *THROUGHPUT* DAN *HOP* DI LINGKUNGAN VANETS

Nama Mahasiswa	**: FATURRAHMAN MA'RUF**
NRP	**: 05111540000027**
Departemen	**: Teknik Informatika FTEIC-ITS**
Dosen Pembimbing 1	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**
Dosen Pembimbing 2	**: Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D**

Abstrak

Vehicular Ad hoc Networks (VANETs) merupakan pengembangan dari Mobile Ad hoc Network (MANET) dimana node 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 salah satu routing protocol yang termasuk dalam klasifikasi reactive routing protocol, sebuah protokol yang hanya akan membuat rute ketika node sumber membutuhkannya. AODV memiliki dua fase, yaitu route discovery dan route maintenance. Route discovery digunakan untuk meminta dan meneruskan informasi rute yang terdiri dari proses pengiriman Route Request (RREQ) dan Route Reply (RREP), sedangkan route maintenance digunakan untuk mengetahui informasi adanya kesalahan pada rute. Pada fase ini terdapat proses pengiriman Route Error (RERR).

Pada kinerja AODV biasa, rute yang dipilih adalah rute dengan jarak terpendek tanpa memedulikan throughput dan hop yang ada di rute tersebut. Jika menggunakan metode pemilihan rute berdasarkan faktor throughput dan hop, maka rute yang dipilih

bukanlah rute dengan jarak terpendek, melainkan rute dengan throughput terbesar dan hop terkecil.

Pada Tugas Akhir ini dilakukan modifikasi pada proses pemilihan rute pada AODV berdasarkan throughput dan hop yaitu dengan cara memambahkan field throughput pada RREQ dan RREP dan menghitung nilai throughput setiap kali melakukan send request atau send reply. Node akan membandingkan nilai throughput paket RREQ atau RREP yang baru dengan nilai throughput paket RREQ atau RREP sebelumnya jika menerima paket RREQ atau RREP dengan sequence number yang sama. Node tersebut dan akan meneruskan paket RREQ atau RREP tersebut hanya jika nilai throughput paket tersebut leih besar dari paket RREQ atau RREP yang sebelumnya. Pemilihan rute akan mempertimbangkan nilai hop count jika nilai throughput paket RREP sama dengan nilai throughput paket sebelumnya.

Hal ini dilakukan agar dapat meningkatkan kinerja protokol AODV untuk mencari rute yang stabil dengan cara memodifikasi beberapa bagian dari mekanisme pengiriman paket RREQ dan RREP. Dari hasil uji coba, AODV yang dimodifikasi pada skenario grid berhasil meningkatkan nilai rata-rata Packet Delivery Ratio (PDR) hingga 7,23%, menurunkan Delay hingga 35,32% dan meningkatkan average throughput 5,93%. Sedangkan pada skenario real berhasil meningkatkan nilai rata-rata Packet Delivery Ratio (PDR) hingga 5,42%, menurunkan Delay hingga 49,36% dan meningkatkan average throughput hingga 8,34%.

Kata kunci: VANETs, AODV, Throughput, Hop

MODIFICATION OF ROUTE SELECTION IN AD HOC ON-DEMAND DISTANCE VECTOR (AODV) BASED ON THROUGHPUT FACTOR AND HOP IN VANET ENVIRONMENT

Student's Name	**: FATURRAHMAN MA'RUF**
Student's ID	**: 05111540000027**
Department	**: Informatics – FTEIC ITS**
First Advisor	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**
Second Advisor	**: Ary Mazharuddin Shiddiqi, S.Kom., M.Comp.Sc., Ph.D**

Abstract

VANETs are an improvement of MANET which have high mobility node characteristic and limited movement pattern. There are many routing protocols that can be implemented on VANETs and one of them is the AODV.

AODV is an example of reactive routing protocol classification, a protocol that will only create a route when the source node needs it. AODV have two phases: the route discovery and route the maintenance. Route discovery is used for requesting and forwarding a route information that consist of Route Request (RREQ) and Route Reply (RREP), meanwhile route maintenance that consist of Route Error (RERR) is used for finding out an error information in route.

In AODV, the choosen route is the route with shortest path without calculating throughput and hop. If route discovery use throughput and hop, then the choosen route might not the route with shortest path. It will choose route with the biggest throughput and smalles hop.

This final project modifies the route selection process in AODV based on thorugput and hop with adding field throughput on

RREQ and RREP and calculating throughput whenever send request or send reply.

This is performed in order to improve the performance of the AODV protocol to find a stable route by modifying some parts of the mechanism for sending RREQ and RREP. Experiment results indicated that the modified AODV in the grid scenario has increased the average value of the Packet Delivery Ratio (PDR) by 7,23%, decreased Delivery Delay value by 35,32%, and the value of average throughput by 5,93 %. While in the real scenario the average value of the Packet Delivery Ratio (PDR) has increased by 5,42% , the Delivery Delay value has decreased by 49,36%, and the value of average throughput has increased by 8,34%.

Keyword: VANETs, AODV, Throughput, Hop

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

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 Faktor *Throughput* Dan *Hop* Di Lingkungan Vanets".**

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

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

1. Allah SWT atas semua rahmat yang diberikan sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Bapak Sutarso dan Ibu Eni Nurhayati selaku kedua orangtua penulis atas segala dukungan berupa doa, moral, dan waktu sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Bapak Dr. Eng. Radityo Anggoro, S.Kom., M.Sc., dan Bapak Ary Mazharuddin Shiddiqi., S.Kom., M.Comp.Sc., Ph.D selaku dosen pembimbing penulis atas nasihat, arahan dan bantuannya sehingga penulis dapat menyeleksaikan Tugas Akhir ini.
4. Teman-teman yang memberikan bantuan secara langsung maupun tidak langsung (Huda, Naufal, Adib, Redo) kepada penulis.
5. Teman-reman Aku Pintar (Mas Gilang, Mas Kevin, Mas Hanif, Tegar, Fadli, Fina, Mbak Nafia, Mas Luqman, Mas Divi, Mas Majid, Mas Kukuh, Mas Ilyas, Mas Alek, Mas Iyus, Mas Arya, Fida) yang selalu mengisi hari-hari penulis selama pengerjaan Tugas Akhir ini.

6. Teman teman HMTC Optimasi, HMTC Inspirasi, Schematics 2016, dan Schematics 2017 yang telah memberikan waktu lebih untuk penulis semasa kuliah di Informatika ITS
7. Juga tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini dan yang telah membaca buku Tugas Akhir ini.

Penulis telah berusaha sebaik-baiknya dalam menyusun Tugas Akhir ini, namun penulis menyadari bahwa masih terdapat kekurangan, kesalahan, maupun kelalaian yang telah penulis lakukan. Oleh karena itu, saran dan kritik yang membangun sangat dibutuhkan untuk penyempurnaan Tugas Akhir ini.

Surabaya, Juni 2020

FATURRAHMAN MA'RUF

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

Mobile Ad Hoc Networks (MANETs) terdiri dari *node wireless* yang berkomunikasi satu sama lain tanpa infrastruktur terpusat [1]. MANETs menarik untuk diteliti karena sifat-sifat *self-configuration*, *self-organization* yang mudah di-*deploy*, dan integrasi dengan jaringan lain seperti Wi-Fi, WiMAX dan jaringan seluler [2]. *Vehicle Ad hoc Networks* (VANETs) merupakan perkembangan dari *Mobile Ad hoc Network* (MANET) dimana setiap node memiliki mobilitas yang tinggi yang menyebabkan perubahan dalam topologi jaringan sehingga kesulitan dalam mengaturnya.

Routing protocol dalam VANET dibedakan menjadi dua protokol, yaitu *proactive* dan *reactive*. *Proactive* adalah protokol yang bekerja dengan cara membentuk *routing table* dan melakukan *update* setiap saat pada selang waktu tertentu tanpa memperhatikan beban jaringan, *bandwidth,* dan ukuran jaringan sehingga menyebabkan meningkatnya *routing overhead*. Sedangkan *reactive* merupakan mekanisme *routing* yang membentuk *routing table* jika ada permintaan pengiriman data atau terjadinya perubahan rute dalam setiap jaringan sehingga menyebabkan mengurangnya *routing overhead*. Contoh *proactive routing protocol* adalah *Destination-Sequenced Distance Vector* (DSDV), dan *Optimized Link State Routing* protocol (OLSR) sedangkan contoh *reactive routing protocol* adalah *Adhoc On-Demand Distance Vector* (AODV), dan *Dynamic Source Routing* (DSR).

Salah satu contoh implementasi protokol AODV adalah diimplementasikannya protokol tersebut kedalam jaringan sensor nirkabel melalui simulasi skenario VANETs. Pada *Ad Hoc On-demand Distance vector* (AODV), pemilihan rute berdasarkan meminimalisir jumlah hop. Meminimalisir jumlah hop berarti

meningkatnya jarak setiap paket dan hal itu mengakibatkan mengurangnya kekuatan sinyal yang menyebabkan meningkatnya rasio *loss* paket. Performa AODV menurun ketika ada rute dengan jumlah hop tinggi dan *throughput* tinggi. *End-to-end delay* untuk rute dengan *throughput* tinggi adalah rendah dibandingkan rute dengan *throughput* rendah. Efek *throughput* suatu rute harus dipertimbangkan ketika memilih rute ketika proses *routing discovery* agar performa AODV lebih baik [3].

Pada Tugas Akhir ini diusulkan suatu mekanisme *routing discovery* pada AODV berdasarkan *throughput* dan hop yang bernama *Modified-AODV* (M-AODV). Hasil akhir yang diharapkan adalah mengetahui perbandingan kinerja antara AODV dan AODV yang sudah dimodifikasi yang diukur berdasarkan *packet delivery ratio* (PDR), *throughput*, *end-to-end delay*.

1.2 Rumusan Masalah

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

1. Bagaimana melakukan pemilihan rute berdasarkan *throughput* dan hop di lingkungan VANETs?
2. Bagaimana peranan M-AODV dalam meningkatkan rasio pengiriman paket?
3. Bagaimana peranan M-AODV terhadap performa AODV yang diukur berdasarkan *packet delivery ratio* (PDR), *throughput* dan *end-to-end delay*?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki batasan sebagai berikut:

1. Jaringan yang digunakan adalah jaringan *Vehicular Ad hoc Networks* (VANETs).

2. Simulasi pengujian jaringan menggunakan *Network Simulator 2* (NS-2).
3. Pembuatan skenario uji coba menggunakan *Simulation of Urban Mobility* (SUMO).

1.4 Tujuan

1. Melakukan pemilihan rute berdasarkan *throughput* dan hop di lingkungan VANETs.
2. Menganalisa peranan M-AODV dalam meningkatkan rasio pengiriman paket.
3. Menganlisa performa AODV yang telah dimodifikasi berdasarkan *packet delivery ratio* (PDR), *throughput* dan *end-to-end delay*

1.5 Manfaat

Manfaat yang diperoleh dari pengerjaan Tugas Akhir ini adalah dapat memberikan informasi tentang dampak dari modifikasi pemilihan rute berdasarkan *throughput* dan *hop* terhadap kinerja *routing protocol* AODV di lingkungan VANETs.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahapan awal dari Tugas Akhir ini adalah penyusunan Proposal Tugas Akhir. Proposal Tugas Akhir berisi pendahuluan, deskripsi dan gagasan metode-metode yang dibuat dalam Tugas Akhir ini. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya Tugas Akhir, rumusan masalah yang diangkat, batasan masalah untuk Tugas Akhir, dan manfaat dari hasil pembuatan Tugas Akhir ini. Selain itu dijabarkan pula

tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan Tugas Akhir.

1.6.2 Studi Literatur

Pada tahap ini, dipelajari sejumlah referensi yang diperlukan dalam melakukan implementasi yaitu mengenai VANETs, AODV, *Network Simulator* NS2, OpenStreetMap, Java OpenStreetMap (JOSM), SUMO, dan AWK.

1.6.3 Analisis dan Desain Sistem

Pada tahap ini dilakukan analisis dari hasil percobaan modifikasi AODV yang dibuat. Data yang dianalis berasal dari perhitungan *Packet Delivery Ratio (*PDR), *End-to-End Delay* paket dari *node* ke *node* lainnya, dan *average throughput*. Hal ini bertujuan untuk merumuskan solusi yang tepat untuk konfigurasi AODV yang dimodifikasi dalam lingkungan topologi MANET. Setelah selesai diaplikasikan pada MANET, dilakukan simulasi yang dilakukan pada VANETs dengan bantuan SUMO.

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

Pada tahap ini dilakukan pengujian menggunakan SUMO, sebuah *traffic generator* untuk membuat simulasi keadaan topologi yang diujikan. Hasil dari SUMO tersebut akan dijalankan

pada NS-2 yang akan menghasilkan *trace file*. *Packet Delivery Ratio, End-to-end Delay, average throughput* akan dihitung dari *trace file* tersebut untuk menguji performa AODV yang telah dimodifikasi.

1.6.6 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku yang menjelaskan seluruh konsep, teori dasar dari metode yang digunakan, implementasi, serta hasil yang telah dikerjakan sebagai dokumentasi dari pelaksanaan Tugas Akhir.

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, average throughput.*).
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.

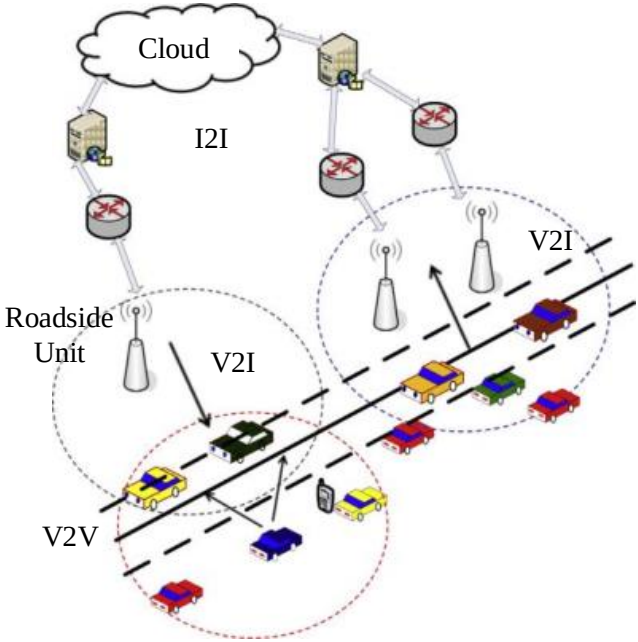
5. Bab V. Uji Coba dan Evaluasi
   Bab ini berisikan hasil uji coba dan evaluasi dari implementasi yang telah dilakukan untuk menyelesaikan masalah yang dibahas pada Tugas Akhir.
6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, 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.

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 VANETs

Vehicular Ad-hoc Networks (VANETs) merupakan pengembangan dari *Mobile Ad-hoc Network* (MANET) dimana pengembangannya difokuskan pada kendaraan (*vehicle*) yang dapat saling berkomunikasi maupun mengirimkan data. VANETs adalah sebuah teknologi baru yang memadukan kemampuan komunikasi nirkabel kendaraan menjadi sebuah jaringan yang bebas infrastuktur serta memiliki karakteristik mobilitas yang sangat tinggi dan terbatas pada pola pergerakannya. *Node* dalam jaringan dianggap sebagai *router* yang bebas bergerak dan bebas menentukan baik menjadi *client* maupun menjadi *router*. Protokol *routing* pada VANETs memiliki dua model yaitu protokol *reactive routing* yang membentuk tabel *routing* hanya saat dibutuhkan dan protokol *proactive routing* yang melakukan pemeliharaan tabel *routing* secara berkala pada waktu tertentu. Pergerakan *node* pada VANETs bisa berubah setiap saat dan terbatas pada rute lalu lintas yang dapat ditentukan dari koordinat peta. Hal ini membuat setiap *node* akan terus memperbarui informasi dalam tabelnya sesuai informasi dari *node* lain. Perubahaan pergerakan pada VANETs menjadi salah satu permasalahan dalam pengiriman paket data sehingga dibutuhkan informasi jarak antar *node*, kecepatan dan *delay* transmisi [4]. Ilustrasi VANETs dapat dilihat pada Gambar 2.1.



Gambar 2.1 Ilustrasi VANETs *[15]*

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

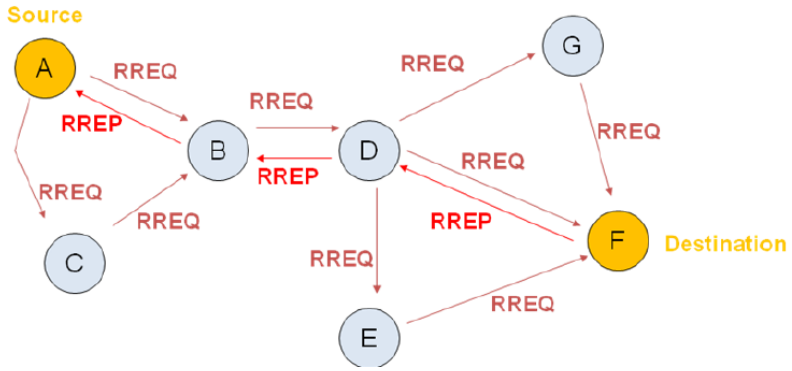
2.2 *Ad-hoc On-demand Distance Vector* (AODV)

Ad-hoc On-demand Distance Vector (AODV) adalah salah satu *routing* protokol yang termasuk dalam klasifikasi *reactive routing protocol*. Sebuah protokol yang hanya membuat sebuah rute saat dibutuhkan. AODV dikembangkan oleh C. E. Perkins, E.M. Belding-Royer dan S. Das pada RFC 3561 [5].

Ciri utama dari AODV adalah menjaga *timer-based state* pada setiap *node* sesuai dengan penggunaan tabel *routing*. Tabel *routing*

akan kadaluarsa jika jarang digunakan. Ada dua tahapan dalam AODV yaitu *route discovery* dan r*oute maintenance*. *Route discovery* memiliki dua pesan yaitu berupa *Route Request* (RREQ) dan *Route Reply* (RREP). Sedangkan *Route maintenance* berupa *Route Error* (RERR).

AODV adalah sebuah metode *routing* pesan antar *node* yang memungkinkan *node-node* tersebut untuk melewatkan pesan melalui lingkungannya ke *node* yang tidak dapat dihubungi secara langsung. AODV melakukan ini dengan cara menemukan rute yang bisa dilalui oleh pesan. Selain itu AODV juga memastikan rute ini tidak mengandung perulangan (*loop*), menangani perubahan rute, dan membuat rute baru apabila terjadi *error* [6]. Ilustrasi pencarian rute oleh AODV dapat dilihat pada Gambar 2.2.



Gambar 2.2 Ilustrasi pencarian rute *routing protocol* AODV *[13]*

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.

Sebagai contoh proses *route discovery* dalam AODV, ilustrasi pada Gambar 2.2 menggambarkan bagaimana *source node*, yaitu *node* A mencari rute untuk menuju *destination node* yaitu *node* F. *Node* A 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* F. Kemudian, jika rute menuju *node* F 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 *Network Simulator-2* (NS-2)

Network Simulator (NS) adalah suatu *interpreter* yang berorientasi objek, dan *discrete event-driven* yang dikembangkan oleh University of California Berkeley dan USC ISI sebagai bagian dari projek *Virtual INternet Testbed* (VINT) [7]. NS yang banyak dikenal dengan NS-2 (versi 2) menjadi salah satu *tool* yang sangat berguna untuk menunjukkan simulasi jaringan melibatkan *Local Area Network* (LAN), *Wide Area Network* (WAN), tapi fungsi dari *tool* ini telah berkembang selama beberapa tahun belakangan untuk

memasukkan jaringan nirkabel (*wireless*) dan juga jaringan *ad hoc* [8].

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

2.3.1 Instalasi

NS-2 membutuhkan beberapa *package* yang harus sudah ter*install* sebelum memulai instalasi NS-2. Untuk meng*install dependency* yang dibutuhkan dapat dilakukan dengan *command* yang ditunjukkan pada Gambar 2.3.

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

Gambar 2.3 Perintah untuk meng*install dependency* NS-2

Setelah meng*install dependency* yang dibutuhkan, ekstrak *package* NS-2 dan ubah baris kode ke-137 pada *file* ls.h di *folder* linkstate menjadi seperti pada Gambar 2.4.

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

Gambar 2.4 Baris kode yang diubah pada file ls.h

Instalasi dengan menjalankan perintah ./install pada folder NS-2.

2.3.2 *Trace File*

Trace file merupakan *file* hasil simulasi yang dilakukan oleh NS-2 dan berisikan informasi detail pengiriman paket data. *Trace file* digunakan untuk menganalisis performa *routing protocol* yang

disimulasikan. Detail penjelasan *trace file* ditunjukkan pada Tabel 2.1.

Tabel 2.1 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*)

2.4 OpenStreetMap

OpenStreetMap (OSM) adalah sebuah proyek berbasis web untuk membuat peta dunia yang gratis dan terbuka yang mendigitalisasi citra satelit dan mengumpulan serta membebaskan data geografis yang tersedia di publik. Melalui *Open Data Commons Open Database License 1.0,* kontributor OSM dapat memiliki, memodifikasi, dan membagikan data peta secara luas. Terdapat beragam jenis peta digital yang tersedia di internet, namun sebagian besar memiliki keterbatasan secara legal maupun teknis. Hal ini membuat masyarakat, pemerintah, peneliti dan akademisi, *inovator,* dan banyak pihak lainnya tidak dapat menggunakan data yang tersedia di dalam peta tersebut secara luas. Di sisi lain, baik peta dasar OSM maupun data yang tersedia di dalamnya dapat diunduh secara gratis dan terbuka, untuk kemudian digunakan untuk diredistribusikan kembali.

Di banyak tempat di dunia ini, terutama di daerah terpencil dan terbelakang secara ekonomi, tidak terdapat insentif komersil sama sekali bagi perusahaan pemetaan untuk mengembangkan data di tempat ini. OSM dapat menjadi jawaban di banyak tempat seperti ini,

baik itu pengembangan ekonomi, tata kota, kontinjensi bencana, maupun untuk berbagai tujuan lainnya [9].

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)

Java OpenStreetMap Editor (JOSM) adalah aplikasi untuk menyunting data yang didapatkan dari OpenStreetMap [10].

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 *Simulation of Urban Mobility* (SUMO)

Simulation of Urban Mobility (SUMO) merupakan paket simulasi lalu lintas yang bersifat *open-source* dimana pengembangannya dimulai pada tahun 2001. Dan semenjak itu SUMO telah berubah menjadi sebuah simulasi lalu lintas dengan kelengkapan fitur dan pemodelannya termasuk kemampuan jalannya jaringan untuk membaca *format* yang berbeda.

SUMO juga memungkinkan untuk mendefinisikan kendaraan dengan sifat tertentu seperti panjang kendaraan, kecepatan maksimum, percepatan dan perlambatannya. SUMO juga menyediakan pilihan bagi pengguna menentukan rute acak untuk kendaraan. Ada juga pilihan yang tersedia untuk model sistem transportasi umum, dimana setiap kendaraan datang dan berangkat sesuai dengan jadwal [11].

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.

2.7 AWK

AWK adalah bahasa pemrograman yang digunakan untuk melakukan *text processing* dan ekstraksi data [12]. AWK merupakan sebuah program filter untuk teks, seperti halnya perintah grep pada terminal linux. AWK dapat digunakan untuk mencari bentuk / model dalam sebuah berkas teks ke dalam bentuk teks lain. AWK dapat juga digunakan untuk melakukan proses aritmatika seperti yang dilakukan oleh perintah expr. AWK sama halnya seperti bahasa shell atau C yang memiliki karakteristik yaitu sebagai *tool* yang cocok untuk *jobs* juga sebagai pelengkap untuk *filter* standar.

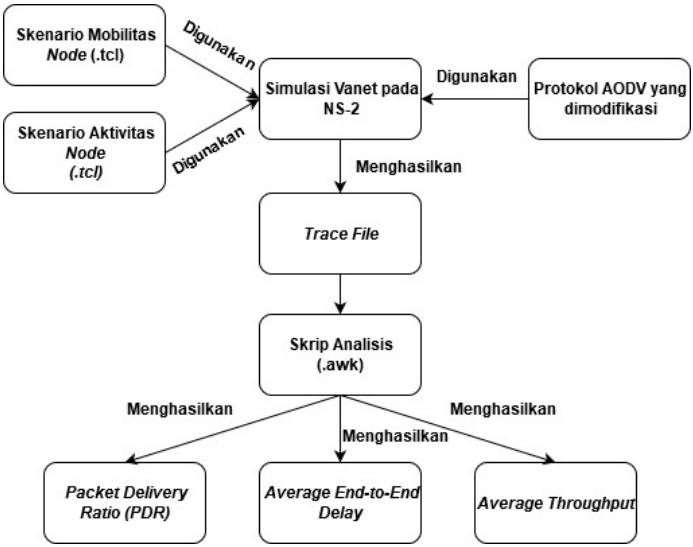
Pada Tugas Akhir ini, AWK digunakan untuk membuat script menghitung *Packet Delivery Ratio (*PDR*), End-to-end Delay, Routing Overhead* (RO), dan *Forwarded Route Request* (RREQ F) dari *trace file* NS2.

BAB III
PERANCANGAN

Perancangan merupakan bagian penting dari pembuatan sistem secara teknis sehingga bab ini secara khusus menjelaskan perancangan sistem yang dibuat dalam Tugas Akhir. 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 Gambar 3.1.



Gambar 3.1 Diagram Rancangan Simulasi AODV Modifikasi [16]

Modifikasi akan diawali dengan menambahkan *field throughput* pada RREQ dan RREP. *Source node* melakukan *broadcast* paket RREQ dan menyimpan nilai *throughput* dari node tersebut. Kemudian menghitung nilai *throughput* setiap kali mengirimkan RREP atau RREP. Saat *forward request,* akan dibandingkan nilai *throughput link* dengan nilai *throughput* RREQ untuk menentukan nilai *throughput RREQ* dan menyimpan nilai *throughput* pada *routing table*. Saat *forward reply*, akan dibandingkan nilai *throughput link* dengan nilai *throughput* RREP untuk menentukan nilai *throughput* RREP. Pemilihan rute akan mempertimbangkan nilai *hop count* jika nilai *throughput* RREP sama dengan nilai *throughput* paket sebelumnya.

Modifikasi yang telah dilakukan akan disimulasikan pada NS-2 dengan peta berbentuk *grid* dan peta *real* pada lingkungan lalu lintas di kota Surabaya. Pembuatan kedua peta tersebut menggunakan bantuan *tools* SUMO. Simulasi tersebut akan memberikan hasil *trace file* yang kemudian dianalisis menggunakan skrip AWK untuk mendapatkan *Packet Delivery Ratio* (PDR), *End-to-end Delay (*E2E), dan *Average Throughput.* Analisis tersebut dapat mengukur performa *routing protocol* AODV yang telah dimodifikasi dibandingkan dengan AODV sebelum dimodifikasi. Analisis ini digunakan untuk mengukur tingkat reliabilitas pengiriman data antara protokol AODV dengan protokol AODV yang dimodifikasi. Daftar istilah yang sering digunakan pada buku Tugas Akhir ini dapat dilihat pada Tabel 3.1.

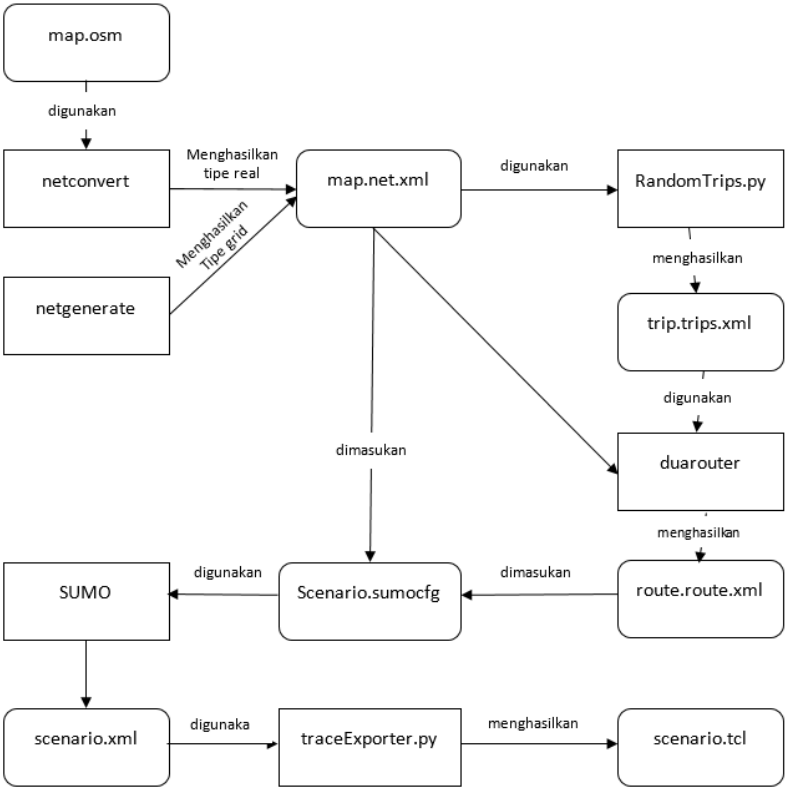
Tabel 3.1 Daftar Istilah

No.	Istilah	Penjelasan
1	AODV	Singkatan dari *Ad hoc On-demand Distance Vector*. Protokol yang digunakan pada Tugas Akhir ini.
2	PDR	*Packet Delivery Ratio*. Salah satu metrik analisis yang diukur. Berupa rasio jumlah pengiriman paket yang terkirim.
3	E2E	*Average End-to-End Delay*. Jeda waktu yang diukur saat paket terkirim.

No.	Istilah	Penjelasan
4	RO	*Routing Overhead*. Jumlah *control packet* yang terkirim
5	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.

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 Surabaya.



Gambar 3.2 Alur perancangan skenario [16]

3.2.1 Perancangan Skenario *Grid*

Perancangan skenario mobilitas *grid* diawali dengan merancang luas area peta *grid* yang dibutuhkan. Luas area tersebut bisa didapatkan dengan cara menentukan terlebih dahulu jumlah titik persimpangan yang diinginkan, sehingga dari jumlah persimpangan tersebut dapat diketahui berapa banyak peta yang dibutuhkan. Dengan mengetahui jumlah petak yang dibutuhkan, dapat ditentukan panjang tiap petak sehingga mendapatkan luas area yang dibutuhkan yaitu

berukuran 1500 m x 1500 m. Dengan 4 titik persimpangan, maka akan didapatkan 9 petak dan panjang tiap peta adalah 500m.

Peta *grid* yang telah ditentukan luasnya tersebut kemudian dibuat dengan menggunakan *tools* SUMO yaitu 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 dan duarouter.

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.

3.2.2 Perancangan Skenario *Real*

Perancangan skenario mobilitas *real* diawali dengan memilih area yang akan dijadikan simulasi. Pada Tugas Akhir ini, digunakan peta dari OpenStreetMap untuk mengambil area yang dijadikan model simulasi. Setelah memilih area, dilakukan pengunduhan dengan menggunakan fitur *export* yang telah disediakan oleh OpenStreetMap. Peta hasil *export* tersebut memiliki ekstensi .osm.

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 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.

3.3 Perancangan Modifikasi *Routing Protocol* AODV

Protokol AODV yang diajukan pada Tugas Akhir ini merupakan modifikasi dari protokol AODV yang mengubah mekanisme *route discovery* pada protokol tersebut. Pada protokol AODV, rute yang dipilih untuk pengiriman paket data adalah rute dengan jarak terpendek. Pada AODV yang dimodifikasi ini, rute akan dipilih berdasarkan *throughput* terbesar dan *hop count* yang lebih kecil.

Throughput didapatkan dengan cara menambahkan *field throughput* pada RREQ dan RREP. Nilai *throughput* yang tersimpan akan dibandingkan setiap kali mengirimkan RREQ atau RREP. Nilai *throughput* yang tersimpan pada paket RREQ atau RREP akan dibandingkan dengan nilai *throughput* pada setiap node yang dilalui. Perbandingan dilakukan untuk menentukan apakah nilai *throughput* dari paket RREQ atau RREP lebih kecil dari nilai *throughhput* node. Jika nilai *throughput* node lebih kecil, maka nilai *throughput* pada paket RREQ atau RREP diperbarui menjadi nilai *throughput* node tersebut. Jika tidak, maka paket RREQ atau RREP langsung diteruskan. Ketika paket RREQ atau RREP dengan *sequence number* yang sama diterima lebih dari 1 kali oleh sebuah node, maka node tersebut akan membandingkan nilai *throughput* paket RREQ atau RREP yang baru dengan nilai *throughput* paket RREQ atau RREP sebelumnya. Node tersebut akan meneruskan paket RREQ atau RREP jika nilai *throughput* paket tersebut lebih besar dari paket RREQ atau RREP sebelumnya.

Ketika paket RREQ mencapai destination node, maka akan dibalas dengan paket RREP yang diinisialisasi dengan nilai *throughput* dari destination node tersebut. Saat *forward reply*, akan dibandingkan nilai *throughput link* dengan nilai *throughput* RREP untuk menentukan nilai *throughput* RREP. Jika *source node* menerima lebih dari 1 paket RREP dengan *sequence destination*

number yang sama, maka *source node* akan menyimpan rute dengan nilai *throughput* yang lebih besar. Pemilihan rute akan mempertimbangkan nilai *hop count* jika nilai *throughput* RREP sama dengan nilai *throughput* paket sebelumnya.

3.4 Perancangan Simulasi pada NS-2

Simulasi VANETs pada NS-2 dilakukan dengan menggabungkan *file* skenario yang telah dibuat menggunakan SUMO dan *file* skrip dengan ekstensi .tcl yang berisikan konfigurasi lingkungan simulasi.

Kode yang diubah diantaranya adalah penghitungan nilai *throughput* ketika mengirimkan paket RREQ dan RREP pada *file* aodv.cc dan penambahan variable *throughput* pada file aodv.h. Pada saat simulasi NS-2 dijalankan, maka *routing protocol* AODV akan mengirim paket data melalui rute dengan nilai *throughput* besar dan *hop* kecil.

3.5 Perancangan Metrik Analisis

Berikut ini merupakan parameter – parameter yang akan dianalisis pada Tugas Akhir ini untuk dapat membandingkan performa dari *routing protocol* AODV yang asli dengan AODV yang telah dimodifikasi:

3.5.1 *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 berhasil pengiriman paket yang dilakukan Rumus untuk menghitung PDR dapat dilihat pada Persamaan 3.1.

$$PDR = \frac{received}{sent} x\ 100\ \% \qquad (3.1)$$

Keterangan:

PDR = *Packet Delivery Ratio*

received = banyak paket data yang diterima

sent = banyak paket data yang dikirimkan

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

Average End-to-End Delay dihitung berdasarkan rata-rata *delay* antara waktu paket data diterima dan waktu paket dikirimkan dalam satuan detik. Delay tiap paket didapat dari rentang waktu antara *node* asal saat mengirimkan paket dan *node* tujuan menerima paket. Delay tiap paket tersebut semua dijumlahkan dan dibagi dengan jumlah paket yang berhasil diterima, maka akan didapatkan rata – rata E2E, yang dapat dihitung dengan Persamaan 3.2.

$$E2E = \frac{\sum_{m=1}^{recvnum} CBRRecvTime - CBRSentTime}{recvnum} \qquad (3.2)$$

Keterangan:

E2E = *End-to-End Delay*

CBRRecvTime = Waktu *node* asal mengirimkan paket

CBRSentTime = Waktu *node* tujuan menerima paket

recvnum = Jumlah paket yang berhasil diterima

3.5.3 *Average Throughput*

Average Throughput adalah jumlah rata-rata yang merepresentasikan data yang sukses diterima oleh semua penerima

per satuan waktu. *Average Throughput* didapatkan dengan membagi jumlah paket diterima dengan selisih antara waktu berhenti dan waktu mulai paket dikirim. Perhitungan *Average Throughput* dapat dilihat dengan Persamaan 3.3.

$$AT = \frac{packetsize}{newTime - oldTime} \quad (3.3)$$

(Halaman ini sengaja dikosongkan)

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.

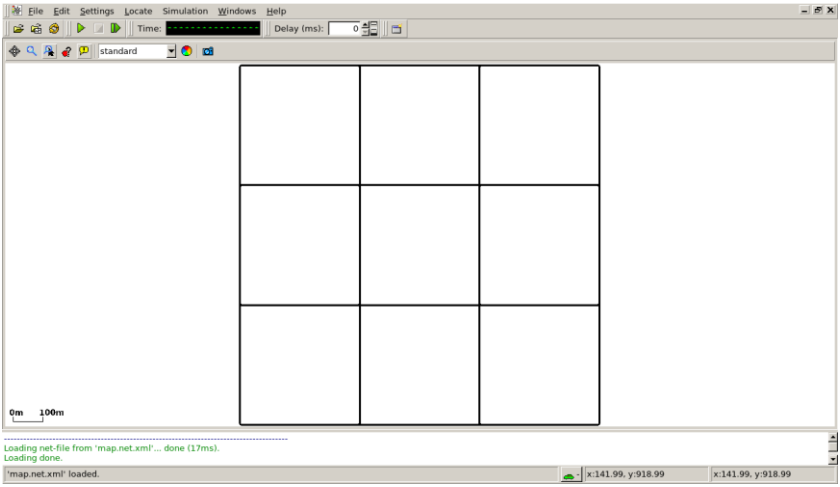
4.1.1 Skenario *Grid*

Dalam mengimplementasikan skenario *grid*, SUMO menyediakan *tools* untuk membuat peta *grid* yaitu netgenerate. Pada Tugas Akhir ini, penulis membuat peta *grid* dengan luas 650 m x 650 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 650 m x 650 m dibutuhkan luas per petak sebesar 200 m x 200 m. Berikut perintah netgenerate untuk membuat peta tersebut dengan kecepatan *default* kendaraan sebesar 20 m/s dapat dilihat pada Gambar 4.1.

```
netgenerate --grid --grid.number=4 --
grid.length=200 --default.speed=20 --
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* seacara 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 48 -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.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="http://sumo.dlr.d
e/xsd/sumoConfiguration.xsd">
   <input>
      <net-file value="map.net.xml"/>
      <route-files value="routes.rou.xml"/>
   </input>
   <time>
      <begin value="0"/>
      <end value="200"/>
   </time>
</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

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

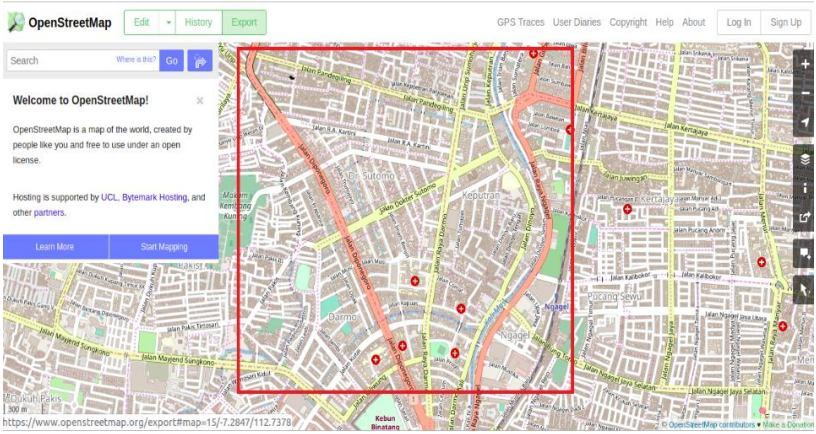
menggunakan NS-2. *Tools* yang digunakn untuk melakukan konversi ini adalah traceExporter. Perintah untuk menggunakan traceExporter dapat dilihat pada Gambar 4.7.

4.1.2 Skenario *Real*

Dalam mengimplementasikan skenario *real*, langkah pertama

Gambar 4.7 Perintah traceExporter

adalah menentukan area yang akan dijadikan area simulasi. Pada Tugas Akhir ini penulis mengambil area jalan sekitar Jl. Dr. Soetomo Surabaya. Area simulasi ditandai di dalam kotak berwarna merah. Setelah menentukan area simulasi, ekspor data peta dari OpenStreetMap seperti yang ditunjukkan pada Gambar 4.8.



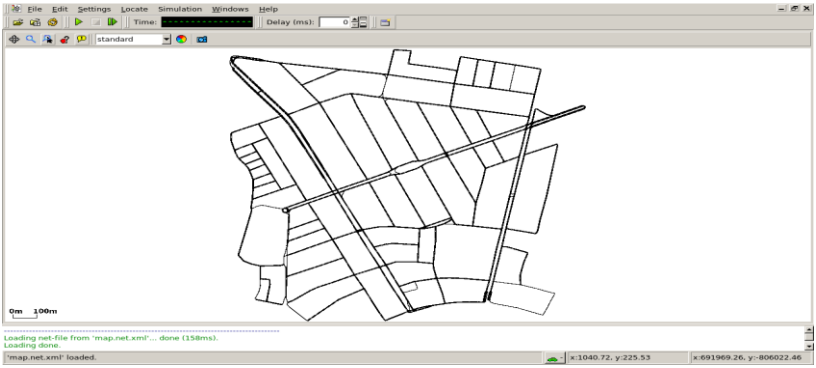
Gambar 4.8 Ekspor Peta dari OpenStreetMap

File hasil ekspor dari OpenStreetMap tersebut adalah *file* peta dengan ekstensi .osm. Kemudian konversi *file* .osm tersebut menjadi peta dalam bentuk *file* berekstensi .xml menggunakan *tools* netconvert dari SUMO. Perintah untuk menggunakan netconvert dapat dilihat pada Gambar 4.9.

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

Gambar 4.9 Perintah netconvert

Hasil konversi peta dari *file* berekstensi .osm menjadi *file* berekstensi .xml dapat dilihat menggunakan *tools* sumo-gui seperti yang ditunjukkan pada Gambar 4.10.



Gambar 4.10 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 Menentukan *Forwarding Node*

Pada Tugas Akhir ini dilakukan modifikasi pada *routing protocol* AODV agar dapat mengurangi jumlah *forwarding node*, yaitu *node* yang bertugas untuk melakukan *rebroadcast* paket RREQ. Hal tersebut dilakukan dengan cara memilih *forwarding node* berdasarkan *clustering* dengan cara membuat *node cluster head* yang ditentukan berdasarkan perhitungan *threshold* jumlah tetangga *node* tersebut dan dihitung berdasarkan pasangan node pada scenario yang bersifat statis, sehingga dapat dilihat peningkatan performa pada *routing* AODV yang telah dimodifikasi.

Implementasi modifikasi routing protocol AODV ini dibagi menjadi 3 bagian yaitu:

- Implementasi Penghitungan *Throughput* di Tiap Node
- Implementasi Penghitungan *Throughput* di Paket RREQ dan RREP
- Implementasi Pemilihan Rute dengan *Throughput* besar dan *Hop* Kecil

Kode implementasi dari routing protocol AODV pada NS-2 versi 2.35 berada pada direktori ns-2.35/aodv. Pada direktori tersebut terdapat beberapa file diantaranya seperti aodv.cc, aodv.h dan sebagainya. Pada Tugas Akhir ini, penulis memodifikasi *file* aodv.cc yang terdapat dalam *folder* ns-2.35/aodv untuk menghitung *throughput* di tiap node, menghitung *throughput* pada paket RREQ dan RREP, dan pemilihan rute dengan *throughput* besar dan *hop* kecil, dan *file* aodv.h yang ada di dalam *folder* ns-2.35/aodv untuk mendaftarkan fungsi dan variabel baru. Pada bagian ini penulis akan menjelaskan langkah – langkah dalam mengimplementasikan modifikasi *routing protocol* AODV dalam memilih *throughput* besar dan *hop* kecil.

4.3 Implementasi Simulasi pada NS-2

Implementasi simulasi VANETs diawali dengan pendeskripsian lingkungan simulasi pada sebuah *file* tcl. *File* ini berisikan konfigurasi setiap *node* dan langkah-langkah yang dilakukan selama simulasi. Potongan konfigurasi lingkungan simulasi dapat dilihat pada Gambar 4.15.

```
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 opt(x)       1500
set opt(y)       1500
set val(ifqlen)  1000
set val(nn)      200
set val(seed)    1.0
set val(adhocRouting) AODV
set val(stop)    200
set val(cp)      "cbr200.tcl"
set val(sc)      "scenario.tcl"
```

Gambar 4.11 Implementasi Simulasi NS-2

Pada konfigurasi dilakukan pemanggilan terhadap *file traffic* yang berisikan konfigurasi *node* asal, *node* tujuan, pengiriman paket, serta *file* skenario yang berisi pergerakan *node* yang telah di*generate* oleh SUMO. Kode implementasi pada NS-2 dapat dilihat pada lampiran A.5 Kode Skenario NS-2.

Konfigurasi untuk *file traffic* bisa dilakukan dengan membuat *file* berekstensi .txt untuk menyimpan konfigurasi tersebut. Pada *file* konfigurasi lingkungan simulasi, *file traffic* tersebut dimasukkan agar

dibaca sebagai *file traffic*. Potongan konfigurasi *file traffic* dapat dilihat pada Gambar 4.16.

```
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(198) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(199) $null_(0)
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_ 1000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 0 "$cbr_(0) start"
$ns_ at 200.0000000000000000 "$cbr_(0) stop"
```

Gambar 4.12 Implementasi Simulasi File Traffic

Pada konfigurasi tersebut, ditentukan *node* sumber dan *node* tujuan pengiriman paket. Pengiriman dimulai pada detik ke- 2.55. Implementasi konfigurasi *file traffic* untuk simulasi pada NS-2 dapat dilihat pada lampiran A.8 Kode Konfigurasi *Traffic*

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 dalam bentuk *file* berekstensi .tr. Dari data *trace file* tersebut, dapat dilakukan analisis performa *routing protocol* dengan mengukur beberapa metrik. Pada Tugas Akhir ini, metrik yang akan dianalisis adalah PDR, E2E, *average thorughput*.

4.4.1 Implementasi *Packet Delivery Ratio* (PDR)

Pada subbab 2.3.2 telah ditunjukkan contoh struktur data *event* yang dicatat dalam *trace file* oleh NS-2. Kemudian, pada persamaan 3.1 telah dijelaskan bagaimana menghitung PDR. Skrip awk untuk menghitung PDR berdasarkan kedua informasi tersebut dapat dilihat pada lampiran A.9 Kode Skrip AWK *Packet Delivery Ratio*.

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 3.1. Pseudocode untuk menghitung PDR dapat dilihat pada Gambar 4.17.

```
sent = 0
received = 0
for i = 1 to the number of rows
     if in a row contains “s” and AGT then
          sent++
     else if in a row contains “r” and AGT then
          received++
     end if
pdr = received / sent
```

Gambar 4.13 Pseudocode untuk Perhitungan PDR

Contoh perintah pengeksekusian skrip awk untuk menganalisis *trace file* adalah `awk -f pdr.awk result.tr`.

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

Skrip awk untuk menghitung E2E dapat dilihat pada lampiran A.10 Kode Skrip AWK Rata-Rata *End-to-End Delay*.

Dalam perhitungan E2E, langkah yang digunakan untuk mendapatkan E2E hampir sama dengan ketika mencari PDR, hanya saja yang perlu diperhatikan adalah waktu dari sebuah *event* yang tercatat pada kolom ke-2 dengan *filter event* pada kolom ke-4 adalah layer AGT dan *event* pada kolom pertama guna membedakan paket 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 syarat 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. *Pseudocode* untuk menghitung rata-rata E2E dapat dilihat pada Gambar 4.18.

```
sum_delay = 0
counter = 0

for i = 1 to the number of rows
     counter++
     if layer == AGT and event == s then
      start_time[packet_id] = time
     else if layer == AGT and event == 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]
```

Gambar 4.14 Pseudocode untuk Perhitungan E2E

Contoh perintah pengeksekusian skrip awk untuk menganalisis *trace file* adalah `awk –f e2e.awk result.tr`

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™ i7-7700HQ CPU @ 2.80GHz
Sistem Operasi	Ubuntu 18.04.2 LTS
Linux Kernel	Linux kernel 4.4
Memori	12.0 GB
Penyimpanan	200 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 menggunakan kode yang terdapat pada lampiran A.9 Kode Skrip

AWK *Packet Delivery Ratio*, A.10 Kode Skrip AWK Rata-Rata *End-to-End Delay*.

Tabel 5.2 Lingkungan Uji Coba

No.	Parameter	Spesifikasi
1	Network simulator	NS-2.35
2	Routing protocol	AODV
3	Waktu simulasi	200 detik
4	Area simulasi	1500 m x 1500 m
5	Jumlah *Node*	50, 100, 150, 200
6	Radius transmisi	400m
7	Kecepatan maksimum	20 m/s
8	Protokol MAC	IEEE 802.11p
9	Model Propagasi	*Two-ray ground*

5.2 Hasil Uji Coba

Hasil uji coba menggunakan *node* sumber dan *node* tujuan yang diletakkan secara statis. Hasil dari scenario *grid* dan scenario *real* untuk Tugas Akhir ini dapat diliht sebagai berikut:

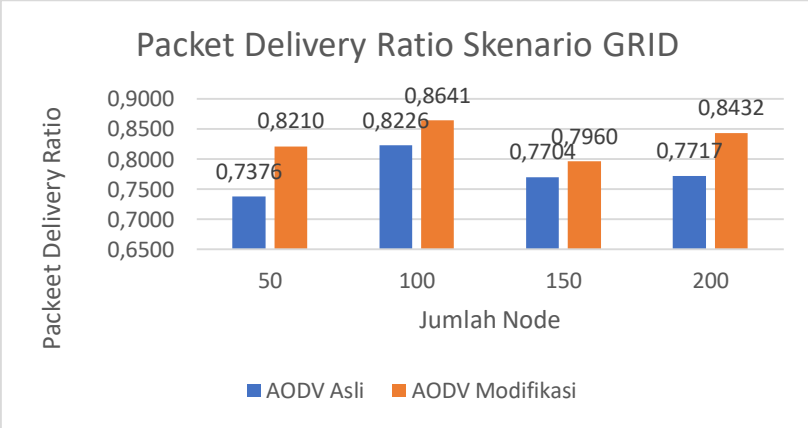
5.2.1 Hasil Uji Coba Skenario *Grid*

Pengujian pada skenario *grid* digunakan untuk melihat perbandingan PDR, E2E, *Average Throughput* antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi dalam pemilihan *node* yang dapat menerima paket *route request*.

Pengambilan data uji PDR, E2E, *Average Throughput* pada skenario *grid* dilakukan sebanyak 10 kali dengan skenario mobilitas *random* pada peta *grid* dengan luas area 1500 m x 1500 m dan *node* sebanyak 50 untuk lingkungan yang jarang, 100 dan 150 *node* untuk lingkungan yang sedang, dan 200 *node* untuk lingkungan yang padat dilakukan pada kecepatan standar yaitu 20 m/s. Hasil analisis dapat dilihat pada Tabel 5.3, Tabel 5.4, Tabel 5.5, dan Tabel 5.6.

Tabel 5.3 Hasil Rata - Rata PDR Skenario *Grid*

Jumlah *Node*	AODV Asli	AODV Modifikasi	Perbedaan
50	73,76%	82,10%	+ 8,34%
100	82,26%	86,41%	+ 4,15%
150	77,04%	79,60%	+ 2,56%
200	77,17%	84,32%	+ 7,15%



Gambar 5.1 Grafik Packet Delivery Ratio Skenario *Grid*

Berdasarkan grafik pada Gambar 5.1 dapat dilihat bahwa routing protocol AODV yang telah dimodifikasi dan juga routing protocol AODV asli mengalami kenaikan yang signifikan pada packet delivery ratio. Pada lingkungan yang jarang dengan jumlah 50 node, menghasilkan perbedaan selisih PDR sebesar 0.0834, atau naik menjadi sekitar 11,30% dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal PDR tersebut. Pada lingkungan yang sedang dengan jumlah 100 node, menghasilkan perbedaan selisih PDR sebesar 0.0415, atau naik menjadi sekitar 5.04% dimana routing protocol AODV yang telah dimodifikasi unggul dalam hal PDR tersebut dari AODV asli. Pada lingkungan yang sedang dengan jumlah 150 node, menghasilkan perbedaan selisih PDR sebesar

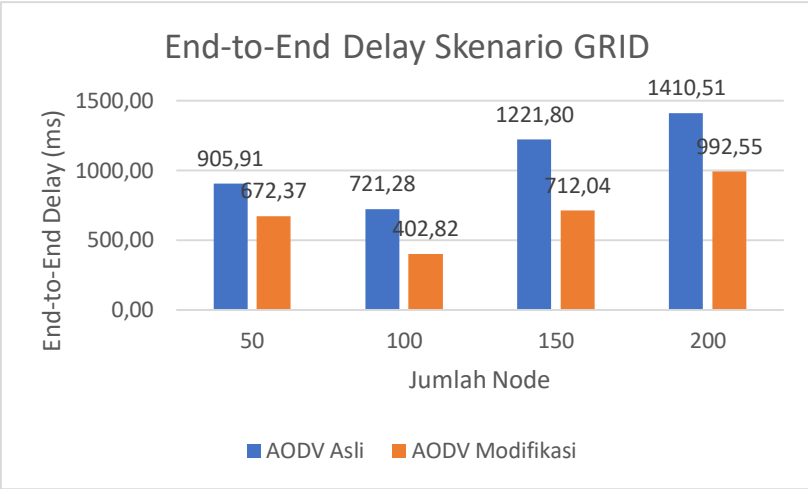
0.0256, atau naik menjadi sekitar 3,32% dimana routing protocol AODV yang telah dimodifikasi juga unggul dalam hal PDR tersebut dari AODV asli. Pada lingkungan yang padat dengan jumlah 200 node, menghasilkan perbedaan selisih PDR sebesar 0.0715, atau naik menjadi sekitar 9,26% dimana routing protocol AODV yang telah dimodifikasi juga unggul dalam hal PDR tersebut.

Dapat dilihat rata-rata kenaikan yang terjadi untuk PDR adalah 7,23%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa pada lingkungan dengan jumlah node 100, menghasilkan PDR yang lebih bagus daripada di lingkungan dengan jumlah node 50, 150, dan 200 untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan PDR yang lebih bagus daripada AODV asli dengan jumlah selisih PDR yang cukup signifikan.

Hasil pengambilan data rata-rata untuk *end-to-end delay* (E2E) pada skenario *grid* dengan jumlah *node* 50, 100, 150, dan 200 dapat dilihat pada Gambar 5.2.

Tabel 5.4 Hasil Rata - Rata E2E Skenario Grid

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	905,91 ms	672,37 ms	- 233,54 ms
100	721,28 ms	402,82 ms	- 318,46 ms
150	1221,80 ms	712,04 ms	- 509,76 ms
200	1410,51 ms	992,55 ms	- 369,93 ms



Gambar 5.2 Grafik End-to-end Delay Skenario *Grid*

Berdasarkan grafik pada Gambar 5.2 dapat dilihat bahwa rata-rata *end-to-end delay* antara routing protocol AODV asli dan AODV yang telah dimodifikasi mengalami penurunan yang signifikan. Pada lingkungan yang jarang dengan jumlah 50 node, terjadi perbedaan selisih *end-to-end delay* sebesar 233,54 ms antara routing protocol AODV asli dengan routing protocol AODV yang telah dimodifikasi atau mengalami penurunan sebesar 25,77%, dimana routing protocol AODV yang telah dimodifikasi unggul dalam hal *end-to-end delay* tersebut. Sedangkan pada lingkungan sedang dengan jumlah 100 node, terjadi perbedaan selisih *end-to-end delay* sebesar 318,46 ms antara routing protocol AODV asli dengan routing protocol AODV yang telah dimodifikasi atau mengalami penurunan sebesar 44,15%, dimana routing protocol AODV yang dimodifikasi lebih unggul dalam hal *end-to-end delay* tersebut. Pada pada lingkungan sedang dengan jumlah 150 node, terjadi perbedaan selisih *end-to-end delay* sebesar 509,76 ms antara routing protocol AODV asli dengan routing protocol AODV yang telah dimodifikasi atau mengalami penurunan sebesar 41,72%, dimana routing protocol AODV yang dimodifikasi lebih unggul dalam hal *end-to-end delay*

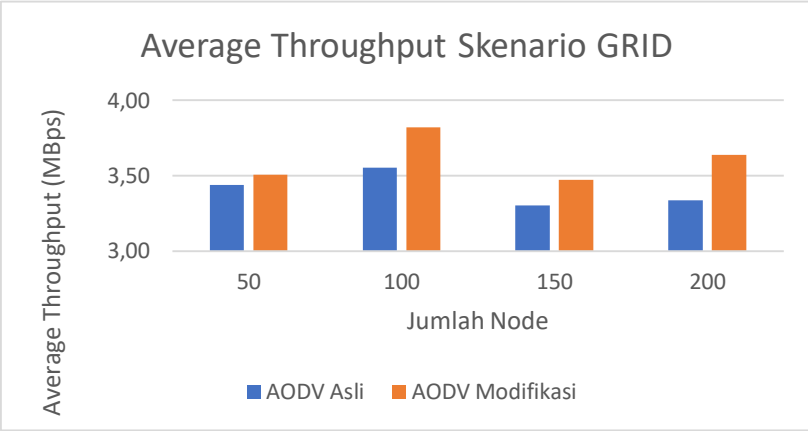
tersebut. Pada lingkungan yang padat dengan jumlah 200 node, terjadi perbedaan selisih *end-to-end delay* sebesar 417,96 ms antara routing protocol AODV asli dengan routing protocol AODV yang telah dimodifikasi atau mengalami penurunan sebesar 29,63%, dimana routing protocol AODV yang telah dimodifikasi jauh lebih unggul dalam hal *end-to-end delay* tersebut.

Jika ketiga lingkungan tersebut dibandingkan,dapat dilihat bahwa semakin padat lingkungannya, maka *end-to-end delay* yang dihasilkan semakin bagus dengan menggunakan *routing protocol* AODV yang telah dimodifikasi. Hal ini bisa terjadi karena dengan AODV modifikasi kali ini rute yang dipilih adalah rute dengan *throughput* besar dan *hop* kecil sehingga paket-paket akan lebih sedikit mengalami adanya delay dalam pengiriman.

Untuk hasil pengambilan data *averae throughput* pada skenario *grid* 50 *node*, 100 *node*, 150 *node* dan 300 *node* dapat dilihat pada Gambar 5.3.

Tabel 5.5 Hasil Rata - Rata *Throughput* Skenario Grid

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	3,44 MB/s	3,51 MB/s	+ 0,07 MB/s
100	3,55 MB/s	3,82 MB/s	+ 0,27 MB/s
150	3,30 MB/s	3,47 MB/s	+ 0,17 MB/s
200	3,34 MB/s	3,64 MB/s	+ 0,30 MB/s



Gambar 5.3 Grafik Aerage Throughput Skenario *Grid*

Berdasarkan grafik pada Gambar 5.3 dapat dilihat bahwa rata-rata *average throughput* antara routing protocol AODV asli dan AODV yang telah dimodifikasi mengalami kenaikan. Pada lingkungan yang jarang dengan jumlah 50 node, menghasilkan perbedaan selisih *average throughput* sebesar 0,07 atau mengalami kenaikan sebesar 2,07%, dimana routing protocol AODV modifikasi unggul dalam hal *average throughput* tersebut karena menghasilkan *average throughput* yang lebih rendah dari *average throughput* AODV asli. Pada lingkungan yang sedang dengan jumlah 100 node, menghasilkan perbedaan selisih *average throughput* sebesar 0.27 atau mengalami kenaikan sebeasr 7,54%, dimana routing protocol AODV yang telah dimodifikasi lebih unggul dari AODV asli. Pada lingkungan yang sedang dengan jumlah 150 node, menghasilkan perbedaan selisih *average throughput* sebesar 0.17 atau mengalami kenaikan sebesar 3,00%, dimana routing protocol AODV yang telah dimodifikasi lebih unggul dari AODV asli. Pada lingkungan yang padat dengan jumlah 200 node, menghasilkan perbedaan selisih *average throughput* sebesar 0,20 atau mengalami kenaikan sebesar 8,98%, dimana *average throughput* AODV yang telah dimodifikasi lebih unggul daripada AODV asli.

Dapat dilihat rata-rata kenaikan yang terjadi untuk *average throughput* adalah sebesar 5,93%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa node dengan jumlah 100 menghasilkan *average* throughput yang lebih tinggi daripada di lingkungan dengan jumlah node 50, 150, 200 untuk *routing* protocol AODV asli maoupun *routing* protocol AODV yang telah dimodifikasi. Dapat dilihat juga bahwa AODV yang telah dimodifikasi mempunyai *average throughput* yang lebih bagus daripada AODV asli.

5.2.2 Hasil Uji Coba Skenario *Real*

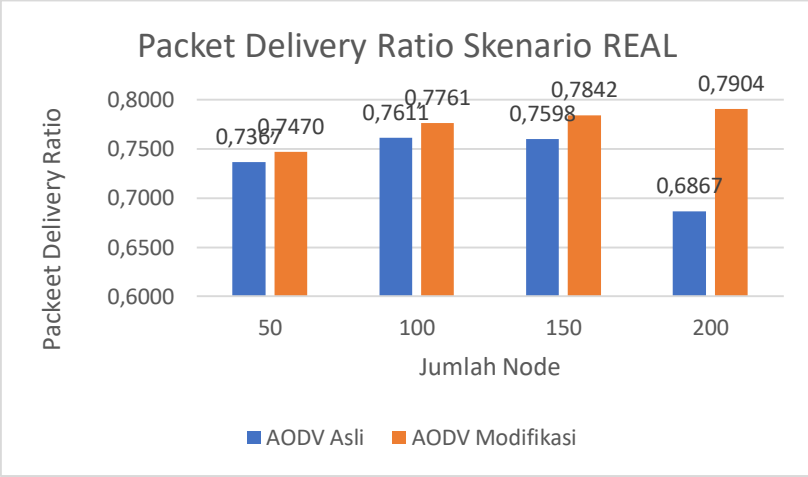
Pengujian pada skenario *real* digunakan untuk melihat perbandingan PDR, E2E, *average throughput* antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi dalam pemilihan *node* yang dapat menerima paket *route request*.

Pengambilan data uji PDR, E2E, *average throughput* pada skenario *real* dilakukan sebanyak 10 kali dengan skenario mobilitas *random* pada peta *real* dengan luas area 1500 m x 1500 m dengan *range transmisi* 400 meter dan *node* sebanyak 50 untuk lingkungan yang jarang, 100 dan 150 *node* untuk lingkungan yang sedang, dan 200 *node* untuk lingkungan yang padat dilakukan pada kecepatan standar yaitu 20 m/s. Untuk uji coba setiap lingkungan menggunakan interval yang berbeda-beda untuk mencari nilai interval yang terbaik dari hasil skenario. Interval waktu yang digunakan adalah 5 detik, 10 detik dan 15 detik. Hasil analisis dapat dilihat pada Tabel 5.6, Tabel 5.7, Tabel 5.9, dan Tabel 5.8.

Tabel 5.6 Hasil Rata - Rata PDR pada Skenario Real

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	0.7367	0,7470	+ 0,0103
100	0.7611	0,7761	+ 0,0150
150	0.7598	0,7842	+ 0,0243
200	0.6867	0,7904	+ 0,1037

Dari data di atas, dibuat grafik yang merepresentasikan hasil perhitungan PDR yang ditunjukkan pada Gambar 5.5.



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

Berdasarkan grafik pada Gambar 5.5 dapat dilihat bahwa routing protocol AODV yang telah dimodifikasi dan juga routing protocol AODV asli mengalami kenaikan pada packet delivery ratio. Pada lingkungan yang jarang dengan jumlah 50 node, menghasilkan perbedaan selisih PDR sebesar 0.0103, atau naik menjadi sekitar 10,30% dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal PDR tersebut. Pada lingkungan yang sedang dengan jumlah 100 node, menghasilkan perbedaan selisih PDR sebesar 0.0150, atau naik menjadi sekitar 1,96% dimana routing protocol AODV yang telah dimodifikasi unggul dalam hal PDR tersebut dari AODV asli. Pada lingkungan yang sedang dengan jumlah 150 node, menghasilkan perbedaan selisih PDR sebesar 0.0243, atau naik menjadi sekitar 3,20% dimana routing protocol AODV yang telah dimodifikasi unggul dalam hal PDR tersebut dari AODV asli. Pada

lingkungan yang padat dengan jumlah 200 node, menghasilkan perbedaan selisih PDR sebesar 0.1037, atau naik menjadi sekitar 15,10 % dimana routing protocol AODV yang telah dimodifikasi juga unggul dalam hal PDR tersebut.

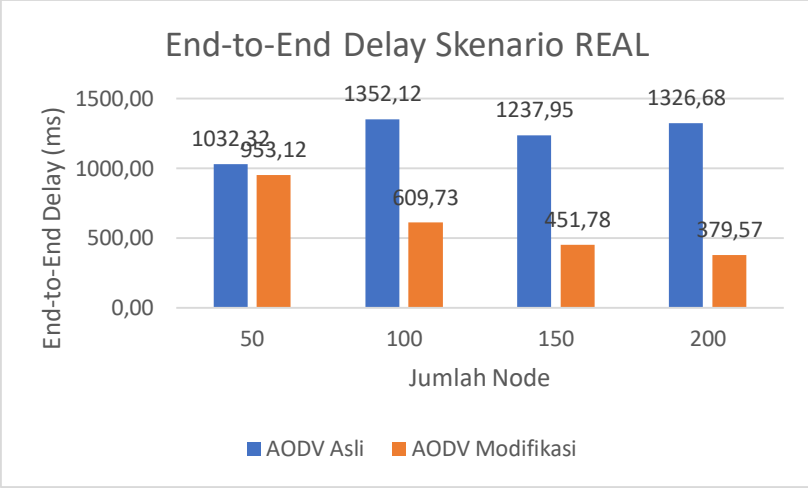
Dapat dilihat rata-rata kenaikan yang terjadi untuk PDR adalah 5,42%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang lebih banyak atau pada lingkungan dengan *node* yang padat, menghasilkan PDR yang lebih bagus daripada di lingkungan dengan jumlah *node* yang jarang maupun yang sedang baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan PDR yang lebih bagus daripada AODV asli dengan jumlah selisih PDR yang cukup signifikan.

Hasil pengambilan data rata-rata untuk *end-to-end delay* (E2E) pada skenario *real* dengan jumlah *node* 50, 100, 150, dan 200 dapat dilihat pada Tabel 5.8.

Tabel 5.7 Hasil Rata -Rata E2E pada Skenario Real

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	1032,32 ms	953,12 ms	- 79,20 ms
100	1352,12 ms	609,73 ms	-742,39 ms
150	1237,95 ms	451,78 ms	-786,17 ms
200	1326,68 ms	379,57 ms	-947,11 ms

Dari data di atas, dibuat grafik yang merepresentasikan hasil perhitungan E2E yang ditunjukkan pada Gambar 5.6.



Gambar 5.5 Grafik *End-to-end Delay* pada Skenario *Real*

Berdasarkan grafik pada Gambar 5.6 dapat dilihat bahwa rata-rata *end-to-end delay* antara routing protocol AODV asli dan AODV yang telah dimodifikasi mengalami perubahan yang cukup signifikan. Pada lingkungan yang jarang dengan jumlah 50 node, terjadi perbedaan selisih *end-to-end delay* sebesar 79,20 ms antara routing protocol AODV asli dengan routing protocol AODV yang telah dimodifikasi atau mengalami penurunan sebesar 7,67%, dimana routing protocol AODV yang telah dimodifikasi lebih unggul dari AODV asli dalam hal *end-to-end delay* tersebut. Sedangkan pada lingkungan sedang dengan jumlah 100 node, terjadi perbedaan selisih *end-to-end delay* sebesar 742,39 ms antara routing protocol AODV asli dengan routing protocol AODV yang telah dimodifikasi atau mengalami penurunan sebesar 54,90%, dimana routing protocol AODV yang dimodifikasi lebih unggul dalam hal *end-to-end delay*. Pada pada lingkungan sedang dengan jumlah 150 node, terjadi perbedaan selisih *end-to-end delay* sebesar 786,17 ms antara routing protocol AODV asli dengan routing protocol AODV yang telah

dimodifikasi atau mengalami penurunan sebesar 63,50%, dimana routing protocol AODV yang dimodifikasi lebih unggul dalam hal *end-to-end delay* tersebut. Pada lingkungan yang padat dengan jumlah 200 node, terjadi perbedaan selisih *end-to-end delay* sebesar 947,11 ms antara routing protocol AODV asli dengan routing protocol AODV yang telah dimodifikasi atau mengalami penurunan sebesar 71,39%, dimana routing protocol AODV yang telah dimodifikasi jauh lebih unggul dalam hal *end-to-end delay* tersebut.

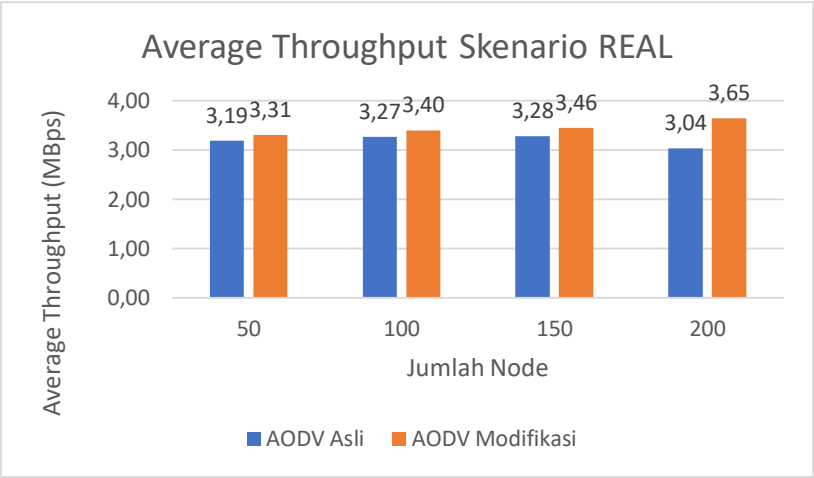
Jika ketiga lingkungan tersebut dibandingkan, dapat dilihat pada semua lingkungan, AODV yang telah dimodifikasi menghasilkan *end-to-end delay* yang lebih baik daripada AODV asli dengan jumlah selisih *end-to-end delay* yang cukup signifkan.

Untuk hasil pengambilan data *average throughput* pada scenario *real* 50 *node*, 100 *node*, 150 *node* dan 300 *node* dapat dilihat pada Tabel 5.10.

Tabel 5.8 Hasil *Average Throughput* pada Skenario Real

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	3,19 MB/s	3,31 MB/s	+0,13 MB/s
100	3,27 MB/s	3,40 MB/s	+0,13 MB/s
150	3,28 MB/s	3,46 MB/s	+0,18 MB/s
200	3,04 MB/s	3,65 MB/s	+0,61 MB/s

Dari data di atas, dibuat grafik yang merepresentasikan hasil perhitungan *average throughput* yang ditunjukkan pada Gambar 5.8.



Gambar 5.6 Grafik *Average Throughput* Skenario *Real*

Berdasarkan grafik pada Gambar 5.8 dapat dilihat bahwa rata-rata *average throughput* antara routing protocol AODV asli dan AODV yang telah dimodifikasi mengalami kenaikan. Pada lingkungan yang jarang dengan jumlah 50 node, menghasilkan perbedaan selisih *average throughput* sebesar 0,13 atau mengalami kenaikan sebesar 3,9%, dimana routing protocol AODV modifikasi unggul dalam hal *average throughput* tersebut karena menghasilkan *average throughput* yang lebih rendah dari *average throughput* AODV asli. Pada lingkungan yang sedang dengan jumlah 100 node, menghasilkan perbedaan selisih *average throughput* sebesar 0.13 atau mengalami kenaikan sebeasr 3,9%, dimana routing protocol AODV yang telah dimodifikasi lebih unggul dari AODV asli. Pada lingkungan yang sedang dengan jumlah 150 node, menghasilkan perbedaan selisih *average throughput* sebesar 0.18 atau mengalami kenaikan sebesar 5,36%, dimana routing protocol AODV yang telah dimodifikasi lebih unggul dari AODV asli. Pada lingkungan yang padat dengan jumlah 200 node, menghasilkan perbedaan selisih *average throughput* sebesar 0,61 atau mengalami kenaikan sebesar 20,15%, dimana *average throughput* AODV yang telah dimodifikasi lebih unggul daripada AODV asli.

Dapat dilihat rata-rata kenaikan yang terjadi untuk *average throughput* adalah sebesar 8,34%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa node dengan jumlah 200 menghasilkan *average* throughput yang lebih tinggi daripada di lingkungan dengan jumlah node 50, 100, 150 untuk *routing* protocol AODV asli maoupun *routing* protocol AODV yang telah dimodifikasi. Dapat dilihat juga bahwa AODV yang telah dimodifikasi mempunyai *average throughput* yang lebih bagus daripada AODV asli.

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. Pemilihan rute berdasarkan *throughput* dan *hop* bisa dilakukan dengan cara mengirim paket RREQ dengan *throughput* lebih besar jika node dengan *sequence number* yang sama menerima paket RREQ lebih dari 1 kali dan memilih *hop* yang lebih kecil jika nilai *throughput* sama dan mengirim paket RREP dengan *throughput* yang lebih besar jika *source node* menerima paket RREP lebih dari 1 kali dan memilih *hop* yang lebih kecil jika nilai *throughput* paket RREP sama dengan paket RREP sebelumnya.
2. Berdasarkan hasil analisis uji coba pada skenario grid, dampak pemilihan rute berdasarkan *throughput* dan *hop* adalah rata-rata peningkatan *Packet Delivery Ratio*(PDR) sebesar 7,23%, rata-rata penurunan *End-to-End Delay* sebesar 35,32%. Rata-rata peningkatan *throughput* adalah 5,93%.
3. Berdasarkan hasil analisis uji coba pada skenario real, dampak pemilihan rute berdasarkan *throughput* dan *hop* adalah rata-rata peningkatan *Packet Delivery Ratio*(PDR) sebesar 5,42%, rata-rata penurunan *End-to-End Delay* sebesar 49,36%. Rata-rata peningkatan *throughput* adalah 8,34%.

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, bisa melakukan uji coba yang lebih banyak.
2. Kedepannya bisa menambahkan aspek lain yang bisa dijadikan acuan untuk diimplementasikan dalam pemilihan rute.

DAFTAR PUSTAKA

[1] "VANET - Vehicle Ad hoc Network," [Online]. Available: http://comp.ist.utl.pt/~rmr/WSN/CaseStudies2007-no/WSN_Transportation/.

[2] J. Harri, F. Filali dan C. Bonnet, "Mobility Models for Vehicular Ad Hoc Network: A Survey and Taxonomy," IEEE, Florida, 2009.

[3] L. U. Khan, S. A. Mahmud, M. H. Zafar, G. M. Khan dan H. S. Al-Raweshidy, "M-AODV: Modified Ad Hoc On-demand distance vector routing scheme," *2014 9th International Symposium on Communication Systems, Networks and Digital Signal Processing, CSNDSP 2014,* pp. 18-22, 2014.

[4] R. Brendha dan V. S. J. Prakash, "A Survey on Routing Protocols for Vehicular Ad hoc Networks," IEEE, Coimbatore, 2017.

[5] E. B.-R. S. D. C. Perkin, "Ad hoc On-Demand Distance Vector (AODV) Routing," *IETF Internet Draft,* 2002.

[6] R. F. Sari dan A. Syarif, "Analisis Kinerja Protokol Routing Ad Hoc On-Demand Distance Vector (AODV) pada Jaringan Ad Hoc," p. 22, October 2010.

[7] Anon, "http://www.isi.edu/nsnam/ns," 2005.

[8] P. Meeneghan dan D. Delaney, "An Introduction to NS Nam and OTcl scripting," April 2004.

[9] "OpenStreetMap," [Online]. Available: https://www.openstreetmap.org/. [Diakses 15 November 2017].

[10] "JOSM," [Online]. Available: https://josm.openstreetmap.de/. [Diakses 15 November 2017].

[11] D. Krajzewics, J. Erdmann, M. Behrisch dan L. Bieker, "Recent Development and Application of SUMO," *International Journal On Advances in Systems and Measurements,* p. 128, December 2012.

[12] "AWK," [Online]. Available: http://tldp.org/LDP/abs/html/awk.html. [Diakses 10 01 2017].

[13] M. Iqbal, M. Shafiq, H. Attaullah, J.-G. Choi, K. Akram dan X. Wang, "Design and Analysis of a Novel Hybrid Wireless Mesh Network Routing Protocol," p. 22, January 2014.

[14] Y. Feng , B. Zhang, S. Chai, L. Cui dan Q. Li, "An Optimized AODV Protocol based on Clustering for WSNs," *6th International Conference on Computer Science and Network Technology (ICCSNT),* 2017.

[15] R. G. Engoulou, M. Bellaiche, S. Pierre dan A. Quintero, "VANET Security Surveys," *Computer Communication,* vol. 44, p. 2, 2014.

[16] R. A. M. I. F. Nutrihadi, "Studi Kinerja VANET Scenario Generators : SUMO dan VanetMobisim untuk Implementasi Routing Protocol AODV menggunakan Network Simulator 2 (NS-2)," vol. 5, no. 1, pp. 1-6, 2016.

LAMPIRAN

A.1 Implementasi File aodv_packet.h

```
struct hdr_aodv_request {
      double     rq_throughput;
}

struct hdr_aodv_reply {
      double     rp_throughput;
}
```

A.2 Implementasi File aodv_rtable.h

```
class aodv_rt_entry {
      double     rt_throughput;
}
```

A.3 Kode Drop RREQ

```
if (rq->rq_hop_count == rt0->rt_hops && rq-
>rq_throughput < rt0->rt_throughput){
    Packet::free(p);
    return;
  }
```

A.3 Kode *Update* RREP *Throughput*

```
if (ih->daddr() == index && rt->rt_seqno ==
rp->rp_dst_seqno && rp->rp_throughput > rt-
>rt_throughput){

    rt->rt_throughput = rp>rp_throughput;
  }
```

A.4 Kode *Drop* RREP

```
if (rp->rp_hop_count > rt->rt_hops){
    Packet::free(p);
    return;
  }
```

A.5 Kode *Update* RREP *Throughput*

```
double packet_size= rp->size();
double throughput=
calc_throughput(packet_size, rp-
>rp_timestamp, CURRENT_TIME);
    if (throughput > rp->rp_throughput){
      rp->rp_throughput=  throughput;
    }

      assert (rt0->rt_flags == RTF_UP);
      rp->rp_hop_count += 1;
      rp->rp_src = index;
      forward(rt0, p, NO_DELAY);

      rt->pc_insert(rt0->rt_nexthop);
    }
    else {
      drop(p, DROP_RTR_NO_ROUTE);
    }
  }
```

A.6 Kode Fungsi Update Routing Table

```
void AODV::rt_update(aodv_rt_entry *rt,
u_int32_t seqnum, u_int16_t metric,
nsaddr_t nexthop, double throughput, double
expire_time) {

  rt->rt_seqno = seqnum;
  rt->rt_hops = metric;
  rt->rt_flags = RTF_UP;
  rt->rt_nexthop = nexthop;
  rt->rt_throughput = throughput;
  rt->rt_expire = expire_time;
}
```

A.7 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 opt(x)      1500;
set opt(y)      1500;
set val(ifqlen) 1000;
set val(nn)     200;
set val(seed)   1.0;
set val(adhocRouting)   AODV;
set val(stop)   200;
set val(cp)     "cbr200.tcl";
set val(sc)     "scenario.tcl";

set ns_         [new Simulator]

# setup topography object

set topo   [new Topography]

# create trace object for ns and nam

set tracefd     [open scenario1.tr w]
set namtrace    [open scenario1.nam w]

$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace
$opt(x) $opt(y)
```

```
# Create God
set god_ [create-god $val(nn)]

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

# 802.11p default parameters
Phy/WirelessPhy set  RXThresh_ 5.57189e-
11 ; #400m
Phy/WirelessPhy set  CSThresh_ 5.57189e-
11 ; #400m

#  Create the specified number of nodes
[$val(nn)] and "attach" them
#  to the channel.
for {set i 0} {$i < $val(nn)} {incr i} {
           set node_($i) [$ns_ node]
           $node_($i) random-motion 0 ;#
disable random motion
}
```

```
# Define node movement model
puts "Loading connection pattern..."
source $val(cp)

# Define traffic model
puts "Loading scenario file..."
source $val(sc)

# Define node initial position in nam

for {set i 0} {$i < $val(nn)} {incr i} {

    # 20 defines the node size in nam,
must adjust it according to your scenario
    # The function must be called after
mobility model is defined

    $ns_ initial_node_pos $node_($i) 20
}

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

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

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

puts "Starting Simulation..."
$ns_ run
```

A.8 Kode Konfigurasi *Traffic*

```
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(198) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(199) $null_(0)
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_ 1000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 0 "$cbr_(0) start"
$ns_ at 200.0000000000000000 "$cbr_(0) stop"
```

A.9 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.10 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;
        }
    }
```

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

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

FATURRAHMAN MA'RUF, lahir di Klaten, 30 Mei 1997. Penulis adalah anak kedua dari dua bersaudara. Penulis menempuh pendidikan sekolah dasar di SD Muhammadiyah 1 Wedi lalu melanjutkan pendidikan sekolah menengah pertama di SMP Negeri 1 Wedi dan penulis menempuh pendidikan menengah atas di SMA Negeri 1 Klaten. Selanjutnya penulis melanjutkan pendidikan sarjana di Departemen Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember Surabaya. Selama kuliah, penulis aktif dalam berbagai organisasi tingkat jurusan.

Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Arsitektur dan Jaringan Komputer (AJK). Sebagai mahasiswa, penulis berperan aktif dalam beberapa organisasi kampus seperti Staf Kesejahteraan Mahasiswa Himpunan Mahasiswa Teknik-Computer (HMTC) 2016, Staf Ahli Staf Kesejahteraan Mahasiswa Himpunan Mahasiswa Teknik-Computer (HMTC) 2017. Selain itu, penulis juga menjadi Staf Perkap SCHEMATICS 2016 dan Staf Ahli Perkap SCHEMATICS 2017, Staf Hubungan Masyarakat SCHEMATICS 2017, dan Staf Ahli Logistic FTIF-Festival (FF) 2017. Penulis pernah melakukan kerja praktik di Krakatau Information Technology pada Juni – Agustus 2018 dan membuat aplikasi SAP ABAP untuk pembuatan laporan. Penulis dapat dihubungi melalui nomor *handphone*: 082214612485 atau *email*: faturrahmanmakruf@gmail.com .