





**ITS**  
Institut  
Teknologi  
Sepuluh Nopember

**TUGAS AKHIR - IF184802**

# **PENGEMBANGAN AODV DENGAN METODE DQUEUE (DQAODV) PADA LINGKUNGAN VANET**

**MUHAMMAD BUDIANA EKA FARUQI**  
**NRP 0511154000084**

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

Dosen Pembimbing II  
Wahyu Suadi, S.Kom., M.M., M.Kom.

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







**TUGAS AKHIR - IF184802**

# **PENGEMBANGAN AODV DENGAN METODE DQUEUE (DQAODV) PADA LINGKUNGAN VANET**

**MUHAMMAD BUDIANA EKA FARUQI**  
**NRP 05111540000084**

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

**Dosen Pembimbing II**  
**Wahyu Suadi, S.Kom., M.M., M.Kom.**

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

*(Halaman ini sengaja dikosongkan)*



**UNDERGRADUATE THESES - IF184802**

# **AODV DEVELOPMENT WITH DQUEUE (DQAODV) METHOD IN THE VANET ENVIRONMENT**

**MUHAMMAD BUDIANA EKA FARUQI**  
**NRP 05111540000084**

**First Advisor**

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

**Second Advisor**

**Wahyu Suadi, S.Kom., M.M., M.Kom.**

**Department of Informatics**

**Faculty of Intelligent Electrical and Informatics Technology**

**Sepuluh Nopember Institute of Technology**

**Surabaya 2019**

***(Halaman ini sengaja dikosongkan)***



## LEMBAR PENGESAHAN

### PENGEMBANGAN AODV DENGAN METODE DQUEUE (DQAODV) PADA LINGKUNGAN VANET

#### TUGAS AKHIR

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


Oleh:

**MUHAMMAD BUDIANA EKA FARUQI**


**NRP: 05111540000084**

Disetujui oleh Pembimbing Tugas Akhir,

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

  
.....  
(Pembimbing 1)

2. Wahyu Suadi, S.Kom., M.M., M.Kom  
(NIP. 197110302002121001)

  
.....  
(Pembimbing 2)

**SURABAYA  
JANUARI, 2020**

*(Halaman ini sengaja dikosongkan)*

## **PENGEMBANGAN AODV DENGAN METODE DQUEUE (DQAODV) PADA LINGKUNGAN VANET**

**Nama Mahasiswa : MUHAMMAD BUDIANA EKA  
FARUQI**  
**NRP : 05111540000084**  
**Departemen : Teknik Informatika FTEI-ITS**  
**Dosen Pembimbing 1 : Dr.Eng. Radityo Anggoro, S.Kom.,  
M.Sc.**  
**Dosen Pembimbing 2 : Wahyu Suadi, S.Kom., M.M.,  
M.Kom.**

### **Abstrak**

Teknologi internet saat ini dapat digunakan sebagai suatu pemecahan suatu masalah. Contoh perkembangannya adalah jaringan nirkabel *ad-hoc*. Seperti contohnya proses penentuan rute perjalanan berhubungan dengan rute pengiriman data informasi dalam jaringan internet. Untuk melakukan pemecahan permasalahan hal tersebut dapat memanfaatkan teknologi jaringan *Ad-Hoc* yang mana mendasari pembuatan Vehicular Ad-Hoc Networks (VANETs). *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 *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, jika suatu *node* sudah mendapatkan paket yang sama dari *intermediate node* yang lain, maka data tentang node tersebut akan dibuang. Pada saat melakukan *unicast* RREP jika terjadi kerusakan *link*, maka *graph* akan terputus karena tidak ada *back-up node*.

Pada Tugas Akhir ini mengimplementasikan metode dqAODV agar saat suatu *node* menerima paket yang sama dari *intermediate node* lain, maka *broadcast ID* dari *node* pengirim paket akan disimpan untuk dijadikan *back-up*. Dari hasil uji coba, AODV yang dimodifikasi pada skenario *grid* belum berhasil meningkatkan nilai rata-rata *Packet Delivery Ratio* (PDR) yang naik menjadi 0,68%, rata-rata *End-to-End Delay* (E2E) naik menjadi 107,53%,tetapi menurunkan *Routing Overhead* hingga 57,66%. Sedangkan pada skenario *real* juga belum berhasil meningkatkan nilai rata-rata *Packet Delivery Ratio* (PDR) yang turun 4,96%,tetapi berhasil menurunkan *End-to-End Delay* (E2E) hingga 7,12%, menurunkan *Routing Overhead* hingga 61,71%.

**Kata kunci:** *VANET, AODV, dqueue, dqAODV*

## **AODV DEVELOPMENT WITH DQUEUE (DQAODV) METHOD IN THE VANET ENVIRONMENT**

**Student's Name** : MUHAMMAD BUDIANA EKA  
FARUQI  
**Student's ID** : 05111540000084  
**Department** : Informatics – FTEI ITS  
**First Advisor** : Dr.Eng. Radityo Anggoro, S.Kom.,  
M.Sc.  
**Second Advisor** : Wahyu Suadi, S.Kom., M.M., M.Kom.

### ***Abstract***

*In this era, Internet technology can be used as problem solving. An example of its development is ad-hoc wireless networks. For example, the process of determining the route of travel is related to sending data information on the internet network. To do this problem solver can use Ad-Hoc network technology which underlies the creation of Ad-Hoc Vehicle Networks (VANETs). Ad hoc Network Vehicle (VANET) is the development of an Ad hoc Mobile Network (MANET) where nodes have characteristics with very high mobility and are limited to their movement patterns. There are many routing protocols that can be implemented on VANET, one of which is Ad hoc On demand Distance Vector (AODV). 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 two 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 AODV performance, if a node has received the same packet from another intermediate node, then data about that node will be issued. When unifying the RREP if a link break occurs, the graph will be broken because there is no backup node.*

*In this final project implements the dqAODV method so that when a node receives the same packet from another intermediate node, the broadcast ID of the packet sending node will be stored for back-up. From the results of the trial, AODV modified in the grid scenario has not succeeded in increasing the average value of Packet Delivery Ratio (PDR) which rose to 0.68%, the average End-to-End Delay (E2E) rose to 107.53% , but decreased Routing Overhead by 57.66%. Whereas in the real scenario it also hasn't succeeded in increasing the average value of Packet Delivery Ratio (PDR) which fell 4.96%, but succeeded in reducing the End-to-End Delay (E2E) to 7.12%, reducing Routing Overhead to 61.71 %.*

***Keyword: VANETs, AODV, dqueue, dqAODV***

## KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul **“Pengembangan AODV Dengan Metode DQUEUE (DQAODV) Pada Lingkungan VANET”**.

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 segala rahmat yang diberikan sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Keluarga penulis terutama Bapak dan Ibu selaku orang tua penulis dan adik-adik penulis atas segala dukungan berupa motivasi, doa, moral, dan material sehingga penulis tetap semangat dan dapat menyelesaikan Tugas Akhir ini.
3. Bapak Dr. Eng. Radityo Anggoro, S.Kom., M.Sc., dan Bapak Wahyu Suadi, S.Kom., M.M., M.Kom. selaku dosen pembimbing penulis atas nasihat, arahan dan bantuannya sehingga penulis dapat menyelesaikan Tugas Akhir ini.
4. Teman-teman dari Warkop WARDUG (Heakal, Penyok, Dito, Gandhi, Irvan, Janar, Beryl, Dio, Maul, Pandu, Faris, Tegar, Cak Tek, dan Mas Atom) yang selalu memberikan semangat, hiburan, dan menjadi tempat bertukar pikiran dan pendapat serta menemani penulis di WARDUG selama penulis berkuliah dan mengerjakan Tugas Akhir ini.
5. Teman-teman dari Karambia Cukia (Zahri, Didi, Harits, Diko, Evan, Arif, Farhan, Adit, Zikrul, Faldo, Syauqi, Yogi, Capaik, Ajin, Mutia, Reisa, Yola, Feby, Yaya, Vivien) yang

telah menemani, memotivasi, memberikan doa, memberikan hiburan di kala penulis sedang jenuh saat pengerjaan Tugas Akhir ini.

6. Teman – teman Informatika 2015 yang telah memberikan semangat dan doa, serta fasilitas kepada penulis dalam mengerjakan 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, Desember 2019

MUHAMMMAD BUDIANA EKA FARUQI



## DAFTAR ISI

Abstrak.....	vii
<i>Abstract</i> .....	ix
KATA PENGANTAR.....	xi
DAFTAR ISI.....	xiii
DAFTAR GAMBAR.....	xvii
DAFTAR TABEL.....	xix
<b>BAB I PENDAHULUAN.....</b>	<b>1</b>
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah .....	2
1.3 Batasan Permasalahan .....	2
1.4 Tujuan.....	3
1.5 Manfaat .....	3
1.6 Metodologi.....	3
1.6.1 Penyusunan Proposal Tugas Akhir.....	3
1.6.2 Studi Literatur .....	4
1.6.3 Analisis dan Desain Sistem.....	4
1.6.4 Implementasi Sistem .....	4
1.6.5 Pengujian dan Evaluasi .....	4
1.6.6 Penyusunan Buku .....	5
1.7 Sistematika Penulisan Laporan .....	5
<b>BAB II TINJAUAN PUSTAKA.....</b>	<b>7</b>
2.1 DqAODV.....	7
2.2 VANETs .....	7
2.3 <i>Ad-hoc On demand Distance Vector (AODV)</i> .....	9
2.4 <i>Network Simulator-2 (NS-2)</i> .....	11
2.4.1 Instalasi .....	122
2.4.2 <i>Trace File</i> .....	12
2.5 OpenStreetMap.....	144
2.6 <i>Java OpenStreetMap Editor (JOSM)</i> .....	155
2.7 <i>Simulation of Urban Mobility (SUMO)</i> .....	155
2.8 AWK.....	177
<b>BAB III PERANCANGAN .....</b>	<b>199</b>

3.1	Deskripsi Umum .....	199
3.2	Perancangan Skenario Mobilitas .....	221
3.2.1	Perancangan Skenario <i>Grid</i> .....	211
3.2.2	Perancangan Skenario <i>Real</i> .....	233
3.3	Perancangan Modifikasi <i>Routing Protocol</i> AODV .....	233
3.3.1	Perancangan Penghitungan <i>Node Degree</i> .....	26
3.4	Perancangan Simulasi pada NS-2 .....	26
3.5	Perancangan Metrik Analisis .....	27
3.5.1	<i>Packet Delivery Ratio</i> (PDR) .....	27
3.5.2	<i>Average End-to-End Delay</i> (E2E) .....	27
3.5.3	<i>Routing Overhead</i> (RO) .....	28
<b>BAB IV</b>	<b>IMPLEMENTASI .....</b>	<b>2929</b>
4.1	Implementasi Skenario Mobilitas .....	29
4.1.1	Skenario <i>Grid</i> .....	2929
4.1.2	Skenario <i>Real</i> .....	33
4.2	Implementasi Modifikasi pada <i>Routing Protocol</i> AODV untuk Meminimalisir Kegagalan RREP saat Mencari Rute Kembali .....	35
4.2.1	Implementasi Perubahan Format Paket RREQ ....	35
4.2.2	Implementasi Metode <i>First In, First Out</i> (FIFO) pada dqAODV .....	36
4.3	Implementasi Simulasi pada NS-2 .....	37
4.4	Implementasi Metrik Analisis .....	38
4.4.1	Implementasi <i>Packet Delivery Ratio</i> (PDR) .....	39
4.4.2	Implementasi <i>Average End-to-End Delay</i> (E2E) .....	40
4.4.3	Implementasi <i>Routing Overhead</i> (RO) .....	41
<b>BAB V</b>	<b>UJICOBA DAN EVALUASI .....</b>	<b>43</b>
5.1	Lingkungan Uji Coba .....	43
5.2	Hasil Uji Coba .....	44
5.2.1	Hasil Uji Coba Skenario <i>Grid</i> .....	44
5.2.2	Hasil Uji Coba Skenario <i>Real</i> .....	50
<b>BAB VI</b>	<b>KESIMPULAN DAN SARAN .....</b>	<b>57</b>
6.1	Kesimpulan .....	57

6.2	Saran.....	57
<b>DAFTAR PUSTAKA .....</b>		<b>59</b>
<b>LAMPIRAN.....</b>		<b>61</b>
A.1	Kode Skenario NS-2 .....	61
A.2	Kode Konfigurasi <i>Traffic</i> .....	64
A.3	Kode Skrip AWK <i>Packet Delivery Ratio</i> .....	65
A.4	Kode Skrip AWK Rata-Rata <i>End-to-End Delay</i> .....	66
A.5	Kode Skrip AWK <i>Routing Overhead</i> .....	67
A.6	Kode Skrip DqAODV .....	68
<b>BIODATA PENULIS .....</b>		<b>71</b>

*(Halaman ini sengaja dikosongkan)*

## DAFTAR GAMBAR

<b>Gambar 2.1</b> Ilustrasi VANETs [2] .....	9
<b>Gambar 2.2</b> Ilustrasi pencarian rute routing protocol AODV [3] 10	
<b>Gambar 2.3</b> Perintah untuk menginstall dependency NS-2 .....	122
<b>Gambar 2.4</b> Baris kode yang diubah pada file ls.h .....	122
<b>Gambar 3.1</b> Diagram Alur Rancangan Simulasi.....	199
<b>Gambar 3.2</b> Alur perancangan skenario Grid dan Real .....	222
<b>Gambar 3.3</b> Algoritma dqAODV.....	254
<b>Gambar 3.4</b> Alur penggunaan DqAODV dalam AODV .....	<b>Error!</b>
<b>Bookmark not defined.5</b>	
<b>Gambar 3.5</b> Modifikasi paket format RREQ.....	<b>Error!</b>
<b>Bookmark not defined.6</b>	
<b>Gambar 4.1</b> Perintah netgenerate .....	2929
<b>Gambar 4.2</b> Hasil Generate Peta Grid.....	300
<b>Gambar 4.3</b> Perintah randomTrips.....	311
<b>Gambar 4.4</b> Perintah duarouter .....	311
<b>Gambar 4.5</b> File Skrip .sumocfg .....	322
<b>Gambar 4.6</b> Perintah SUMO untuk membuat skenario .xml .....	322
<b>Gambar 4.7</b> Perintah traceExporter .....	333
<b>Gambar 4.8</b> Ekspor Peta dari OpenStreetMap .....	333
<b>Gambar 4.9</b> Perintah netconvert .....	344
<b>Gambar 4.10</b> Hasil Konversi Peta Real .....	344
<b>Gambar 4.11</b> Potongan Kode Modifikasi Fungsi recvRequest().....	366
<b>Gambar 4.12</b> Implementasi Simulasi NS-2.....	<b>Error!</b>
<b>Bookmark not defined.37</b>	
<b>Gambar 4.13</b> Implementasi Simulasi File Traffic.....	38
<b>Gambar 4.14</b> Pseudocode untuk Perhitungan PDR .....	39
<b>Gambar 4.15</b> Pseudocode untuk Perhitungan E2E .....	370
<b>Gambar 4.16</b> Pseudocode Perhitungan Routing Overhead.....	381
<b>Gambar 5.1</b> Grafik Packet Delivery Ratio Skenario Grid .....	445
<b>Gambar 5.2</b> Grafik End-to-end Delay Skenario Grid .....	4747
<b>Gambar 5.3</b> Grafik Routing Overhead Skenario Grid .....	4949
<b>Gambar 5.4</b> Grafik Packet Delivery Ratio Skenario Real .....	51
<b>Gambar 5.5</b> Grafik End-to-end Delay pada Skenario Real.....	53

<b>Gambar 5.6</b>	Grafik Routing Overhead Skenario Real .....	55
-------------------	---	----

*(Halaman ini sengaja dikosongkan)*

## DAFTAR TABEL

<b>Tabel 2.1</b> Struktur Paket RREQ.....	10
<b>Tabel 2.2</b> Detail Penjelasan Trace File AODV .....	1313
<b>Tabel 3.1</b> Daftar Istilah .....	2020
<b>Tabel 5.1</b> Spesifikasi Perangkat yang Digunakan .....	433
<b>Tabel 5.2</b> Lingkungan Uji Coba .....	4444
<b>Tabel 5.3</b> Hasil Rata - Rata PDR Skenario Grid.....	455
<b>Tabel 5.4</b> Hasil Rata - Rata E2E Skenario Grid .....	4646
<b>Tabel 5.5</b> Hasil Rata - Rata RO Skenario Grid.....	48
<b>Tabel 5.6</b> Hasil Rata - Rata PDR pada Skenario Real .....	51
<b>Tabel 5.7</b> Hasil Rata -Rata E2E pada Skenario Real .....	5252
<b>Tabel 5.8</b> Hasil Rata - Rata RO Skenario Real .....	5454

*(Halaman ini sengaja dikosongkan)*



# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Saat ini, perkembangan teknologi informasi dan komunikasi sudah mengalami kemajuan yang didukung dengan perkembangan dunia internet yang sangat pesat. Teknologi internet saat ini juga digunakan sebagai suatu pemecahan suatu masalah. Seperti contohnya pada permasalahan di jalan raya dengan melakukan pencarian rute tercepat agar sampai tujuan. Proses penentuan rute perjalanan berhubungan dengan rute pengiriman data dalam jaringan internet. Untuk melakukan pemecahan permasalahan hal tersebut dapat memanfaatkan teknologi jaringan *Ad-Hoc* yang mana mendasari pembuatan *Vehicle Ad-Hoc Networks* (VANETs). *Vehicle Ad hoc Networks* (VANETs) merupakan perkembangan dari *Mobile Ad hoc Network* (MANET) dimana setiap node memiliki mobilitas yang tinggi dan dikhususkan untuk jaringan yang dinamis. [1].

*Routing protocol* dalam VANET dibedakan menjadi dua model, yaitu *proactive* dan *reactive routing*. *Proactive routing* adalah protokol yang bekerja dengan cara melakukan *update* tabel *routing* setiap saat pada waktu tertentu tanpa memperhatikan beban jaringan, *bandwidth* dan ukuran jaringan. Sedangkan *reactive routing* adalah merupakan mekanisme *routing* yang membentuk tabel *routing* hanya jika ada permintaan pengiriman data atau paket [2].

Salah satu contoh *reactive routing protocol* adalah *routing protocol* AODV. Contoh penggunaan protokol AODV adalah diimplementasikannya protokol tersebut kedalam jaringan sensor nirkabel melalui simulasi 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. AODV bekerja

dengan tiga tipe kontrol pesan untuk menemukan dan menjaga rute, yaitu *Route Request*(RREQ), *Route Reply*(RREP), *Route Error*(RRER). Mekanisme pertama *routing protocol* AODV adalah proses menemukan rute untuk mengirim paket dari *node* sumber menuju *node* tujuan. *Node* sumber melakukan *broadcast* RREQ ke semua *node* tetangga, proses ini berlanjut hingga *node* tujuan ditemukan. Lalu *node* tujuan maupun *node* yang mengarah ke *node* tujuan melakukan *unicast* pesan RREP ke *node* sebelumnya yang berdekatan dari mana *node* tersebut mendapatkan pesan RREQ terlebih dahulu. Dengan ini maka jalur dari sumber ke *node* tujuan akan terbentuk. Ketika *node* menemukan kerusakan dalam rute aktif, pesan *error* akan dihasilkan. Lalu *node* tempat kerusakan terjadi akan mengirim pesan RRER kepada *node* selanjutnya yang berada pada jalur *node* tujuan[10].

Pada Tugas Akhir ini penulis akan melakukan modifikasi pada *routing protocol* AODV dengan mengimplementasikan konsep dqAODV (antrian paket) berdasarkan *Packet Delivery Ratio*, *End-to-End Delay*, dan *Routing Overhead*. Dengan implementasi ini diharapkan akan mengurangi kegagalan saat RREP mencari rute kembali menuju *node* sumber dan dapat meningkatkan performa AODV pada VANET.

## 1.2 Rumusan Masalah

Tugas Akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana meminimalisir kegagalan saat RREP mencari rute kembali ke *node* sumber ?
2. Bagaimana dampak penggunaan dqAODV terhadap performa AODV pada VANET?

## 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).

## **1.4 Tujuan**

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Meminimalisir kegagalan RREP dalam mencari rute kembali menuju *node* sumber.
2. Meningkatkan hasil *throughput* daripada AODV biasa.

## **1.5 Manfaat**

Dengan dibuatnya Tugas Akhir ini diharapkan akan memberikan manfaat dalam meningkatkan performa AODV yang telah dimodifikasi.

## **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.

### **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 dianalisis berasal dari perhitungan *Packet Delivery Ratio*, *packet loss*, dan *throughput*. Hal ini bertujuan untuk merumuskan solusi yang tepat untuk konfigurasi AODV yang dimodifikasi dalam lingkungan topologi MANET. Setelah selesai diaplikasikan pada MANET, dilakukan simulasi yang dilakukan pada VANETs dengan bantuan SUMO.

### **1.6.4 Implementasi Sistem**

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

## **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 DqAODV**

DqAODV merupakan sebuah metode modifikasi pada AODV yang digunakan untuk menyimpan *broadcast ID node*. Pada AODV biasa, saat sebuah *node* sudah menerima paket RREQ lalu menerima paket yang sama dari *node* lain, maka paket tersebut akan dibuang. Jika terjadi kerusakan pada saat *unicast RREP* saat mencari rute kembali menuju *node* sumber, pesan *error* akan dihasilkan dan *node* tempat terjadi kerusakan akan mengirimkan paket RRER ke *node* selanjutnya pada rute aktif menuju *node* sumber. Pada DqAODV, *broadcast ID* dari *node* lain yang mengirimkan paket RREQ akan disimpan dalam sebuah list di *node* penerima. Nantinya jika terjadi kerusakan, DqAODV akan menyediakan *node* cadangan yang sudah disimpan di *node* tempat kerusakan terjadi.

#### **2.2 VANET**

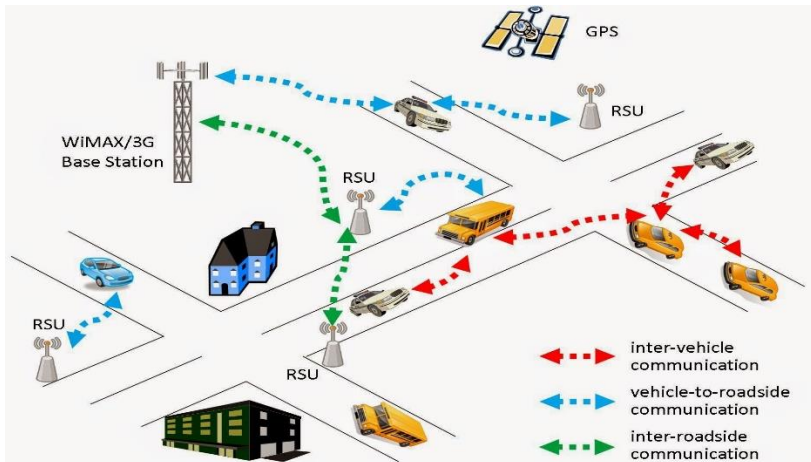
*Vehicular Ad-hoc Network* (VANET) merupakan sekumpulan *node* yang bergerak secara dinamis (*mobile*) sebagai pengembangan dari *Mobile Ad Hoc Network* (MANET) yang mempertimbangkan semua kendaraan dalam jaringan sebagai *node* tersebut untuk berkomunikasi dengan kendaraan lainnya pada radius tertentu [2]. Pada VANET maupun MANET, *node* yang bergerak tergantung kepada *ad hoc routing protocol* untuk menentukan bagaimana proses pengiriman data dari *node* asal ke *node* tujuan. Meskipun menggunakan *routing protocol* yang sama, namun VANET memiliki karakter yang berbeda dengan MANET karena VANET memiliki batasan pergerakan dan kecepatan yang tinggi yang menciptakan keunikan karakteristik dari VANET. VANET merupakan

pengembangan dari *Mobile Ad-hoc Network* (MANET) dimana pengembangannya difokuskan pada kendaraan (*vehicle*) dan *infrastructure* yang dapat saling berkomunikasi maupun mengirimkan data sebagai pengembangan *Intelligent Transport System* (ITS) dengan tujuan meningkatkan keselamatan dan kenyamanan berkendara. Komunikasi Wireless ini meliputi komunikasi *Inter-Vehicle Communication* (IVC), *Vehicle to Roadside* (V2R), atau *Roadside to Roadside* (R2R)..

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. Perubahan pergerakan pada VANETs menjadi salah satu permasalahan dalam pengiriman paket data sehingga dibutuhkan informasi jarak antar *node*, kecepatan dan *delay* transmisi [3]. Ilustrasi VANETs dapat dilihat pada Gambar 2.1.

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





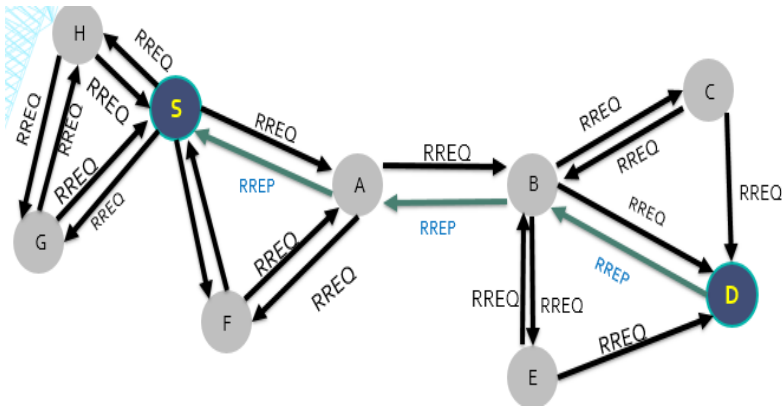
Gambar 2.1 Ilustrasi VANET [2]

### 2.3 Ad-hoc On demand Distance Vector (AODV)

*Ad-hoc On demand Distance Vector (AODV)* adalah salah satu *routing* protokol yang dirancang untuk jaringan *ad-hoc mobile* dan termasuk dalam klasifikasi *reactive routing protocol*. Dimana merupakan algoritman *routing* permintaan, yang berarti sebuah protokol yang hanya membuat sebuah rute antara node hanya saat dibutuhkan. AODV menggunakan table routing satu *entry* untuk setiap tujuan, tanpa menggunakan *routing*.

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 mempercayakan pada tabel *routing* untuk menyebarkan *Route Reply* kembali ke sumber dan mengarahkan paket menuju tujuan. Ciri utama dari AODV adalah menjaga *timer-based state* pada setiap *node* sesuai dengan penggunaan tabel *routing*.

AODV adalah sebuah metode *routing* pesan antar *node* yang memungkinkan *node-node* tersebut untuk melewati 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* [4]. Ilustrasi pencarian rute oleh AODV dapat dilihat pada Gambar 2.2. Struktur paket RREQ dapat dilihat pada Tabel 2.1.



**Gambar 2.2** Ilustrasi pencarian rute *routing protocol* AODV [11]

**Tabel 2.1** Struktur Paket RREQ

<i>source_addr</i>	<i>source_sequence_#</i>	<i>broadcast_id</i>
<i>dest_addr</i>	<i>dest_sequence_#</i>	<i>hop_cnt</i>

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

## 2.4 Network Simulator-2 (NS-2)

*Network Simulator 2* (NS-2) merupakan sebuah network simulator yang dibuat dengan tujuan riset dan pendidikan. Pada awalnya, NS dibangun sebagai varian dari *Real Network Simulator* pada tahun 1989 di University of California Berkeley dan USC ISI sebagai bagian dari proyek *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* , *Wide Area Network* (WAN), dan telah berkembang selama beberapa tahun belakangan untuk memasukkan

jaringan nirkabel (*wireless*) dan juga jaringan *ad hoc*. NS-2 memiliki beberapa fitur kelebihan yang dapat dimanfaatkan dalam pemodelan dan pengujian VANET [5].

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 untuk mengukur performa *routing* protokol AODV yang dimodifikasi.

### 2.4.1 Instalasi

NS-2 membutuhkan beberapa *package* yang harus sudah terinstall sebelum memulai instalasi NS-2. Untuk menginstall *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 menginstall *dependency* NS-2

Setelah menginstall *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*

### 2.4.2 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 2.2

**Tabel 2.2** Detail Penjelasan Trace File AODV

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

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

## 2.5 OpenStreetMap

OpenStreetMap (OSM) adalah sebuah proyek kolaboratif berbasis web untuk membuat dan membangun peta geografis seluruh dunia yang gratis dan terbuka, dibangun sepenuhnya oleh sukarelawan dengan melakukan survei dan mengumpulkan data menggunakan perangkat GPS, fotografi udara, yang dimanfaatkan untuk beragam kebutuhan termasuk navigasi. 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 didistribusikan kembali.

OSM dapat menjadi jawaban di banyak tempat seperti ini, baik itu pengembangan ekonomi, tata kota, kontinjensi bencana, maupun untuk berbagai tujuan lainnya [6].

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.

## 2.6 *Java OpenStreetMap Editor (JOSM)*

*Java OpenStreetMap Editor (JOSM)* adalah aplikasi desktop berbasis java dan dapat dioperasikan pada system operasi seperti Windows, MacOS, dan Linux. JOSM adalah alat penyunting data yang didapatkan dari OpenStreetMap [7].

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.

## 2.7 *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 [8].

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.

## 2.8 AWK

AWK adalah bahasa pemrograman yang digunakan untuk melakukan *text processing* dan ekstraksi data [9]. 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* (E2E), dan *Routing Overhead* (RO) dari *trace file* NS2

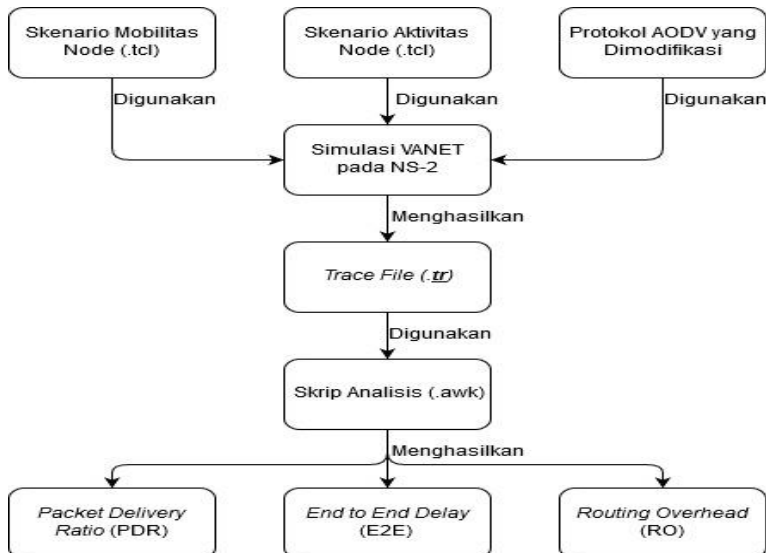
*(Halaman ini sengaja dikosongkan)*

## 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 dilakukan modifikasi pada *routing protocol* AODV dengan mengimplementasikan konsep dqAODV yang akan disimulasikan pada NS-2. Diagram dari rancangan simulasi dari AODV asli dan AODV modifikasi dapat dilihat pada Gambar 3.1.



**Gambar 3.1** Diagram Alur Rancangan Simulasi

Modifikasi akan diawali dengan broadcast paket RREQ dari *node* sumber. Setiap *node* yang menerima paket RREQ dari *node* tetangga dimana paket tersebut sudah didapat dari *node* lain, maka *node* tersebut akan menyimpan IP dari *node* pengirim kedalam dqAODV (antrian). Lalu selama proses *unicast* RREP dilakukan, apabila terjadi kerusakan pada *link*, *node* yang melakukan *unicast* akan mengambil salah satu IP *node* dari dalam fungsi dqAODV yang telah dibuat sebelumnya dan melakukan *re-unicast* paket RREP.

Modifikasi yang telah dilakukan akan disimulasikan pada NS-2 dengan peta berbentuk *grid* dan peta *real* pada lingkungan lalu lintas di kota Surabaya. Pembuatan kedua peta tersebut menggunakan bantuan *tools* SUMO. Simulasi tersebut akan memberikan hasil *trace file* yang kemudian dianalisis menggunakan skrip AWK untuk mendapatkan *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2E), dan *Routing Overhead* (RO). 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 <i>Ad hoc On-demand Distance Vector</i> . Protokol yang digunakan pada Tugas Akhir ini.
2	PDR	<i>Packet Delivery Ratio</i> . Salah satu metrik analisis yang diukur. Berupa rasio jumlah pengiriman paket yang terkirim.
3	E2E	<i>Average End-to-End Delay</i> . Jeda waktu yang diukur saat paket terkirim.
4	RO	<i>Routing Overhead</i> . Jumlah <i>control packet</i> yang terkirim

No.	Istilah	Penjelasan
5	RREQ	<i>Route Request</i> . Paket <i>request</i> pada AODV yang dikirim untuk mendapatkan rute.
6	RREP	<i>Route Reply</i> . Paket <i>reply</i> pada AODV yang dikirim ke <i>node</i> sumber melalui rute yang sudah terbuat.

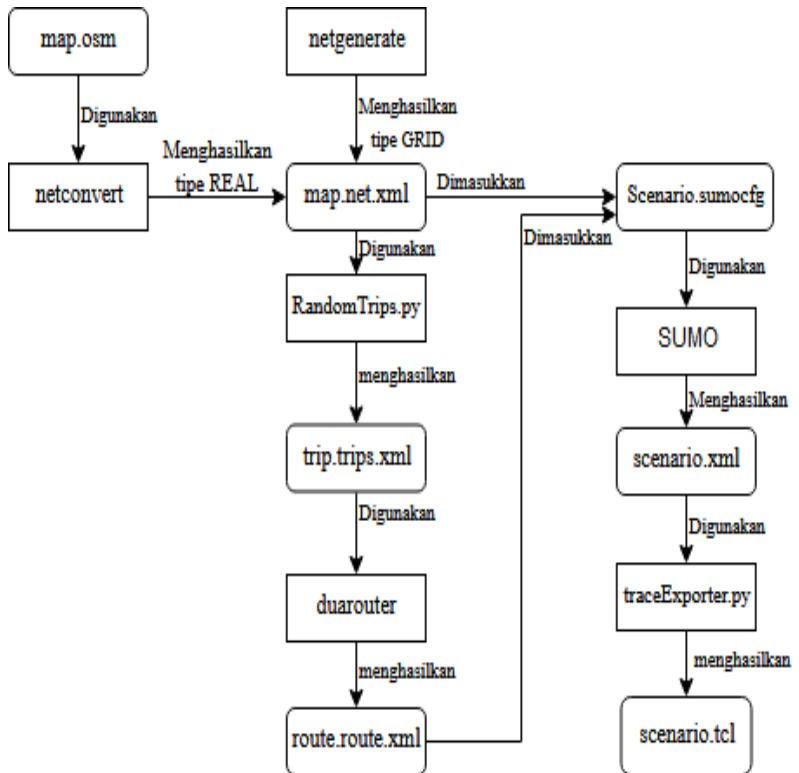
### 3.2 Perancangan Skenario Mobilitas

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

#### 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 700 m x 700 m. Dengan 4 titik persimpangan, maka akan didapatkan 9 petak dan panjang tiap petak adalah 200 m.



**Gambar 3.2** Alur perancangan skenario *Grid* dan *Real*

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 berkektensi .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* yaitu pada *header* RREQ yang akan dijelaskan nanti pada bab 3.3.1. *Node* sumber akan melakukan

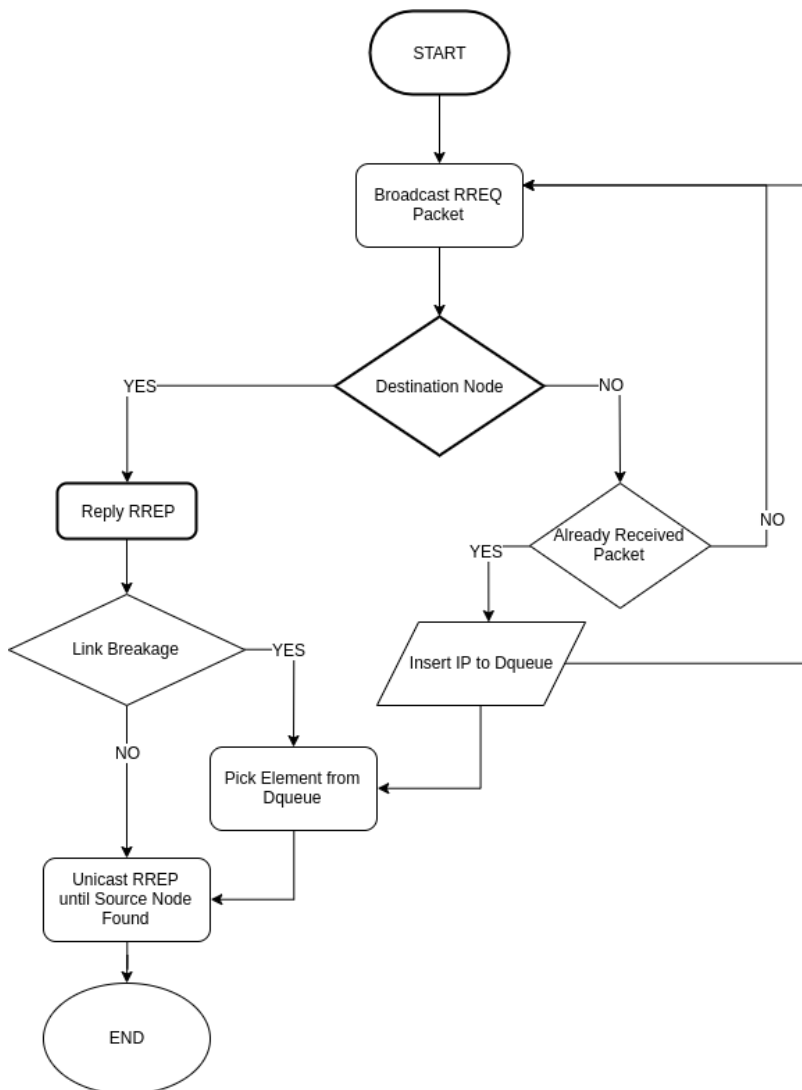
*broadcast* RREQ ke seluruh *node* tetangga, apabila *intermediate node* telah menerima paket RREQ dari *node* lain, maka *node* tersebut akan memasukkan *Broadcast ID* dari *sender node* ke dalam dqAODV.

Selanjutnya *broadcast* RREQ dilanjutkan hingga *node* tujuan ditemukan. Lalu, *node* tujuan dan *intermediate node* akan melakukan *unicast* RREP hingga *node* sumber ditemukan. Pada saat *unicast* RREP, jika terjadi kerusakan *link*, maka akan diambil *node* tetangga terdekat yang berada dalam *queue*. Diagram rancangan modifikasi AODV dapat dilihat pada Gambar 3.3. Algoritma dqAODV dapat dilihat pada gambar 3.4.

### Gambar 3.3 Algoritma dqAODV

1. Until destination found continue broadcast RREP
2. If node n is destination,  
    Reply RREP packet
3. Else  
    If node (n-m) already receive the packet, Insert into dqAODV.  
    Else Intermediate node rebroadcast packet.
4. Until source node get RREP, continue unicast  
    If link breakage, pick element from dqAODV.  
    Else  
        If node source, then n ready to send data.  
        Otherwise unicast RREP.

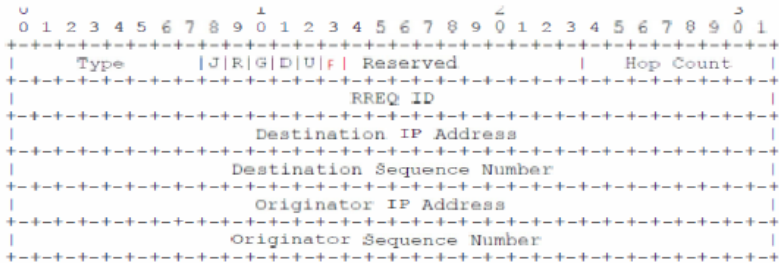




**Gambar 3.4** Alur penggunaan DqAODV dalam AODV

### 3.3.1 Perancangan Format *Route Request* (RREQ)

Pada *header* RREQ, ditambahkan sebuah *flag* baru yaitu “F”. Jika “F” di-*set* pada *node* tujuan, *Broadcast ID node* yang menuju ke *node* tujuan setelah *node* lain sudah sampai lebih dulu akan disimpan pada fungsi dqAODV. *Broadcast ID node* yang disimpan pada fungsi dqAODV akan berurutan masuk ke fungsi sesuai dengan jarak terdekat ke *node* tujuan. Saat *node* tujuan melakukan *unicast* paket RREP, jika terjadi kerusakan pada link, maka sesuai dengan *flag* “F” akan diambil *broadcast ID node* tetangga yang berdekatan dari dalam dqAODV. Pengambilan *broadcast ID* dari dqAODV menggunakan metode *First In, First Out* (FIFO) karena *broadcast ID* yang masuk dalam dqAODV akan berurutan sesuai dengan jarak terdekat ke *node* tujuan.



**Gambar 3.5** Modifikasi Paket Format RREQ

### 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 adalah fungsi *recvRequest* pada *file* aodv.cc. Pada saat simulasi NS-2 dijalankan, maka setiap *node* pada *routing protocol* AODV akan memeriksa apakah sudah menerima paket yang sama.

### 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.

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

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.

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

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.

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

Keterangan:

$Packet\ sent$  = Jumlah kontrol paket RREQ, RREP, dan RRER  
yang dikirimkan

$sentnum$  = Jumlah total control paket yang terkirim

## 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.

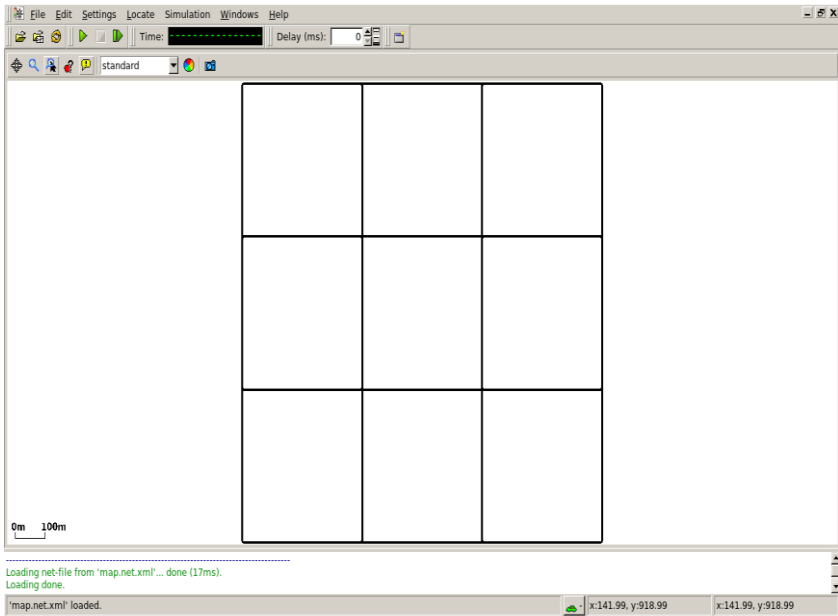
##### 4.1.1 Skenario *Grid*

Dalam mengimplementasikan skenario *grid*, SUMO menyediakan *tools* untuk membuat peta *grid* yaitu *netgenerate*. Pada Tugas Akhir ini, penulis membuat peta *grid* dengan luas 700 m x 700 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 700 m x 700 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 *digenerate* tidak akan melenceng dari jalur peta yang sudah *digenerate* menggunakan *tools* randomTrips. Selanjutnya untuk menjadikan peta dan pergerakan *node* yang telah *digenerate* 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 digunakn untuk melakukan konversi



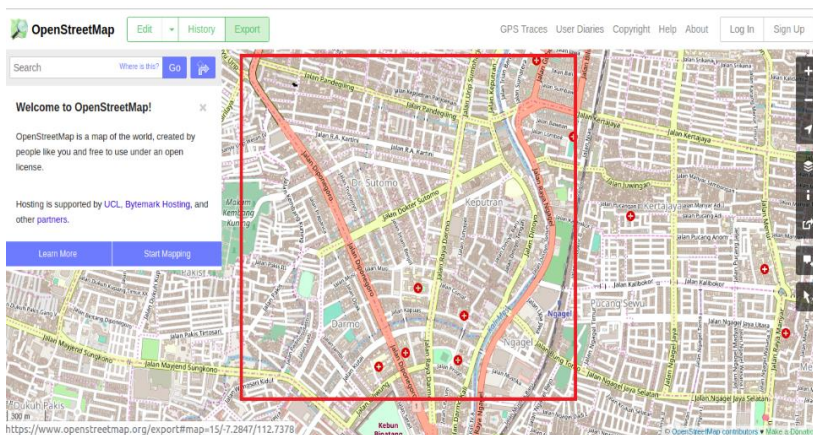
ini adalah traceExporter. Perintah untuk menggunakan traceExporter dapat dilihat pada Gambar 4.7.

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

**Gambar 4.7** Perintah traceExporter

## 4.1.2 Skenario *Real*

Dalam mengimplementasikan skenario *real*, langkah pertama adalah menentukan area yang akan dijadikan area simulasi. Pada Tugas Akhir ini penulis mengambil area jalan sekitar Jl. 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.



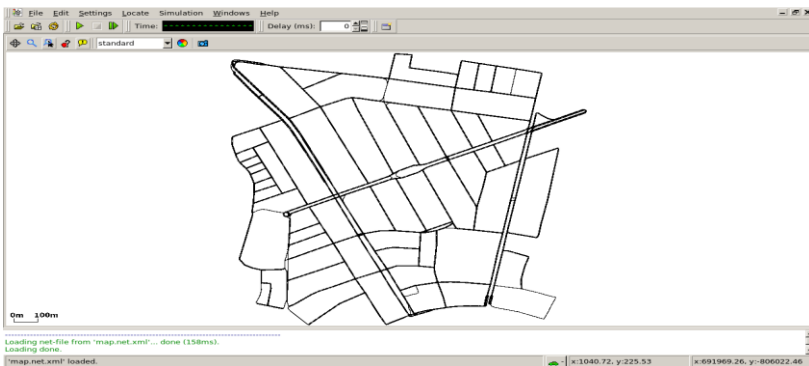
**Gambar 4.8** Ekspor Peta dari OpenStreetMap

*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 Meminimalisir Kegagalan RREP saat Mencari Rute Kembali

Pada Tugas Akhir ini dilakukan modifikasi pada *routing protocol AODV* agar dapat meminimalisir kegagalan RREP saat mencari rute kembali ke *node* sumber.

Implementasi modifikasi *routing protocol AODV* ini dibagi menjadi 2 bagian yaitu:

- Implementasi Pengubahan Format Paket RREQ
- Implementasi Metode *First in, First Out* (FIFO) pada dqAODV

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 mengimplementasikan metode dqAODV dan *file* aodv.h yang ada di dalam *folder* ns-2.35/aodv untuk mendaftarkan *flag* baru. Pada bagian ini penulis akan menjelaskan langkah – langkah dalam mengimplementasikan modifikasi *routing protocol AODV* untuk meminimalisir kegagalan RREP saat mencari rute kembali menuju *node* sumber.

### 4.2.1 Implementasi Pengubahan Format Paket RREQ

Langkah yang dilakukan untuk menerapkan metode dqAODV seperti yang telah dirancang pada subbab 3.3.1 adalah dengan cara menangkap menambahkan *flag* baru “F” pada header RREQ. *Flag* “F” akan menyimpan *broadcast ID* dari *node* yang masuk ke dalam dqAODV. Pada tugas akhir ini, penulis memanfaatkan fungsi *recvRequest* untuk menerapkan dqAODV pada *file* aodv.cc. Untuk potongan kode tersebut bisa dilihat pada Gambar 4.11.

```

1. if (id_lookup(rq->rq_src, rq->rq_bcast_id)){
2.     //DqAODV Code
3.     //Create Queue
4.     //Enqueue Broadcast ID
5.     Packet::free(p);
6.     rt_update(aodv_rt_entry *rt_queue,
               u_int32_t seqnum, u_int16_t metric,
               nsaddr_t nexthop, double expire_time);
7.     return;
8. }

```

**Gambar 4.11** Potongan Kode Modifikasi Fungsi *recvRequest()*

#### 4.2.2 Implementasi Metode *First In, First Out* (FIFO) pada dqAODV

Pada tahap selanjutnya menerapkan metode *First In, First Out* pada dqAODV agar pemilihan *supply node* pada saat terjadi kerusakan *link* sesuai dengan urutan masuk *node* ke dalam *list* dqAODV. *Node* yang pertama masuk ke dalam *list* dqAODV adalah *node* dengan jarak terdekat setelah *node* pengirim sebelumnya.

Fungsi dqAODV yang menggunakan metode FIFO ini ditambahkan pada kode sumber aodv.cc yang terletak pada ns2.35/aodv. Potongan kode untuk fungsi dqAODV dengan metode FIFO dapat dilihat pada lampiran A.6 Kode Skrip DqAODV.

### 4.3 Implementasi Simulasi pada NS-2

Implementasi simulasi VANETs diawali dengan pendeskripsian lingkungan simulasi pada sebuah *file tcl*. *File* ini berisikan konfigurasi setiap *node* dan langkah-langkah yang dilakukan selama simulasi. Potongan konfigurasi lingkungan simulasi dapat dilihat pada Gambar 4.12.

```
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

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 digenerate oleh SUMO. Kode implementasi pada NS-2 dapat dilihat pada lampiran A.1 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.2 Kode Konfigurasi *Traffic*

#### 4.4 Implementasi Metrik Analisis

Simulasi yang telah dijalankan oleh NS-2 menghasilkan sebuah *trace file* yang berisikan data mengenai apa saja yang terjadi selama simulasi dalam bentuk *file* berekstensi .tr. Dari data *trace file* tersebut, dapat dilakukan analisis performa *routing protocol* dengan mengukur beberapa metrik. Pada Tugas Akhir ini, metrik yang akan dianalisis adalah PDR, E2E, dan RO.

#### 4.4.1 Implementasi *Packet Delivery Ratio* (PDR)

Pada subbab 2.4.2 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.3 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 menunjukkan *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 menunjukkan 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`.

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

Skrip awk untuk menghitung E2E dapat dilihat pada lampiran A.4 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.`



#### 4.4.3 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. Perhitungan RO telah dijelaskan pada persamaan 3.3. Skrip AWK untuk menghitung RO dapat dilihat pada lampiran A.3 Kode Skrip AWK *Packet Delivery Ratio*

```
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

#### A.4 Kode Skrip AWK Rata-Rata *End-to-End Delay*

#### A.5 Kode Skrip AWK *Routing Overhead*

. *Pseudocode* untuk menghitung RO dapat dilihat pada Gambar 4.16.

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

*(Halaman ini sengaja dikosongkan)*

## **BAB V**

### **UJICOBADA 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

<b>Komponen</b>	<b>Spesifikasi</b>
<b>CPU</b>	Intel(R) Core™ i5-6200HQ CPU @ 2.30GHz
<b>Sistem Operasi</b>	Ubuntu 18.04.2 LTS
<b>Linux Kernel</b>	Linux kernel 4.4
<b>Memori</b>	4.0 GB

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

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

Parameter lingkungan uji coba yang digunakan pada NS-2 dapat 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, dan RO menggunakan kode yang terdapat pada lampiran A.3 Kode Skrip AWK *Packet Delivery Ratio*, A.4 Kode Skrip AWK *Rata-Rata End-to-End Delay*, dan A.3 Kode Skrip AWK *Packet Delivery Ratio*

#### A.4 Kode Skrip AWK Rata-Rata *End-to-End Delay*

#### A.5 Kode Skrip AWK *Routing Overhead*

**Tabel 5.2** Lingkungan Uji Coba

No.	Parameter	Spesifikasi
1	Network simulator	NS-2.35
2	<i>Routing protocol</i>	AODV dan dqAODV
3	Waktu simulasi	200 detik
4	Area simulasi	700 m x 700 m (Grid) 1500 m x 1500 m (Real)
5	Jumlah <i>Node</i>	50, 100, 150, 200
6	Radius transmisi	400m
7	Kecepatan maksimum	20 m/s
8	Paket Interval	1 paket/detik
9	Protokol MAC	IEEE 802.11p
10	Model Propagasi	<i>Two-ray ground</i>

## 5.2 Hasil Uji Coba

Hasil uji coba menggunakan *node* sumber dan *node* tujuan yang diletakkan secara statis. Hasil dapat dilihat sebagai berikut:

### 5.2.1 Hasil Uji Coba Skenario *Grid*

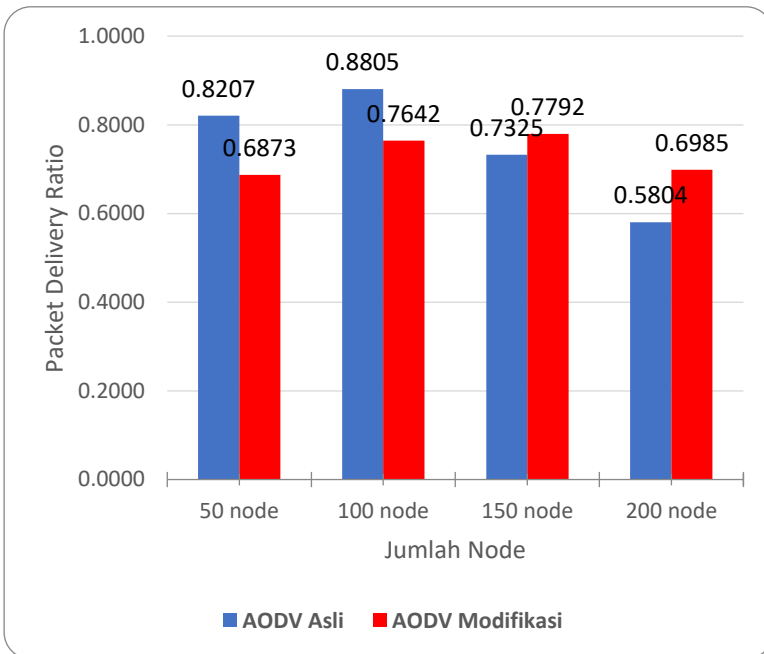
Pengujian pada skenario *grid* digunakan untuk melihat perbandingan PDR, E2E, dan RO antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi dalam pemilihan *node* yang dapat menerima paket *route request*.

Pengambilan data uji pada skenario *grid* dilakukan sebanyak 10 kali dengan skenario mobilitas *random* pada peta *grid* dengan luas

area 700 m x 700m m dan *node* sebanyak 50 untuk lingkungan jarang, 100 dan 150 *node* untuk lingkungan yang sedang, dan 200 *node* untuk lingkungan padat dilakukan pada kecepatan standar yaitu 20 m/s. Hasil analisis dapat dilihat pada Tabel 5.3, Tabel 5.4, dan Tabel 5.5.

**Tabel 5.3** Hasil Rata - Rata PDR Skenario *Grid*

	50 node	100 node	150 node	200 node
AODV Asli	0.8207	0.8805	0.7325	0.5804
AODV Modifikasi	0.6873	0.7642	0.7792	0.6985
Perubahan	-0.1334	-0.1163	0.0467	0.1181
Kenaikan	-16.25%	-13.21%	6.38%	20.36%
Rata-rata	-0.68%			



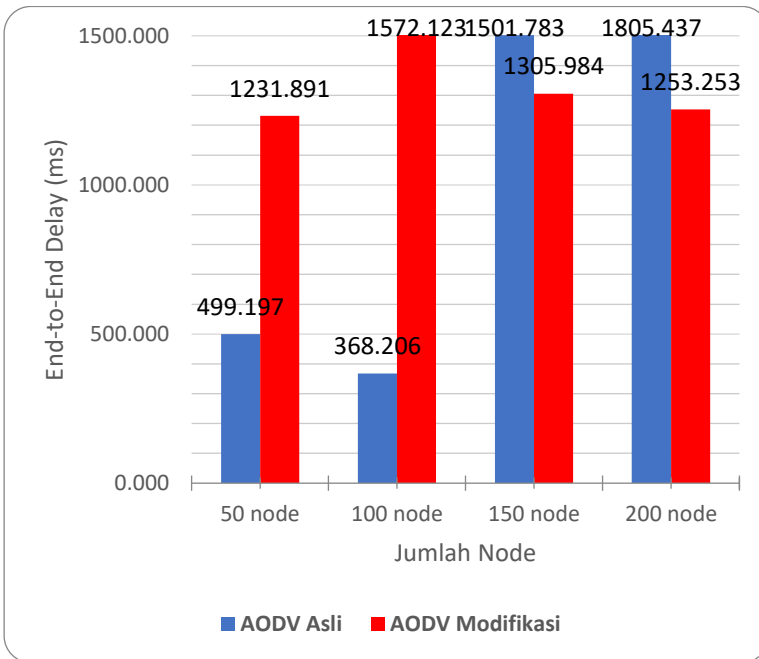
**Gambar 5.1** Grafik Packet Delivery Ratio Skenario *Grid*

Berdasarkan grafik pada Gambar 5.1 dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi dan juga *routing protocol* AODV asli mengalami kenaikan pada jumlah node yang besar. Pada lingkungan yang jarang dengan jumlah 50 node, menghasilkan perbedaan selisih PDR sebesar 0.1334, atau turun menjadi sekitar 16.25% dimana *routing protocol* AODV yang telah dimodifikasi belum unggul dalam hal PDR tersebut. Pada lingkungan yang sedang dengan jumlah 100 node, menghasilkan perbedaan selisih PDR sebesar 0.1163, atau turun menjadi sekitar 13.21% dimana *routing protocol* AODV yang telah dimodifikasi belum unggul dalam hal PDR tersebut dari AODV asli. Pada lingkungan yang sedang dengan jumlah 150 node, menghasilkan perbedaan selisih PDR sebesar 0.0467, atau naik menjadi sekitar 6.38% 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 0.1181, atau naik menjadi sekitar 20.36% dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal PDR tersebut.

Pada Gambar 5.1 juga dapat dilihat, walaupun AODV modifikasi mengalami penurunan total dari AODV asli sebesar 0,68%, tetapi AODV modifikasi lebih stabil dibandingkan AODV asli yang semakin menurun ketika jumlah node semakin padat.

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.

	50 node	100 node	150 node	200 node
AODV Asli	499.197	368.206	1501.783	1805.437
AODV Modifikasi	1231.891	1572.123	1305.984	1253.253
Perubahan	-732.694	-1203.917	195.799	552.184
Penurunan	-146.77%	-326.97%	13.04%	30.58%
Rata-rata	-			

**Tabel 5.4** Hasil Rata - Rata E2E Skenario Grid**Gambar 5.2** Grafik End-to-end Delay Skenario *Grid*

Berdasarkan grafik pada Gambar 5.2 pada lingkungan yang jarang dengan jumlah 50 node, terjadi perbedaan selisih *end-to-end delay* sebesar 732.694 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami kenaikan sebesar 146.77%, dimana *routing protocol* AODV yang telah dimodifikasi belum unggul dalam hal *end-to-end delay* tersebut. Sedangkan pada lingkungan sedang dengan jumlah 100 node, terjadi perbedaan selisih *end-to-end delay* sebesar 1,203.917 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami kenaikan sebesar 326.97%, dimana *routing protocol* AODV yang dimodifikasi belum unggul dalam hal *end-to-end delay* tersebut. Pada pada lingkungan sedang

dengan jumlah 150 node, terjadi perbedaan selisih *end-to-end delay* sebesar 195.799 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 13.04%, 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 552.184 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 30.58%, dimana *routing protocol* AODV yang telah dimodifikasi jauh lebih unggul dalam hal *end-to-end delay* tersebut.

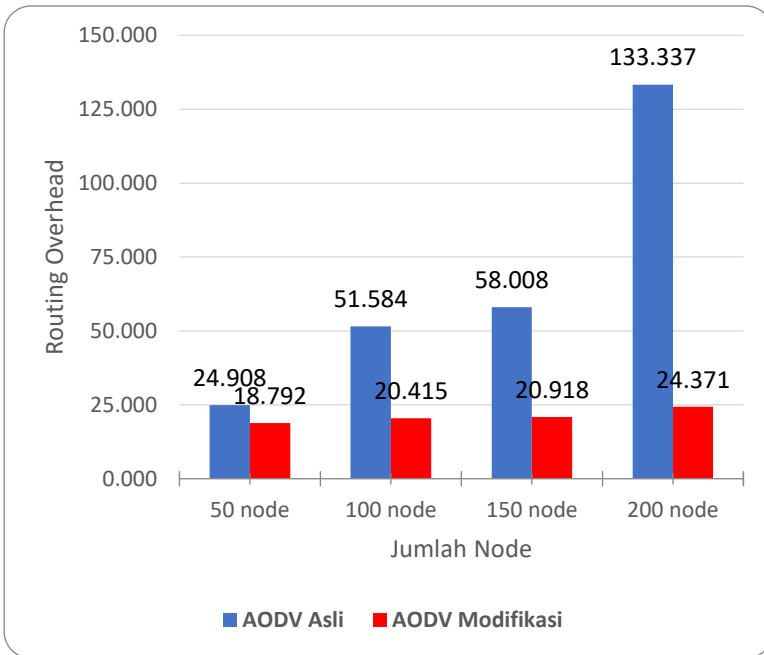
Jika ketiga lingkungan tersebut dibandingkan, AODV yang telah dimodifikasi lebih unggul pada lingkungan yang lebih padat dan lebih stabil dalam hal *end-to-end delay*. Dapat dilihat bahwa dengan jumlah node jarang, sedang dan padat menghasilkan *end-to-end delay* yang lebih baik dan dapat dilihat pula bahwa 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. Hasil rata – rata E2E tidak dapat dianalisis karena terjadi fluktuasi dan tidak stabil. Hal ini dikarenakan waktu *delay* tergantung dari rata – rata waktu paket yang terkirim. Semakin banyak paket yang terkirim, maka semakin beragam *delay*nya.

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 RO Skenario Grid

	50 node	100 node	150 node	200 node
AODV Asli	24.908	51.584	58.008	133.337
AODV Modifikasi	18.792	20.415	20.918	24.371
Perubahan	6.116	31.169	37.090	108.966
Penurunan	24.55%	60.42%	63.94%	81.72%
Rata-rata	57.66%			





**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 perubahan yang signifikan. Pada lingkungan yang jarang dengan jumlah 50 node, menghasilkan perbedaan selisih *routing overhead* sebesar 6.116 atau mengalami penurunan sebesar 24.55%, dimana *routing protocol* AODV modifikasi unggul dalam hal *routing overhead* tersebut karena menghasilkan *routing overhead* yang lebih rendah dari routing AODV asli. Pada lingkungan yang sedang dengan jumlah 100 node, menghasilkan perbedaan selisih *routing overhead* sebesar 31.169 atau mengalami penurunan sebesar 60.42%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam mereduksi *routing*

*overhead* dari AODV asli. Pada lingkungan yang sedang dengan jumlah 150 node, menghasilkan perbedaan selisih *routing overhead* sebesar 37.090 atau mengalami penurunan sebesar 63.94%, dimana *routing protocol* AODV yang telah dimodifikasi mengalami keunggulan dalam hal *routing overhead* tersebut dari AODV asli. Pada lingkungan yang padat dengan jumlah 200 node, menghasilkan perbedaan selisih *routing overhead* sebesar 108.966 atau mengalami penurunan sebesar 81.72%, 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 57.66%. 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 modifikasi lebih unggul di semua lingkungan dengan jumlah selisih *routing overhead* yang cukup signifikan. Semakin rendah nilai RO maka *packet drop* semakin rendah dikarenakan pencarian rute lebih cepat atau optimal dilakukan seiring bertambahnya kepadatan node.

### 5.2.2 Hasil Uji Coba Skenario Real

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

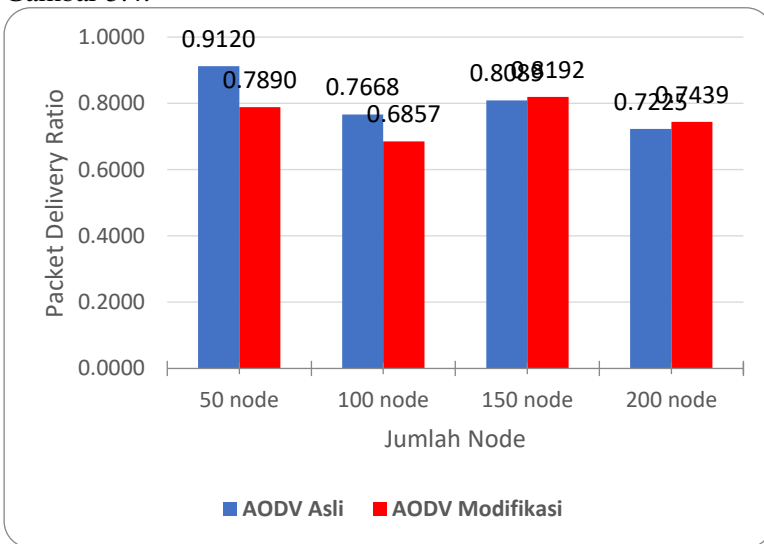
Pengambilan data uji PDR, E2E, dan RO pada skenario *real* dilakukan sebanyak 10 kali dengan skenario mobilitas *random* pada peta *real* dengan luas area 1500 m x 1500 m dengan *range transmisi*

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 hasil analisis Skenario dengan peta *Real* dengan *node* 50, 100, 150 dan 200 dapat dilihat pada Tabel 5.6, Tabel 5.7, dan Tabel 5.8.

**Tabel 5.6** Hasil Rata - Rata PDR pada Skenario Real

	50 node	100 node	150 node	200 node
AODV Asli	0.9120	0.7668	0.8089	0.7225
AODV Modifikasi	0.7890	0.6857	0.8192	0.7439
Perubahan	-0.1230	-0.0811	0.0103	0.0214
Kenaikan	-13.49%	-10.57%	1.27%	2.96%
Rata-rata	-4.96%			

Dari data pada Tabel 5.9, dibuat grafik yang merepresentasikan hasil perhitungan PDR yang ditunjukkan pada Gambar 5.4.



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

Berdasarkan grafik pada Gambar 5.5 dapat dilihat bahwa pada lingkungan yang jarang dengan jumlah 50 node, menghasilkan perbedaan selisih PDR sebesar 0.1230, atau turun menjadi sekitar 13.49% dimana *routing protocol* AODV yang telah dimodifikasi belum unggul dalam hal PDR tersebut. Pada lingkungan yang sedang dengan jumlah 100 node, menghasilkan perbedaan selisih PDR sebesar 0.0811, atau turun menjadi sekitar 10.57% dimana *routing protocol* AODV yang telah dimodifikasi belum unggul dalam hal PDR tersebut dari AODV asli. Pada lingkungan yang sedang dengan jumlah 150 node, menghasilkan perbedaan selisih PDR sebesar 0.0103, atau turun menjadi sekitar 1.27% 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 0.0214, atau turun menjadi sekitar 2.96% dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal PDR tersebut.

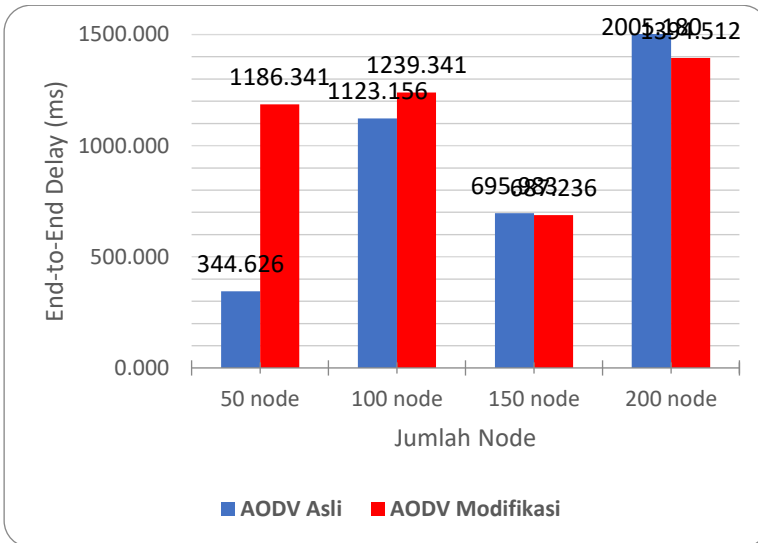
Pada Gambar 5.4 juga dapat dilihat, walaupun AODV modifikasi mengalami penurunan rata-rata sebesar 4.96% tetapi AODV modifikasi lebih stabil dibandingkan AODV asli yang semakin naik ketika jumlah node semakin padat.

Hasil pengambilan data rata-rata untuk *end-to-end delay* (E2E) pada skenario *real* dengan jumlah *node* 50, 100, 150, dan 200 dapat dilihat pada Tabel 5.10.

**Tabel 5.7** Hasil Rata -Rata E2E pada Skenario Real

	50 node	100 node	150 node	200 node
AODV Asli	344.626	1123.156	695.983	2005.180
AODV Modifikasi	1186.341	1239.341	687.236	1394.512
Perubahan	-841.716	-116.185	8.747	610.668
Penurunan	-244.24%	-10.34%	1.26%	30.45%
Rata-rata	7.12%			

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



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

Berdasarkan grafik pada Gambar 5.6 dapat dilihat bahwa rata-rata *end-to-end delay* antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami perubahan yang fluktuatif. Pada lingkungan yang jarang dengan jumlah 50 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 841.716 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami kenaikan sebesar 244.24%, dimana *routing protocol* AODV asli lebih unggul dari AODV yang telah dimodifikasi dalam hal *end-to-end delay* tersebut. Sedangkan pada lingkungan sedang dengan jumlah 100 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 116.185 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 10.34%, dimana *routing protocol* AODV asli lebih unggul dalam hal *end-to-end delay*. Pada

lingkungan sedang dengan jumlah 150 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 8.747 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 1.26%, 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 610.668 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 30.45%, dimana *routing protocol* AODV yang telah dimodifikasi jauh lebih unggul dalam hal *end-to-end delay* tersebut.

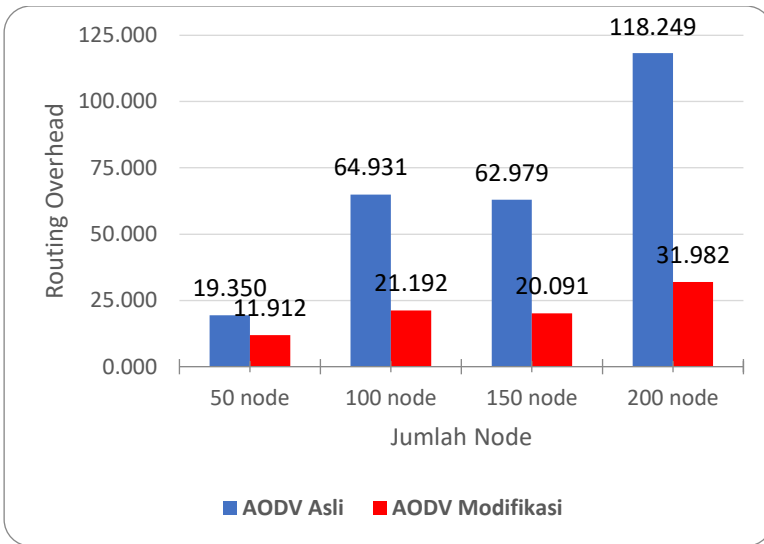
Jika ketiga lingkungan tersebut dibandingkan, memang pada lingkungan yang jarang, AODV asli lebih unggul dari AODV yang telah dimodifikasi dalam hal *end-to-end delay*. Namun, dapat dilihat juga bahwa dengan jumlah *node* sedang dan padat menghasilkan *end-to-end delay* yang lebih baik dan dapat dilihat pula bahwa 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 menyediakan node cadangan sehingga paket-paket akan lebih sedikit mengalami adanya delay dalam pengiriman. Hasil rata – rata *delay* adalah 7.12%.

Untuk hasil pengambilan data *routing overhead* pada skenario *real* 50 *node*, 100 *node*, 150 *node* dan 200 *node* dapat dilihat pada Tabel 5.11.

**Tabel 5.8** Hasil Rata - Rata RO Skenario Real

	50 node	100 node	150 node	200 node
AODV Asli	19.350	64.931	62.979	118.249
AODV Modifikasi	11.912	21.192	20.091	31.982
Perubahan	7.438	43.739	42.888	86.267
Penurunan	38.44%	67.36%	68.10%	72.95%
Rata-rata	61.71%			

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



**Gambar 5.6** Grafik *Routing Overhead* Skenario Real

Berdasarkan grafik pada Gambar 5.7 dapat dilihat bahwa rata-rata *routing overhead* antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami penurunan yang signifikan. Pada lingkungan yang jarang dengan jumlah 50 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 7.438 atau mengalami penurunan sebesar 38.44%, dimana *routing protocol* AODV modifikasi unggul dalam hal *routing overhead* tersebut karena menghasilkan *routing overhead* yang lebih rendah dari *routing* AODV yang asli. Pada lingkungan yang sedang dengan jumlah 100 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 43.739 atau mengalami penurunan sebesar 67.36%, dimana *routing protocol* AODV yang telah dimodifikasi mengalami keunggulan

dalam hal routing overhead tersebut dari AODV asli. Pada lingkungan yang sedang dengan jumlah 150 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 42.888 atau mengalami penurunan sebesar 68.10%, dimana *routing protocol* AODV yang telah dimodifikasi mengalami keunggulan dalam hal routing overhead tersebut dari AODV asli. Pada lingkungan yang padat dengan jumlah 200 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 86.267 atau mengalami penurunan sebesar 72.95%, dimana *routing protocol* AODV yang telah dimodifikasi juga lebih unggul daripada AODV yang asli dalam hal routing overhead tersebut.

Dapat dilihat rata-rata penurunan yang terjadi untuk *routing overhead* adalah sebesar 61.71%. Jika 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 modifkikasi lebih unggul di semua lingkungan dengan jumlah selisih *routing overhead* yang cukup signifikan.



## **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. Dengan menggunakan metode DqAODV, AODV modifikasi berhasil meminimalisir kegagalan saat RREP mencari rute kembali dilihat dari rata-rata *Packet Delivery Ratio* turun menjadi 0.68% pada skenario *grid* dan turun menjadi 4.96% pada skenario *real* namun stabil seiring dengan bertambah padatnya node.
2. Dengan penggunaan DqAODV pada lingkungan VANET berhasil meningkatkan performa AODV dilihat rata-rata *Routing Overhead* yang turun menjadi 57.66% pada skenario *grid* dan turun menjadi 61.71% pada skenario *real*.

#### **6.2 Saran**

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

1. Menambahkan aspek lain pada dqAODV seperti mengurangi terjadinya kerusakan link agar dqAODV dapat bekerja lebih cepat.
2. Lebih banyak uji coba yang dilakukan untuk mendapatkan hasil yang lebih akurat.

***(Halaman ini sengaja dikosongkan)***

## DAFTAR PUSTAKA

- [1] "VANET - Vehicle Ad hoc Network," [Online]. Available: [http://comp.ist.utl.pt/~rmr/WSN/CaseStudies2007-no/WSN\\_Transportation/](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] R. Brendha dan V. S. J. Prakash, "A Survey on Routing Protocols for Vehicular Ad hoc Networks," IEEE, Coimbatore, 2017.
- [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. Meenaghan 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] I. Chlamtac, M. Conti, J. Liu, "Mobile ad hoc networking:imperatives and challenges, Ad Hoc Networks",pp. 13-64, 2003.
- [11] Y. Feng , B. Zhang, S. Chai, L. Cui dan Q. Li, "An Optimized AODV Protocol based on Clustering for WSNs," *6th International Conference on Computer Science and Network Technology (ICCSNT)*, 2017.
- [12] R. G. Engoulou, M. Bellaiche, S. Pierre dan A. Quintero, "VANET Security Surveys," *Computer Communication*, vol. 44, p. 2, 2014.
- [13] X. Shen, Y. Wu, Z. Xu dan X. Lind, "AODV-PNT: An improved version of AODV routing protocol with predicting node trend in VANET," dalam *The 7th IEEE/International Conference on Advanced Infocomm Technology*, Fuzhou, China , 2014.

## LAMPIRAN

### A.1 Kode Skenario NS-2

```
set val(chan) Channel/WirelessChannel;
set val(prop) Propagation/TwoRayGround;
set val(netif) Phy/WirelessPhy;
set val(mac) Mac/802_11;
set val(ifq) Queue/DropTail/PriQueue;
set val(ll) LL;
set val(ant) Antenna/OmniAntenna;
set 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) "scenarioreal.tcl";

set ns_ [new Simulator]

# setup topography object

set topo [new Topography]

# create trace object for ns and nam

set tracefd [open scenariol.tr w]
set namtrace [open scenariol.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  CStresh_ 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.2 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.3 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.4 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.5 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.7 Kode Skrip DqAODV

```

class QNode {
public:
    int key;
    QNode* next;
};
class Queue {
public:
    QNode *front, *rear;
};
QNode* newNode(int rq_src)
{
    QNode* temp = new QNode();
    temp->key = rq_src;
    temp->next = NULL;
    return temp;
}
Queue* createQueue()
{
    Queue* q = new Queue();
    q->front = q->rear = NULL;
    return q;
}
void enqueue(Queue* q, int rq_src)
{
    QNode* temp = newNode(rq_src);
    if (q->rear == NULL) {
        q->front = q->rear = temp;
        return;
    }
    q->rear->next = temp;
    q->rear = temp;
}

```

```
QNode* deQueue(Queue* q)
{
    // If queue is empty, return NULL.
    if (q->front == NULL)
        return NULL;
    QNode* temp = q->front;
    delete(temp);
    q->front = q->front->next;
    if (q->front == NULL)
        q->rear = NULL;
    return temp;
}
```

x

***(Halaman ini sengaja dikosongkan)***

## BIODATA PENULIS



**Muhammad Budiana Eka Faruqi**, lahir di Dumai, 7 Maret 1998. Penulis adalah anak pertama dari empat bersaudara. Penulis menempuh pendidikan sekolah dasar di SDN 005 Teluk Binjai Kota Dumai, lalu melanjutkan pendidikan sekolah menengah pertama di SMP Binaan Khusus Kota Dumai dan penulis menempuh pendidikan menengah atas di SMA Negeri 2 Dumai. Selanjutnya penulis

melanjutkan pendidikan sarjana di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Arsitektur dan Jaringan Komputer (AJK). Sebagai mahasiswa, penulis berperan aktif dalam beberapa organisasi kampus seperti Ikatan Mahasiswa Minang Surabaya (IMAMI Surabaya), dan pernah mengisi acara pada ITS Expo 2016. Selain itu, penulis juga menjadi staf Dana dan Usaha SCHEMATICS 2016 dan SCHEMATICS 2017. Penulis pernah melakukan kerja praktik di PT. Pertamina (Persero) Kota Dumai pada Juni – Agustus 2018 dan membuat aplikasi berbasis *web Phonebook* Pertamina Dumai. Penulis dapat dihubungi melalui nomor *handphone*: 082140907201 atau *email*: [budifaruci@gmail.com](mailto:budifaruci@gmail.com).