TUGAS AKHIR – IF184802

MODIFIKASI PEMILIHAN RUTE *AD-HOC ON DEMAND DISTANCE VECTOR* (AODV) BERDASARKAN FAKTOR *DELAY* DI LINGKUNGAN VANETS

MUHAMMAD ADIB ARINANDA
NRP 05111540000111

Dosen Pembimbing I
Ir. F.X. Arunanto, M.Sc.

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

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR – IF184802

MODIFIKASI PEMILIHAN RUTE *AD-HOC ON DEMAND DISTANCE VECTOR* (AODV) BERDASARKAN FAKTOR *DELAY* DI LINGKUNGAN VANETS

MUHAMMAD ADIB ARINANDA
NRP 05111540000111

Dosen Pembimbing I
Ir. F.X. Arunanto, M.Sc.

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

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – IF184802

MODIFICATION OF ROUTE SELECTION IN AD-HOC ON DEMAND DISTANCE VECTOR (AODV) BASED ON DELAY FACTOR IN VANETS

MUHAMMAD ADIB ARINANDA
NRP 05111540000111

First Advisor
Ir. F.X. Arunanto, M.Sc.

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

Department of Informatics
Faculty of Information Technology and Communication
Sepuluh Nopember Institute of Technology
Surabaya 2019

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

MODIFIKASI PEMILIHAN RUTE *AD-HOC ON DEMAND DISTANCE VECTOR* (AODV) BERDASARKAN FAKTOR *DELAY* 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 Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
MUHAMMAD ADIB ARINANDA
NRP: 05111540000111

Disetujui oleh Pembimbing Tugas Akhir:

1. Ir. F.X. Arunanto, M.Sc. .....
(NIP. 195701011983031004) (Pembimbing 1)

2. Dr.Eng. Radityo Anggoro, S.Kom., M.Sc. .....
(NIP. 198410162008121002) (Pembimbing 2)

SURABAYA
JUNI, 2019

(Halaman ini sengaja dikosongkan)

MODIFIKASI PEMILIHAN RUTE *AD-HOC ON DEMAND DISTANCE VECTOR* (AODV) BERDASARKAN FAKTOR *DELAY* DI LINGKUNGAN VANETS

Nama Mahasiswa : MUHAMMAD ADIB ARINANDA
NRP : 05111540000111
Departemen : Informatika FTIK-ITS
Dosen Pembimbing 1 : Ir. F.X. Arunanto, M.Sc.
Dosen Pembimbing 2 : Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.

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 delay yang terjadi di rute tersebut. Jika menggunakan metode pemilihan rute berdasarkan faktor delay, maka rute yang dipilih bukanlah rute dengan jarak terpendek, melainkan rute dengan total delay terkecil.

Pada Tugas Akhir ini, diusulkan suatu algoritma route discovery yang bernama Delay-based AODV (AODV-D) untuk

mendapatkan rute terbaik berdasarkan delay di protokol AODV. Pada algoritma ini, destination node akan mengirim route reply ke node yang memiliki delay terkecil, dan ketika ada route request yang sampai ke destination node dengan rute yang lebih pendek, maka destination node tidak memasukkan rute tersebut ke routing table.

Hal ini dilakukan agar dapat meningkatkan kinerja protokol AODV untuk mencari rute dengan total delay terkecil dengan cara memodifikasi beberapa bagian dari mekanisme pengiriman paket RREP. Dari hasil uji coba, AODV yang dimodifikasi pada skenario grid berhasil meningkatkan nilai rata-rata Packet Delivery Ratio (PDR) hingga 3,15%, menurunkan Delivery Delay hingga 25,90%, dan penurunan nilai rata-rata Routing Overhead (RO) hingga 8,51%. Sedangkan pada skenario real berhasil meningkatkan nilai rata-rata Packet Delivery Ratio (PDR) hingga 6,13%, menurunkan Delivery Delay hingga 26,38%, dan penurunan nilai rata-rata Routing Overhead (RO) hingga 13,03%.

Kata kunci: VANETs, AODV, Delay, Route Discovery

MODIFICATION OF ROUTE SELECTION IN AD-HOC ON DEMAND DISTANCE VECTOR (AODV) BASED ON DELAY FACTOR IN VANETS

Student's Name	**:**	**MUHAMMAD ADIB ARINANDA**
Student's ID	**:**	**05111540000111**
Department	**:**	**Informatics – FTIK ITS**
First Advisor	**:**	**Ir. F.X. Arunanto, M.Sc.**
Second Advisor	**:**	**Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**

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 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 2 phase which are route discovery and route 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 normal AODV performance, it will take a lot of time to do Re-Broadcast to all nodes. If you use the Clustering method by selecting Cluster Head based on the degree of neighboring nodes, then when doing Rebroadcast it is not sent to all nodes, but only through Cluster Head nodes and neighboring nodes from the previous route. After that the process of building the shortest route will be built.

In this Final Project proposed an route discovery algorithm called Delay-based AODV (AODV-D) to obtain the best route based on delay in the AODV protocol. In this algorithm, destination node

will send route reply to node with smallest delay, and if there is a route request received at destination node with shorter route path, the destination node will not insert the route to the routing table.

This is done 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 packages. From the test results, modified AODV in the grid scenario has increased the average value of the Packet Delivery Ratio (PDR) by 3,15%, decreased Delivery Delay value by 19,25%, and the value of Routing Overhead (RO) has decreased by 8,51%. While in the real scenario the average value of the Packet Delivery Ratio (PDR) has increased by 6,13% , decreased Delivery Delay value by 26,38%, and the value of Routing Overhead (RO) has decreased by 13,03%.

Keyword: VANETs, AODV, Delay, Route Discovery

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 *Ad-hoc On Demand Distance Vector* (AODV) Berdasarkan Faktor *Delay* 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 Yusmiril Rodhyan Noor dan Ibu Fadma Noor Diana selaku kedua orangtua penulis atas segala dukungan berupa motivasi, doa, moral, dan material sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Saudara penulis dan keluarga kecilnya, Muhammad Alfi Arinanda, Melisa Rahmadini, dan Lanika Jauza Almahira, atas segala dukungan yang telah diberikan sehingga penulis tetap semangat dalam mengerjakan Tugas Akhir ini.
4. Bapak Ir. F.X. Arunanto, M.Sc. dan Bapak Dr. Eng. Radityo Anggoro, S.Kom., M.Sc. selaku dosen pembimbing penulis atas nasihat, arahan dan bantuannya sehingga penulis dapat menyeleksaikan Tugas Akhir ini.
5. Sahabat penulis, Annisa Humaira, yang tengah menempuh pendidikan sarjana di Moscow, Russia, yang selalu menghibur, menyemangati, menemani, dan mendukung penulis selama masa perkuliahan dan selama penulisan Tugas Akhir ini, dikala susah maupun senang.

6. Teman-teman dari THEREDSTC (Hilman, Rage, Botak, Sekbay, Romi, Otong, Ichsan, Pius, Purnomo, Teja, Isye, Haura dan Hanif) yang selalu menghibur, menyemangati, dan mendukung penulis di tiap Game Week dan European Nights.
7. Rakhma Rufaida Hanum yang telah menemani, menyemangati, dan banyak membantu penulis selama masa perkuliahan khususnya di awal-awal masa perkuliahan.
8. Teman-teman dari GAES x KONS (Zahri, Yuga, Ivan, Abyan, Unggul, Fajar, Pradipta, GD Wahyu, Redo, Akram, Pandito, Rafi, Pandu, Yoza, Narendra) yang selalu ada saat dibutuhkan, memberikan hiburan kepada penulis, dan juga menjadi keluarga baru penulis saat berkuliah di Departemen Informatika ITS.
9. Teman-teman multichat Keresahan Tugas Akhir (Naufal, Huda, Ilham Penyok, dan Redo) yang selalu menyemangati penulis agar bisa menyelesaikan Tugas Akhir ini tepat waktu.
10. Teman-teman dari Depan KBJ x Warkop NF (Zahri, Tegar, Djohan, Naufal, Ichsan, Aqil, Arya, Illham, Yasin, Faiq, dan Adam) yang selalu ada ketika penulis sedang jenuh dengan perkuliahan, mengisi hari-hari penulis dengan obrolan-obrolan kecil namun berbobot di depan KBJ maupun warkop.
11. Teman-teman dari keluarga besar Laboratorium AJK (Ilham Penyok, Fuad, Didin, Awan, Satriya, Daus, Nahda, dan Hana) dan Laboratorium MI (Huda, Unggul, Ivan, Yudhis, Rezky, Yola, Ajeng, Nafi, dan Salma) yang telah menemani, memotivasi, memberikan doa, memberikan hiburan di kala penulis sedang jenuh saat pengerjaan Tugas Akhir ini, serta menyediakan fasilitas selama penulis mengerjakan Tugas Akhir ini.
12. Teman-teman Schematics 2016 dan Schematics 2017 yang sudah memberikan kesibukan lebih untuk penulis semasa kuliah di Informatika ITS.

13. 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, Mei 2019

MUHAMMAD ADIB ARINANDA

(Halaman ini sengaja dikosongkan)

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

Kemajuan terbaru dalam perangkat keras, perangkat lunak dan teknologi komunikasi memungkinkan adanya desain dan implementasi berbagai macam jaringan yang diimplementasikan di berbagai jenis lingkungan. Salah satu jaringan yang bertumbuh pesat dan mendapat perhatian cukup besar adalah *Vehicular Ad-Hoc Networks*(VANETs). *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. Topologi yang dinamis ini merupakan perbedaan mendasar antara VANET dan MANET. Tingkat mobilitas yang tinggi pada setiap node juga dapat menyebabkan adanya rute terputus dikarenakan node keluar dari jangkauan sinyal transmisi [1].

Routing protocol dalam VANET dibedakan menjadi dua model, yaitu *proactive* dan *reactive routing*. *Proactive routing* 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. Sedangkan *reactive routing* adalah merupakan mekanisme *routing* yang membentuk *routing table* jika ada permintaan pengiriman data atau terjadinya perubahan rute dalam setiap jaringan. 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 pengimplementasian protokol AODV adalah diimplementasikannya protokol tersebut kedalam jaringan sensor nirkabel melalui simulasi skenario VANETs. *Adhoc on-Demand Distance Vector Protocol* merupakan salah satu routing protocol reaktif yang dikenal luas, sehingga sudah banyak penelitian untuk yang memodifikasi untuk meningkatkan kinerjanya. Pencarian rute menjadi suatu mekanisme yang penting untuk mendukung mobilitas di VANET. Pemilihan rute yang stabil saat proses pencarian rute sangat diperlukan untuk memperpanjang waktu penggunaan rute. Salah satu cara dapat dilakukan adalah dengan memilih rute yang memiliki kemungkinan kecil terputus [2]. Pada kinerja AODV biasa, rute yang dipilih adalah rute dengan jumlah *hop* paling sedikit. Rute dengan jumlah *hop* paling sedikit tersebut belum tentu merupakan rute dengan *delay* terkecil. Dengan menggunakan metode *route discovery* berdasarkan *delay*, diharapkan rute dengan *delay* terkecil bisa didapatkan untuk menunjang pengiriman paket data dari *source node* ke *destination node*[3] .

Pada Tugas Akhir ini, diusulkan suatu algoritma *route discovery* yang bernama *Delay-based* AODV (AODV-D) untuk mendapatkan rute terbaik berdasarkan *delay* di protokol AODV. Pada algoritma ini, *destination node* akan mengirim *route reply* ke *node* yang memiliki *delay* terkecil, dan ketika ada *route request* yang sampai ke *destination node*

dengan rute yang lebih pendek, maka *destination node* tidak memasukkan rute tersebut ke *routing table*.

1.2 Rumusan Masalah

Tugas Akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana melakukan pemilihan rute berdasarkan *Delay* di lingkungan VANETs?
2. Bagaimana peranan AODV-D terhadap performa protokol AODV secara keseluruhan diukur berdasarkan *Packet Delivery Ratio, End-to-end Delay*, dan *Routing Overhead*?

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. *Routing protocol* yang diujicobakan yaitu AODV.
3. Simulasi pengujian jaringan menggunakan *Network Simulator 2* (NS-2).
4. Pembuatan skenario uji coba menggunakan *Simulation of Urban Mobility* (SUMO).

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Melakukan pemilihan rute berdasarkan *Delay* di lingkungan VANETs.
2. Menganalisa performa AODV yang telah dimodifikasi berdasarkan matriks *Packet Delivery Ratio* (PDR), *End-to-end Delay*, dan *Routing Overhead*.

1.5 Manfaat

Manfaat yang diperoleh dari pengerjaan Tugas Akhir ini adalah dapat memberikan informasi tentang dampak dari modifikasi dengan melalukan pemilihan *rute* berdasarkan *delay* 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), *Routing Overhead* (RO), dan *End-to-End Delay* paket dari *node* ke *node* lainnya. 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, dan Routing Overhead* 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, 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.

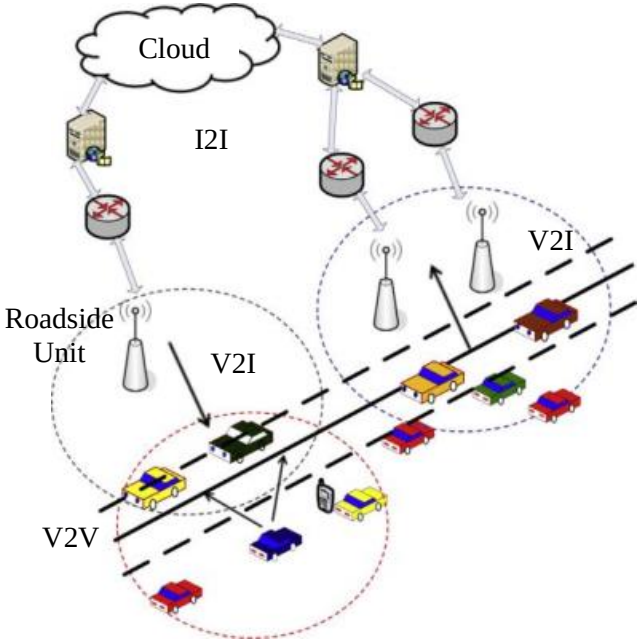
(Halaman ini sengaja dikosongkan)

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 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 [4]

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

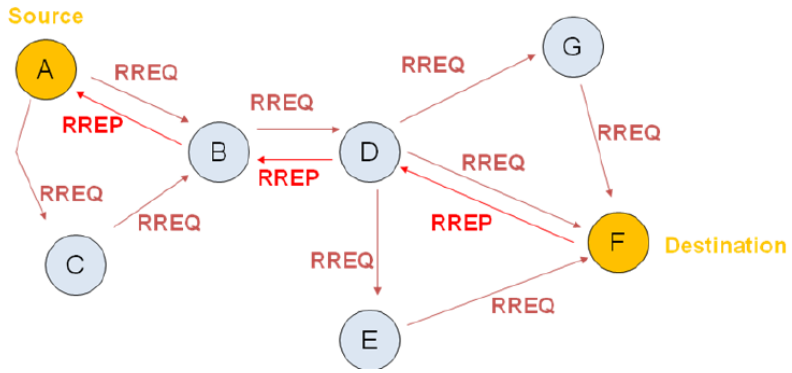
2.2 *Ad-hoc On demand Distance Vector* (AODV)

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.

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 *route 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*[5]. Ilustrasi pencarian rute oleh AODV dapat dilihat pada Gambar 2.2.



Gambar 2.2 Ilustrasi pencarian rute *routing protocol* AODV *[12]*

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). 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*[6].

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, dibangun sepenuhnya oleh sukarelawan dengan melakukan survei menggunakan GPS, mendigitalisasi citra satelit dan mengumpulkan 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[7].

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 *real* VANETs.

2.5 *Java OpenStreetMap Editor* (JOSM)

Java OpenStreetMap Editor (JOSM) adalah aplikasi untuk menyunting data yang didapatkan dari OpenStreetMap[8].

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[9].

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[10]. 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.

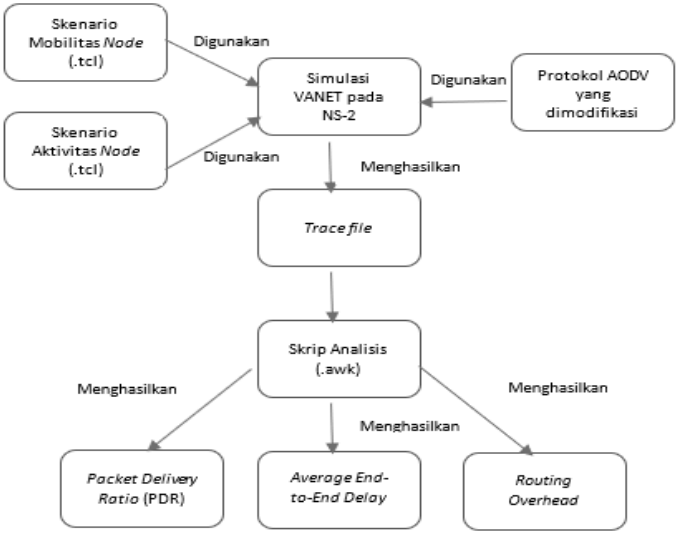
Pada Tugas Akhir ini, AWK digunakan untuk membuat *script* menghitung *Packet Delivery Ratio (*PDR*), End-to-end Delay*, *Routing Overhead* (RO), *Forwarded Route Request* (RREQ F), dan *Forwarded Route Reply* (RREP F) dari *trace file* NS2.

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

Modifikasi diawali dengan mendapatkan *delay* di tiap *node* perantara (*one-hop node*). *Delay* didapatkan dengan mengurangi waktu saat paket RREQ diterima di suatu *node* dengan waktu saat paket RREQ diterima di *node* sebelumnya, atau dengan waktu saat paket RREQ pertama kali dikirim apabila *node* sebelumnya merupakan *source node*. Untuk setiap *hop*, *delay* tersebut ditambahkan terus-menerus agar didapatkan nilai *total delay* di *destination node*. Ketika paket RREQ diterima di *destination node*, *destination node* akan meng*update routing table*, kemudian *destination node* akan mengirim paket RREP ke *node* dengan *total delay* terkecil. Namun, agar *destination node* tidak mengupdate *routing table* ketika menemukan rute dengan *hop count* lebih kecil dan mengirim paket RREP ke *node* tersebut, maka fungsi untuk meng*update routing table* ketika ditemukan rute dengan *hop count* lebih kecil perlu dihapus.

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), *Routing Overhead* (RO), *Forwarded Route Request* (RREQ F), dan *Forwarded Route Reply* (RREP F). 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

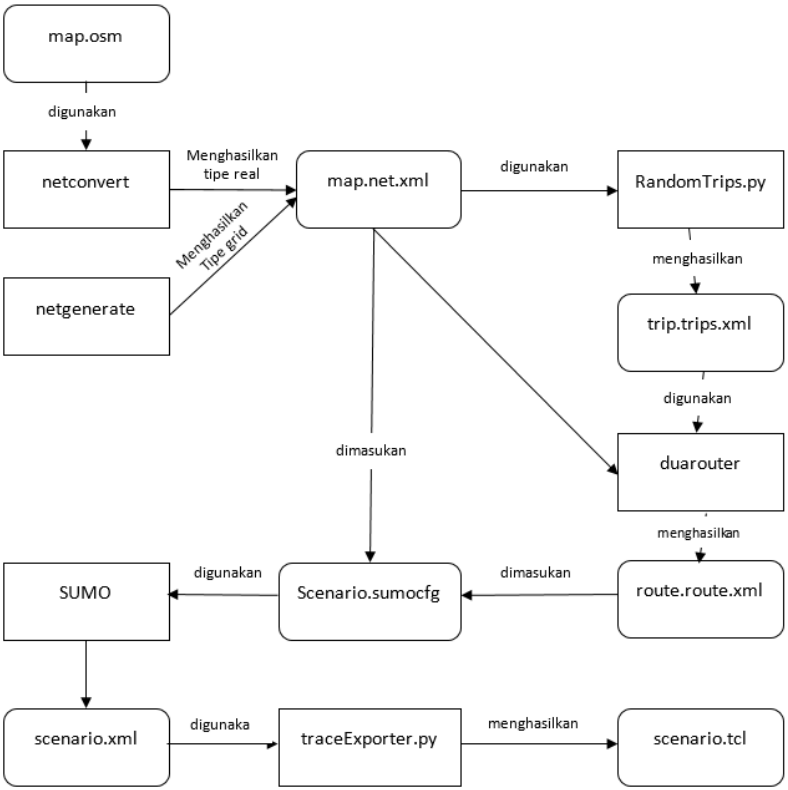
No.	Istilah	Penjelasan
1	AODV	Singkatan dari *Ad hoc On-demand Distance Vector*. Protokol yang digunakan pada Tugas Akhir ini.

No.	Istilah	Penjelasan
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.
4	RO	*Routing Overhead*. Jumlah *control packet* yang terkirim
5	RREQ	*Route Request*. Paket *request* pada AODV yang dikirim untuk mendapatkan rute.
6	RREQ F	*Forwarded Route Request*. Jumlah forwarding node yang bertugas untuk mengirim ulang (rebroadcast) RREQ.
7	RREP	*Route Reply*. Paket *reply* pada AODV yang dikirim ke *node* sumber melalui rute yang sudah terbuat.
8	RREP F	*Forwarded Route Reply*. Jumlah forwarding node yang bertugas untuk mengirim ulang (rebroadcast) RREP.

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

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 berekstensi .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 *hop count* terkecil atau rute dengan jarak terpendek. Pada AODV yang dimodifikasi ini, rute dengan jarak terpendek akan diabaikan apabila ditemukan rute dengan *total delay* terkecil.

Delay di tiap rute didapatkan dengan cara menjumlahkan *delay* yang terjadi di tiap *node*. Variabel *current delay* menyimpan *delay* yang terjadi di tiap *node*, dan *current delay* di suatu *node* akan dijumlahkan dengan *current delay* di *node* setelahnya untuk kemudian dimasukkan ke variabel *total delay*, hingga paket RREQ sampai di *destination node*, di tiap rutenya. Di *destination node*, *routing table* akan diupdate dan paket RREP akan dikirim melalui rute dengan *total delay* paling kecil.

Selanjutnya, pada protokol AODV, setiap ada paket RREQ yang sampai di *destination node* dan memiliki *hop count* lebih kecil, maka *destination node* akan mengupdate *routing table* yang kemudian *destination node* akan mengirim paket RREP melalui rute tersebut, dimana rute tersebut belum tentu merupakan rute dengan *total delay* terkecil. Untuk menghindari hal tersebut, fungsi *update routing table* ketika ada rute baru dengan jumlah *hop count* lebih kecil sampai di *destination node* perlu dihapus, agar *destination node* tidak mengupdate *routing table*nya.

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 penjumlahan nilai *delay* dan penghapusan fungsi *update routing table* ketika paket RREQ dengan rute yang memiliki *hop count* lebih kecil sampai di *destination node* pada *file* aodv.cc, dan penambahan variabel *current delay* dan *total delay* pada *file* aodv.h. Pada saat simulasi NS-2 dijalankan, maka *routing protocol* AODV akan mengirim paket data melalui *rute* dengan nilai *total delay* terkecil.

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 *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 3.3.

$$RO = \sum_{m=1}^{sentnum} packet\ sent \quad (3.3)$$

3.5.4 *Forwarded Route Request* (RREQ F)

Forwarded Route Request adalah jumlah paket kontrol *route request* yang di*forward* per data paket ke *node* tujuan selama simulasi terjadi. RREQ F didapatkan dengan menjumlahkan semua paket *control routing* yang ditransmisikan khususnya *route request* bagian *forwarding* (RREQ F). Perhitungan RREQ F dapat dilihat dengan persamaan 3.4

$$Forwarded\ Route\ Request = \sum_{n=1}^{rreqsent} packet\ sent \quad (3.4)$$

3.5.5 *Forwarded Route Reply* (RREP F)

Forwarded Route Reply adalah jumlah paket kontrol *route reply* yang di*forward* per data paket ke *node* sumber selama simulasi terjadi. RREP F didapatkan dengan menjumlahkan semua paket *control routing* yang ditransmisikan khususnya *route reply* bagian *forwarding* (RREP F). Perhitungan RREP F dapat dilihat dengan persamaan 3.5

$$Forwarded\ Route\ Reply = \sum_{n=1}^{rrepsent} packet\ sent \quad (3.5)$$

(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.

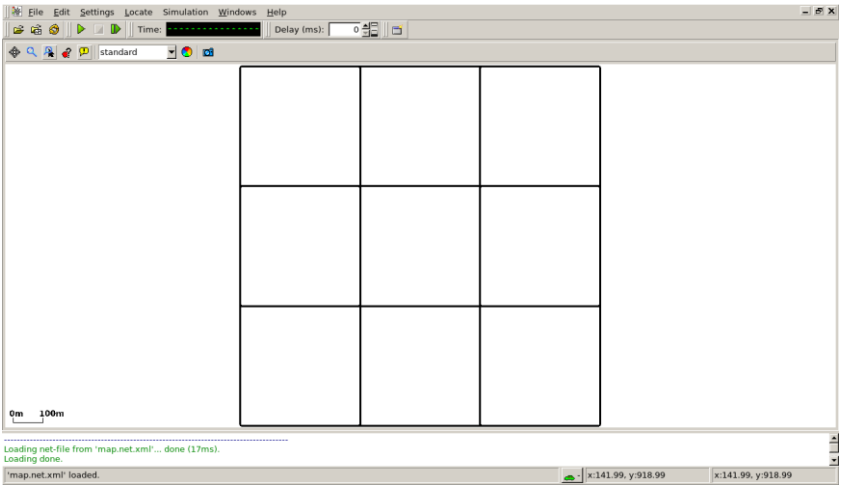
4.1.1 Skenario *Grid*

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, maka 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* 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 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.de/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 menggunakan NS-2. *Tools* yang digunakan untuk melakukan

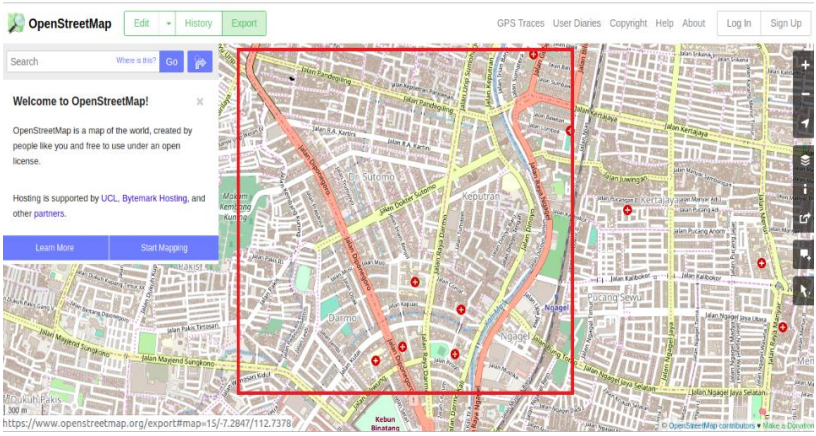
konversi ini adalah traceExporter. Perintah untuk menggunakan traceExporter dapat dilihat pada Gambar 4.7.

```
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. 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.



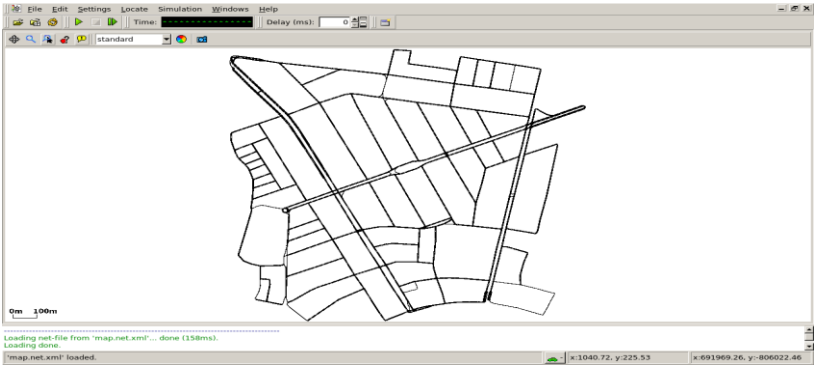
Gambar 4.8 Ekspor Peta dari OpenStreetMap

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 Rute dengan *Total Delay* Terkecil

Pada Tugas Akhir ini dilakukan modifikasi pada routing protocol AODV agar dapat mengurangi nilai End-to-End Delay, yaitu waktu yang dibutuhkan oleh suatu paket untuk diterima destination node dari source node. Hal tersebut dilakukan dengan cara memilih rute dengan nilai total delay terkecil, dan mengabaikan rute dengan hop count terkecil jika total delay yang dimiliki rute tersebut tidak lebih kecil dari rute yang didapatkan sebelumnya, sehingga dapat dilihat peningkatan performa pada routing protocol AODV yang telah dimodifikasi.

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

- Implementasi Penghitungan *Delay* di Tiap *Node*
- Implementasi Penghitungan *Total Delay* Tiap Rute
- Implementasi Pemilihan Rute dengan *Total Delay* Terkecil

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 *delay* di tiap *node*, *total delay* di tiap rute, dan pemilihan rute dengan *total delay* terkecil, 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 untuk mengurangi nilai *End-to-End Delay*.

4.3 Implementasi Fungsi Update Routing Table

Agar rute yang dipilih adalah rute dengan delay terkecil, maka perlu dilakukan pengubahan di fungsi update routing table, dimana pada AODV asli routing table akan diupdate ketika ditemukan rute dengan hop count lebih kecil dari rute yang sudah ada di routing table.

Pengubahan ini dilakukan pada fungsi `recvRequest()` yang terletak pada kode sumber aodv.cc yang terletak pada ns2.35/aodv. Pada bagian percabangan saat akan masuk fungsi update routing table, kondisi yang membandingkan jumlah hop count dari rute yang sudah ada di routing table dengan jumlah hop count dari rute yang baru masuk dihapus. Potongan kode untuk fungsi update routing table dapat dilihat pada Gambar 4.11

```
if ( (rq->rq_src_seqno > rt0->rt_seqno ) ||
(rq->rq_src_seqno == rt0->rt_seqno)  ) {
rt_update(rt0, rq->rq_src_seqno, rq-
>rq_hop_count, ih->saddr(),
             max(rt0->rt_expire,
(CURRENT_TIME + REV_ROUTE_LIFE)) );
     if (rt0->rt_req_timeout > 0.0) {
       rt0->rt_req_cnt = 0;
       rt0->rt_req_timeout = 0.0;
       rt0->rt_req_last_ttl = rq-
>rq_hop_count;
       rt0->rt_expire = CURRENT_TIME +
ACTIVE_ROUTE_TIMEOUT;
     }

     assert (rt0->rt_flags == RTF_UP);
     Packet *buffered_pkt;
     while ((buffered_pkt =
rqueue.deque(rt0->rt_dst))) {
```

```
        if (rt0 && (rt0->rt_flags == RTF_UP))
{
          assert(rt0->rt_hops != INFINITY2);
          forward(rt0, buffered_pkt,
NO_DELAY);
        }
      }
    }
```

Gambar 4.11 Potongan Kode Fungsi Update Routing Table

4.4 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

```
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 4.12 Implementasi Simulasi NS-2

dilakukan selama simulasi. Potongan konfigurasi lingkungan simulasi dapat dilihat pada Gambar 4.12.

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 4.13.

```
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 4.13 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.4 Kode Konfigurasi *Traffic*

4.5 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, dan RO, *Forwarded Route Reques*t (RREQ F), dan *Forwarded Route Reply* (RREP F).

4.5.1 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.5 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 4.14.

```
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 4.14 Pseudocode untuk Perhitungan PDR

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

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

Skrip awk untuk menghitung E2E dapat dilihat pada lampiran A.6 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 4.15.

```
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 4.15 Pseudocode untuk Perhitungan E2E

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

4.5.3 Implementasi *Routing Overhead* (RO)

Seperti yang telah dijelaskan sebelumnya, *routing overhead* merupakan jumlah dari paket kontrol *routing* baik itu RREQ, RREP, maupun RERR. Dengan begitu, untuk mendapatkan RO yang perlu dilakukan adalah menjumlahkan tiap paket dengan *filter event sent* pada kolom pertama dan *event layer* RTR pada kolom ke-4.

```
ro = 0
for i = 1 to the number of rows
     if in a row contains “s” and RTR then
        ro++
     end if
```

Gambar 4.16 Pseudocode Perhitungan Routing Overhead

Perhitungan RO telah dijelaskan pada persamaan 3.3. Skrip AWK untuk menghitung RO dapat dilihat pada lampiran A.7 Kode Skrip AWK *Routing Overhead*. *Pseudocode* untuk menghitung RO dapat dilihat pada Gambar 4.16.

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

4.5.4 Implementasi *Forwarded Route Request*

Forwarded Route Request (RREQ F) adalah jumlah paket *control route request* yang diteruskan per-data paket ke *node* tujuan selama simulasi terjadi. Dan dapat dikatakan bahwa RREQ F adalah jumlah forwarding node yang bertugas untuk mengirim ulang (*rebroadcast*) RREQ. Dengan begitu, untuk mendapatkan RREQ F yang perlu dilakukan adalah menjumlahkan tiap paket dengan *filter event sent* pada kolom pertama, *event layer* RTR pada kolom ke-4, dan *event layer route request* pada kolom ke-25. Penyaringan juga dilakukan pada kolom ke-3 yang menunjukan *node id*. Selama *node* bukan *node* sumber, maka akan terus dilakukan penjumlahan baris yang terdapat pada *file* dengan ekstensi .tr. Perhitungan RREQ F telah dijelaskan pada persamaan 3.4. Skrip AWK untuk menghitung RREQ F dapat dilihat pada lampiran

A.8 Kode *Skrip AWK Forwarded Route Request*. *Pseudocode* untuk menghitung RREQ F dapat dilihat pada Gambar 4.17.

```
rreqf = 0
for i = 1 to the number of rows
     if packet is AODV and RREQ and not source
node then
        rreqf++
     end if
```

Gambar 4.17 Pseudocode Perhitungan RREQ F

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

4.5.5 Implementasi *Forwarded Route Reply*

Forwarded Route Reply (RREP F) adalah jumlah paket *control route reply* yang diteruskan per-data paket ke *node* tujuan selama simulasi terjadi. Dan dapat dikatakan bahwa RREP F adalah jumlah forwarding node yang bertugas untuk mengirim ulang (rebroadcast) RREP. Dengan begitu, untuk mendapatkan RREP F yang perlu dilakukan adalah menjumlahkan tiap paket dengan *filter event sent* pada kolom pertama, *event layer* RTR pada kolom ke-4, dan *event layer route reply* pada kolom ke-25. Penyaringan juga dilakukan pada kolom ke-3 yang menunjukan *node id*. Selama *node* bukan *node* destinasi, maka akan terus dilakukan penjumlahan baris yang terdapat pada *file* dengan ekstensi .tr. Perhitungan RREP F telah dijelaskan pada persamaan 3.4. Skrip AWK untuk menghitung RREP F dapat dilihat pada lampiran

A.8 Kode Skrip *AWK Forwarded Route Request*. *Pseudocode* untuk menghitung RREP F dapat dilihat pada Gambar 4.18.

```
rrepf = 0
for i = 1 to the number of rows
     if packet is AODV and RREP and not
destination node then
        rrepf++
     end if
```

Gambar 4.18 Pseudocode Perhitungan RREP F

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

(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™ i5-7400 CPU @ 3.00GHz
Sistem Operasi	Ubuntu 18.04.2 LTS
Linux Kernel	Linux kernel 4.18
Memori	8.0 GB
Penyimpanan	208,9 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 dilihat 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, RO, RREQ F, dan RREP F menggunakan kode yang terdapat pada lampiran A.5 Kode Skrip AWK *Packet Delivery Ratio*, A.6 Kode

Skrip AWK Rata-Rata *End-to-End Delay*, A.7 Kode Skrip AWK *Routing Overhead*, A.10 Kode Skrip Awk *Forwarded Route Request*, dan A.11 Kode Skrip *AWK Forwarded Route Reply*.

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	650 m x 650 m, 1500 m x 1500 m
5	Jumlah *Node*	50, 100, 150, 200
6	Radius transmisi	400m
7	Kecepatan maksimum	20 m/s

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 dilihat sebagai berikut:

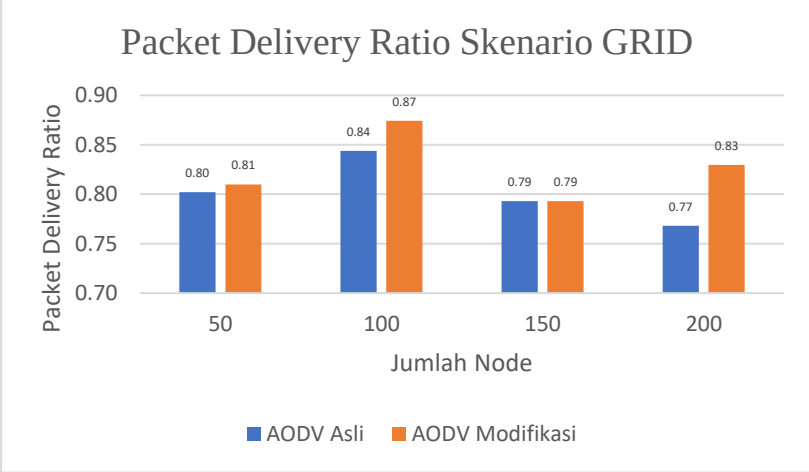
5.2.1 Hasil Uji Coba Skenario *Grid*

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

Pengambilan data uji PDR, E2E, RO, RREQ F dan RREP F pada skenario *grid* dilakukan sebanyak 10 kali dengan skenario mobilitas *random* pada peta *grid* dengan luas area 650 m x 650 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, Tabel 5.6, dan Tabel 5.7.

Tabel 5.3 Hasil Rata - Rata Packet Delivery Ratio Skenario *Grid*

Jumlah *Node*	AODV Asli	AODV Modifikasi	Perbedaan
50	0,80	0,81	+ 0,01
100	0,84	0,87	+ 0,03
150	0,79	0,79	+ 0,00
200	0,77	0,83	+ 0,06



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 perubahan yang fluktuatif pada *packet delivery ratio*. Pada lingkungan yang jarang dengan jumlah 50 *node*, menghasilkan perbedaan selisih PDR sebesar 0.01, atau naik sekitar 0,98% 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 0.03, atau naik sekitar 3,60% dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal PDR tersebut dari

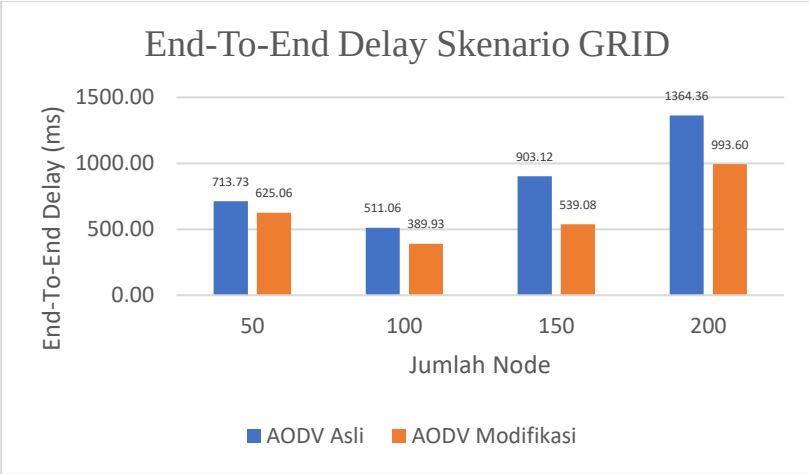
AODV asli. Pada lingkungan yang sedang dengan jumlah 150 *node*, *routing protocol* AODV asli dan yang telah dimodifikasi tidak menghasilkan perbedaan PDR. Pada lingkungan yang padat dengan jumlah 200 *node*, menghasilkan perbedaan selisih PDR sebesar 0.06, atau naik sekitar 8,00% dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal PDR tersebut.

Dapat dilihat rata-rata kenaikan yang terjadi untuk PDR adalah 3,15%. 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 End-to-end Delay Skenario Grid

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	713,73 ms	625,06 ms	- 88,67 ms
100	511,06 ms	389,93 ms	- 121,13 ms
150	903,12 ms	539,08 ms	- 364,04 ms
200	1.364,36 ms	993,60 ms	- 370,76 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 cukup signifikan. Pada lingkungan yang jarang dengan jumlah 50 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 88,67 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 14,19%, dimana *routing protocol* AODV yang sudah 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 121,13 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 23,70%, dimana *routing protocol* AODV yang telah 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 364,04 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 40,31%, dimana *routing protocol* AODV yang telah 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 370,76 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 27,17%, dimana *routing protocol* AODV yang telah dimodifikasi jauh lebih unggul dalam hal *end-to-end delay* tersebut.

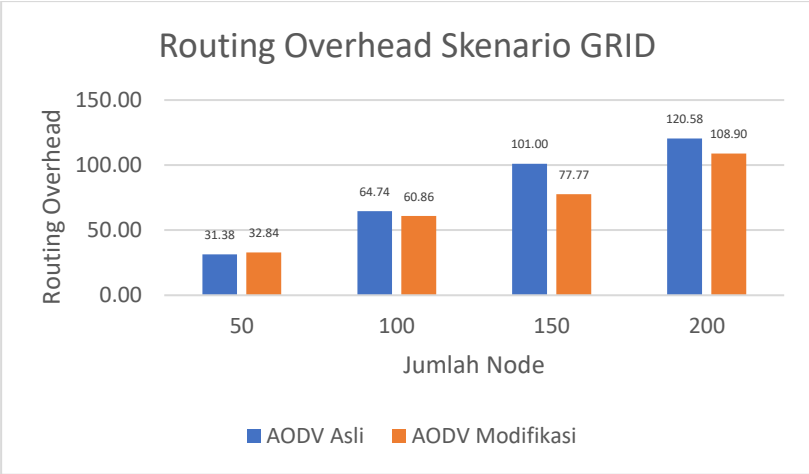
Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa semakin padat lingkungannya, maka *end-to-end delay* yang dihasilkan semakin bagus dengan menggunakan *routing protocol* AODV yang telah dimodifikasi.

Hal ini bisa terjadi karena dengan AODV modifikasi kali ini rute yang dipilih adalah rute dengan *total delay* terkecil sehingga meskipun didapatkan rute dengan jarak terpendek namun memiliki *total delay* yang lebih besar, maka rute tersebut tidak akan dipilih.

Untuk hasil pengambilan data *routing overhead* pada skenario *grid* 50 *node*, 100 *node*, 150 *node* dan 200 *node* dapat dilihat pada Gambar 5.3.

Tabel 5.5 Hasil Rata - Rata Routing Overhead Skenario Grid

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	31,38	32,84	+ 1,45
100	64,74	60,86	- 3,88
150	101,00	77,77	- 23,22
200	120,58	108,90	- 11,67



Gambar 5.3 Grafik Routing Overhead Skenario *Grid*

Berdasarkan grafik pada Gambar 5.3 dapat dilihat bahwa rata-rata *routing overhead* antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami kenaikan yang signifikan. Pada lingkungan yang jarang dengan jumlah 50 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 1,45 atau mengalami kenaikan sebesar 4,63%, dimana *routing protocol* AODV asli unggul dalam hal *routing overhead* tersebut karena menghasilkan *routing overhead* yang lebih rendah dari *routing protocol* AODV yang telah dimodifikasi. Pada lingkungan yang sedang dengan jumlah 100 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 3,88 atau mengalami penurunan sebesar 5,99%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *routing overhead* tersebut dari AODV asli. Pada lingkungan yang sedang dengan jumlah 150 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 23,22 atau mengalami penurunan sebesar 22,99%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *routing overhead* tersebut dari AODV asli. Pada lingkungan yang padat dengan jumlah 200 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 11,67 atau mengalami penurunan sebesar

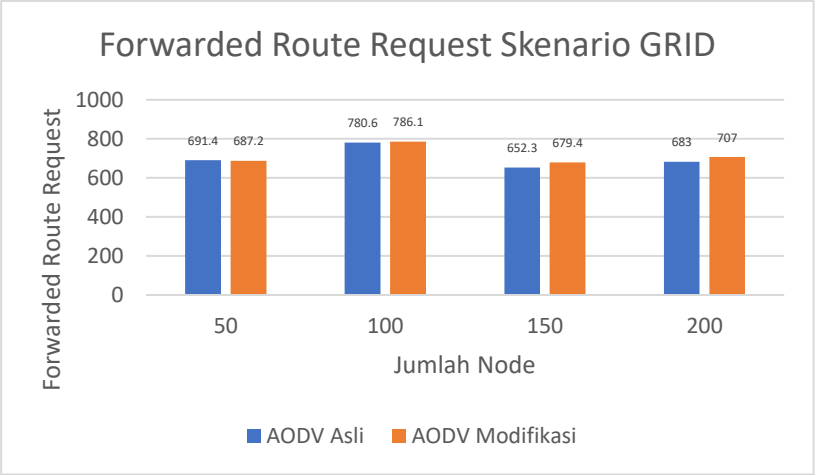
9,68%, dimana *routing protocol* AODV yang telah dimodifikasi lebih baik daripada AODV asli dalam hal *routing overhead* tersebut.

Dapat dilihat rata-rata penurunan yang terjadi untuk *routing overhead* adalah sebesar 8,51%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang lebih sedikit atau pada lingkungan dengan *node* yang jarang, menghasilkan *routing overhead* yang lebih bagus atau lebih sedikit daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan *routing overhead* yang lebih sedikit atau dalam hal ini berarti lebih unggul di semua lingkungan dengan jumlah selisih *routing overhead* yang cukup signifikan, kecuali di lingkungan dengan jumlah *node* yang sedang.

Untuk hasil pengambilan data *forwarded route request* (RREQ F) pada skenario *grid* 50 *node*, 100 *node*, 150 *node* dan 200 *node* dapat dilihat pada Gambar 5.4.

Tabel 5.6 Hasil Rata – Rata Forwarded Route Request Skenario Grid

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	691,4	687,2	- 4.2
100	780,6	786,1	+ 5.5
150	652,3	679,4	+ 27.1
200	683,0	707,0	+ 24



Gambar 5.4 Grafik Forwarded Route Request Skenario *Grid*

Berdasarkan grafik pada Gambar 5.4 dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi dan juga *routing protocol* asli memiliki jumlah *forwarded route request* (RREQ F) yang cukup stabil. Pada lingkungan yang jarang dengan jumlah 50 *node*, menghasilkan perbedaan selisih *forwarded route request* sebesar 4,2 atau mengalami penurunan sebesar 0,61%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *forwarded route request* tersebut karena menghasilkan *forwarded route request* yang lebih rendah dari *routing protocol* AODV asli. Pada lingkungan yang sedang dengan jumlah 100 *node*, menghasilkan perbedaan selisih *forwarded route request* sebesar 5,5 atau mengalami peningkatan sebesar 0,70%, dimana *routing protocol* AODV asli unggul dalam hal *forwarded route request* tersebut dari *forwarded route request* AODV yang telah dimodifikasi. Pada lingkungan yang sedang dengan jumlah 150 *node*, menghasilkan perbedaan selisih *forwarded route request* sebesar 27,1 atau mengalami peningkatan sebesar 4,15%, dimana *routing protocol* AODV asli lebih unggul dalam hal *forwarded route request* tersebut

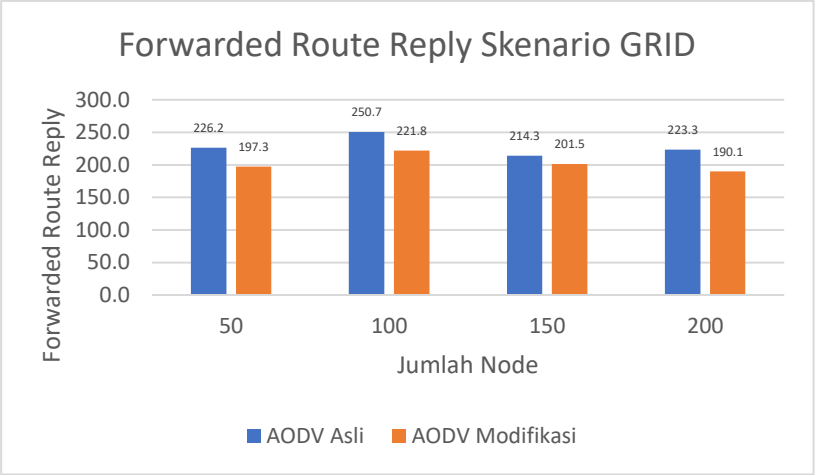
dari *forwarded route request* AODV yang telah dimodifikasi. Pada lingkungan yang padat dengan jumlah 200 *node*, menghasilkan perbedaan selisih *forwarded route request* sebesar 24 atau mengalami peningkatan sebesar 3,51%, dimana *routing protocol* AODV asli juga unggul dalam hal *forwarded route request* tersebut.

Dapat dilihat rata-rata peningkatan yang terjadi untuk *forwarded route request* adalah sebesar 1,94%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang jarang, menghasilkan *forwarded route request* yang lebih bagus atau lebih sedikit daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. Dapat dilihat pula bahwa AODV asli menghasilkan *forwarded route request* yang lebih bagus atau dalam hal ini lebih rendah daripada AODV yang telah dimodifikasi, kecuali di lingkungan dengan jumlah node yang jarang.

Untuk hasil pengambilan data *forwarded route reply* (RREP F) pada skenario *grid* 50 *node*, 100 *node*, 150 *node* dan 200 *node* dapat dilihat pada Gambar 5.5.

Tabel 5.7 Hasil Rata - Rata Forwarded Route Reply Skenario Grid

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	226,2	197,3	- 28,9
100	250,7	221,8	- 28,9
150	214,3	201,5	- 12,8
200	223,3	190,1	- 33,2



Gambar 5.5 Grafik Forwarded Route Request Skenario *Grid*

Berdasarkan grafik pada Gambar 5.5 dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi dan juga *routing protocol* asli memiliki perubahan jumlah *forwarded route reply* (RREP F) yang fluktuatif. Pada lingkungan yang jarang dengan jumlah 50 *node*, menghasilkan perbedaan selisih *forwarded route reply* sebesar 28,9 atau mengalami penurunan sebesar 12,78%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *forwarded route reply* tersebut karena menghasilkan *forwarded route reply* yang lebih rendah dari *routing protocol* AODV asli. Pada lingkungan yang sedang dengan jumlah 100 *node*, menghasilkan perbedaan selisih *forwarded route reply* sebesar 28,9 atau mengalami penurunan sebesar 11,53%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *forwarded route reply* tersebut dari *forwarded route reply* AODV asli. Pada lingkungan yang sedang dengan jumlah 150 *node*, menghasilkan perbedaan selisih *forwarded route reply* sebesar 12,8 atau mengalami penurunan sebesar 5,97%, dimana *routing protocol* AODV yang telah dimodifikasi lebih unggul dalam hal *forwarded route reply* tersebut

dari *forwarded route reply* AODV asli. Pada lingkungan yang padat dengan jumlah 200 *node*, menghasilkan perbedaan selisih *forwarded route request* sebesar 33,2 atau mengalami penurunan sebesar 14,88%, dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal *forwarded route reply* tersebut.

Dapat dilihat bahwa AODV asli menghasilkan *forwarded route reply* yang lebih bagus atau dalam hal ini lebih rendah daripada AODV yang telah dimodifikasi.

5.2.2 Hasil Uji Coba Skenario *Real*

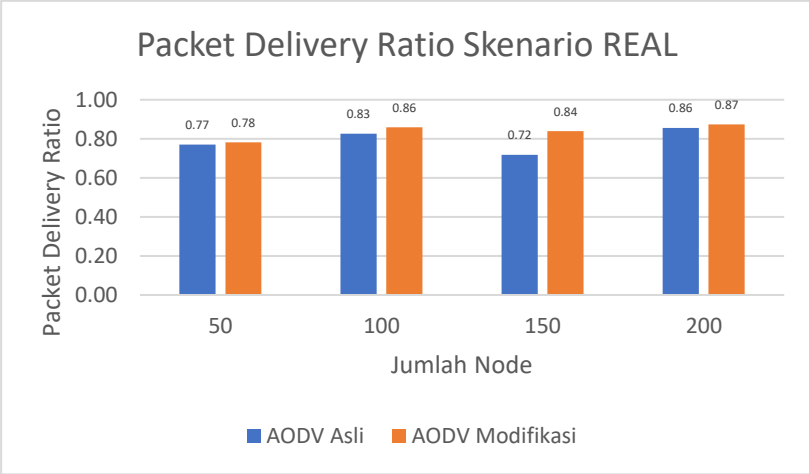
Pengujian pada skenario *real* digunakan untuk melihat perbandingan PDR, E2E, RO, RREQ F, dan RREP F antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi dalam pemilihan *node* yang dapat menerima paket *route request*.

Pengambilan data uji PDR, E2E, RO, RREQ F, dan RREP F 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 5.8, Tabel 5.9, Tabel 5.10, Tabel 5.11, dan Tabel 5.12.

Tabel 5.8 Hasil Rata - Rata *Packet Delivery Ratio* pada Skenario *Real*

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	0,77	0,78	+ 0,01
100	0,83	0,86	+ 0,03
150	0,72	0,84	+ 0,12
200	0,86	0,87	+ 0,01

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



Gambar 5.6 Grafik *Packet Delivery Ratio* Skenario *Real*

Berdasarkan grafik pada Gambar 5.6 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 0,01, atau naik menjadi sekitar 1,45% 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 0,03, atau mengalami peningkatan sekitar 3,98% 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 0,12, atau mengalami peningkatan sekitar 16,96% 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 0,02, atau mengalami peningkatan sekitar 2,13% dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal PDR tersebut.

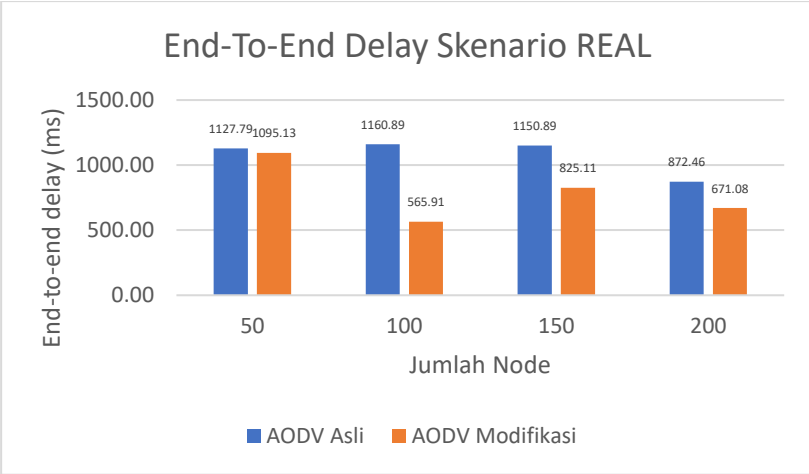
Dapat dilihat rata-rata kenaikan yang terjadi untuk PDR adalah 6,13%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* 100, menghasilkan PDR yang lebih bagus daripada di lingkungan dengan jumlah *node* 50, 150 maupun 200 untuk *routing protocol* AODV asli maupun *routing 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 *real* dengan jumlah *node* 50, 100, 150, dan 200 dapat dilihat pada Gambar 5.7.

Tabel 5.9 Hasil Rata -Rata *End-to-End Delay* pada Skenario *Real*

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	1.127,79 ms	1.095,13 ms	- 32,65 ms
100	1.160,89 ms	565,91 ms	- 594,97 ms
150	1150,89 ms	825,11 ms	- 325,77 ms
200	872,46 ms	671,08 ms	- 201,39 ms

Dari data di atas, dibuat grafik yang merepresentasikan hasil perhitungan E2E yang ditunjukkan pada Gambar 5.7.



Gambar 5.7 Grafik *End-to-end Delay* pada Skenario *Real*

Berdasarkan grafik pada Gambar 5.7 dapat dilihat bahwa rata-rata *end-to-end delay* antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami penurunan yang cukup signifikan. Pada lingkungan yang jarang dengan jumlah 50 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 32,65 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 2,90%, 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 594,97 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 51,25%, dimana *routing protocol* AODV yang dimodifikasi jauh lebih unggul dalam hal *end-to-end delay*. Pada pada lingkungan sedang dengan jumlah 150 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 325,77 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 28,31%, 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 201,39 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 23,08%, dimana *routing protocol* AODV yang telah dimodifikasi lebih unggul dalam hal *end-to-end delay* tersebut.

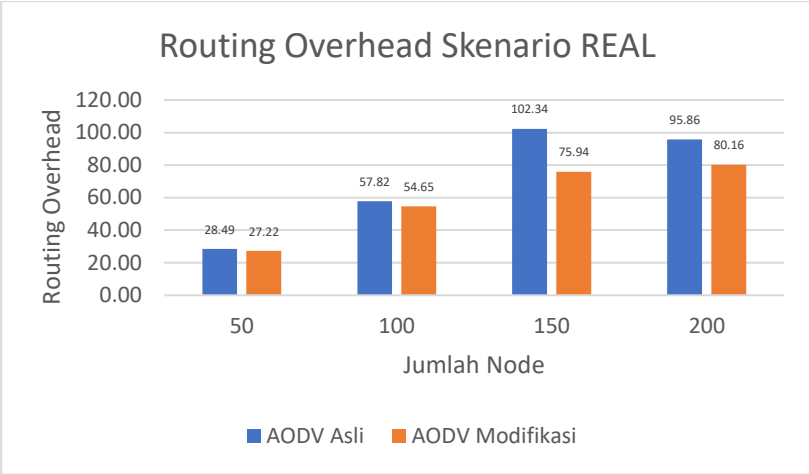
Jika ketiga lingkungan tersebut dibandingkan, dapat dilihat pada semua lingkungan, AODV yang telah dimodifikasi menghasilkan *end-to end delay* yang lebih baik daripada AODV asli dengan jumlah selisih *end-to-end delay* yang cukup signifikan. Hal ini bisa terjadi karena dengan AODV modifikasi kali ini rute yang dipilih adalah rute dengan *total delay* terkecil sehingga meskipun didapatkan rute dengan jarak terpendek namun memiliki *total delay* yang lebih besar, maka rute tersebut tidak akan dipilih.

Untuk hasil pengambilan data *routing overhead* pada skenario *real* 50 *node*, 100 *node*, 150 *node* dan 200 *node* dapat dilihat pada Gambar 5.8.

Tabel 5.10 Hasil Rata - Rata *Routing Overhead* Skenario *Real*

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	28,49	27,22	- 1,27
100	57,82	54,65	- 3,17
150	102,34	75,94	- 26,40
200	95,86	80,16	- 15,71

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



Gambar 5.8 Grafik *Routing Overhead* Skenario *Real*

Berdasarkan grafik pada Gambar 5.8 dapat dilihat bahwa rata-rata *routing overhead* antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami kenaikan yang cukup signifikan. Pada lingkungan yang jarang dengan jumlah 50 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 1,26 atau mengalami penurunan sebesar 4,44%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *routing overhead* tersebut karena menghasilkan *routing overhead* yang lebih rendah dari *routing protocol* AODV asli. Pada lingkungan yang sedang dengan jumlah 100 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 3,17 atau mengalami penurunan sebesar 5,49%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *routing overhead* tersebut dari AODV asli. Pada lingkungan yang sedang dengan jumlah 150 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 26,40 atau mengalami penurunan sebesar 25,79%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *routing overhead* tersebut dari AODV asli. Pada lingkungan yang padat dengan jumlah 200 *node*,

menghasilkan perbedaan selisih *routing overhead* sebesar 15,71 atau mengalami penurunan sebesar 16,39%, dimana *routing protocol* AODV yang telah dimodifikasi lebih unggul daripada AODV asli dalam hal *routing overhead* tersebut.

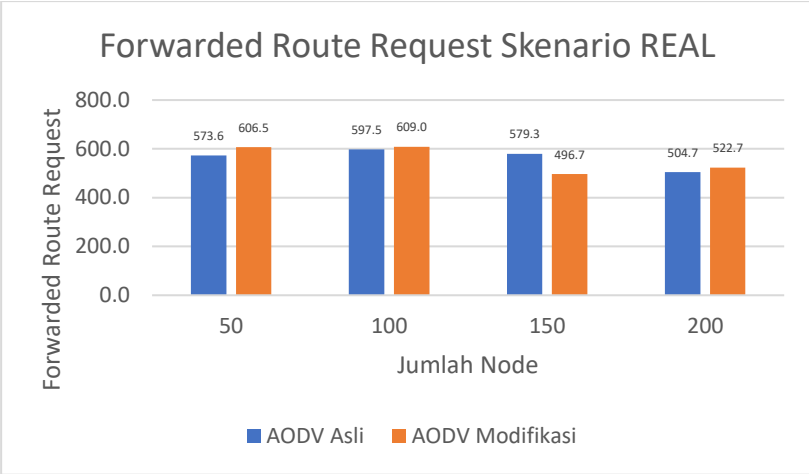
Dapat dilihat rata-rata penurunan yang terjadi untuk *routing overhead* adalah sebesar 13,03 %. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang lebih sedikit atau pada lingkungan dengan *node* yang jarang, menghasilkan *routing overhead* yang lebih bagus atau lebih sedikit daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan *routing overhead* yang lebih sedikit atau dalam hal ini AODV yang telah dimodifikasi unggul di semua lingkungan dengan jumlah selisih *routing overhead* yang cukup signifikan.

Untuk hasil pengambilan data *forwarded route request* (RREQ F) pada skenario *real* 50 *node*, 100 *node*, 150 *node* dan 200 *node* dapat dilihat pada Gambar 5.9.

Tabel 5.11 Hasil Rata - Rata *Forwarded Route Request* pada Skenario *Real*

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	573,6	606,5	+ 32,9
100	597,5	609,0	+ 11,5
150	579,3	496,7	- 82,6
200	504,7	522,7	+ 18,0

Dari data di atas, dibuat grafik yang merepresentasikan hasil perhitungan RREQ F yang ditunjukkan pada Gambar 5.9.



Gambar 5.9 Grafik *Forwarded Route Request* Skenario *Real*

Berdasarkan grafik pada Gambar 5.9 dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi dan juga *routing protocol* asli relatif tidak mengalami perubahan *forwarded route request* (RREQ F). Pada lingkungan yang jarang dengan jumlah 50 *node*, menghasilkan perbedaan selisih *forwarded route request* sebesar 32,9 atau mengalami peningkatan sebesar 5,74%, dimana *routing protocol* AODV asli unggul dalam hal *forwarded route request* tersebut karena menghasilkan *forwarded route request* yang lebih rendah dari *routing protocol* AODV yang telah dimodifikasi. Pada lingkungan yang sedang dengan jumlah 100 *node*, menghasilkan perbedaan selisih *forwarded route request* sebesar 11,5 atau mengalami peningkatan sebesar 1,92%, dimana *routing protocol* AODV asli juga unggul dalam hal *forwarded route request* tersebut dari *forwarded route request* AODV yang telah dimodifikasi. Pada lingkungan yang sedang dengan jumlah 150 *node*, menghasilkan perbedaan selisih *forwarded route request* sebesar 82,6 atau mengalami penurunan sebesar 14,26%, dimana *routing protocol* AODV yang telah dimodifikasi lebih unggul dalam hal *forwarded route request* tersebut dari *forwarded route request* AODV asli. Pada lingkungan yang padat dengan jumlah 200 *node*, menghasilkan

perbedaan selisih *forwarded route request* sebesar 18,0 atau mengalami peningkatan sebesar 3,57%, dimana *routing protocol* AODV asli juga unggul dalam hal *forwarded route request* tersebut.

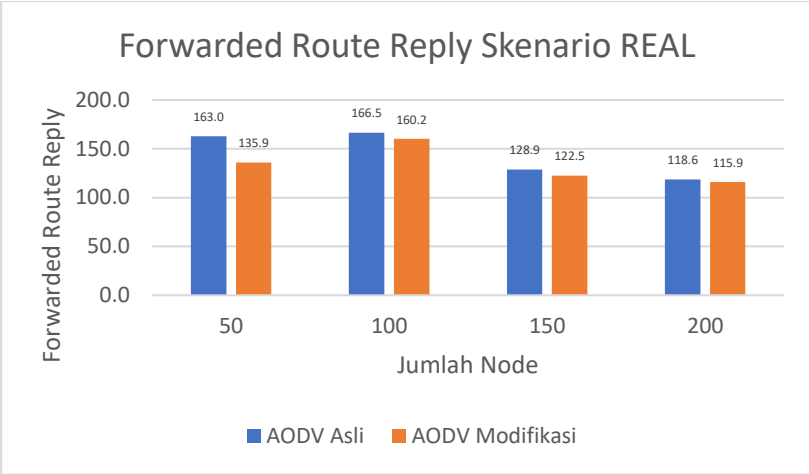
Dapat dilihat rata-rata penurunan yang terjadi untuk *forwarded route request* adalah sebesar 0,76% untuk hasil rata-rata dari 50 *node*, 100 *node*, 150 *node*, dan 200 *node*. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang padat, menghasilkan *forwarded route request* yang lebih bagus atau lebih sedikit 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 asli menghasilkan *forwarded route request* yang lebih bagus atau dalam hal ini lebih sedikit daripada AODV yang telah dimodifikasi dengan jumlah selisih *forwarded route request* yang cukup signifikan.

Untuk hasil pengambilan data *forwarded route reply* (RREP F) pada scenario *real* 50 *node*, 100 *node*, 150 *node* dan 200 *node* dapat dilihat pada Gambar 5.10.

Tabel 5.12 Hasil Rata - Rata RREP F pada Skenario Real

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	163,0	135,9	- 27,1
100	166,5	160,2	- 6,3
150	128,9	122,5	- 6,4
200	118,6	115,9	- 2,7

Dari data di atas, dibuat grafik yang merepresentasikan hasil perhitungan RREP F yang ditunjukkan pada Gambar 5.10.



Gambar 5.10 Grafik *Forwarded Route Reply* Skenario *Real*

Berdasarkan grafik pada Gambar 5.10 dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi dan juga *routing protocol* asli relatif tidak mengalami perubahan *forwarded route reply* (RREP F). Pada lingkungan yang jarang dengan jumlah 50 *node*, menghasilkan perbedaan selisih *forwarded route request* sebesar 27,1 atau mengalami penurunan sebesar 19,94%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *forwarded route reply* tersebut karena menghasilkan *forwarded route reply* yang lebih sedikit dari *routing protocol* AODV asli. Pada lingkungan yang sedang dengan jumlah 100 *node*, menghasilkan perbedaan selisih *forwarded route reply* sebesar 6,3 atau mengalami penurunan sebesar 3,78%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *forwarded route reply* tersebut dari *forwarded route reply* AODV asli. Pada lingkungan yang sedang dengan jumlah 150 *node*, menghasilkan perbedaan selisih *forwarded route reply* sebesar 6,4 atau mengalami penurunan sebesar 4,97%, dimana *routing protocol* AODV yang telah dimodifikasi lebih unggul dalam hal *forwarded route reply* tersebut dari *forwarded route reply* AODV asli. Pada lingkungan yang padat dengan jumlah 200 *node*, menghasilkan perbedaan selisih *forwarded route reply* sebesar 2,7 atau mengalami

penurunan sebesar 2,28%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *forwarded route reply* tersebut.

Dapat dilihat rata-rata penurunan yang terjadi untuk *forwarded route reply* adalah sebesar 6,91% untuk hasil rata-rata dari 50 *node*, 100 *node*, 150 *node*, dan 200 *node*. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang padat, menghasilkan *forwarded route reply* yang lebih bagus atau lebih sedikit 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 *forwarded route reply* yang lebih bagus atau dalam hal ini lebih sedikit 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 *delay* di lingkungan VANETs bisa dilakukan dengan cara mengirim paket RREP melalui rute dengan paket RREQ yang pertama kali sampai di *destination node* dan menghapus fungsi *update routing table* ketika rute baru dengan jumlah *hop count* lebih kecil ditemukan.
2. Dampak pemilihan rute berdasarkan *delay* terhadap performa protokol AODV pada skenario *grid* adalah rata-rata peningkatan *Packet Delivery Ratio* (PDR) sebesar 3,15%, rata-rata penurunan *End-to-End Delay* sebesar 25,90%, rata-rata penurunan *Routing Overhead* sebesar 8,51%, rata-rata peningkatan *Forwarded Route Request* (RREQ F) sebesar 1,94%, dan rata-rata penurunan *Forwarded Route Reply* (RREP F) sebesar 11,29%.
3. Dampak pemilihan rute berdasarkan *delay* terhadap performa protokol AODV pada skenario *real* adalah rata-rata peningkatan *Packet Delivery Ratio* (PDR) sebesar 6,13%, rata-rata penurunan *End-to-End Delay* sebesar 26,38%, rata-rata penurunan *Routing Overhead* sebesar 13,03%, rata-rata penurunan *Forwarded Route Request* (RREQ F) sebesar 0,76%, dan rata-rata penurunan *Forwarded Route Reply* (RREP F) sebesar 6,91%.

6.2 Saran

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

1. Lebih banyak uji coba yang dilakukan untuk mendapatkan hasil yang lebih akurat, seperti lebih dari 10 skenario untuk tiap jumlah *node*nya.
2. Menambahkan mekanisme untuk meng-*generate traffic delay* dengan *processing* dan *queueing delay* pada *node-node* yang ada.

DAFTAR PUSTAKA

[1] "VANET - Vehicle Ad hoc Network," [Online]. Available: http://comp.ist.utl.pt/~rmr/WSN/CaseStudies2007-no/WSN_Transportation/. [Diakses 15 November 2017].

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

[3] B. Roy, S. Banik, N. Chaki dan B. Saha, " QAODV: An AODV Based Routing Protocol for QoS Parameters", 2010.

[4] 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.

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

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

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

[8] 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.

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

[10] 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.

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

LAMPIRAN

A.1 Kode Fungsi Delay

```
recv_time = CURRENT_TIME;
current_delay = recv_time - rq-
>rq_recv_time;
rq->rq_recv_time = recv_time;
rq->rq_total_delay = current_delay + rq-
>rq_prev_delay;
rq->rq_prev_delay = current_delay;

DelayData data;
data.recv_time = CURRENT_TIME;
data.current_delay = data.recv_time -
delaydata[rq->rq_src].recv_time;
data.total_delay = data.current_delay +
delaydata[rq->rq_src].total_delay;
delaydata[index] = data;
```

A.2 Kode Fungsi update routing table

```
aodv_rt_entry *rt0;

   rt0 = rtable.rt_lookup(rq->rq_src);
   if(rt0 == 0) {
     rt0 = rtable.rt_add(rq->rq_src);
   }

   rt0->rt_expire = max(rt0->rt_expire,
(CURRENT_TIME + REV_ROUTE_LIFE));

   if (rq->rq_src_seqno > rt0->rt_seqno )
||(rq->rq_src_seqno == rt0->rt_seqno) {
rt_update(rt0, rq->rq_src_seqno, rq-
>rq_hop_count, ih->saddr(),
             max(rt0->rt_expire,
(CURRENT_TIME + REV_ROUTE_LIFE)) );
     if (rt0->rt_req_timeout > 0.0) {
       rt0->rt_req_cnt = 0;
       rt0->rt_req_timeout = 0.0;
       rt0->rt_req_last_ttl = rq-
>rq_hop_count;
       rt0->rt_expire = CURRENT_TIME +
ACTIVE_ROUTE_TIMEOUT;
     }
```

A.3 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.4 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.5 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.6 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";
}
```

A.7 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;
}
```

A.8 Kode Skrip AWK *Forwarded Route Request*

```
BEGIN {
      rt_forward = 0;
}
{
            if (($1 == "s") && ($4 ==
"RTR") && ($7 == "AODV") && ($25 ==
"(REQUEST)") && ($3 != "_58_")){
                  rt_forward++;
            }
      }
}
END {
      printf "Forwarded Route Request\t=
%d \n", rt_forward;
}
```

A.9 Kode Skrip AWK *Forwarded Route Reply*

```
BEGIN {
      rt_forward = 0;
}
{
            if (($1 == "s") && ($4 ==
"RTR") && ($7 == "AODV") && ($25 ==
"(REPLY)") && ($3 != "_58_")){
                  rt_forward++;
            }
      }
}
END {
      printf "Forwarded Route Reply\t= %d
\n", rt_forward;
}
```

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

MUHAMMAD ADIB ARINANDA, lahir di Balikpapan, 15 Juni 1997. Penulis adalah anak kedua dari dua bersaudara. Penulis menempuh pendidikan sekolah dasar di SDIT Istiqomah Balikpapan lalu melanjutkan pendidikan sekolah menengah pertama di SMPIT Istiqomah Balikpapan dan penulis menempuh pendidikan menengah atas di SMA Negeri 1 Balikpapan. Selanjutnya penulis melanjutkan pendidikan sarjana di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya. Selama kuliah, penulis aktif dalam berbagai organisasi baik tingkat jurusan maupun universitas.

Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Arsitektur dan Jaringan Komputer (AJK). Sebagai mahasiswa, penulis berperan aktif dalam beberapa organisasi kampus seperti Koordinator Perkap SCHEMATICS 2016 dan Wakil Ketua SCHEMATICS 2017. Penulis pernah melakukan kerja praktik di PT. Aplikanusa Lintasarta Jakarta pada Januari 2018 dan membuat fungsi macro untuk untuk menunjang pembuatan laporan finansial perusahaan di Microsoft Excel. Selain itu penulis juga pernah melakukan magang di E-Life Solutions PLT Johor Bahru, Malaysia pada Juli – Agustus 2018 dan membuat aplikasi berbasis web Smart Medical Equipment Monitoring System. Penulis dapat dihubungi melalui nomor *handphone*: 08115408066 atau *email*: arinanda.adib@gmail.com.