



TUGAS AKHIR - IF184802

# IMPLEMENTASI METODE *ENHANCED-ANT* PADA *MULTIPATH* AODV DI VANETS

SATRIA CHANDRA YUDHA WIBOWO  
NRP 05111540000127

Dosen Pembimbing I  
Ir. Muchammad Husni, M.Kom.

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

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









**TUGAS AKHIR - IF184802**

# **IMPLEMENTASI METODE *ENHANCED-ANT* PADA *MULTIPATH* AODV DI VANETS**

**SATRIA CHANDRA YUDHA WIBOWO**  
**NRP 05111540000127**

**Dosen Pembimbing I**  
**Ir. Muchammad Husni, M.Kom.**

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

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

*(Halaman ini sengaja dikosongkan)*



**UNDERGRADUATE THESES - IF184802**

# **IMPLEMENTATION OF *ENHANCED-ANT* METHOD ON *MULTIPATH* AODV IN VANETS**

**SATRIA CHANDRA YUDHA WIBOWO**  
**NRP 05111440000127**

**First Advisor**

**Ir. Muchammad Husni, M.Kom.**

**Second Advisor**

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

**Department of Informatics**

**Faculty of Information Technology and Communication**

**Sepuluh Nopember Institute of Technology**

**Surabaya 2019**

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



## LEMBAR PENGESAHAN

### IMPLEMENTASI METODE *ENHANCED-ANT* PADA MULTIPATH AODV DI VANETS

#### TUGAS AKHIR

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

Oleh:

**SATRIA CHANDRA YUDHA WIBOWO**

**NRP: 05111540000127**

Disetujui oleh Pembimbing Tugas Akhir:

1. Ir. Muchammad Husni, M.Kom.  
(NIP. 196002211984031001)

(Pembimbing 1)

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

(Pembimbing 2)

**SURABAYA  
JUNI, 2019**

*(Halaman ini sengaja dikosongkan)*

## IMPLEMENTASI METODE *ENHANCED-ANT* PADA MULTIPATH AODV DI VANETS

Nama Mahasiswa : SATRIA CHANDRA YUDHA  
WIBOWO  
NRP : 05111540000127  
Departemen : Informatika FTIK-ITS  
Dosen Pembimbing 1 : Ir. Muchammad Husni, M. Kom.  
Dosen Pembimbing 2 : Dr. Eng. Radityo Anggoro, S.Kom.,  
M.Sc.

### Abstrak

*Vehicular Ad hoc Networks* (VANETs) adalah pengembangan dari *Mobile Ad hoc Network* (MANET). Dalam VANETs, *node* memiliki ciri khas dengan mobilitas yang sangat tinggi dan terbatas pada pola pergerakannya berdasarkan rute yang dibentuk. Ada banyak *routing protocol* yang dapat diimplementasikan pada VANETs, salah satunya adalah *Ad hoc On demand Distance Vector* (AODV).

AODV adalah *routing protocol* yang termasuk dalam tipe *reactive*, yaitu protokol yang hanya akan membuat rute ketika *node* sumber membutuhkan. AODV memiliki dua fase, yaitu *route discovery* dan *route maintenance*. *Route discovery* digunakan untuk mencari rute dan meneruskan paket melalui rute yang telah dibentuk. Fase ini 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 fase *route discovery* protokol AODV asli, penentuan rute terbaik dilakukan berdasarkan paket RREQ yang datang pertama kali. Hal ini menyebabkan paket RREQ yang lebih baik namun datang terlambat akan tereliminasi dan menyebabkan kualitas pengiriman paket menurun.

Pada Tugas Akhir ini mengusulkan sebuah modifikasi dalam *route discovery* yang bernama *Enhanced-Ant Multipath*. Metode ini akan menyeleksi rute yang ditemukan menggunakan 4 parameter nilai yang mempengaruhi kualitas pengiriman paket, yaitu *received signal strength*, *congestion node*, *residual energy node* dan *length of node*. Ke-empat parameter ini akan dikalkulasikan berupa *pheromone count*. *Pheromone count* akan bertugas sebagai penentu rute pertama yang digunakan node tujuan untuk mengirim RREP. Metode ini akan diimplementasikan dalam cara pengiriman *multipath*. Jika dalam AODV biasa RREQ hanya diterima sekali dan sisanya akan di-*drop*, dalam metode ini beberapa RREQ akan diterima semua dan RREP dikirim melalui semua rute yang tersedia.

Hal ini dilakukan agar dapat meningkatkan kualitas pengiriman paket AODV dengan cara memilih node-node yang telah dipilih melalui empat parameter nilai yang mempengaruhi kualitas pengiriman paket. Dari hasil uji coba, AODV yang dimodifikasi pada skenario grid meningkatkan nilai rata-rata *Packet Delivery Ratio* (PDR) sebesar 12.20%, mengurangi *Delay* sebesar 21.64%, mengurangi *Routing Overhead* sebesar 15.94%, dan meningkatkan nilai *Node Survive Percentage* hingga 1.45%. sedangkan pada skenario *real* meningkatkan nilai rata-rata *Packet Delivering Ratio* (PDR) hingga 16.69%, mengurangi *Delay* sebesar 48.14%, mengurangi *Routing Overhead* (RO) sebesar 20.35% dan meningkatkan *Node Survive Percentage* sebesar 37.53%.

**Kata kunci:** *VANETs, AODV, Ant Colony Optimization Enhanced-Ant, Multipath AODV*

## **IMPLEMENTATION OF ENHANCED-ANT METHOD ON MULTIPATH AODV IN VANETS**

**Student's Name** : SATRIA CHANDRA YUDHA  
WIBOWO  
**Student's ID** : 05111540000127  
**Department** : Informatics – FTIK ITS  
**First Advisor** : Ir. Muchammad Husni, M. Kom.  
**Second Advisor** : Dr. Eng. Radityo Anggoro, S.Kom.,  
M.Sc.

### ***Abstract***

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

*AODV is an example of reactive classification, a protocol that will only form a route when the source node needs it. AODV have 2 phase which are route discovery and route maintenance. Route discovery is the phase which used for requesting and forwarding a route information that consist of Route Request (RREQ) and Route Reply (RREP), meanwhile route maintenance that consist of Route Error (RERR) is used for finding out an error information in route.*

*In normal AODV route discovery phase, determining the best route is using the first RREQ packet coming. It will become the better RREQ packet is dropped and affect lower quality of service packet.*

*In this Final Project propose a modification name Enhanced-Ant multipath. This method will select a route found by 4 parameters value who affect the quality of service packet, such as received signal strength, congestion node, residual energy node and length of node. These four parameters will be calculate as pheromone count. Pheromone count is the parameters value who*

*determine the first route RREP will send by the destination node. This method will implemented with multipath way. If the normal AODV RREP will send once and the rest will drop, with this method RREP will send all of them by all routes available.*

*With this modification, hope that quality of service of packet sending will increase by filtering the four parameters value. This is done in order to improve the performance of the AODV protocol to find the stable route by choosing the node who meet the requierement and sort all the route list by the highest pheromone value. From the test results, modified in grid scenario has increased the average value of the Packet Delivery Ratio (PDR) by 12.20%, decreased Delivery Delay by 21.64%, decreased Routing Overhead (RO) by 15.94% and increased the Node Survived Ratio by 1.45%. While in the real scenario the average value of the Packet Delivery Ratio (PDR) has increased by 16.69%, decreased Delivery Delay by 48.14%, decreased Routing Overhead (RO) by 20.35%, and increase Node Survive Percentage by 37.53%.*

***Keyword: VANETs, AODV, Ant Colony Optimization, Enhanced-Ant, Multipath AODV***

## KATA PENGANTAR

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul *“Implementasi Metode Enhanced-Ant Pada Multipath AODV Di VANETs”*.

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

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

1. Allah SWT atas semua rahmat yang diberikan sehingga penulis dapat menyelesaikan Tugas Akhir ini.
2. Papa Harfuddin Daing dan Mama Rinasari Kuswardiana selaku kedua orangtua penulis atas segala dukungan berupa motivasi, doa, moral, dan material sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Adik penulis (Saskia dan Sharga) selaku adik penulis atas segala dukungan yang telah diberikan sehingga penulis tetap semangat dalam mengerjakan Tugas Akhir ini.
4. Bapak Dr. Eng. Radityo Anggoro, S.Kom., M.Sc., dan Bapak Ir. Muchammad Husni, M. Kom. selaku dosen pembimbing penulis atas nasihat, arahan dan bantuannya sehingga penulis dapat menyelesaikan Tugas Akhir ini.
5. Bapak Dr.Eng. Darlis Herumurti, S.Kom., M.Kom. selaku kepala Departemen Informatika ITS.
6. Rekan-rekan kerja saya dari Locomotive Production (Arman, Budi, Helmy, Faris, Darma, Falah, Dyah, Ivana, Kartika) yang selalu memberikan semangat, selalu memberikan hiburan kepada penulis ketika mengerjakan Tugas Akhir.

7. Teman-teman dari kosan muslim (Budi, Falah, Hilmi, Mail, Tubin, Ardy) yang menjadi teman ketika penulis tinggal di Surabaya, selalu menghibur penulis dan selalu ada ketika penulis membutuhkan sesuatu selama mengerjakan Tugas Akhir ini.
8. Teman teman HMTC Optimasi, HMTC Inspirasi, Kepanitian di lingkup ITS (ITS EXPO 2016 dan 2017, Gemastik, Gerigi dan Ini Lho ITS!) yang sudah memberikan kesibukan lebih untuk penulis semasa kuliah di Informatika ITS.
9. Juga tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini dan yang telah membaca buku Tugas Akhir ini.

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

Surabaya, Juni 2019

SATRIA CHANDRA YUDHA WIBOWO



## DAFTAR ISI

<b>Abstrak</b>	<b>vii</b>
<b>Abstract</b>	<b>ix</b>
<b>KATA PENGANTAR</b>	<b>xi</b>
<b>DAFTAR ISI</b>	<b>xiii</b>
<b>DAFTAR GAMBAR</b>	<b>xvii</b>
<b>DAFTAR TABEL</b>	<b>xix</b>
<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 Perangkat Lunak	4
1.6.4 Implementasi Perangkat Lunak	4
1.6.5 Pengujian dan Evaluasi	4
1.6.6 Penyusunan Buku Tugas Akhir	5
1.7 Sistematika Penulisan Laporan	5
<b>BAB II TINJAUAN PUSTAKA</b>	<b>7</b>
2.1 Vehicular Ad Hoc Networks (VANETs)	7
2.2 <i>Ad-hoc On demand Multipath Distance Vector</i> (AOMDV)	8
2.3 <i>Network Simulator-2 (NS-2)</i>	9
2.3.1 Instalasi	10
2.3.2 <i>Trace File</i>	10
2.4 OpenStreetMap	12
2.5 <i>Java OpenStreetMap Editor (JOSM)</i>	12
2.6 <i>Simulation of Urban Mobility (SUMO)</i>	13
2.7 AWK	14
<b>BAB III PERANCANGAN</b>	<b>17</b>
3.1 Deskripsi Umum	17

3.2	Perancangan Skenario Mobilitas .....	21
3.2.1	Perancangan Skenario <i>Grid</i> .....	23
3.2.2	Perancangan Skenario <i>Real</i> .....	23
3.3	Perancangan Modifikasi <i>Routing Protocol</i> AOMDV ....	24
3.3.1	Perancangan Penghitungan <i>Threshold RSS</i> dan <i>RE25</i>	
3.3.2	Perancangan Penghitungan <i>Pheromone Count</i> .....	26
3.3.3	Perancangan Pengiriman RREP .....	28
3.4	Perancangan Simulasi pada NS-2 .....	29
3.5	Perancangan Metrik Analisis .....	29
3.5.1	<i>Packet Delivery Ratio</i> (PDR) .....	29
3.5.2	<i>Average End-to-End Delay</i> (E2E).....	30
3.5.3	<i>Routing Overhead</i> .....	30
3.5.4	<i>Node Survive Percentage</i> .....	31
<b>BAB IV IMPLEMENTASI .....</b>		<b>33</b>
4.1	Implementasi Skenario Mobilitas .....	33
4.1.1	Skenario <i>Grid</i> .....	33
4.1.2	Skenario <i>Real</i> .....	36
4.2	Implementasi Modifikasi pada <i>Routing Protocol</i> AODV untuk Menentukan <i>Forwarding Node</i> .....	38
4.2.1	Implementasi Modifikasi <i>Routing Table</i> .....	39
4.2.2	Penghitungan <i>Threshold RSS</i> dan <i>RE</i> .....	41
4.2.3	Implementasi Penghitungan <i>Pheromone Count</i> ....	42
4.2.4	Implementasi Multi-reply RREP .....	43
4.3	Implementasi Simulasi pada NS-2.....	45
4.4	Implementasi Metrik Analisis .....	46
4.4.1	Implementasi <i>Packet Delivery Ratio</i> (PDR).....	47
4.4.2	Implementasi <i>Average End-to-End Delay</i> (E2E)....	48
4.4.3	Implementasi <i>Routing Overhead</i> (RO).....	49
4.4.4	Implementasi <i>Node Survived Percentage</i> .....	49
<b>BAB V .....</b>		<b>51</b>
5.1	Lingkungan Uji Coba .....	51
5.2	Hasil Uji Coba.....	52
5.2.1	Hasil Uji Coba Skenario <i>Grid</i> .....	52
5.2.2	Hasil Uji Coba Skenario <i>Real</i> .....	60
<b>BAB VI KESIMPULAN DAN SARAN.....</b>		<b>69</b>

6.1	Kesimpulan .....	69
6.2	Saran .....	69
<b>DAFTAR PUSTAKA .....</b>		<b>71</b>
<b>LAMPIRAN .....</b>		<b>73</b>
A.1	Kode Fungsi RecvRequest() .....	73
A.2	Kode Fungsi AOMDV_LINK_DISJOINT_PATH.....	80
A.3	Kode Fungsi sendreply() .....	81
A.4	Kode Skenario NS-2.....	83
A.5	Kode Konfigurasi <i>Traffic</i> .....	86
A.6	Kode Skrip AWK <i>Packet Delivery Ratio</i> .....	87
A.7	Kode Skrip AWK Rata-Rata <i>End-to-End Delay</i> .....	88
A.8	Kode Skrip AWK <i>Routing Overhead</i> .....	89
A.9	Kode Skrip AWK <i>Node Survived Percentage</i> .....	90
<b>BIODATA PENULIS .....</b>		<b>91</b>

*(Halaman ini sengaja dikosongkan)*

## DAFTAR GAMBAR

<b>Gambar 2.1</b> Ilustrasi VANETs [16] .....	7
<b>Gambar 2.2</b> Ilustrasi pencarian rute routing protocol AOMDV [12].....	8
<b>Gambar 2.3</b> Perintah untuk menginstall dependency NS-2 .....	10
<b>Gambar 2.4</b> Baris kode yang diubah pada file ls.h .....	10
<b>Gambar 3.1</b> Diagram Rancangan Simulasi AOMDV Modifikasi .....	17
<b>Gambar 3.2</b> Diagram <i>flowchart</i> proses RREQ .....	19
<b>Gambar 3.3</b> Diagram <i>flowchart</i> proses RREP .....	20
<b>Gambar 3.4</b> Alur perancangan skenario .....	22
<b>Gambar 3.5</b> <i>Pseudocode</i> Penghitungan threshold RE dan RSS .....	26
<b>Gambar 3.6</b> <i>Pseudocode</i> penghitungan Pheromone count .....	27
<b>Gambar 3.7</b> <i>Pseudocode</i> penentuan rute dan pengaplikasian multi-reply .....	28
<b>Gambar 4.1</b> Perintah netgenerate .....	33
<b>Gambar 4.2</b> Hasil Generate Peta Grid.....	34
<b>Gambar 4.3</b> Perintah randomTrips .....	34
<b>Gambar 4.4</b> Perintah duarouter .....	35
<b>Gambar 4.5</b> File Skrip .sumocfg .....	35
<b>Gambar 4.6</b> Perintah SUMO untuk membuat skenario .xml .....	36
<b>Gambar 4.7</b> Perintah traceExporter.....	36
<b>Gambar 4.8</b> Ekspor Peta dari OpenStreetMap .....	37
<b>Gambar 4.9</b> Perintah netconvert.....	37
<b>Gambar 4.10</b> Hasil Konversi Peta Real.....	37
<b>Gambar 4.11</b> Potongan modifikasi paket RREQ.....	40
<b>Gambar 4.12</b> Potongan modifikasi paket RREP .....	40
<b>Gambar 4.13</b> Potongan modifikasi <i>Routing Table</i> .....	41
<b>Gambar 4.14</b> Potongan modifikasi Kode Fungsi threshold RE dan RSS .....	42
<b>Gambar 4.15</b> Potongan Kode Penghitungan <i>pheromone count</i> .....	43
<b>Gambar 4.16</b> Potongan Kode <i>multi-reply</i> AOMDV .....	44
<b>Gambar 4.17</b> Implementasi Simulasi NS-2 .....	45
<b>Gambar 4.18</b> Implementasi Simulasi File Traffic .....	46

<b>Gambar 4.19</b> <i>Pseudocode</i> untuk Penghitungan PDR .....	47
<b>Gambar 4.20</b> <i>Pseudocode</i> untuk Penghitungan E2E .....	48
<b>Gambar 4.21</b> <i>Pseudocode</i> untuk Penghitungan RO .....	49
<b>Gambar 4.22</b> <i>Pseudocode</i> untuk Penghitungan <i>node survive percentage</i> .....	50
<b>Gambar 5.1</b> Grafik PDR pada Skenario Grid .....	53
<b>Gambar 5.2</b> Grafik E2E pada Skenario Grid .....	55
<b>Gambar 5.3</b> Grafik RO pada Skenario Grid .....	57
<b>Gambar 5.4</b> Grafik Node Survive Percentage pada Skenario Grid .....	59
<b>Gambar 5.5</b> Grafik PDR pada Skenario Real .....	61
<b>Gambar 5.6</b> Grafik E2E pada Skenario Real .....	63
<b>Gambar 5.7</b> Grafik RO pada Skenario Real .....	65
<b>Gambar 5.8</b> Grafik Node Survive Percentage pada Skenario Real .....	67

## DAFTAR TABEL

<b>Tabel 2.1</b> Detail Penjelasan Trace File AODV .....	11
<b>Tabel 3.1</b> Daftar Istilah .....	21
<b>Tabel 3.2</b> Struktur Paket RREQ.....	24
<b>Tabel 3.3</b> Struktur Paket RREP .....	24
<b>Tabel 3.4</b> Struktur <i>Routing Table</i> .....	24
<b>Tabel 5.1</b> Spesifikasi Perangkat yang Digunakan .....	51
<b>Tabel 5.2</b> Lingkungan Uji Coba.....	52
<b>Tabel 5.3</b> Hasil Rata - Rata PDR Skenario Grid .....	53
<b>Tabel 5.4</b> Hasil Rata - Rata E2E Skenario Grid .....	55
<b>Tabel 5.5</b> Hasil Rata - Rata RO Skenario Grid .....	57
<b>Tabel 5.6</b> Hasil Rata - Rata Node Survive Percentage Skenario Grid .....	59
<b>Tabel 5.7</b> Hasil Rata - Rata PDR pada Skenario Real .....	61
<b>Tabel 5.8</b> Hasil Rata -Rata E2E pada Skenario Real.....	63
<b>Tabel 5.9</b> Hasil Rata - Rata RO Skenario Real .....	65
<b>Tabel 5.10</b> Hasil Rata - Rata Node Survive Percentage pada Skenario Real .....	67

*(Halaman ini sengaja dikosongkan)*



# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Di era perkembangan teknologi yang sangat pesat, teknologi jaringan nirkabel menjadi sangat populer karena kemampuannya untuk dapat beradaptasi dalam berbagai macam situasi. Banyak studi mengenai arsitektur jaringan yang bersifat *ad hoc* untuk mendukung kemampuan komunikasi antar perangkat bergerak, arsitektur ini dinamakan *Mobile Ad hoc Network* (MANET) [1] [2] [3]. Lalu dibutuhkan sebuah jaringan yang berfungsi untuk mengatur lalu-lintas antar perangkat agar *quality of service* (QoS) antar perangkat dapat terjaga dengan baik. Dibangunlah implementasi dari MANET yang bernama *Vehicular Ad hoc Network* (VANETs).

Di dalam lingkungan VANETs, menjaga kualitas pengiriman paket adalah salah satu masalah yang umum terjadi karena berbagai kondisi di lingkungan. Terdapat berbagai macam jenis *routing protocol* yang tersedia. Namun dari semua protokol yang tersedia, tidak ada yang bisa memenuhi beberapa syarat kualitas pengiriman paket sekaligus.

Di dalam Tugas Akhir ini, penulis akan mengimplementasikan sebuah modifikasi dari *protocol Ad Hoc On Demand Distance Vector* (AODV) yang dikombinasikan dengan konsep *Ant Colony Optimization* (ACO) [4]. Penulis akan mengimplementasikan konsep *pheromone count* dari ACO ke dalam *routing table* AODV dan menyimpan beberapa parameter nilai yang dibutuhkan sebagai syarat kualitas pengiriman paket yang nantinya akan dikalkulasikan berupa *pheromone count*.

Dalam fase *route discovery*, node pengirim *broadcast* sebuah paket ReqAnt di dalam jaringan. Paket ini akan

menyebar ke semua rute dan mencari informasi parameter nilai yang dibutuhkan sebagai syarat kualitas pengiriman paket antara lain: *end-to-end reliability route*, *congestion route*, *residual energy node*, dan *length of node*. Setelah itu dalam destinasi *pheromone count* akan dikalkulasi berdasarkan keempat nilai tersebut dan akan dikirim ulang menuju node asal melalui paket RepAnt. Node pengirim akan menyimpan semua rute yang tersedia beserta dengan jumlah kalkulasi *pheromone count* nya. Node penerima akan mengirimkan RepAnt melalui semua rute yang tersimpan. [5].

## 1.2 Rumusan Masalah

Tugas Akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana mengimplementasikan metode *Enhanced-Ant* Multipath dalam protokol AOMDV menggunakan *network simulator* NS-2?
2. Bagaimana menganalisis pengaruh penambahan metode *Enhanced-Ant* pada AOMDV berdasarkan *Routing Overhead*, *Packet Delivery Ratio*, *Node Survive Percentage* dan *End-to-End Delay* dalam pengiriman paket?

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

5. Analisis perbedaan hasil percobaan didasarkan pada *Routing Overhead*, *Packet Delivery Ratio*, *Node Survive Percentage* dan *End-to-End Delay*.

## 1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah untuk mengetahui pengaruh penambahan metode Enhanced-Ant pada Multipath AODV berdasarkan *routing overhead*, *packet delivery ratio*, *node survive percentage* dan *end-to-end delay*.

## 1.5 Manfaat

Dengan dibuatnya Tugas Akhir ini adalah untuk merancang sebuah modifikasi protokol jaringan VANETs yang lebih *reliable* dalam mengirimkan paket dari node sumber menuju node tujuan.

## 1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

### 1.6.1 Penyusunan Proposal Tugas Akhir

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

pula sub bab jadwal kegiatan yang menjabarkan jadwal pengerjaan tugas akhir.

### **1.6.2 Studi Literatur**

Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam melakukan analisis yaitu mengenai *network simulator* NS-2, *Vehicular Ad hoc Networks* (VANETs), *routing protocol* AOMDV, *Ant Colony Optimization* (ACO), SUMO, OpenStreetMap dan AWK.

### **1.6.3 Analisis dan Desain Perangkat Lunak**

Pada tahap ini dilakukan analisis dari hasil penelitian yang dibuat. Data yang dianalisis berasal dari simulasi Enhanced-Ant Multipath pada *routing protocol* AOMDV dengan menghitung *Routing Overhead*, *Packet Delivery Ratio*, *Delay*, dan *End-to-End Delay* pengiriman paket dari node ke node lainnya, hingga masuk ke dalam tahap pengumpulan data untuk dianalisis.

### **1.6.4 Implementasi Perangkat Lunak**

Pada tahap ini dilakukan perancangan model jaringan dalam penerapan *Enhanced-Ant Multipath* pada *routing protocol* AOMDV di dalam lingkungan VANETs menggunakan generator SUMO. Setelah di generate, akan disimulasikan kedalam NS-2. Kemudian ditarik kesimpulan mengenai performansinya yang dibandingkan oleh beberapa protokol.

### **1.6.5 Pengujian dan Evaluasi**

Pengujian dilakukan dengan *VANETs simulator generator* dan *traffic generator* untuk membuat simulasi keadaan topologi untuk diujikan. Kemudian simulasi yang dibuat pada *VANETs simulator generator* dan *traffic generator* tersebut dijalankan pada *network simulator* NS-2 dan akan menghasilkan

*trace file*. Dari *trace file* tersebut akan dihitung *routing overhead*, *packet delivery ratio*, *delay*, dan *end-to-end delay* pengiriman paket untuk menguji performa *routing protocol* adaptif yang dibuat. Keempat parameter ini akan dianalisis perbandingannya untuk beberapa konfigurasi tertentu.

### 1.6.6 Penyusunan Buku Tugas Akhir

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 berisi 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, AOMDV, 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 AOMDV, serta perancangan metrik analisis (*Routing Overhead*, *Packet Delivery Ratio*, *Node Survive Percentage*, dan *End-to-End Delay*).

#### 4. Bab IV. Implementasi

Bab ini berisi implementasi yang berbentuk kode sumber dari proses modifikasi protokol AOMDV, pembuatan simulasi pada NS2, SUMO, dan penghitungan metrik analisis.

#### 5. Bab V. Uji Coba dan Evaluasi

Bab ini berisi 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

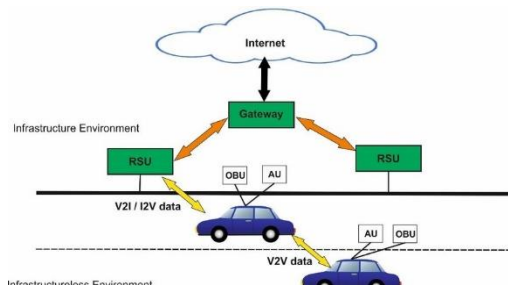
### TINJAUAN PUSTAKA

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

#### 2.1 Vehicular Ad Hoc Networks (VANETs)

Vehicular Ad Hoc Networks (VANETs) adalah teknologi baru yang memadukan jaringan ad hoc, *Wireless LAN* (WLAN), dan teknologi seluler untuk menciptakan komunikasi antar kendaraan yang cerdas dan meningkatkan keselamatan dan efisiensi lalu lintas jalan. VANETs dibedakan dari jenis jaringan ad hoc lainnya berdasarkan arsitektur jaringannya yang hibrida, karakteristik gerakan node, dan skenario aplikasinya yang baru. VANETs menciptakan banyak tantangan yang unik tentang penelitian teknologi jaringan dan desain routing protocol yang efisien untuk VANETs merupakan hal yang krusial. [6]

VANETs terdiri dari komunikasi *vehicle-to-vehicle* dan *vehicle-to-infrastructure* berdasarkan teknologi jaringan nirkabel area lokal. Beberapa aplikasi (contoh: peringatan tabrakan dan informasi lalu lintas area lokal untuk pengemudi), sumber daya (spektrum berlisensi, sumber daya yang dapat diisi ulang), dan lingkungan (contoh: pola arus lalu lintas kendaraan, permasalahan privasi). [7] Ilustrasi VANETs dapat dilihat pada Gambar 2.1.



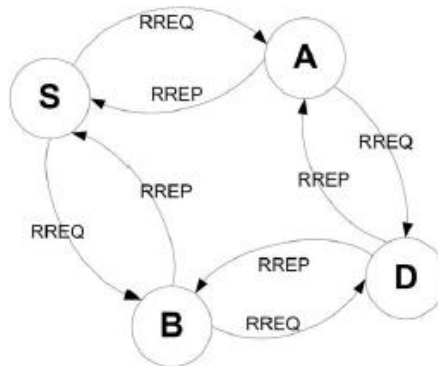
**Gambar 2.1** Ilustrasi VANETs [16]

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

## 2.2 Ad-hoc On demand Multipath Distance Vector (AOMDV)

*Ad hoc On demand Multipath Distance Vector* (AOMDV) adalah *routing protocol* pengembangan dari AODV. Protokol ini bersifat multipath, berbeda dengan AODV yang unipath. Multipath akan membentuk semua rute yang loop-free dan disjoint path. AOMDV akan mengirimkan RREP dan RREQ lebih banyak dari AODV, berakibat pada *Routing Overhead* yang lebih besar. [8]

Pencarian rute dalam AOMDV adalah sebagai berikut. Ketika sumber membutuhkan rute ke tujuan tertentu, ia akan mengirimkan sebuah paket permintaan rute (RREQ) di jaringan ad hoc. Node yang menerima paket tersebut akan mengirimkan balik ke sumber menggunakan RREP sebelumnya dan dilakukan terus menerus oleh node yang ditemukan hingga menemukan node tujuan. Ilustrasi pencarian rute oleh AODV dapat dilihat pada Gambar 2.2.



**Gambar 2.2** Ilustrasi pencarian rute *routing protocol* AOMDV [8]



Sebagai contoh proses *route discovery* dalam AOMDV, ilustrasi pada Gambar 2.2 menggambarkan bagaimana *source node*, yaitu *node S* mencari rute untuk menuju *destination node* yaitu *node D*. *Node S* akan membuat paket RREQ dan melakukan *broadcast* kepada semua *node tetangganya (intermediate node)*. Saat paket RREQ tiba di *intermediate node*, RREQ akan mencatat semua entry yang dibutuhkan dalam *routing table* dan terus mengirimkan paket RREQ tersebut menuju *node tetangganya* dan membuat *reverse path* menuju *source node*, hingga paket tersebut tiba di *destination node*. Saat paket tiba di node tujuan, node D akan mengirimkan RREP menuju *source node* melalui semua rute yang telah dibentuk melalui *routing table* yang telah di-entry saat RREP melakukan pencarian rute.

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

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

NS-2 adalah sebuah *discrete event simulator* yang dirancang untuk membantu mensimulasikan jaringan komputer dengan tujuan riset dan pendidikan. Pada awalnya, NS dibangun sebagai varian dari REAL network simulator pada 1989. Lalu pada 1995 DARPA mendukung pengembangan NS melalui VINT project. NS-2 menggunakan bahasa pemrograman C++ dan OTcl. Bahasa yang digunakan untuk implementasi jaringan adalah C++ sedangkan OTcl digunakan untuk membentuk skenario simulasi jaringan [9].

Pada Tugas Akhir ini, NS-2 digunakan untuk melakukan simulasi lingkungan VANETs menggunakan protokol AOMDV yang sudah dimodifikasi. *Trace file* yang dihasilkan oleh NS-2 juga digunakan untuk mengukur performa *routing protocol* AOMDV yang sudah dimodifikasi.

### 2.3.1 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*

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

### 2.3.2 Trace File

*Trace file* merupakan *file* hasil simulasi yang dilakukan oleh NS-2 dan berisikan informasi detail pengiriman paket data. [10] *Trace file* digunakan untuk menganalisis performa *routing protocol* yang disimulasikan. Detail penjelasan *trace file* ditunjukkan pada Tabel 2.1.

**Tabel 2.1** Detail Penjelasan *Trace File* AODV

Kolom ke-	Penjelasan	Isi
1	<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 Num.</i>	Nomor paket
7	<i>Packet Type</i>	AOMDV : paket <i>routing</i> AOMDV 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 source, destination, dan nexthop</i>	IP	[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> )
----	--	----	--

## 2.4 OpenStreetMap

OpenStreetMap (OSM) adalah sebuah proyek berbasis web untuk membuat peta dunia yang gratis dan terbuka, dibangun sepenuhnya oleh sukarelawan dengan melakukan survei menggunakan GPS, mendigitalisasi citra satelit dan mengumpulkan serta membebaskan data geografis yang tersedia di publik.

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

## 2.5 Java OpenStreetMap Editor (JOSM)

Java OpenStreetMap (JOSM) adalah sebuah aplikasi berbasis Java yang dapat dioperasikan pada sistem operasi Windows, Mac OS dan Linux. yang digunakan untuk menyunting data dari OpenStreetMap (OSM). Aplikasi ini tidak membutuhkan internet dalam penggunaannya. Aplikasi ini dapat diunduh melalui [josm.openstreetmap.de](http://josm.openstreetmap.de) [11].

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

## 2.6 Simulation of Urban Mobility (SUMO)

*Simulation of Urban Mobility* (SUMO) adalah sebuah aplikasi simulasi *multi-modal traffic* yang bersifat *open source*. SUMO mensimulasikan kondisi lalu-lintas dari kendaraan melalui jaringan yang diberikan. Kendaraan ini dapat bergerak sendiri secara individu dan dapat memiliki rute sendiri. [12]

Simulasi dalam SUMO digunakan untuk meneliti pengaruh pengembangan *Enhanced-Ant Multipath* pada jaringan yang bergerak di kondisi riil. SUMO digunakan untuk membuat peta berbentuk grid dan menkonversikannya kedalam format yang dapat digunakan dalam *network simulator* NS-2.

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

- netgenerate  
netgenerate merupakan *tool* yang berfungsi untuk membuat peta berbentuk seperti *grid*, *spider*, dan bahkan *random network*. Sebelum proses netgenerate, pengguna dapat menentukan kecepatan maksimum jalan dan membuat *traffic light* pada peta. Hasil dari netgenerate ini berupa *file* dengan ekstensi *.net.xml*. Pada Tugas Akhir ini netgenerate digunakan untuk membuat peta skenario *grid*.
- netconvert  
netconvert merupakan program CLI yang berfungsi untuk melakukan konversi dari peta seperti OpenStreetMap menjadi format *native* SUMO. Pada Tugas Akhir ini penulis menggunakan netconvert untuk mengonversi peta dari OpenStreetMap.
- randomTrips.py  
*Tool* dalam SUMO untuk membuat rute acak yang akan dilalui oleh kendaraan dalam simulasi.
- duarouter

*Tool* dalam SUMO untuk melakukan penghitungan rute berdasarkan definisi yang diberikan dan memperbaiki kerusakan rute.

- sumo  
Program yang melakukan simulasi lalu lintas berdasarkan data-data yang didapatkan dari netgenerate (skenario *grid*) atau netconvert dari randomTrips.py. Hasil simulasi dapat di-*export* ke sebuah *file* untuk dikonversi menjadi format lain.
- sumo-gui  
GUI untuk melihat simulasi yang dilakukan oleh SUMO secara grafis.
- traceExporter.py  
*Tool* yang bertujuan untuk mengonversi *output* dari sumo menjadi format yang dapat digunakan pada *simulator* lain. Pada Tugas Akhir ini penulis menggunakan traceExporter.py untuk mengonversi data menjadi format .tcl yang dapat digunakan pada NS-2

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

## 2.7 AWK

AWK adalah bahasa pemrograman yang digunakan untuk melakukan *text processing* dan ekstraksi data. 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 [13].

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

*(Halaman ini sengaja dikosongkan)*

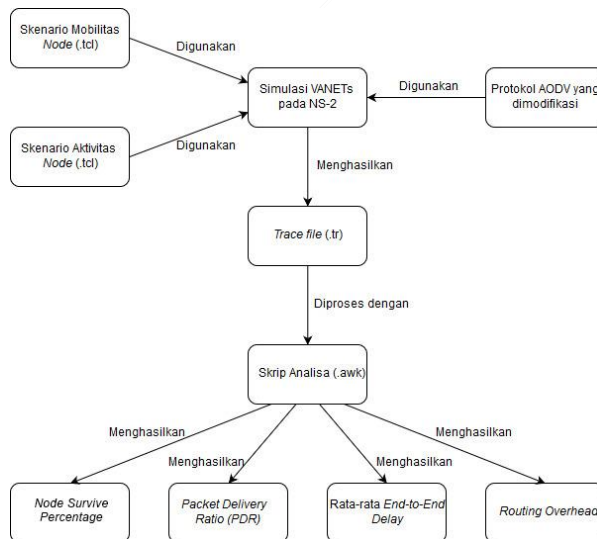


## BAB III PERANCANGAN

Perancangan merupakan bagian krusial dalam menyusun sistem secara teknis sehingga bab ini secara khusus akan menjelaskan perancangan sistem dalam Tugas Akhir. Bab ini akan berawal dari deskripsi umum, perancangan skenario, alur hingga implementasinya.

### 3.1 Deskripsi Umum

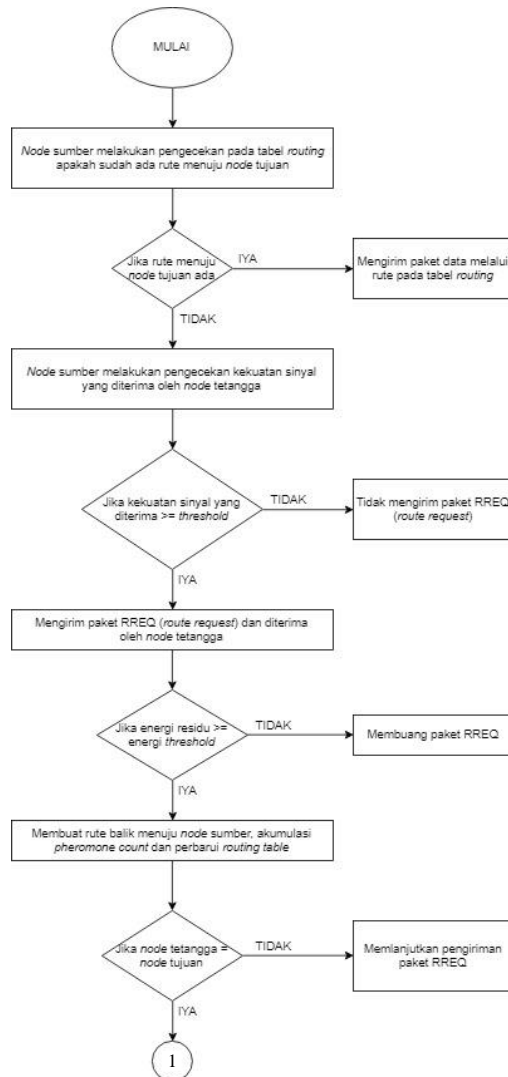
Pada Tugas Akhir ini akan diimplementasikan *routing protocol* AOMDV dengan memodifikasi pada bagian proses *route discovery* yang dijalankan pada simulator NS-2. Diagram dari rancangan simulasi dari AOMDV asli dan AOMDV modifikasi dapat dilihat pada **Gambar 3.1**.



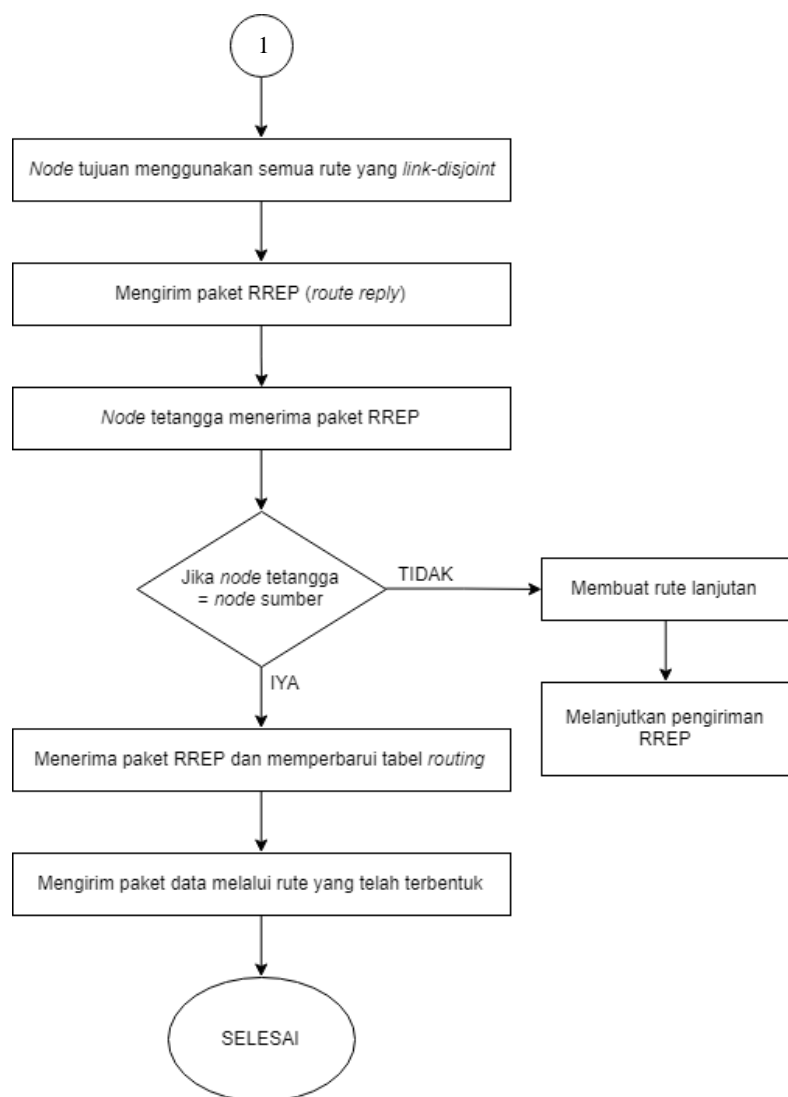
**Gambar 3.1** Diagram Rancangan Simulasi AOMDV Modifikasi

*Enhanced-Ant* Multipath merupakan pengembangan dari protokol AOMDV. *Enhanced-Ant* mengambil karakteristik *route discovery* dari Ant Colony Optimization (ACO) dimana dalam menentukan rute yang dipilih, protokol akan memilih rute terbaik berdasarkan nilai pheromone count tertinggi. Pengembangan protokol ini bertujuan untuk memenuhi beberapa syarat kualitas pengiriman paket sekaligus yang tidak dapat dicapai oleh protokol yang ada. *Pheromone count* adalah sebuah parameter nilai yang dipengaruhi oleh 4 syarat kualitas pengiriman paket yaitu: *end-to-end reliability route*, *congestion route*, *residual energy node*, dan *length of node*. Diagram rancangan *flowchart* dari proses RREQ dan RREP dapat dilihat pada **Gambar 3.2** dan **Gambar 3.3**.

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



**Gambar 3.2** Diagram *flowchart* proses RREQ



**Gambar 3.3** Diagram *flowchart* Proses RREP

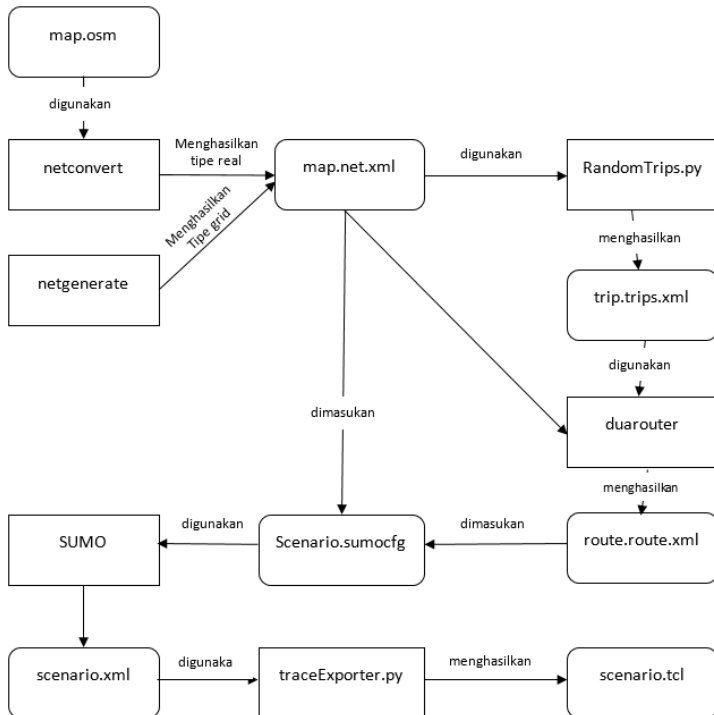
**Tabel 3.1** Daftar Istilah

<b>No.</b>	<b>Istilah</b>	<b>Penjelasan</b>
1	AOMDV	Singkatan dari <i>Ad hoc On-demand Multipath 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	<i>Routing Overhead</i>	Besar jumlah paket kontrol informasi yang dikirimkan node untuk mencari jalur paket data dan menggunakan rute tersebut.
5	<i>Node Survive Percentage</i>	Persentase node yang masih hidup setelah skenario berlangsung.
6	<i>Pheromone Count</i>	Parameter nilai hasil kalkulasi beberapa syarat pengiriman paket yang menjadi penentu <i>route discovery</i> Protokol pada Tugas Akhir ini.
7	RSS	<i>Received Signal Strength</i> . Kekuatan sinyal yang diterima oleh node.
8	RE	<i>Residual Energy</i> . Sisa kekuatan (power) sebuah node untuk menerima paket.
9	Cn	<i>Congestion Node</i> . Panjang antrian paket dalam sebuah jaringan .
10	Hn	<i>Hop number</i> . Jumlah <i>hop</i> yang dilalui antara node pengirim dan node penerima.

### 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. Diagram dari alur perancangan skenario dapat dilihat pada **Gambar 3.4**.



**Gambar 3.4** Alur perancangan skenario

### 3.2.1 Perancangan Skenario *Grid*

Perancangan skenario mobilitas *grid* diawali dengan merancang luas area peta *grid* yang dibutuhkan. Luas area tersebut bisa didapatkan dengan cara menentukan terlebih dahulu jumlah titik persimpangan yang diinginkan, sehingga dari jumlah persimpangan tersebut dapat diketahui berapa banyak peta yang dibutuhkan. Dengan mengetahui jumlah petak yang dibutuhkan, dapat ditentukan panjang tiap petak sehingga mendapatkan luas area yang dibutuhkan yaitu berukuran 1500 m x 1500 m. Dengan 4 titik persimpangan, maka akan didapatkan 9 petak dan panjang tiap peta adalah 500m.

Peta *grid* yang telah ditentukan luasnya tersebut kemudian dibuat dengan menggunakan *tools* SUMO yaitu *netgenerate*. Selain titik persimpangan dan panjang tiap petak *grid*, dibutuhkan juga pengaturan kecepatan kendaraan menggunakan *tools* tersebut. Peta *grid* yang dihasilkan oleh *netgenerate* akan memiliki ekstensi *.net.xml*. Peta *grid* ini kemudian digunakan untuk membuat pergerakan *node* dengan *tools* SUMO yaitu menggunakan *tools* *randomTrips* dan *duarouter*.

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

### 3.2.2 Perancangan Skenario *Real*

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

disediakan oleh OpenStreetMap. Peta hasil *export* tersebut memiliki ekstensi .osm.

Setelah mendapatkan peta area yang akan dijadikan simulasi, peta tersebut dikonversi ke dalam bentuk *file* dengan ekstensi .net.xml menggunakan *tools* SUMO yaitu netconvert. Tahap berikutnya memiliki tahapan yang sama seperti merancang skenario *grid*, yaitu membuat pergerakan *node* menggunakan randomTrips dan duarouter. Kemudian dilakukan penggabungan *file* peta *real* yang sudah dikonversi ke dalam *file* dengan ekstensi .net.xml dan *file* pergerakan *node* yang sudah dibuat sebelumnya. Hasil dari penggabungan tersebut merupakan *file* skenario 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 AOMDV

Protokol AOMDV yang diajukan pada Tugas Akhir ini merupakan modifikasi dari protokol AOMDV yang mengubah mekanisme *route discovery* pada protokol tersebut. Pada modifikasi ini, *Enhanced-Ant* Multipath akan membatasi *forwarding node* menggunakan *threshold* dan menyeleksi rute berdasarkan nilai *pheromone count*-nya. Saat *node* asal mengirimkan paket RREQ, setiap paket di-broadcast menuju node-node tetangga. Saat paket RREQ tiba di *intermediate node*, *node* akan dicek apakah sudah terbentuk *reverse path* dari RREQ sebelumnya. Lalu RREQ akan menghitung apakah *Received Signal Strength* (RSS) dan *Residual Energy* (RE) diatas *threshold* yang telah ditentukan. Hal ini dilakukan untuk menyaring *forwarding node* yang layak untuk dilakukan pengiriman paket.

Setelah itu, RREQ akan menghitung ke-empat parameter dari *node* tersebut antara lain, RSS, RE, *Congestion Number* dan *Hop Count* nya. Ke-empat nilai ini akan dihitung melalui rumus *pheromone count* dan hasilnya akan disimpan ke dalam paket.



*Pheromone count* akan terus diakumulasikan ketika RREQ tiba di *intermediate node* selanjutnya hingga tiba di *node* tujuan.

Pada protokol AODV, apabila *node* tujuan menerima paket RREQ ganda, maka paket yang tidak terpilih akan di-drop. Pada modifikasi ini, saat paket RREQ tiba di *node* tujuan, semua rute yang terbentuk akan disimpan dan dibalas dengan paket RREP. Untuk memenuhi modifikasi ini, dilakukan modifikasi dalam *routing table* pada *header* RREQ dan RREP. Hal ini diperlukan untuk menyimpan semua nilai yang dibutuhkan dalam penghitungan *pheromone count* dan menyimpan nilai *pheromone count* dalam paket. Perubahan struktur dapat dilihat pada **Tabel 3.2**, **Tabel 3.3** dan **Tabel 3.4**.

**Tabel 3.2** Struktur paket RREQ

Src_a ddres	Src_s eqno	Dst_ad dress	Dst_s eqno	Hop_c ount	Signal_s trength	Pheromon e_count
----------------	---------------	-----------------	---------------	---------------	---------------------	---------------------

**Tabel 3.3** Struktur paket RREP

Src_addr ess	Dst_addr ess	Dst_seq no	Lifeti me	Hop_co unt	Pheromone_c ount
-----------------	-----------------	---------------	--------------	---------------	---------------------

**Tabel 3.4** Struktur *routing table*

Dst_addr ess	Seq no	Hop_co unt	Advertised_ hops	Path_l ist	Pheromone_c ount
-----------------	-----------	---------------	---------------------	---------------	---------------------

### 3.3.1 Perancangan Penghitungan *Threshold RSS* dan *RE*

Pada Tugas Akhir ini, penghitungan *threshold* RSS dan RE sebuah *node* dilakukan untuk menyaring *forwarding node* yang siap untuk digunakan pada jaringan untuk mengirim paket. Untuk mendapatkan nilai RSS, modifikasi dilakukan dengan memanggil variabel `txinfo_RxPr` yang dapat dipanggil melalui `Packet *p`. lalu RSS yang telah didapatkan dikonversikan menjadi satuan dBm.

Selanjutnya, untuk mendapatkan nilai RE sebuah *node*, modifikasi akan dilakukan dengan menggunakan *header*

*mobilenode.h* yang berisi fungsi-fungsi MANET dan energy sebuah node. Dari header tersebut, digunakan fungsi `energy_model()->energy()` untuk mendapatkan nilai dari RE sebuah node.

Setelah mendapatkan kedua parameter ini, paket akan dilakukan penyaringan dengan *threshold* yang telah ditetapkan. Apabila salah satu dari kedua parameter ini tidak ada yang mencapai *threshold*, maka paket akan di-drop. Semua modifikasi ini dilakukan pada file *aomdv.cc* yang terletak di dalam *folder ns-2.35/aomdv*. Serta penambahan parameter fungsinya pada file *aomdv.h* yang terletak di *folder ns-2.35/aomdv*. *Pseudocode* dapat dilihat pada **Gambar 3.5**.

```

receive_RREQ (packet p)
    if(rq->rq_dst != this_address)
        if(rq->rq_min_life <= RE_THR)
            discard RREQ;
            return;
        endif
        if (signal_thr >= iRss)
            discard RREQ;
            return;
        endif
    endif
endif

```

**Gambar 3.5** *Pseudocode* Penghitungan *threshold* RE dan RSS

### 3.3.2 Perancangan Penghitungan *Pheromone Count*

Setelah mendapatkan parameter RSS dan RE, modifikasi akan dilakukan untuk mendapatkan *Congestion node* dan *Hop number*. Ke-empat parameter ini dibutuhkan untuk menghitung *pheromone count* yang nantinya akan digunakan untuk menentukan rute pertama yang digunakan *node* tujuan untuk mengirimkan RREP.

*Congestion node* didapatkan dari parameter *len\_* yang di-*parsing* dari file *aomdv\_rqueue.cc* yang terletak di *folder ns-*

2.35/aomdv serta penambahan parameter `queue_length` yang digunakan untuk *men-parsing* variabel `len_` pada file `aomdv_rqueue.h` yang terletak di *folder* `ns-2.35/aomdv`.

Selanjutnya, *Hop number* telah dihitung sebelumnya oleh paket RREQ saat paket dikirimkan, oleh karena itu modifikasi mengambil parameter itu yang digunakan untuk di-kalkulasikan dengan ketiga parameter tersebut.

Setelah senua nilai itu didapatkan, keempat nilai tersebut akan dikalkulasi menggunakan rumus pada persamaan 3.1. *Pseudocode* dapat dilihat pada **Gambar 3.6**.

$$PC_{ij} = \frac{Rn_{ij} \times En_j}{Cn_j \times Hn_{ij}} \quad (3.1)$$

*PC* = Pheromone count

*Rn* = Received signal strength

*En* = Residual energy

*Cn* = Congestion node

*Hn* = Hop number count

```
pheromone_count = (RSS * RE) / (queue_len *
hop_count);

if(pheromone_count != NULL)
    phcount = rq->rq_pheromone_count +
pheromone_count;
else
    phcount = 0 + pheromone_count;
endif
rq->rq_pheromone_count = phcount;
```

**Gambar 3.6** *Pseudocode* penghitungan Pheromone count

### 3.3.3 Perancangan Pengiriman RREP

Setelah melakukan penghitungan *pheromone count*, *node* tujuan akan menerima semua RREQ yang masuk. Lalu *node* tujuan akan mengirimkan RREP melalui rute pertama berdasarkan *pheromone count* tertinggi. Node tujuan akan mengirimkan RREP selanjutnya melalui rute kedua dan seterusnya. Modifikasi yang dilakukan adalah mengaktifkan fitur *multi-reply* yang tersedia dalam protokol AOMDV pada *file aomdv.cc* di *folder ns-2.35/aomdv*. Pada modifikasi ini *multi-reply* yang digunakan adalah *link disjoint*. *Link-Disjoint* adalah metode *multi-reply* pada AOMDV yang berdasarkan pada garis rute yang dilalui. *Link-disjoint* akan menyimpan semua rute yang berbeda jalurnya namun tetap sama nodenya. *Pseudocode* mengenai pemilihan rute pertama dapat dilihat pada **Gambar 3.7**.

```
Dest_receive_RREQ (packet p)
    if (rq->rq_dst == this_address)
        rp_phcount = max(rq_phcount);
        send RREP (p, rp_phcount,
            rp_bcastid);

    #ifdef MULTI_REPLY
    If (rq_bcastid->count==false)
        rq_bcastid->count=true;
        advertised_hops =
        path_max_hopcount();
        sendreply (p, rp_phcount,
            rp_bcastid); #endif
```

**Gambar 3.7** *Pseudocode* penentuan rute dan pengaplikasian *multi-reply*

### 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.

### 3.5 Perancangan Metrik Analisis

Berikut ini merupakan parameter – parameter yang akan dianalisis pada Tugas Akhir ini untuk dapat membandingkan performa dari *routing protocol* AOMDV yang asli dengan AOMDV 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 3.2.

$$PDR = \frac{received}{sent} \times 100 \% \quad (3.2)$$

Keterangan

PDR	= <i>Packet Delivery Ratio</i>
<i>received</i>	= banyak paket data yang diterima
<i>sent</i>	= 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 3.3.

$$E2E = \frac{\sum_{m=1}^{recvnum} CBRRecvTime - CBRSentTime}{recvnum} \quad (3.3)$$

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

*Routing Ovehead* dihitung berdasarkan jumlah paket kontrol informasi yang dikirimkan *node* untuk mencari jalur atau rute yang digunakan untuk menemukan *node* tujuan paket data. Paket kontrol yang dikirimkan berupa *route request* (RREQ), *route reply* (RREP), dan *route error* (RRER). Penghitungan *Route Overhead* dapat dilihat pada persamaan 3.4.

$$RO = \sum_{m=1}^{sentnum} packet\ sent \quad (3.4)$$

### 3.5.4 *Node Survive Percentage*

*Node Survive Percentage* dihitung berdasarkan persentase node yang dapat bertahan dari total node yang ada ketika simulasi berakhir. Rumus untuk menghitung *Node Survive Percentage* dapat dilihat pada persamaan 3.5.

$$\text{Percentage of Node Survived} = \frac{\text{No.nodes survived}}{\text{Total no.of nodes}} \times 100$$

(3.5)

*(Halaman ini sengaja dikosongkan)*



## BAB IV IMPLEMENTASI

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

### 4.1 Implementasi Skenario Mobilitas

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

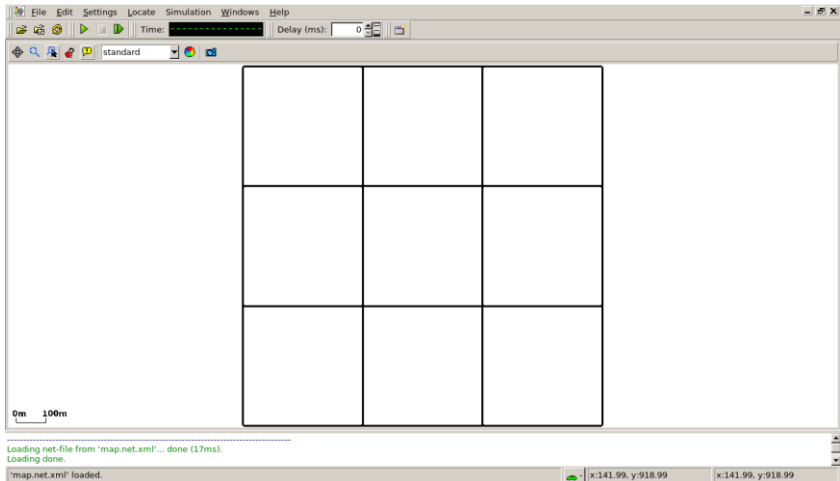
#### 4.1.1 Skenario *Grid*

Dalam mengimplementasikan skenario *grid*, SUMO menyediakan *tools* untuk membuat peta *grid* yaitu *netgenerate*. Pada Tugas Akhir ini, penulis membuat peta *grid* dengan luas 650 m x 650 m yang terdiri dari titik persimpangan antara jalan vertikal dan jalan horisontal sebanyak 4 titik x 4 titik. Dengan jumlah titik persimpangan sebanyak 4 titik tersebut, maka terbentuk 9 buah petak. Sehingga untuk mencapai luas area sebesar 650 m x 650 m dibutuhkan luas per petak sebesar 200 m x 200 m. Berikut perintah *netgenerate* untuk membuat peta tersebut dengan kecepatan *default* kendaraan sebesar 20 m/s dapat dilihat pada **Gambar 4.1**.

```
netgenerate --grid --grid.number=4 --  
grid.length=200 --default.speed=20 --  
tls.guess=1 --output-file=map.net.xml
```

**Gambar 4.1** Perintah *netgenerate*

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



**Gambar 4.2** Hasil *Generate Peta Grid*

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

```
python $SUMO_HOME/tools/randomTrips.py -n
map.net.xml -e 48 -l --trip-
attributes="departLane=\"best\"
departSpeed=\"max\"
departPos=\"random_free\"" -o trip.trips.xml
```

**Gambar 4.3** Perintah randomTrips

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

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

**Gambar 4.4** Perintah duarouter

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

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

**Gambar 4.5** File Skrip .sumocfg

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

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

**Gambar 4.6** Perintah SUMO untuk membuat skenario .xml

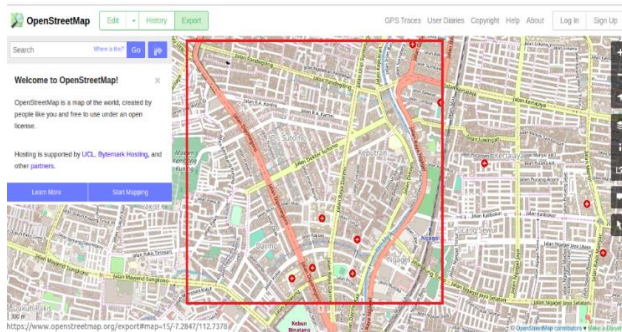
*File* skenario berekstensi .xml selanjutnya dikonversi ke dalam bentuk *file* berekstensi .tcl agar dapat disimulasikan menggunakan NS-2. *Tools* yang digunakan untuk melakukan konversi ini adalah traceExporter. Perintah untuk menggunakan traceExporter dapat dilihat pada **Gambar 4.7**.

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

**Gambar 4.7** Perintah traceExporter

#### 4.1.2 Skenario *Real*

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



**Gambar 4.8** Ekspor Peta dari OpenStreetMap

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

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

**Gambar 4.9** Perintah netconvert

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



**Gambar 4.10** Hasil Konversi Peta *Real*

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

#### **4.2 Implementasi Modifikasi pada *Routing Protocol AODV* untuk Menentukan *Forwarding Node***

Pada Tugas Akhir ini dilakukan modifikasi pada *routing protocol* AOMDV untuk melakukan penyingkapan *node* berdasarkan RSS dan RE, penghitungan *congestion node* dan *hop number* dan penghitungan *pheromone count* sehingga dapat meningkatkan kualitas pengiriman paket dan melakukan *multi-reply* pada pengiriman RREP oleh *node* tujuan untuk mengimplementasikan *Enhanced-Ant* pada AOMDV.

Implementasi modifikasi *routing protocol* AOMDV ini dibagi menjadi 4 bagian yaitu:

- Implementasi Modifikasi Struktur *Routing Table*
- Implementasi Penghitungan *threshold* RSS dan RE
- Implementasi Penghitungan *pheromone count*
- Implementasi Multi-reply RREP

Kode implementasi dari *routing protocol* AOMDV pada NS-2 versi 2.35 berada pada direktori ns-2.35/aomdv. Pada direktori tersebut terdapat beberapa file diantaranya seperti aomdv.cc, aomdv.h dan sebagainya. Pada Tugas Akhir ini, penulis memodifikasi *file* aomdv.cc yang terdapat dalam *folder* ns-

2.35/aomdv untuk mengaktifkan multi-reply pada AOMDV, mengimplementasikan penghitungan *pheromone count*, file *aomdv.h* yang ada di dalam folder *ns-2.35/aomdv* untuk mendaftarkan fungsi baru, file *aomdv\_packet.h* untuk menambahkan *field* baru pada paket RREQ dan RREP dan file *aomdv\_rqueue.cc* untuk menggunakan variable *len\_* sebagai parameter *Congestion node* pada node. Pada bagian ini penulis akan menjelaskan langkah – langkah dalam mengimplementasikan modifikasi *routing protocol* AOMDV untuk melakukan modifikasi struktur *routing table*, penghitungan *threshold* RE dan RSS, kalkulasi *pheromone count* dan mengaktifkan *multi-reply* pada paket RREP di protokol AOMDV.

#### 4.2.1 Implementasi Modifikasi *Routing Table*

Pada implementasi *routing protocol* AOMDV modifikasi ini, terdapat 2 *field* baru yang perlu ditambahkan dalam paket RREQ yaitu *rq\_rss* dan *rq\_pheromone\_count* dan terdapat 1 *field* baru yang perlu ditambahkan dalam paket RREP yaitu *rp\_pheromone\_count*. Modifikasi ini semua dilakukan dalam file *aomdv\_packet.h* di struct *hdr\_aomdv\_request* dan *hdr\_aomdv\_reply*. File ini terdapat dalam folder *ns-2.35/aomdv*. Untuk potongan kode modifikasi RREQ dapat dilihat pada **Gambar 4.11**. Untuk potongan kode modifikasi RREP dapat dilihat pada **Gambar 4.12**.

Serta dilakukan juga penambahan atribut *rt\_pheromone\_count* pada class *aomdv\_rt\_entry* di file *aomdv\_rtable.h*. Untuk potongan kode pada modifikasi *routing table* dapat dilihat pada **Gambar 4.13**.

```

struct hdr_aomdv_request {
    u_int8_t      rq_type;
    u_int8_t      reserved[2];
    u_int8_t      rq_hop_count;
    u_int32_t     rq_bcast_id;
    nsaddr_t      rq_dst;
    u_int32_t     rq_dst_seqno;
    nsaddr_t      rq_src;
    u_int32_t     rq_src_seqno;
    double        rq_min_life;
    double        rq_rss;
    double
rq_pheromone_count;
    double        rq_timestamp;
    nsaddr_t      rq_first_hop; }

```

**Gambar 4.11** Potongan modifikasi paket RREQ

```

struct hdr_aomdv_reply {
    u_int8_t      rp_type;
    u_int8_t      reserved[2];
    u_int8_t      rp_hop_count;
    nsaddr_t      rp_dst;
    u_int32_t     rp_dst_seqno;
    nsaddr_t      rp_src;
    double        rp_lifetime;
    double        rp_pheromone_count;

    double        rp_timestamp;
    u_int32_t     rp_bcast_id;
    nsaddr_t      rp_first_hop; }

```

**Gambar 4.12** Potongan modifikasi paket RREP



```

LIST_ENTRY(aomdv_rt_entry) rt_link;
nsaddr_t          rt_dst;
u_int32_t          rt_seqno;
u_int16_t          rt_hops;
u_int16_t          rt_advertised_hops;
int                rt_last_hop_count;
aomdv_paths        rt_path_list;
u_int32_t          rt_highest_seqno_heard;
int                rt_num_paths_;
bool               rt_error;
double             rt_pheromone_count;

```

**Gambar 4.13** Potongan modifikasi *routing table*

#### 4.2.2 Penghitungan *Threshold* RSS dan RE

Langkah awal yang dilakukan untuk menghitung *threshold* RSS dan RE seperti yang telah dirancang pada subbab 3.3.1 adalah menyiapkan fungsi-fungsi yang dibutuhkan untuk mendapatkan parameter sinyal dan energi pada node.

Energi dalam node dapat diaktifkan dengan menyertakan fungsi-fungsi energi dalam *header* `mobilenode.h`. Protokol `mobilenode` merupakan protokol yang digunakan MANET untuk menjalankan protokolnya. Karena dalam modifikasi ini dibutuhkan parameter energi, oleh karena itu `mobilenode` dibutuhkan untuk menyertakan parameter energi. Lalu parameter sinyal dapat ditemukan dalam *header* paket yang digunakan protokol AOMDV. Setelah kedua parameter ini ditemukan.

Dalam proses *route discovery*, node asal akan men-*broadcast* RREQ menuju node tetangganya. Setelah RREQ tiba di node tetangga, node akan dicek apakah sinyal dan energinya diatas *threshold* yang ditentukan. Apabila tidak memenuhi syarat, paket akan di-drop. Namun apabila memenuhi syarat, node akan melanjutkan ke proses selanjutnya. Modifikasi yang dilakukan terdapat di *file* `aomdv.cc` dalam fungsi `recvrequest()`. File

ini terdapat dalam *folder* ns-2.35/aomdv. Untuk potongan kode tersebut dapat dilihat pada **Gambar 4.14**. Kode selengkapnya dapat dilihat di **Lampiran A1**.

```

Void AOMDV::recvRequest(Packet *p) {
    iEnergy= iNode->energy_model()-
>energy();
    rec_power = p->txinfo_.RxPr;
    if (rec_power == 0){
        iRss = -200; //- (infinite)}
    else{
        iRss = 10*log10(rec_power) + 30;}
    if (rq->rq_min_life > iEnergy && iEnergy
> 0){
        rq->rq_min_life = iEnergy; }

    if (iRss != -200 && iRss < 0) {
        rq->rq_rss = iRss; }

    if(rq->rq_dst != index){
        if(rq->rq_min_life <= re_thr){
            Packet::free(p);
            return;}}}

```

**Gambar 4.14** Potongan modifikasi Kode Fungsi *threshold* RE dan RSS

#### 4.2.3 Implementasi Penghitungan *Pheromone Count*

Pada tahap selanjutnya setelah dapat mengetahui nilai parameter RSS dan RE, RREQ akan menghitung *pheromone count* menggunakan 4 parameter nilai yang diperlukan. Karena RSS dan RE sudah ada, selanjutnya adalah mendapatkan nilai *Congestion node* dan *Hop number*. *Congestion node* didapatkan dari fungsi `queue_length()` dalam *file* `aomdv_queue.cc` yang terdapat dalam *folder* ns-2.35/aomdv. lalu *hop number* didapatkan dari

parameter paket RREQ yang telah sampai di node tersebut. Selanjutnya node akan menghitung berdasarkan rumus *pheromone count* yang telah dirancang di subbab 3.3.2.

Penghitungan ini dilakukan pada fungsi `recvrequest()` yang terletak pada kode sumber `aomdv.cc` yang terletak pada `ns-2.35/aomdv`. Potongan kode untuk proses penghitungan *pheromone count* dapat dilihat pada **Gambar 4.15**. Kode selengkapnya dapat dilihat di **Lampiran A1**.

```

queuelen = rqueue.queue_length(index);
if(queuelen == 0) {
    queuelen = 1;
}
pheromone_count = ((iRss + 91) * iEnergy)
/ (queuelen * rq->rq_hop_count);

if(rq->rq_pheromone_count != NULL) {
    rq_phcount = rq-
>rq_pheromone_count + pheromone_count;
} else {
    rq_phcount = 0 + pheromone_count;
}
rq->rq_pheromone_count =
rq_phcount;

```

**Gambar 4.15** Potongan Kode Penghitungan *pheromone count*

#### 4.2.4 Implementasi Multi-reply RREP

Di tahap selanjutnya, setelah semua RREQ mencapai node tujuan dan nilai *pheromone count* pada semua RREQ telah terkumpul, RREP akan dikirim oleh node tujuan pada rute pertama dengan *pheromone count* tertinggi. Lalu dilanjut melalui rute kedua dan seterusnya.

Untuk melakukan fungsi *multi-reply*, modifikasi menggunakan protokol AOMDV yang telah tersedia di *folder ns-*

2.35/aomdv. AOMDV adalah versi modifikasi *multipath* dari protokol AODV. *Multi-reply* yang diaktifkan adalah AOMDV\_LINK\_DISJOINT\_PATHS seperti yang telah dirancang pada subbab 3.3.3. fitur ini tersedia dalam fungsi `recvrequest()` yang terletak pada kode sumber `aomdv.cc` yang terletak pada `ns-2.35/aomdv`. Potongan kode untuk proses AOMDV\_LINK\_DISJOINT\_PATHS dapat dilihat pada **Gambar 4.16**

```
#ifdef AOMDV_LINK_DISJOINT_PATHS
    if (b->count == 0) {
        b->count = 1;
        if (rt->rt_advertised_hops ==
            INFINITY)
            rt->rt_advertised_hops = rt-
                >path_get_max_hopcount();

        AOMDV_Path* forward_path = NULL;
        AOMDV_Path *r = rt-
            >rt_path_list.lh_first;
        rt->rt_error = true;

        sendReply(rq->rq_src,
            rt->rt_advertised_hops, rq->rq_dst,
            rt->rt_seqno,
            forward_path->expire - CURRENT_TIME,
            rq->rq_timestamp, ih->saddr(),
            rq->rq_bcast_id, forward_path-
                >lasthop);
    }
#endif
```

**Gambar 4.16** Potongan Kode *multi-reply* AOMDV

### 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 4.17**.

```
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)         50
set val(seed)       1.0
set val(adhocRouting) AOMDV
set val(stop)       200
set val(cp)         "cbr50.tcl"
set val(sc)         "scenario.tcl"
```

**Gambar 4.17** 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.4 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 4.18**.

```
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(48) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(49) $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 4.18** 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.5 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, *Routing Overhead*, dan *Node Survived Percentage*.

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

Pada subbab 2.3.2 telah ditunjukkan contoh struktur data *event* yang dicatat dalam *trace file* oleh NS-2. Kemudian, pada persamaan 3.1 telah dijelaskan bagaimana menghitung PDR. Skrip awk untuk menghitung PDR berdasarkan kedua informasi tersebut dapat dilihat pada Lampiran A.6 Kode Skrip AWK *Packet Delivery Ratio*.

PDR didapatkan dengan cara menghitung setiap baris terjadinya *event* pengiriman dan penerimaan paket data yang dikirim melalui agen pada *trace file*. Skrip menyaring setiap baris yang mengandung *string* AGT karena kata kunci tersebut menunjukan *event* yang berhubungan dengan paket komunikasi data. Penghitungan dilakukan dengan menjumlahkan paket yang dikirimkan dan paket yang diterima dengan menggunakan karakter pada kolom pertama sebagai *filter*. Kolom pertama menunjukan event yang terjadi dari sebuah paket. Setelah itu nilai PDR dihitung dengan cara persamaan 3.1. Pseudocode untuk menghitung PDR dapat dilihat pada **Gambar 4.19**.

```

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 4.19** Pseudocode untuk Menghitung PDR

Contoh perintah pengesekusian 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.7 Kode Skrip AWK Rata-Rata *End-to-End Delay*.

Dalam penghitungan 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 4.20**.

```
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 4.20** Pseudocode untuk Penghitungan Rata-Rata 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 *Routing Overhead* dapat dilihat pada lampiran A.8 Kode Skrip AWK *Routing Overhead*. *Pseudocode* untuk menghitung *Routing Overhead* dapat dilihat pada **Gambar 4.21**.

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

**Gambar 4.21** Pseudocode untuk Penghitungan *Route Overhead*

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

#### 4.4.4 Implementasi *Node Survived Percentage*

*Node survive percentage* adalah persentase jumlah node yang masih hidup ketika simulasi telah selesai. Untuk mendapatkan jumlah node yang masih hidup adalah menyaring node yang masih aktif dan dibagi dengan jumlah node yang di-set sebelum simulasi. Penghitungan *node survive percentage* telah dijelaskan pada persamaan 3.5. Skrip awk untuk menghitung *node survive percentage* dapat dilihat pada lampiran

A.9 Kode Skrip AWK *Node Survived Percentage* . *Pseudocode* untuk menghitung *node survive percentage* dapat dilihat pada **Gambar 4.22**.

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

```
while [ "$i" -le "$1" ];  
do  
    if($4 == "-n")  
        sort -n  
        !a[$1]++  
        if($2!=0)  
            s++  
        return  
    endif  
endif  
nsp = s/NR*100
```

**Gambar 4.22** Pseudocode Penghitungan *Node Survived Percentage*

## 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

Komponen	Spesifikasi
CPU	Intel(R) Core™ i5-4690K CPU @ 3.50GHz
Sistem Operasi	Ubuntu 18.04.2 LTS
Linux Kernel	Linux kernel 4.4
Memori	16.0 GB
Penyimpanan	128 GB

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

- SUMO versi 0.32.0 untuk pembuatan skenario mobilitas VANETs.
- JOSM versi 10301 untuk penyuntingan peta OpenStreetMap.
- NS-2 versi 2.35 untuk simulasi skenario VANETs.

Parameter lingkungan uji coba yang digunakan pada NS-2 dapat dilihat pada Tabel 5.2. Pengujian dilakukan dengan menjalankan skenario yang disimulasikan pada NS-2. Dari simulasi tersebut dihasilkan sebuah *trace file* dengan ekstensi .tr yang akan dianalisis dengan bantuan skrip awk untuk mendapatkan PDR, E2E, RO, dan *node survive percentage* menggunakan kode yang terdapat pada lampiran A.6 Kode Skrip AWK Packet

*Delivery Ratio*, A.7 Kode Skrip AWK Rata-Rata *End-to-End Delay*, A.8 Kode Skrip AWK , dan A.9 Kode Skrip Awk *Node Survived Percentage*.

**Tabel 5.2** Lingkungan Uji Coba

No.	Parameter	Spesifikasi
1	Network simulator	NS-2.35
2	Routing protocol	AOMDV
3	Waktu simulasi	200 detik
4	Area simulasi	1500 m x 1500 m
5	Jumlah <i>Node</i>	150, 200, 250, 300
6	Radius transmisi	400m
7	Kecepatan maksimum	20 m/s
8	Protokol MAC	IEEE 802.11p
9	Energi awal	100 Joule
10	Konsumsi Energi saat menerima	8.0 Watt
11	Konsumsi Energi saat mengirim	6.0 Watt

## 5.2 Hasil Uji Coba

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

### 5.2.1 Hasil Uji Coba Skenario *Grid*

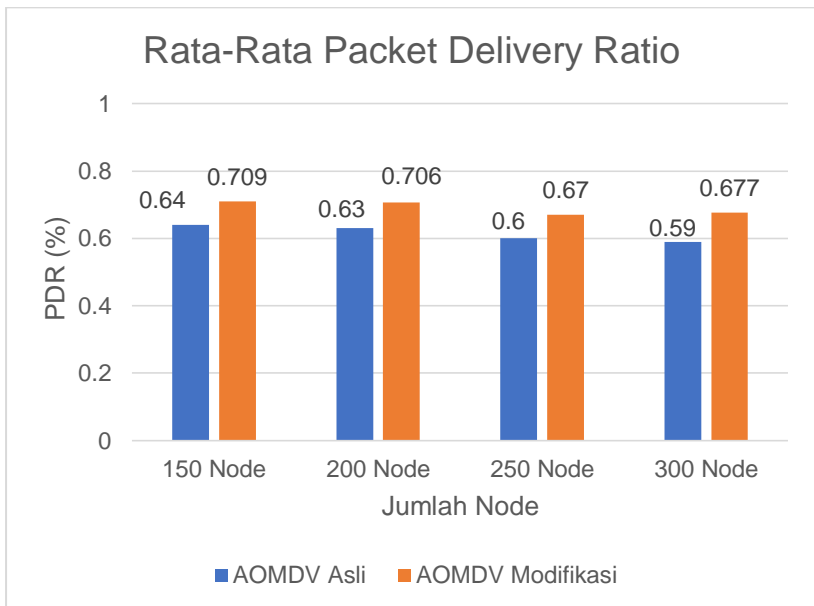
Pengujian pada skenario *grid* digunakan untuk melihat perbandingan PDR, E2E, RO, dan *node survive percentage* antara *routing protocol* AOMDV asli dan AOMDV yang telah dimodifikasi dalam pemilihan *node* yang dapat menerima paket *route request*.

Pengambilan data uji PDR, E2E, RO, dan *node survive percentage* pada skenario *grid* dilakukan sebanyak 10 kali dengan skenario mobilitas *random* pada peta *grid* dengan luas area 1500 m

x 1500 m dan *node* sebanyak 150 untuk lingkungan yang jarang, 200 dan 250 *node* untuk lingkungan yang sedang, dan 300 *node* untuk lingkungan yang padat dilakukan pada kecepatan standar yaitu 20 m/s. Hasil analisis dapat dilihat pada **Tabel 5.3**, **Tabel 5.4**, **Tabel 5.5**, dan **Tabel 5.6**.

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

<b>Jumlah Node</b>	<b>AOMDV Asli</b>	<b>AOMDV Modifikasi</b>	<b>Perbedaan</b>
150	0.6441364	0.7090764	0.06494
200	0.6291178	0.7060256	0.0769078
250	0.6015149	0.6704338	0.0689189
300	0.5887309	0.6772582	0.0885273



**Gambar 5.1** Grafik PDR pada Skenario *Grid*

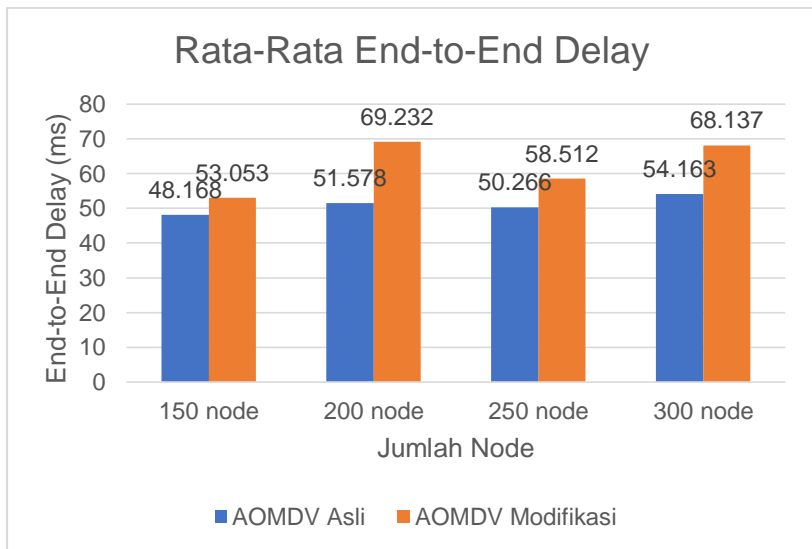
Berdasarkan grafik pada **Gambar 5.1** dapat dilihat bahwa routing protocol AOMDV yang telah dimodifikasi dan juga routing protocol AOMDV asli mengalami kenaikan yang sangat signifikan pada packet delivery ratio. Pada lingkungan yang jarang dengan jumlah 150 node, menghasilkan perbedaan selisih PDR sebesar 0.064, atau naik menjadi sekitar 10.08% dimana *routing protocol* AOMDV yang modifikasi unggul dalam hal PDR tersebut. Pada lingkungan yang sedang dengan jumlah 200 node, menghasilkan perbedaan selisih PDR sebesar 0,0769, atau naik menjadi sekitar 12.22% dimana routing protocol AOMDV yang modifikasi unggul dalam hal PDR tersebut dari AOMDV yang asli. Pada lingkungan yang sedang dengan jumlah 250 node, menghasilkan perbedaan selisih PDR sebesar 0.0689, atau naik menjadi sekitar 11,46% dimana routing protocol AOMDV yang modifikasi juga unggul dalam hal PDR tersebut dari AOMDV yang asli. Pada lingkungan yang padat dengan jumlah 300 node, menghasilkan perbedaan selisih PDR sebesar 0.0885, atau turun menjadi sekitar 15.04% dimana routing protocol AOMDV yang modifikasi juga unggul dalam hal PDR tersebut.

Dapat dilihat rata-rata kenaikan yang terjadi untuk PDR adalah 12.20%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang lebih sedikit atau pada lingkungan dengan *node* yang jarang, menghasilkan PDR yang lebih bagus daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat baik untuk *routing protocol* AOMDV asli maupun *routing protocol* AOMDV yang telah dimodifikasi. Dapat dilihat pula bahwa AOMDV yang telah dimodifikasi menghasilkan PDR yang lebih baik daripada AOMDV asli dengan jumlah selisih PDR yang cukup signifikan.

Hasil pengambilan data rata-rata untuk *end-to-end delay* (E2E) pada skenario *grid* dengan jumlah *node* 150, 200, 250, dan 300 dapat dilihat pada **Gambar 5.2**.

**Tabel 5.4** Hasil Rata - Rata E2E Skenario Grid

Jumlah Node	AOMDV Asli	AOMDV Modifikasi	Perbedaan
150	48.16788	53.05307	4.88519
200	51.57868	69.23182	17.6531
250	50.26623	58.51187	8.24564
300	54.16259	68.13712	13.9745

**Gambar 5.2** Grafik E2E pada Skenario Grid

Berdasarkan grafik pada **Gambar 5.2** dapat dilihat bahwa rata-rata *end-to-end delay* antara routing protocol AOMDV asli dan AOMDV yang telah dimodifikasi mengalami kenaikan yang sangat jauh perbedaannya. Pada lingkungan yang jarang dengan jumlah 150 node, terjadi perbedaan selisih *end-to-end delay* sebesar 4.885 ms antara routing protocol AOMDV asli dengan routing protocol AOMDV yang telah dimodifikasi atau mengalami kenaikan sebesar 10.14%, dimana routing protocol AOMDV yang asli unggul jauh dalam hal *end-to-end delay* tersebut. Sedangkan

pada lingkungan sedang dengan jumlah 200 node, terjadi perbedaan selisih *end-to-end delay* sebesar 17.653 ms antara routing protocol AOMDV asli dengan routing protocol AOMDV yang telah dimodifikasi atau mengalami kenaikan signifikan sebesar 34.23%, dimana routing protocol AOMDV yang asli masih unggul jauh dalam hal *end-to-end delay* tersebut. Pada pada lingkungan sedang dengan jumlah 250 node, terjadi perbedaan selisih *end-to-end delay* sebesar 8.245 ms antara routing protocol AOMDV asli dengan routing protocol AOMDV yang telah dimodifikasi atau masih mengalami kenaikan sebesar 16.40%, dimana routing protocol AOMDV yang asli masih unggul telak dalam hal *end-to-end delay* tersebut. Pada lingkungan yang padat dengan jumlah 300 node, terjadi perbedaan selisih *end-to-end delay* sebesar 13.974 ms antara routing protocol AOMDV asli dengan routing protocol AOMDV yang telah dimodifikasi atau mengalami kenaikan signifikan sebesar 2787.01%, dimana routing protocol AOMDV yang asli masih jauh lebih unggul dalam hal *end-to-end delay* tersebut.

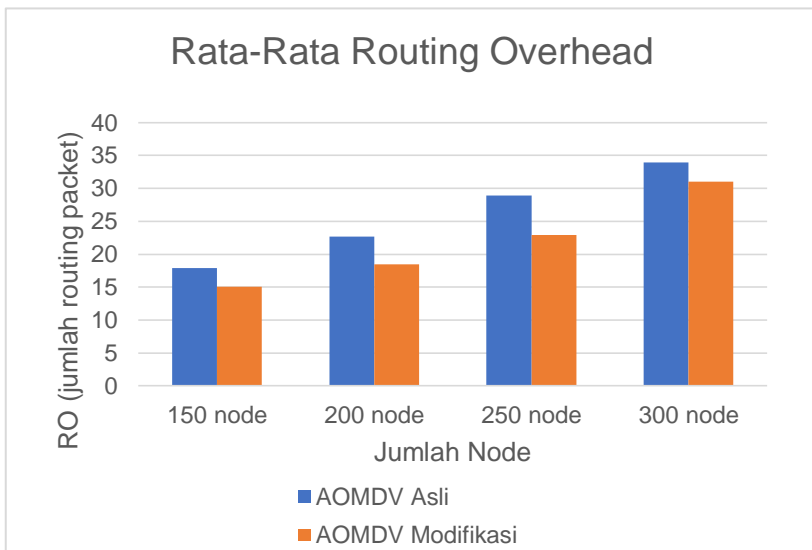
Dapat dilihat rata-rata kenaikan yang terjadi dalam *end-to-end delay* adalah 15.94 %. Jika ketiga lingkungan tersebut dibandingkan, memang pada lingkungan yang jarang, sedang, maupun padat routing protocol AOMDV yang telah dimodifikasi tidak pernah unggul dalam hal *end-to-end delay*. Dapat dilihat bahwa dengan jumlah node jarang, sedang dan padat menghasilkan *end-to-end delay* yang lebih buruk dan dapat dilihat pula bahwa AOMDV yang telah dimodifikasi menghasilkan *end-to end delay* yang lebih buruk daripada AOMDV asli dengan jumlah selisih *end-to-end delay* yang cukup sangat signifikan. Hal ini bisa terjadi karena dengan AODV modifikasi kali ini menggunakan metode - *multi-reply* yang menggunakan semua rute yang tersedia dan menggunakan 4 seleksi kualitas paket sehingga terjadi *delay* yang sangat lama.



Untuk hasil pengambilan data *routing overhead* pada skenario *grid 150 node*, *200 node*, *250 node* dan *300 node* dapat dilihat pada **Gambar 5.3**.

**Tabel 5.5** Hasil Rata - Rata RO Skenario Grid

Jumlah Node	AOMDV Asli	AOMDV Modifikasi	Perbedaan
150	17.92335	15.06794	2.85541
200	22.69837	18.49802	4.20035
250	28.87885	22.91849	5.96036
300	33.9475	30.99684	2.95066



**Gambar 5.3** Grafik RO pada Skenario Grid

Berdasarkan grafik pada **Gambar 5.3** dapat dilihat bahwa rata-rata *routing overhead* antara routing protocol AOMDV asli dan AOMDV yang telah dimodifikasi mengalami penurunan yang signifikan. Pada lingkungan yang jarang dengan jumlah 150 node, menghasilkan perbedaan selisih *routing overhead* sebesar 2.855

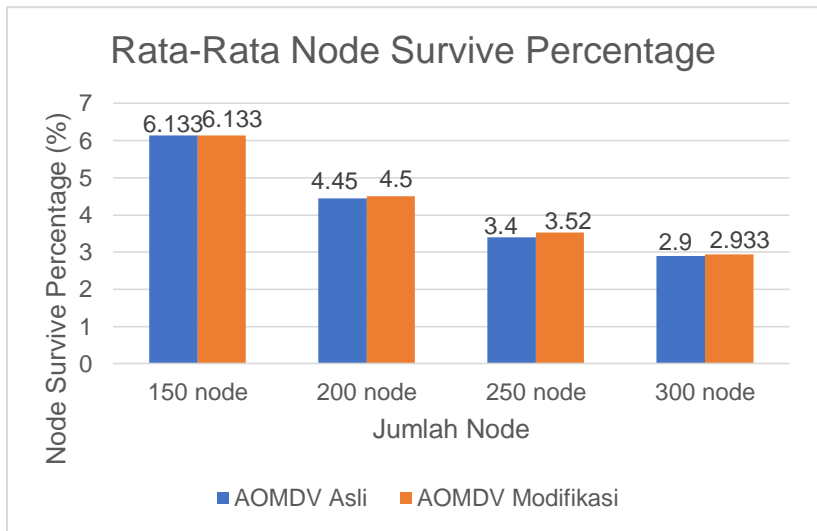
atau mengalami kenaikan sebesar 15.93%, dimana routing protocol AOMDV modifikasi unggul dalam hal *routing overhead* tersebut karena menghasilkan *routing overhead* yang lebih rendah dari routing AOMDV yang asli. Pada lingkungan yang sedang dengan jumlah 200 node, menghasilkan perbedaan selisih *routing overhead* sebesar 4.200 atau mengalami penurunan sebesar 18.51%, dimana routing protocol AOMDV yang telah dimodifikasi unggul dalam hal routing overhead tersebut dari AOMDV asli. Pada lingkungan yang sedang dengan jumlah 250 node, menghasilkan perbedaan selisih *routing overhead* sebesar 5.960 atau mengalami penurunan sebesar 20.64%, dimana routing protocol AOMDV yang telah dimodifikasi unggul dalam hal routing overhead tersebut dari AOMDV asli. Pada lingkungan yang padat dengan jumlah 300 node, menghasilkan perbedaan selisih *routing overhead* sebesar 2.950 atau mengalami penurunan sebesar 8.69%, dimana routing protocol AOMDV asli juga lebih unggul daripada AOMDV yang telah dimodifikasi dalam hal routing overhead tersebut.

Dapat dilihat rata-rata kenaikan yang terjadi untuk *routing overhead* adalah sebesar 15.94%. 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* baik AOMDV asli maupun AOMDV modifikasi. Dapat dilihat pula bahwa AOMDV yang telah dimodifikasi menghasilkan *routing overhead* yang lebih sedikit atau dalam hal ini AOMDV modifikasi lebih unggul di semua lingkungan dengan jumlah selisih *routing overhead* yang cukup signifikan. Hal ini dikarenakan dalam pencarian rute, diinisiasi dengan seleksi node berdasarkan RSS, RE, *hop count* dan *congestion node* Serta melakukan *multi-reply* pada RREP. Oleh karena itu paket yang perlu dikirimkan lebih sedikit karena sudah dibatasi.

Untuk hasil pengambilan data *Node Survive Percentage* pada skenario *grid 150 node*, *200 node*, *250 node* dan *300 node* dapat dilihat pada **Gambar 5.4**.

**Tabel 5.6** Hasil Rata - Rata Node Survive Percentage Skenario Grid

Jumlah <i>Node</i>	AOMDV Asli	AOMDV Modifikasi	Perbedaan
150	6.133334	6.133334	0
200	4.45	4.5	0.05
250	3.4	3.52	0.12
300	2.900001	2.933332	0.033331



**Gambar 5.4** Grafik Node Survive Percentage pada Skenario *Grid*

Berdasarkan grafik pada **Gambar 5.4**, dapat dilihat bahwa *routing protocol* AOMDV yang telah dimodifikasi dan juga *routing protocol* asli mengalami kenaikan yang tidak signifikan. Pada lingkungan yang jarang dengan jumlah *150 node*, tidak terjadi perbedaan hasil baik *routing protocol* AOMDV yang telah dimodifikasi maupun AOMDV asli. Pada lingkungan yang sedang

dengan jumlah 200 *node*, menghasilkan perbedaan selisih *node survive percentage* sebesar 0.05 atau mengalami peningkatan sebesar 1%, dimana *routing protocol* AOMDV yang telah dimodifikasi juga unggul dalam hal *node survive percentage* tersebut dari AOMDV asli. Pada lingkungan yang sedang dengan jumlah 250 *node*, menghasilkan perbedaan selisih sebesar 0.12 atau mengalami kenaikan sebesar 3.53%, dimana *node survive percentage* AOMDV modifikasi lebih unggul dari AOMDV yang telah dimodifikasi. Pada lingkungan yang padat dengan jumlah 300 *node*, menghasilkan perbedaan selisih *node survive percentage* sebesar 0.03 atau mengalami penurunan sebesar 1.15%, dimana *routing protocol* AOMDV yang telah dimodifikasi juga unggul dalam hal *node survive percentage* tersebut.

Dapat dilihat rata-rata peningkatan yang terjadi untuk *node survive percentage* adalah sebesar 1.45%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang jarang, menghasilkan *node survive percentage* yang lebih bagus atau lebih besar persentasenya daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat baik untuk *routing protocol* AOMDV asli maupun *routing protocol* AOMDV yang telah dimodifikasi. Hal ini dikarenakan dalam *routing protocol* AOMDV modifikasi, *node* diseleksi berdasarkan syarat kualitas pengiriman paket. Oleh karena itu, dalam pencarian rute, hanya menggunakan *node* diatas *threshold* yang ditentukan untuk menjaga kualitas paket.

### 5.2.2 Hasil Uji Coba Skenario *Real*

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

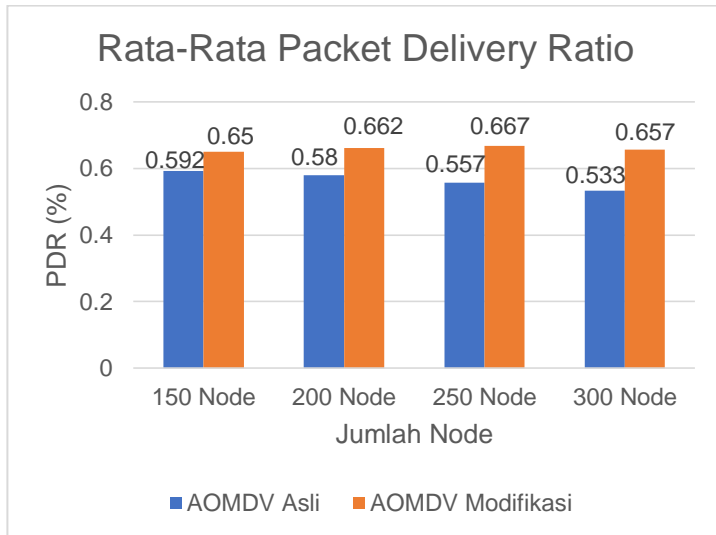
Pengambilan data uji PDR, E2E, RO, dan *node survive percentage* pada skenario *real* dilakukan sebanyak 10 kali dengan

skenario mobilitas *random* pada peta *grid* dengan luas area 650 m x 650 m dan *node* sebanyak 50 untuk lingkungan yang jarang, 100 dan 150 *node* untuk lingkungan yang sedang, dan 200 *node* untuk lingkungan yang padat dilakukan pada kecepatan standar yaitu 20 m/s. Hasil analisis dapat dilihat pada **Tabel 5.7**, **Tabel 5.8**, **Tabel 5.9**, dan **Tabel 5.10**.

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

Jumlah Node	AOMDV Asli	AOMDV Modifikasi	Perbedaan
150	0.592887	0.6509165	0.0580295
200	0.580812	0.6622734	0.0814614
250	0.557912	0.6678756	0.1099636
300	0.5338388	0.6578221	0.1239833

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



**Gambar 5.5** Grafik PDR pada Skenario *Real*

Berdasarkan grafik pada **Gambar 5.5** dapat dilihat bahwa *routing protocol* AOMDV yang telah dimodifikasi dan juga *routing protocol* AOMDV asli mengalami peningkatan pada *packet delivery ratio*. Pada lingkungan yang jarang dengan jumlah 150 node, menghasilkan perbedaan selisih PDR sebesar 0.058, atau naik sebesar sekitar 9.79% dimana *routing protocol* AOMDV yang modifikasi unggul dalam hal PDR tersebut. Pada lingkungan yang sedang dengan jumlah 200 node, menghasilkan perbedaan selisih PDR sebesar 0.081, atau turun menjadi sekitar 14.03% dimana *routing protocol* AOMDV yang telah dimodifikasi unggul dalam hal PDR tersebut dari AOMDV asli. Pada lingkungan yang sedang dengan jumlah 250 node, menghasilkan perbedaan selisih PDR sebesar 0.109, atau naik menjadi sekitar 19.71% dimana *routing protocol* AOMDV yang telah dimodifikasi juga unggul dalam hal PDR tersebut dari AOMDV asli. Pada lingkungan yang padat dengan jumlah 300 node, menghasilkan perbedaan selisih PDR sebesar 0.123, atau naik menjadi sekitar 23.22% dimana *routing protocol* AOMDV yang telah dimodifikasi juga unggul dalam hal PDR tersebut.

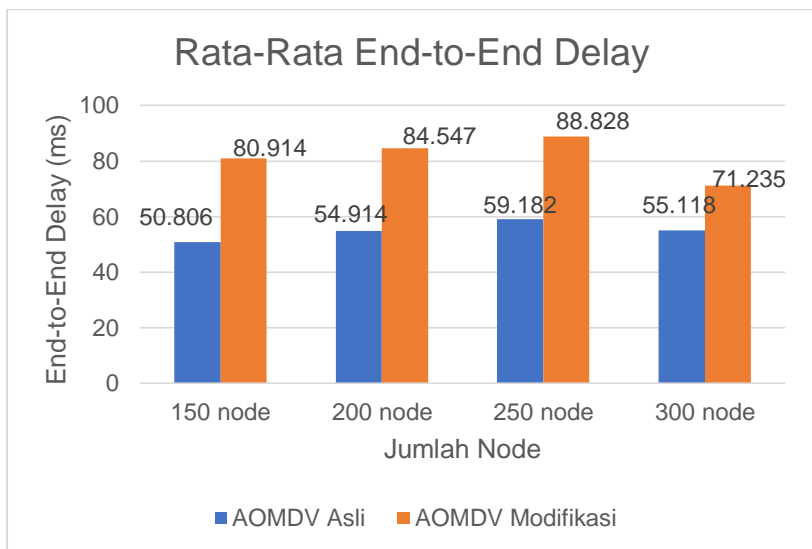
Dapat dilihat rata-rata kenaikan yang terjadi untuk PDR adalah 16.69%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa diantara 3 lingkungan baik pada lingkungan yang jarang, sedang dan padat menghasilkan PDR yang nilainya tidak jauh beda baik untuk *routing protocol* AOMDV asli maupun *routing protocol* AOMDV yang telah dimodifikasi. Dapat dilihat pula bahwa AOMDV modifikasi menghasilkan PDR yang lebih bagus daripada AOMDV yang asli dengan jumlah selisih PDR yang cukup signifikan.

Hasil pengambilan data rata-rata untuk *end-to-end delay* (E2E) pada skenario *real* dengan jumlah *node* 150, 200, 250, dan 300 dapat dilihat pada **Gambar 5.6**.

**Tabel 5.8** Hasil Rata -Rata E2E pada Skenario Real

Jumlah Node	AOMDV Asli	AOMDV Modifikasi	Perbedaan
150	50.8067	80.91468	30.10798
200	54.91464	84.5476	29.63296
250	59.18219	88.8285	29.64631
300	55.11811	71.2354	16.11729

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

**Gambar 5.6** Grafik E2E pada Skenario Real

Berdasarkan grafik pada **Gambar 5.6** dapat dilihat bahwa rata-rata *end-to-end delay* antara routing protocol AOMDV asli dan AOMDV yang telah dimodifikasi mengalami kenaikan yang signifikan. Pada lingkungan yang jarang dengan jumlah 150 node, terjadi perbedaan selisih *end-to-end delay* sebesar 30.107 ms antara routing protocol AOMDV asli dengan routing protocol

AOMDV yang telah dimodifikasi atau mengalami peningkatan sebesar 59.26%, dimana routing protocol AOMDV asli unggul dalam hal *end-to-end delay* tersebut. Sedangkan pada lingkungan sedang dengan jumlah 200 node, terjadi perbedaan selisih *end-to-end delay* sebesar 29.632 ms antara routing protocol AOMDV asli dengan routing protocol AOMDV yang telah dimodifikasi atau mengalami peningkatan sebesar 53.96%, dimana routing protocol AOMDV yang asli lebih unggul dalam hal *end-to-end delay* tersebut. Pada pada lingkungan sedang dengan jumlah 250 node, terjadi perbedaan selisih *end-to-end delay* sebesar 29.696 ms antara routing protocol AOMDV asli dengan routing protocol AOMDV yang telah dimodifikasi atau mengalami penurunan sebesar 50.09%, dimana routing protocol AOMDV yang asli lebih unggul dalam hal *end-to-end delay* tersebut. Pada lingkungan yang padat dengan jumlah 300 node, terjadi perbedaan selisih *end-to-end delay* sebesar 16.11 ms antara routing protocol AOMDV asli dengan routing protocol AOMDV yang telah dimodifikasi atau mengalami peningkatan sebesar 29.24%, dimana routing protocol AOMDV asli unggul dalam hal *end-to-end delay* tersebut.

Jika ketiga lingkungan tersebut dibandingkan, memang pada lingkungan yang jarang, sedang, maupun padat routing protocol AOMDV asli selalu lebih unggul 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 asli menghasilkan *end-to end delay* yang lebih baik daripada AOMDV yang dimodifikasi dengan jumlah selisih *end-to-end delay* yang cukup signifikan. Hal ini bisa terjadi karena dengan AODV modifikasi kali ini menggunakan metode - *multi-reply* yang menggunakan semua rute yang tersedia dan menggunakan 4 seleksi kualitas paket sehingga terjadi penambahan waktu *delay*.

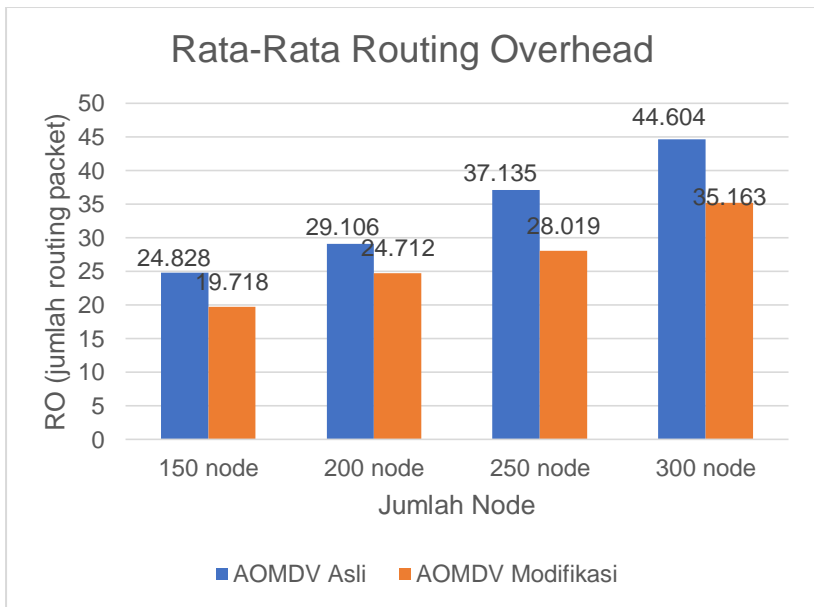
Untuk hasil pengambilan data *routing overhead* pada skenario *real* 150 node, 200 node, 250 node dan 300 node dapat dilihat pada **Gambar 5.7**.



**Tabel 5.9** Hasil Rata - Rata RO Skenario Real

<b>Jumlah Node</b>	<b>AOMDV Asli</b>	<b>AOMDV Modifikasi</b>	<b>Perbedaan</b>
150	24.82883	19.71834	5.11049
200	29.10639	24.71271	4.39368
250	37.13556	28.01914	9.11642
300	44.60469	35.16327	9.44142

Dari data di atas, dibuat grafik yang merepresentasikan hasil penghitungan RO yang ditunjukkan pada **Gambar 5.7**.

**Gambar 5.7** Grafik RO pada Skenario *Real*

Berdasarkan grafik pada **Gambar 5.7** dapat dilihat bahwa rata-rata *routing overhead* antara routing protocol AOMDV asli dan AOMDV yang telah dimodifikasi mengalami penurunan yang

signifikan. Pada lingkungan yang jarang dengan jumlah 150 node, menghasilkan perbedaan selisih *routing overhead* sebesar 5.110 atau mengalami kenaikan sebesar 20.58%, dimana *routing protocol* AOMDV modifikasi unggul dalam hal *routing overhead* tersebut karena menghasilkan *routing overhead* yang lebih rendah dari *routing* AOMDV yang asli. Pada lingkungan yang sedang dengan jumlah 200 node, menghasilkan perbedaan selisih *routing overhead* sebesar 4.393 atau mengalami kenaikan sebesar 15.10%, dimana *routing protocol* AOMDV asli mengalami kekalahan dalam hal *routing overhead* tersebut dari AODV yang modifikasi. Pada lingkungan yang sedang dengan jumlah 250 node, menghasilkan perbedaan selisih *routing overhead* sebesar 9.116 atau mengalami penurunan sebesar 24.55%, dimana *routing protocol* AOMDV asli mengalami kekalahan dalam hal *routing overhead* tersebut dari AODV modifikasi. Pada lingkungan yang padat dengan jumlah 300 node, menghasilkan perbedaan selisih *routing overhead* sebesar 9.441 atau mengalami penurunan sebesar 24.55%, dimana *routing protocol* AODV modifikasi juga lebih unggul daripada AODV yang asli dalam hal *routing overhead* tersebut.

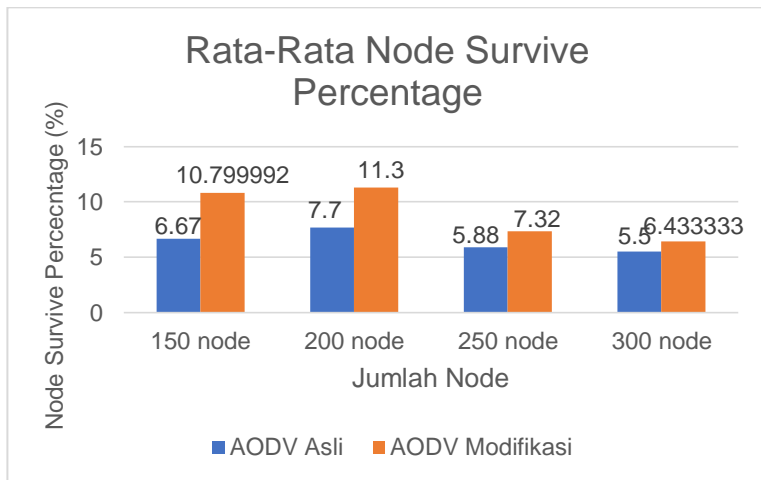
Dapat dilihat rata-rata penurunan yang terjadi untuk *routing overhead* adalah sebesar 20.35%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang lebih sedikit atau pada lingkungan dengan *node* yang jarang, menghasilkan *routing overhead* yang lebih bagus atau lebih sedikit daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat baik untuk *routing protocol* AOMDV asli maupun *routing protocol* AOMDV yang telah dimodifikasi. Dapat dilihat pula bahwa AOMDV yang telah dimodifikasi menghasilkan *routing overhead* yang lebih sedikit atau dalam hal ini AOMDV modifikasi lebih unggul di semua lingkungan dengan jumlah selisih *routing overhead* yang cukup signifikan. Hal ini dikarenakan dalam pencarian rute, diinisiasi dengan seleksi *node* berdasarkan RSS, RE, *hop count* dan *congestion node* Serta

melakukan *multi-reply* pada RREP sehingga menyebabkan berkurangnya paket yang dikirimkan.

Untuk hasil pengambilan data *node survive percentage* pada scenario *real* 150 node, 200 node, 250 node dan 300 node dapat dilihat pada **Gambar 5.8**.

**Tabel 5.10** Hasil Rata - Rata *Node Survive Percentage* pada Skenario Real

Jumlah Node	AOMDV Asli	AOMDV Modifikasi	Perbedaan
150	6.67	10.799999	4.12999
200	7.7	11.3	3.6
250	5.88	7.32	1.44
300	5.5	6.433333	0.93333



**Gambar 5.8** Grafik *Node Survive Percentage* pada Skenario Real

Dari data di atas, dibuat grafik yang merepresentasikan hasil penghitungan *node survive percenctage* yang ditunjukkan pada **Gambar 5.8**.

Berdasarkan grafik pada **Gambar 5.8** dapat dilihat bahwa *routing protocol* AOMDV yang telah dimodifikasi dan juga *routing protocol* asli mengalami perubahan *node survive percentage* yang signifikan. Pada lingkungan yang jarang dengan jumlah 150 *node*, menghasilkan perbedaan selisih *node survive percentage* sebesar 4.13 atau mengalami kenaikan sebesar 61.92%, dimana *routing protocol* AOMDV yang telah dimodifikasi unggul dalam hal *node survive percentage* tersebut karena menghasilkan *node survive percentage* yang lebih tinggi dari *routing protocol* AOMDV asli. Pada lingkungan yang sedang dengan jumlah 200 *node*, menghasilkan perbedaan selisih *node survive percentage* sebesar 3.60 atau mengalami kenaikan sebesar 46.95%, dimana *routing protocol* AOMDV yang telah dimodifikasi juga unggul dalam hal *node survive percentage* tersebut dari *node survive percentage* AOMDV asli. Pada lingkungan yang sedang dengan jumlah 250 *node*, menghasilkan perbedaan selisih *node survive percentage* sebesar 1.44 atau mengalami kenaikan sebesar 24.49%, dimana *routing protocol* AOMDV modifikasi lebih unggul dalam hal *node survive percentage* tersebut dari *node survive percentage* AOMDV yang asli. Pada lingkungan yang padat dengan jumlah 300 *node*, menghasilkan perbedaan selisih *node survive percentage* sebesar 0.93 atau mengalami penurunan sebesar 16.97%, dimana *routing protocol* AOMDV yang telah dimodifikasi juga unggul dalam hal *node survive percentage* tersebut.

Dapat dilihat rata-rata kenaikan yang terjadi untuk *node survive percentage* adalah sebesar 37.53%. Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang padat, menghasilkan *node survive percentage* yang lebih bagus atau lebih tinggi daripada di lingkungan dengan jumlah *node* yang jarang maupun yang sedang baik untuk *routing protocol* AOMDV asli maupun *routing protocol* AOMDV yang telah dimodifikasi. Hal ini dikarenakan dalam *routing protocol* AODV modifikasi, *forwarding node* dibatasi berdasarkan kualitas sinyal dan energi sehingga *node* yang digunakan dalam rute dalam kualitas bagus dan menggunakan *node* seefektif mungkin.

## **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 *multipath*, *Enhanced-Ant AOMDV* yang dimodifikasi sudah berhasil membatasi node yang digunakan rute untuk mengirim paket berdasarkan *Received Signal Strength*, *Residual Energy*, *Hop Count* dan *Congestion Node* dalam mengirim paket RREQ.
2. Dampak implementasi *multipath* terhadap performa protokol *Enhanced-Ant AODV* pada skenario grid adalah rata – rata kenaikan *Packet Delivery Ratio* (PDR) sebesar 12.20%, rata – rata kenaikan *End-to-end Delay* sebesar 21.64%, rata – rata penurunan *Routing Overhead* (RO) sebesar 15.94% dan rata – rata kenaikan *node survive percentage* sebesar 1.45%.
3. Dampak implementasi *multipath* terhadap performa protokol *Enhanced-Ant AODV* pada skenario real adalah rata – rata kenaikan *Packet Delivery Ratio* (PDR) sebesar 16.69%, rata – rata kenaikan *End-to-end Delay* sebesar 48.14%, rata – rata penurunan *Routing Overhead* (RO) sebesar 20.35% dan rata – rata kenaikan *node survive percentage* sebesar 37.53%.

#### **6.2 Saran**

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

1. Lebih banyak uji coba yang dilakukan untuk mendapatkan hasil yang lebih akurat, seperti lebih dari 10 skenario untuk tiap jumlah nodenya.
2. Melakukan uji coba menggunakan nilai energi yang heterogen agar menghasilkan lingkungan simulasi yang organik.

## DAFTAR PUSTAKA

- [1] K. Pahlavan dan P. Krishnamurthy, *Principles of wireless networks: A unified approach*, Prentice Hall, 2011.
- [2] S. Giordano, *Mobile Ad Hoc Networks*, John Wiley & Sons, Inc., 2002, pp. 325-346.
- [3] M. Yadav, K. Arya dan V. Rishiwal, "Improved Ant Colony Optimization Technique for Mobile Adhoc Networks," dalam *International Conference on Computer Science and Information Technology*, 2011.
- [4] M. Doringo dan S. T, "Ant Colony Optimization," dalam *6th International Conference, ANTS*, Brussels, 2008.
- [5] D. Sarkar, S. Choudhury dan A. Majumder, "Enhanced-Ant-AODV for optimal route selection in mobile ad-hoc network," *Journal of King Saud University - Computer and Information Sciences*, 2018.
- [6] F. Li dan W. Yu, "Routing in Vehicular Ad Hoc Networks: A Survey," *IEEE Vehicular Technology Magazine* vol. 2 no.2, pp. 12-22, 2007.
- [7] H. Hartenstein dan L. Laberteaux, "A Tutorial Survey on Vehicular Ad Hoc Network," *IEEE Communications Magazine* vol. 46 no. 6, pp. 164-171, 2008.
- [8] R. Anggoro, T. Kitasuka dan R. Nakamura, "Performance Evaluation of AODV and AOMDV," dalam *2012 Third International Conference on Networking and Computing*, Kumamoto, 2012.
- [9] S. McCanne dan S. Floyd, "The Network Simulator- NS-2," [Online]. Available: <https://www.isi.edu/nsnam/ns/>. [Diakses 30 December 2018].
- [10] P. Meenaghan dan D. Delaney, "An Introduction to NS Nam and OTcl scripting," April 2004.

- [11] "JOSM," [Online]. Available:  
<https://josm.openstreetmap.de/>. [Diakses 2018  
 December 2018].
- [12] R. Hilbrich, "SUMO - Simulation of Urban MObility,"  
 [Online]. Available: <http://www.sumo.dlr.de/>. [Diakses  
 30 December 2018].
- [13] "AWK," [Online]. Available:  
<http://tldp.org/LDP/abs/html/awk.html>. [Diakses 10  
 01 2017].
- [14] "OpenStreetMap," [Online]. Available:  
<https://www.openstreetmap.org/>. [Diakses 15  
 November 2017].
- [15] 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.
- [16] G. Kumar, R. Saha dan M. Kumar Rai, "Multidimensional  
 Security Provision for Secure Communication in  
 Vehicular Ad hoc Networks using Hierarchical  
 Structure and End-to-End Authentication," *IEEE  
 Access*, 2018.
- [17] C. Perkins dan E. Royer, "Ad hoc On-demand Distance  
 Vector Routing," dalam *2nd IEEE Workshop on  
 Mobile Computing Systems and Applications*, 1999.
- [18] V. Sameswari dan R. E, "Shortest Route Discovery Using  
 Hybrid (Aodv And Olsr) Routing Protocols in  
 Manet," dalam *Shortest Route Discovery Using  
 Hybrid (Aodv And Olsr) Routing Protocols in Manet*,  
 Nagercoil, 2014.



## LAMPIRAN

### A.1 Kode Fungsi RecvRequest()

```

AOMDV::recvRequest(Packet *p) {
    struct hdr_ip *ih = HDR_IP(p);
    struct hdr_aomdv_request *rq =
HDR_AOMDV_REQUEST(p);
    aomdv_rt_entry *rt;
    AOMDVBroadcastID* b = NULL;
    bool kill_request_propagation =
false;
    AOMDV_Path* reverse_path = NULL;

    iNode=      (MobileNode *)
(Node::get_node_by_address (index) );
    xpos=      iNode->X();
    ypos=      iNode->Y();
    iEnergy=    iNode->energy_model()-
>energy();

    rec_power = p->txinfo_.RxPr;
    if (rec_power == 0)
    {
        iRss = -200; //(infinite)
    }
    else{
        iRss = 10*log10(rec_power) + 30;
    }
    queuelen = rqueue.queue_length(index);
    if(queuelen == 0) {
        queuelen = 1;
    }
}

```

```

    if (rq->rq_min_life > iEnergy && iEnergy
> 0){
        rq->rq_min_life = iEnergy;
    }
    if (iRss != -200 && iRss < 0) {
        rq->rq_rss = iRss;
    }
    if(rq->rq_dst != index){
        if(rq->rq_min_life <= re_thr){
            Packet::free(p);
            return;
        }
        else {
            pheromone_count = ((iRss + 91) *
iEnergy) / (queuelen * rq->rq_hop_count);

            if(rq->rq_pheromone_count != NULL) {
                rq_phcount = rq-
>rq_pheromone_count + pheromone_count;
            } else {
                rq_phcount = 0 + pheromone_count;
            }
            rq->rq_pheromone_count = rq_phcount;
        }
    } else {
        if(rq->rq_pheromone_count != NULL) {
            rq_phcount = rq->rq_pheromone_count
+ pheromone_count;
        } else {
            rq_phcount = 0 + pheromone_count;
        }
        rq->rq_pheromone_count = rq_phcount;
    }
}

```

```

if(rq->rq_src == index) {
    Packet::free(p);
    return;
}
if ( (b = id_get(rq->rq_src, rq-
>rq_bcast_id)) == NULL) {

    // Cache the broadcast ID
    id_insert(rq->rq_src, rq-
>rq_bcast_id);
    b = id_get(rq->rq_src, rq-
>rq_bcast_id);
}
else
    kill_request_propagation =
true;
if (rq->rq_hop_count == 0)
    rq->rq_first_hop = index;

    aomdv_rt_entry *rt0; // rt0 is the
reverse route

    rt0 = rtable.rt_lookup(rq->rq_src);
    if(rt0 == 0) { /* if not in the route
table */
        // create an entry for the
reverse route.
        rt0 = rtable.rt_add(rq-
>rq_src);}
        if (rt0->rt_seqno < rq-
>rq_src_seqno) {
            rt0->rt_seqno = rq-
>rq_src_seqno;
            rt0->rt_advertised_hops =
INFINITY;

```

```

rt0->path_delete(); // Delete all previous
paths to RREQ source
        rt0->rt_flags = RTF_UP;
        reverse_path = rt0-
>path_insert(ih->saddr(), rq-
>rq_hop_count+1, CURRENT_TIME +
REV_ROUTE_LIFE, rq->rq_first_hop);
        // CHANGE
        rt0->rt_last_hop_count = rt0-
>path_get_max_hopcount();
        // CHANGE }
        else if ( (rt0->rt_seqno == rq-
>rq_src_seqno) && (rt0->rt_advertised_hops
> rq->rq_hop_count) ) {
            AOMDV_Path* erp=NULL;

            assert(rt0->rt_flags ==
RTF_UP); // Make sure path is up
            if ((reverse_path = rt0-
>disjoint_path_lookup(ih->saddr(), rq-
>rq_first_hop))) {
                assert(reverse_path-
>hopcount == (rq->rq_hop_count+1));
                reverse_path->expire =
max(reverse_path->expire, (CURRENT_TIME +
REV_ROUTE_LIFE));
            }
            else if (rt0->new_disjoint_path(ih-
>saddr(), rq->rq_first_hop)) {
                if ( (rt0->rt_num_paths_ <
aomdv_max_paths_) && (((rq->rq_hop_count +
1) - rt0->path_get_min_hopcount()) <=
aomdv_prim_alt_path_len_diff_)
                    ) {

```

```

reverse_path = rt0->path_insert(ih-
>saddr(), rq->rq_hop_count+1, CURRENT_TIME
+ REV_ROUTE_LIFE, rq->rq_first_hop);
rt0->rt_last_hop_count = rt0-
>path_get_max_hopcount();
    }
if (((rq->rq_hop_count + 1) - rt0-
>path_get_min_hopcount()) >
aomdv_prim_alt_path_len_diff) {
    Packet::free(p);
    return;}}

else if ( (rq->rq_dst == index) && ( ((erp
= rt0->path_lookup_lasthop(rq-
>rq_first_hop)) == NULL) || ((rq-
>rq_hop_count+1) > erp->hopcount)) ) {
    Packet::free(p);
    return;}}

else {
    Packet::free(p);
    return;
}

if (rt0->rt_flags == RTF_UP) {
    // Reset the soft state
    rt0->rt_req_timeout = 0.0;
    rt0->rt_req_last_ttl = 0;
    rt0->rt_req_cnt = 0;

    Packet *buffered_pkt;
    while ((buffered_pkt =
rqueue.dequeue(rt0->rt_dst))) {
        if (rt0 && (rt0->rt_flags
== RTF_UP)) {
            forward(rt0,
buffered_pkt, NO_AOMDV_DELAY);}}

```

```

if(rq_phcount != NULL) {
    rt0->rt_pheromone_count = rq_phcount;
}

    rt = rtable.rt_lookup(rq->rq_dst);
    if (rq->rq_dst == index) {

if (seqno < rq->rq_dst_seqno) {
    //seqno = max(seqno, rq-
>rq_dst_seqno)+1;
    seqno = rq->rq_dst_seqno
+ 1; //CHANGE (replaced above line with
this one) }
    if (seqno%2)
        seqno++;
    sendReply(rq->rq_src,0, index,
        seqno,
        MY_ROUTE_TIMEOUT,
        rq->rq_timestamp,
        ih->saddr(),
        rq->rq_bcast_id,
        ih->saddr());
    Packet::free(p); }
    else if ( rt &&(rt->rt_flags ==
RTF_UP) &&(rt->rt_seqno >= rq-
>rq_dst_seqno) ) {

        assert ((rt->rt_seqno%2) == 0);
if (reverse_path)
    #ifdef AOMDV_NODE_DISJOINT_PATHS;

else { if (kill_request_propagation) {
    Packet::free(p);
    return;}
    else {ih->saddr() = index; if (rt) ;
        rq->rq_dst_seqno = max(rt-
>rt_seqno, rq->rq_dst_seqno);

```

```
if (rt0->rt_advertised_hops == INFINITY)
    rt0->rt_advertised_hops = rt0-
    >path_get_max_hopcount();
    rq->rq_hop_count = rt0-
    >rt_advertised_hops;
#ifdef AOMDV_NODE_DISJOINT_PATHS
    rq->rq_first_hop = (rt0-
    >path_find())->lasthop;
#endif // AOMDV_NODE_DISJOINT_PATHS

    forward((aomdv_rt_entry*) 0, p,
    AOMDV_DELAY);
```

## A.2 Kode Fungsi AOMDV\_LINK\_DISJOINT\_PATH

```

AOMDV_Path* forward_path = NULL;
AOMDV_Path *r = rt->rt_path_list.lh_first;

for(; r; r = r->path_link.le_next) {
    if (b->forward_path_lookup(r->nexthop,
r->lasthop) == NULL) {
        forward_path = r;
        break;
    }
    if ( forward_path &&
        (b->reverse_path_lookup(reverse_path-
>nexthop, reverse_path->lasthop) == NULL) ) {

        b->reverse_path_insert(reverse_path-
>nexthop, reverse_path->lasthop);
        b->forward_path_insert(forward_path-
>nexthop, forward_path->lasthop);

        if (rt->rt_advertised_hops == INFINITY)
            rt->rt_advertised_hops = rt-
>path_get_max_hopcount();

        rt->rt_error = true;
        sendReply(rq->rq_src,rt->rt_advertised_hops,
            rq->rq_dst, rt->rt_seqno,
            forward_path->expire -
CURRENT_TIME, rq->rq_timestamp, ih->saddr(),
            rq->rq_bcast_id,
            forward_path->lasthop);
    }
}

```



### A.3 Kode Fungsi sendreply()

```

Packet *p = Packet::alloc();
    struct hdr_cmh *ch = HDR_CMH(p);
    struct hdr_ip *ih = HDR_IP(p);
    struct hdr_aomdv_reply *rp =
HDR_AOMDV_REPLY(p);
    aomdv_rt_entry *rt =
rtable.rt_lookup(ipdst);

    rp->rp_type = AOMDVTYPE_RREP;
    //rp->rp_flags = 0x00;
    rp->rp_hop_count = hop_count;
    rp->rp_dst = rpdst;
    rp->rp_dst_seqno = rpseq;
    rp->rp_src = index;
    rp->rp_lifetime = lifetime;
    rp->rp_timestamp = timestamp;
    rp->rp_bcast_id = bcast_id;
    rp->rp_first_hop = rp_first_hop;
    rq->rq_min_life = iEnergy;
    rq->rq_rss = iRss;

    if(rp->rp_dst == index) {
        rp->rp_pheromone_count = rt-
>rt_pheromone_count;
    }

    // ch->uid() = 0;
    ch->ptype() = PT_AOMDV;
    ch->size() = IP_HDR_LEN + rp->size();
    ch->iface() = -2;
    ch->error() = 0;
    ch->addr_type() = NS_AF_INET;

```

```
ch->next_hop_ = nexthop;  
  
    ch->xmit_failure_ =  
aomdv_rt_failed_callback;  
    ch->xmit_failure_data_ = (void*) this;  
  
        ih->saddr() = index;  
        ih->daddr() = ipdst;  
        ih->sport() = RT_PORT;  
        ih->dport() = RT_PORT;  
        ih->ttl_ = NETWORK_DIAMETER;  
  
        Scheduler::instance().schedule(target_,  
p, 0.);
```

## A.4 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)        50;
set val(seed)      1.0;
set val(adhocRouting)  AOMDV;
set val(stop)      200;
set val(cp)        "cbr50.tcl";
set val(sc)        "scenario.tcl";

set ns_            [new Simulator]

# setup topography object

set topo          [new Topography]

# create trace object for ns and nam

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

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

```

```

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

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

# 802.11p default parameters
Phy/WirelessPhy set  RXThresh_ 5.57189e-
11 ; #400m
Phy/WirelessPhy set  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.5 Kode Konfigurasi *Traffic*

```

set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(48) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(49) $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"

set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(46) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(47) $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"

set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(44) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(45) $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"

```

```

set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(42) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(43) $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.6 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.7 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.8 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.9 Kode Skrip AWK *Node Survived Percentage***

```
BEGIN {  
    ns = 0;  
}  
awk '{ if($4 == "-n") print $5"\t"$7 }' |  
sort -n | awk '!a[$1]++' | awk  
'{if($2!=0)s++}  
ns = s/NR*100;  
END  
{  
    Print 'Percentage of Node Survived = %d',  
    ns}'
```

## BIODATA PENULIS



**SATRIA CHANDRA YUDHA WIBOWO**, lahir di Malang, 24 Januari 1998. Penulis adalah anak pertama dari tiga bersaudara. Penulis menempuh pendidikan sekolah dasar di SD Blimbing 3 Malang lalu melanjutkan pendidikan sekolah menengah pertama di SMP Negeri 3 Malang dan penulis menempuh pendidikan menengah atas di SMA Negeri 3 Malang. Selanjutnya penulis melanjutkan pendidikan sarjana

di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember. Semasa kuliah, penulis aktif dalam berbagai organisasi dan kepanitiaan baik tinggal jurusan maupun universitas.

Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Arsitektur dan Jaringan Komputer (AJK). Sebagai mahasiswa, penulis berperan aktif dalam beberapa organisasi kampus seperti staf ahli Media dan Informasi Himpunan Teknik-Computer (HMTC) ITS, staf ahli *Motion Picture* ITS EXPO 2017, kepala departemen dokumentasi Gemastik 11, dan staf SCHEMATICS 2017. Penulis saat ini bekerja sebagai Manager di CV. Cakralaksana Sejahtera. Penulis dapat dihubungi melalui email: [satriachandrayw@gmail.com](mailto:satriachandrayw@gmail.com).

