



TUGAS AKHIR - KI141502

IMPLEMENTASI *ALTERNATIVE PATH* PADA AODV DI LINGKUNGAN VANET

ILYAS BINTANG PRAYOGI
NRP 05111440000157

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

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

Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

(Halaman ini sengaja dikosongkan)

RBTC



TUGAS AKHIR - KI141502

IMPLEMENTASI *ALTERNATIVE PATH* PADA AODV DI LINGKUNGAN VANET

**ILYAS BINTANG PRAYOGI
NRP 05111440000157**

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

**Dosen Pembimbing II
Ir. F.X. Arunanto, M.Sc.**

**Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018**

(Halaman ini sengaja dikosongkan)

RBTC



UNDERGRADUATE THESES - KI141502

IMPLEMENTATION OF *ALTERNATIVE PATH* ON AODV IN VANETS ENVIRONMENT

**ILYAS BINTANG PRAYOGI
NRP 05111440000157**

First Advisor

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

Second Advisor

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

Department of Informatics

Faculty of Information and Communication Technology

Sepuluh Nopember Institute of Technology

Surabaya 2018

(Halaman ini sengaja dikosongkan)

RBTC

LEMBAR PENGESAHAN

IMPLEMENTASI *ALTERNATIVE PATH* PADA AODV DI LINGKUNGAN VANET

TUGAS AKHIR

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

Oleh:

ILYAS BINTANG PRAYOGI
NRP 05111440000157

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr.Eng. Radityo Anggoro, S.Kom, M.Sc.
(NIP. 198410162008121002) (Pembimbing 1)
2. Ir. F.X. Arunanto, M.Sc.
(NIP. 195701011983031004) (Pembimbing 2)



SURABAYA
JUNI, 2018

(Halaman ini sengaja dikosongkan)

RBTC

IMPLEMENTASI *ALTERNATIVE PATH* PADA AODV DI LINGKUNGAN VANET

Nama Mahasiswa : Ilyas Bintang Prayogi
NRP : 5114100157
Departemen : Informatika FTIK-ITS
Dosen Pembimbing 1 : Dr.Eng. Radityo Anggoro, S.Kom.,
M.Sc.
Dosen Pembimbing 2 : Ir. F.X. Arunanto, M.Sc.

Abstrak

Mobile Ad hoc Network (MANET) terdiri dari sekumpulan perangkat nirkabel (*node network*) yang bergerak secara acak dan berkomunikasi satu sama lain tanpa adanya infrastruktur yang tetap. Karena sifatnya yang dinamis, koneksi pada jaringan sering berubah-ubah, *node network* bisa bertambah atau berkurang sewaktu-waktu, dan rute terkadang rusak.

Vehicular Ad hoc Network (VANET) hampir sama dengan MANET. Hal yang membedakan adalah VANET memiliki *node network* yang berkecepatan tinggi dan pola pergerakannya terbatas.

Routing protocol berperan penting dalam mendapatkan performa jaringan yang lebih baik. Ada banyak sekali *routing protocol* yang dapat diimplementasikan pada VANET, salah satunya adalah AODV. AODV termasuk dalam *routing protocol* yang reaktif, yang artinya *node network* hanya mencari rute saat dibutuhkan saja.

AODV memiliki tiga fase, yaitu *route discovery*, *packet forwarding*, dan *route mintenance*. *Route discovery* merupakan fase pencarian rute dengan mengirim dan meneruskan *Route Request* (RREQ) secara *broadcast* untuk menemukan *destination node* dan konfirmasi dalam bentuk *Route Reply* (RREP) oleh *destination node* menuju *source node* melalui rute yang dilalui RREQ lalu rute terbentuk. *Packet forwarding* merupakan fase pengiriman paket dari *source node* ke *destination node* melalui rute

yang telah terbentuk. *Route maintenance* merupakan fase yang terjadi pada saat rute rusak dengan mengirimkan *Route Error* (RRER) kepada *source node*.

Pada Tugas Akhir ini akan dilakukan modifikasi pada fase *route discovery* dengan membuat jalur alternatif. Jalur alternatif dibuat dengan memberikan kesempatan *node* untuk meneruskan RREQ sebanyak dua kali. Sehingga terbentuk dua jalur yang berbeda setiap *nodenya*. Dari hasil uji coba, PDR meningkat sebanyak 7,25%, E2E berkurang sebanyak 56,5%, dan *packet loss* berkurang sebanyak 49,5% daripada AODV murni. Tetapi masih terdapat kelemahan pada *routing overhead* yang bertambah sebanyak 4% dan RREQ F yang bertambah sebanyak 12% daripada AODV murni.

Kata kunci: MANET, VANET, AODV, Routing Protocol.

IMPLEMENTATION OF ALTERNATIVE PATH ON AODV IN VANETS ENVIRONMENT

Student's Name : Ilyas Bintang Prayogi
Student's ID : 5114100157
Department : Informatika FTIK-ITS
First Advisor : Dr.Eng. Radityo Anggoro, S.Kom,
M.Sc.
Second Advisor : Ir. F.X. Arunanto, M.Sc.

Abstract

Mobile Ad hoc Network (MANET) consists of a set of wireless devices that move randomly and inter-connected. Due to its dynamic nature, network connections are change frequently, network nodes may increase or decrease at times, and routes may damaged.

Ad hoc Network Vehicle (VANET) is similar to MANET. The difference is that VANET has high-speed nodes and limited movement patterns.

Routing protocols are required in better network performance. There are many routing protocols that can be implemented on VANET, one of which is AODV. AODV is included in a reactive routing protocol, which means the network node only looks for the route when it's needed.

AODV has three phases: route discovery, packet forwarding, and route mintenance. Route discovery is a search phase by sending and rebroadcasting a Route Request (RREQ) to find the destination node and confirm it in the Route Reply (RREP) form by the destination node to the source node through the RREQ route. Packet forwarding is the packet delivery phase from the source node to the destination node through the established route. Route maintenance is a phase that occurs when a route is broken by sending Route Error (RRER) to the source node.

In this Final Project will be modified in the phase route discovery by creating an alternative path. An alternative path is

created by allowing the node to rebroadcast RREQ twice. So that two different path are formed every node. From the results, PDR increased by 7.25%, E2E reduced by 56.5%, and packet loss was reduced by 49.5% over pure AODV. But there is still a weakness in routing overhead that increases by 4% and RREQ F increases by 12% over pure AODV.

Keyword: MANET, VANET, AODV, Routing Protocol.

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Puji syukur kepada Allah swt, berkat segala rahmat dan hidayah-Nya penulis dapat menyelesaikan tugas akhir yang berjudul

“Implementasi *Alternative Path* pada AODV di Lingkungan VANET”

Harapan dari penulis, semoga apa yang tertulis pada buku tugas akhir ini dapat bermanfaat untuk pengembangan ilmu pengetahuan selanjutnya dan memberikan kontribusi yang nyata.

Dalam pelaksanaan dan pengerjaan tugas akhir ini sangat banyak bantuan kepada penulis dari berbagai pihak, penulis ingin berterima kasih kepada:

1. Allah SWT, karena atas izin-Nya penulis dapat menyelesaikan tugas akhir dengan baik.
2. Orang tua penulis atas segala dukungan berupa doa, bantuan moral, dan material selama penulis belajar di Departemen Informatika ITS.
3. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom., selaku Kepala Departemen Informatika ITS.
4. Bapak Dr.Eng. Radityo Anggoro, S.Kom., M.Sc. selaku Koordinator Tugas Akhir di Teknik Informatika ITS.
5. Bapak Dr.Eng. Radityo Anggoro, S.Kom., M.Sc. dan Bapak Ir. F.X. Arunanto, M.Sc. selaku pembimbing tugas akhir yang telah memberikan banyak waktu untuk berdiskusi dan arahan kepada penulis untuk menyelesaikan tugas akhir.
6. Bapak dan Ibu Dosen di Jurusan Teknik Informatika yang telah memberikan ilmu selama penulis kuliah di Teknik Informatika
7. Seluruh Staf dan karyawan Teknik Informatika yang telah memberikan bantuan selama penulis kuliah di Teknik Informatika.

8. Rekan-rekan di laboratorium Arsitektur Jaringan Komputer yang saling mendukung dan menyemangati selama pengerjaan tugas akhir.
9. Rekan-rekan sepejuangan topik Arsitektur Jaringan Komputer yang saling mendukung dan menyemangati selama pengerjaan tugas akhir.
10. Seluruh rekan-rekan satu angkatan TC 2014 yang saya banggakan.

Penulis telah berusaha sebaik-baiknya dalam menyusun buku tugas akhir ini. Namun penulis memohon maaf apabila terdapat kekurangan atau kesalahan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya. Semoga buku ini dapat memberikan manfaat pada pembaca. Aamiin.

Surabaya, 6 Juni 2018

Ilyas Bintang Prayogi

DAFTAR ISI

Abstrak.....	vii
Abstract	ix
KATA PENGANTAR	xi
DAFTAR ISI.....	xiii
DAFTAR GAMBAR.....	xvii
DAFTAR TABEL.....	xix
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Permasalahan	2
1.4 Tujuan	2
1.5 Manfaat.....	2
1.6 Metodologi	3
1.6.1 Penyusunan Proposal Tugas Akhir	3
1.6.2 Studi Literatur	3
1.6.3 Implementasi Sistem.....	3
1.6.4 Pengujian dan Evaluasi.....	4
1.6.5 Penyusunan Buku	4
1.7 Sistematika Penulisan Laporan	4
BAB II TINJAUAN PUSTAKA.....	7
2.1 <i>Mobile Ad hoc Network</i> (MANET)	7
2.2 <i>Vehicular Ad hoc Network</i> (VANET)	7
2.3 <i>Ad hoc On demand Distance Vector</i> (AODV)	7
2.4 <i>Network Simulator 2</i> (NS-2)	8
2.4.1 Instalasi.....	8
2.4.2 <i>Trace File</i>	9
2.5 OpenStreetMap.....	11
2.6 Java OpenStreetMap Editor (JOSM).....	11
2.7 Simulation of Urban Mobility (SUMO).....	12
2.8 AWK	12
BAB III PERANCANGAN.....	15
3.1 Deskripsi Umum	15
3.2 Perancangan Alternative Path pada AODV	18

3.2.1	Perancangan Mekanisme Flag RREQ.....	20
3.2.2	Perancangan Mekanisme <i>Broadcast Node</i>	21
3.3	Perancangan Skenario Mobilitas	22
3.3.1	Perancangan Skenario Grid.....	22
3.3.2	Perancangan Skenario Real.....	24
3.4	Perancangan Simulasi pada NS-2	25
3.5	Perancangan Metrik Analisis.....	26
3.5.1	<i>Packet Delivery Ratio</i> (PDR).....	26
3.5.2	<i>Average End-to-End Delay</i> (E2E).....	27
3.5.3	<i>Routing Overhead</i> (RO).....	27
3.5.4	<i>Forwarded Route Request</i> (RREQ F)	27
3.5.5	<i>Packet Loss</i>	27
BAB IV IMPLEMENTASI.....		29
4.1	Implementasi Modifikasi pada <i>Routing Protocol</i> AODV untuk Memilih <i>Forwarding Node</i>	29
4.1.1	Implementasi Mekanisme <i>Flag RREQ</i>	30
4.1.2	Implementasi Mekanisme <i>Broadcast Node</i>	31
4.2	Implementasi Skenario Mobilitas	33
4.2.1	Skenario <i>Grid</i>	33
4.2.2	Skenario <i>Real</i>	37
4.3	Implementasi Simulasi pada NS-2	38
4.4	Implementasi Metrik Analisis	40
4.4.1	Implementasi <i>Packet Delivery Ratio</i>	41
4.4.2	Implementasi Rata-Rata <i>End-to-End Delay</i>	41
4.4.3	Implementasi <i>Routing Overhead</i>	41
4.4.4	Implementasi <i>Forwarded Route Request</i>	41
4.4.5	Implementasi <i>Packet Loss</i>	42
BAB V UJI COBA DAN EVALUASI.....		43
5.1	Lingkungan Uji Coba	43
5.2	Hasil Uji Coba	43
5.2.1	Hasil Uji Coba Skenario <i>Grid</i>	44
5.2.2	Hasil Uji Coba Skenario <i>Real</i>	52
BAB VI KESIMPULAN DAN SARAN.....		61
6.1	Kesimpulan.....	61
6.2	Saran.....	61

DAFTAR PUSTAKA	63
LAMPIRAN	65
A.1 Kode Fungsi AODV::recvRequest ()	65
A.2 Kode Skenario NS-2.....	69
A.3 Kode Konfigurasi <i>Traffic</i>	72
A.4 Kode Skrip AWK <i>Packet Delivery Ratio</i>	72
A.5 Kode Skrip AWK Rata-Rata <i>End-to-End Delay</i>	73
A.6 Kode Skrip AWK <i>Routing Overhead</i>	74
A.7 Kode Skrip AWK <i>Forwarded Route Request</i>	75
A.8 Kode Skrip AWK <i>Packet Loss</i>	75
BIODATA PENULIS	77

(Halaman ini sengaja dikosongkan)

RBTC

DAFTAR GAMBAR

Gambar 3.1 Diagram Rancangan Simulasi dengan Routing Protocol AODV Murni.....	16
Gambar 3.2 Diagram Rancangan Simulasi dengan Routing Protocol AODV Modifikasi	17
Gambar 3.3 AODV dengan Alternative Path.....	18
Gambar 3.4 AODV tanpa Alternative Path.....	19
Gambar 3.5 Diagram Alir Alternative Path.....	20
Gambar 3.6 Pseudocode Mekanisme Flag RREQ pada Node.....	21
Gambar 3.7 Pseudocode Mekanisme Broadcast Node.....	22
Gambar 3.8 Alur Pembuatan Skenario Mobilitas Grid	23
Gambar 3.9 Alur Pembuatan Skenario Mobilitas Real	25
Gambar 4.1 Potongan kode untuk deklarasi variable flag.....	30
Gambar 4.2 Potongan kode untuk untuk mekanisme flag.....	30
Gambar 4.3 Potongan kode deklarasi variable broadcast node ...	31
Gambar 4.4 Potongan kode untuk untuk mekanisme broadcast node	32
Gambar 4.5 Potongan kode update neighbor	33
Gambar 4.6 Kode sumber netgenerate	34
Gambar 4.7 Hasil Generate Peta Grid	34
Gambar 4.8 Perintah randomTrips	35
Gambar 4.9 Perintah duarouter	35
Gambar 4.10 File Skrip .sumocfg	36
Gambar 4.11 Perintah SUMO	36
Gambar 4.12 Perintah traceExporter	37
Gambar 4.13 Ekspor Peta dari OpenStreetMap	37
Gambar 4.14 Perintah netconvert	38
Gambar 4.15 Hasil Konversi Peta Real.....	38
Gambar 4.16 Konfigurasi Lingkungan Simulasi.....	39
Gambar 4.17 Konfigurasi Traffic	40
Gambar 5.1 PDR Skenario Grid.....	45
Gambar 5.2 E2E Delay Skenario Grid	46
Gambar 5.3 Hasil Routing Overhead Skenario Grid.....	48
Gambar 5.4 Hasil RREQ F Skenario Grid	49
Gambar 5.5 Hasil Packet Loss Skenario Grid	51

Gambar 5.6 Hasil PDR Skenario Real.....53

Gambar 5.7 E2E Delay Skenario Real54

Gambar 5.8 Hasil Routing Overhead Skenario Real56

Gambar 5.9 Hasil RREQ F Skenario Real57

Gambar 5.10 Hasil Packet Loss Skenario Real59

RBTC

DAFTAR TABEL

Tabel 2.1 Penjelasan Trace File NS-2	9
Tabel 3.1 Daftar Istilah.....	17
Tabel 3.2 Parameter Lingkungan Simulasi dengan Skenario.....	26
Tabel 5.1 Spesifikasi Perangkat yang Digunakan	43
Tabel 5.2 PDR Skenario Grid.....	44
Tabel 5.3 E2E Delay Skenario Grid.....	46
Tabel 5.4 Routing Overhead Skenario Grid.....	47
Tabel 5.5 Hasil RREQ F Skenario Grid	49
Tabel 5.6 Hasil Packet Loss Skenario Grid.....	50
Tabel 5.7 Hasil PDR Skenario Real	52
Tabel 5.8 E2E Delay Skenario Real	54
Tabel 5.9 Routing Overhead Skenario Real	55
Tabel 5.10 Hasil RREQ F Skenario Real	57
Tabel 5.11 Hasil Packet Loss Skenario Real.....	58

(Halaman ini sengaja dikosongkan)

RBTC

BAB I

PENDAHULUAN

Bab ini membahas tentang latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir. Harapannya penjelasan pada bab ini dapat menjadi gambaran tugas akhir secara umum.

1.1 Latar Belakang

Mobile Ad hoc Network (MANET) terdiri dari sekumpulan perangkat nirkabel yang bergerak secara acak dan berkomunikasi satu sama lain tanpa adanya infrastruktur yang tetap. Karena sifatnya yang dinamis, koneksi pada jaringan sering berubah-ubah, *node network* bisa bertambah atau berkurang sewaktu-waktu, dan rute terkadang rusak.

Vehicular Ad hoc Network (VANET) hampir sama dengan MANET. Hal yang membedakan adalah VANET memiliki *node network* yang berkecepatan tinggi dan pola pergerakannya terbatas.

Dewasa ini, VANET menjadi topik hangat para peneliti dan pengembang. Banyak *routing protocol* yang dikembangkan saat ini. *Routing protocol* berperan penting dalam mendapatkan performa jaringan yang lebih baik. Untuk menemukan *routing protocol* yang efektif para peneliti memodifikasi *routing protocol* yang telah ada. Salah satu dari modifikasi *routing protocol* yang akan diangkat menjadi topik di tugas akhir ini adalah *alternative path* AODV (AODV-AP) yang merupakan pengembangan dari AODV. AODV-AP mengembangkan AODV dengan memberikan jalur alternatif pada setiap *node network* pada jaringan. Sehingga ketika jalur utama rusak terdapat jalur alternatif yang dapat digunakan, hal itu lebih baik daripada mencari rute baru yang dapat membebani jaringan dan menambah *delay*. Dengan modifikasi ini

diharapkan *routing protocol* yang dimaksudkan dapat meningkatkan kinerja routing pada lingkungan VANET.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana implementasi *alternative path* pada AODV di lingkungan VANET?
2. Bagaimana dampak penambahan *alternative path* pada AODV di lingkungan VANET?

1.3 Batasan Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Lingkungan yang digunakan adalah jaringan VANET.
2. *Routing protocol* yang digunakan merupakan modifikasi dari AODV yaitu *Alternative Path AODV* (AODV-AP).
3. Pengujian jaringan dilakukan menggunakan aplikasi *Network Simulator 2* (NS-2).
4. Pembuatan skenario uji coba menggunakan *Simulation of Urban Mobility* (SUMO).

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah meningkatkan kinerja dengan metode *alternative path* pada AODV di lingkungan VANET.

1.5 Manfaat

Pengerjaan tugas akhir ini dilakukan dengan harapan dapat menjadi pedoman penelitian untuk mengembangkan metode

penambahan *alternative path* pada AODV di lingkungan VANET di masa yang akan datang.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahap awal yang dilakukan dalam pengerjaan tugas akhir ini adalah penyusunan proposal tugas akhir. Di dalam proposal diajukan suatu gagasan untuk melakukan implementasi *alternative path* pada AODV di lingkungan VANET. Proposal tugas akhir berisi pendahuluan, deskripsi, dan gagasan metode-metode yang dibuat dalam tugas akhir ini. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, manfaat dan tujuan dari hasil pembuatan tugas akhir ini. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

1.6.2 Studi Literatur

Pada tahap ini, dilakukan pencarian dan pembelajaran literatur yang berhubungan dengan *routing protocol* AODV, *Vehicular Ad Hoc Networks* (VANET), *Network Simulator 2* (NS2), *Simulation of Urban Mobility* (SUMO), Java OpenStreetMap (JOSM), dan AWK. Literatur yang digunakan berupa: buku referensi, jurnal, dan dokumentasi internet.

1.6.3 Implementasi Sistem

Pada tahap ini dilakukan implementasi metode yang telah diajukan. Untuk mengimplementasikan metode penulis menggunakan NS-2 sebagai alat untuk mensimulasikan jaringan, bahasa C/C++ sebagai bahasa pemrograman, SUMO dan JOSM sebagai perangkat lunak untuk membuat topologi lingkungan VANET.

1.6.4 Pengujian dan Evaluasi

Pada tahap ini dilakukan pengujian dengan SUMO untuk membuat topologi lingkungan VANET. Kemudian topologi yang telah dibuat dijalankan dengan NS-2 sehingga menghasilkan *trace file*. Lalu dari *trace file* bisa dianalisis performanya.

1.6.5 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku sebagai dokumentasi dari pelaksanaan tugas akhir yang mencakup seluruh konsep, teori, implementasi, hasil yang telah dikerjakan, dan kesimpulan.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

1. Bab I. Pendahuluan

Bab ini membahas tentang latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

2. Bab II. Tinjauan Pustaka

Bab ini membahas tentang teori, metode, algoritma, dan *tools* yang digunakan dalam pengerjaan tugas akhir ini.

3. Bab III. Perancangan

Bab ini membahas tentang perancangan implementasi sistem yang akan dibuat pada tugas akhir ini. Bagian yang dibahas

pada bab ini berawal dari deskripsi umum, perancangan skenario, perancangan simulasi, dan perancangan metrik analisis.

4. Bab IV. Implementasi

Pada bab ini membahas tentang implementasi dari perancangan yang sudah dilakukan pada bab perancangan. Implementasi berupa kode sumber untuk membangun program.

5. Bab V. Pengujian dan Evaluasi

Bab ini membahas tentang uji coba dan evaluasi dari implementasi yang telah dijelaskan pada bab implementasi. Hasil uji coba kemudian dievaluasi sehingga menghasilkan kesimpulan pada bab selanjutnya.

6. Bab VI. Kesimpulan dan Saran

Bab ini membahas tentang kesimpulan yang diperoleh dari tugas akhir yang telah dikerjakan dan saran untuk pengembangan tugas akhir ini di masa depan.

7. Daftar Pustaka

Pada *point* ini berisi daftar pustaka yang dijadikan literatur dalam Tugas Akhir.

8. Lampiran

Pada *point* ini dilampirkan kode sumber program yang dipakai.

(Halaman ini sengaja dikosongkan)

RBTC

BAB II

TINJAUAN PUSTAKA

Bab ini membahas tentang teori, metode, algoritma, dan *tools* yang digunakan dalam pengerjaan tugas akhir ini. Pembahasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan riset yang berkaitan.

2.1 *Mobile Ad hoc Network (MANET)*

Mobile Ad hoc Network (MANET) terdiri dari sekumpulan perangkat nirkabel yang bergerak secara acak dan berkomunikasi satu sama lain tanpa adanya infrastruktur yang tetap. Karena sifatnya yang dinamis, koneksi pada jaringan sering berubah-ubah, *node network* bisa bertambah atau berkurang sewaktu-waktu, dan rute terkadang rusak.

2.2 *Vehicular Ad hoc Network (VANET)*

Vehicular Ad hoc Network (VANET) hampir sama dengan MANET. Hal yang membedakan adalah VANET memiliki *node* yang berkecepatan tinggi dan pola pergerakannya terbatas.

Routing protocol berperan penting dalam mendapatkan performa jaringan yang lebih baik. Ada banyak sekali *routing protocol* yang dapat diimplementasikan pada VANET, salah satunya adalah AODV. AODV termasuk dalam *routing protocol* yang reaktif, yang artinya *node network* hanya mencari rute saat dibutuhkan saja.

2.3 *Ad hoc On demand Distance Vector (AODV)*

Ad-hoc On demand Distance Vector (AODV) adalah salah satu *routing protocol* yang dibuat khusus untuk jaringan Ad hoc yang bergerak. AODV termasuk dalam *routing protocol* yang

reaktif, artinya *routing protocol* hanya mencari rute pada saat membutuhkan saja.

AODV memiliki tiga fase, yaitu *route discovery*, *packet forwarding*, dan *route maintenance*. *Route discovery* merupakan fase pencarian rute dengan mengirim dan meneruskan *Route Request* (RREQ) secara *broadcast* untuk menemukan *destination node* dan konfirmasi dalam bentuk *Route Reply* (RREP) oleh *destination node* menuju *source node* melalui rute yang dilalui RREQ lalu rute terbentuk. *Packet forwarding* merupakan fase pengiriman paket dari *source node* ke *destination node* melalui rute yang telah terbentuk. *Route maintenance* merupakan fase yang terjadi pada saat rute rusak dengan mengirimkan *Route Error* (RREP) kepada *source node*.

2.4 Network Simulator 2 (NS-2)

NS-2 adalah sebuah perangkat lunak yang digunakan untuk mensimulasikan jaringan. NS-2 menggunakan dua bahasa utama, yaitu bahasa C/C++ dan *Object-oriented Tool Command Language* (OTCL). Pada NS-2, C++ digunakan untuk mendefinisikan mekanisme internal dari objek simulasi seperti *routing protocol*, sedangkan OTCL digunakan untuk mendefinisikan lingkungan atau topologi. Setelah simulasi dilakukan NS-2 memberikan simulasi berupa *file* NAM dan *trace file*.

Pada Tugas Akhir ini, NS-2 digunakan sebagai simulator untuk mensimulasikan lingkungan VANETs menggunakan *routing protocol* AODV yang sudah dimodifikasi. *Trace file* yang dihasilkan oleh NS-2 juga digunakan sebagai informasi untuk mengukur performa *routing protocol* AODV yang sudah dimodifikasi.

2.4.1 Instalasi

Berikut ini adalah langkah-langkah instalasi NS-2:

1. Install beberapa *dependencies package* yang harus dipasang terlebih dahulu sebelum memulai instalasi NS-2. Berikut ini adalah *command* untuk menginstall *dependencies package*:

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

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 perintah berikut:

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

3. Mulai instalasi NS-2 dengan menjalankan perintah *./install* pada folder NS-2.

2.4.2 Trace File

Trace file merupakan *file* yang dihasilkan NS-2 ketika selesai melakukan simulasi. *Trace file* berisi informasi detail alur pengiriman paket data. *Trace file* digunakan untuk menganalisis performa *routing protocol* yang disimulasikan. Detail penjelasan *trace file* ditunjukkan pada Tabel 2.1

Tabel 2.1 Penjelasan Trace File NS-2

Kolom	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>

Kolom	Penjelasan	Isi
3	<i>ID Node</i>	<i>_x_</i> : identifier dari <i>node</i> yang melakukan <i>event</i>
4	<i>Layer</i>	<i>Network layer</i> tempat terjadi <i>event</i> AGT: <i>application</i> RTR: <i>routing</i> LL: <i>link layer</i> IFQ: <i>packet queue</i> MAC: <i>MAC</i> PHY: <i>physical</i>
5	<i>Flag</i>	---: Tidak ada
6	<i>Sequence Number</i>	<i>Sequence number</i> dari paket atau nomor urut paket
7	Tipe Paket	AODV: paket <i>routing AODV</i> 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: <i>MAC ACK</i> 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 asal d: <i>IP header</i>
10	<i>Flag</i>	-----: Tidak ada
11	Detail <i>IP source</i> , <i>destination</i> , dan <i>nexthop</i>	[a:b c:d e f] a: <i>IP source node</i> b: <i>port source node</i>

Kolom	Penjelasan	Isi
		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>next hop</i> (jika 0 berarti <i>node</i> 0 atau <i>broadcast</i>)

2.5 OpenStreetMap

OpenStreetMap (OSM) adalah proyek web untuk membuat peta yang bebas dan terbuka dari peta seluruh dunia. OpenStreetMap dibangun sepenuhnya oleh sukarelawan yang melakukan survey dengan GPS, *digitizing aerial imagery*, dan mengumpulkan serta membebaskan sumber data geografis publik yang ada.

Menggunakan Lisensi Open Database Commons Open Database 1.0, kontributor OSM dapat memiliki, memodifikasi, dan berbagi data pemetaan ke publik. Ada banyak pilihan peta digital yang tersedia di internet, tetapi kebanyakan dari mereka memiliki batasan legal atau teknis. Ini menyulitkan orang, juga bagi pemerintah, peneliti dan akademisi, inovator, dan banyak pemangku kepentingan lainnya untuk secara bebas dan terbuka menggunakan data yang tersedia di peta. Di sisi lain, peta dasar dan data di OSM dapat diunduh untuk penggunaan dan redistribusi lebih lanjut.

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

2.6 Java OpenStreetMap Editor (JOSM)

Java OpenStreetMap Editor (JOSM) adalah aplikasi yang dikembangkan oleh Immanuel Scholz. Aplikasi JOSM digunakan untuk menyunting data yang didapatkan dari OpenStreetMap. Aplikasi JOSM dapat diunduh pada alamat web <https://josm.openstreetmap.de>. Penulis menggunakan aplikasi ini untuk menyunting dan merapikan peta yang diunduh dari OpenStreetMap.

Pada Tugas Akhir ini, penulis menggunakan data yang telah diambil dari OSM lalu disunting. Map yang disunting yaitu menghilangkan dan juga menyambungkan jalan yang ada, dan menghilangkan gedung-gedung yang ada di map.

2.7 Simulation of Urban Mobility (SUMO)

Simulation of Urban Mobility atau disingkat SUMO merupakan sebuah aplikasi simulator lalu lintas jalan raya yang *open source*, *microscopic*, dan *multi-modal*. SUMO mensimulasikan bagaimana pergerakan lalu lintas yang terdiri dari beberapa kendaraan yang berjalan pada jalan raya yang telah ditentukan. Simulasi SUMO dapat menunjukkan beberapa topik manajemen lalu lintas dalam skala besar. SUMO murni *microscopic*, yang artinya setiap kendaraan dimodelkan secara eksplisit, memiliki rutenya sendiri, dan bergerak secara individu dalam jaringan.

Pada Tugas Akhir ini, penulis menggunakan SUMO untuk membuat skenario VANETs, peta area simulasi, dan pergerakan *node* sehingga menyerupai keadaan sebenarnya lalu lintas. Pada pembuatan skenario VANETs di tugas akhir ini. SUMO akan menghasilkan pergerakan *node* yang acak sehingga tidak akan sama setiap skenarionya.

2.8 AWK

AWK adalah bahasa pemrograman yang dirancang untuk pemrosesan teks dan biasanya digunakan untuk ekstraksi data dan

alat pelaporan. AWK merupakan fitur standar dari kebanyakan sistem operasi Unix-like.

Bahasa AWK adalah bahasa skrip *data-driven* yang terdiri dari serangkaian tindakan yang harus diambil terhadap aliran data tekstual. AWK dapat dijalankan langsung pada *file* atau digunakan sebagai bagian dari *pipeline* dengan tujuan mengekstrak atau mengubah teks, seperti menghasilkan diformat laporan.

Pada tugas akhir ini AWK digunakan untuk mengekstrak data dari *trace file* menghasilkan data performa dari *routing protocol* modifikasi.

(Halaman ini sengaja dikosongkan)

RBTC

BAB III

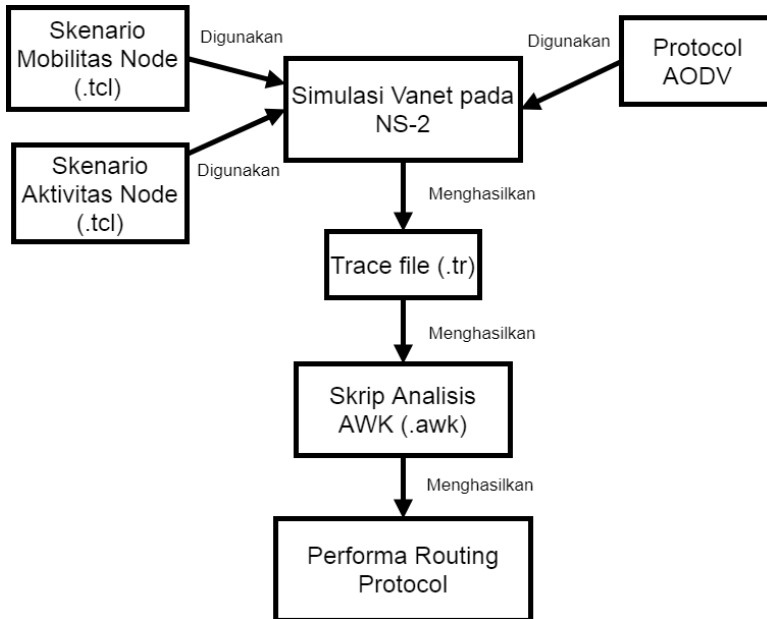
PERANCANGAN

Bab ini membahas tentang perancangan implementasi sistem yang akan dibuat pada tugas akhir ini. Bagian yang dibahas pada bab ini berawal dari deskripsi umum, perancangan skenario, perancangan simulasi, dan perancangan metrik analisis.

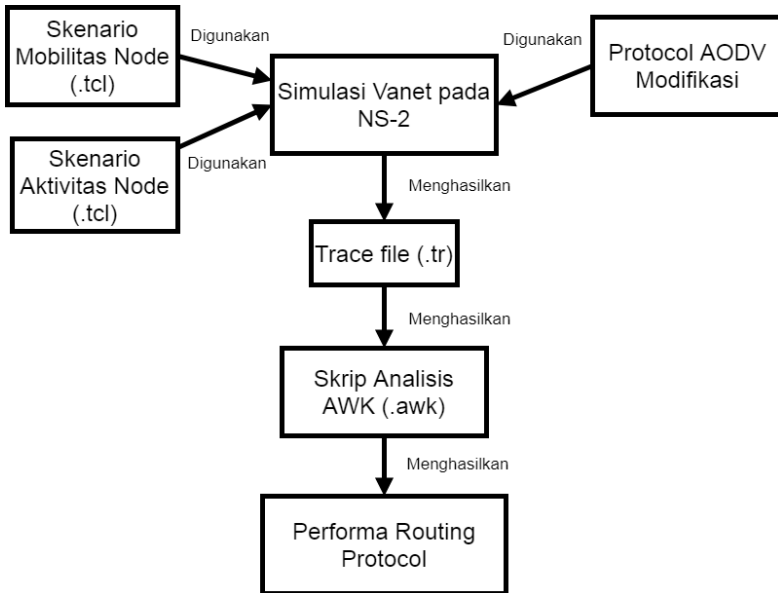
3.1 Deskripsi Umum

Pada Tugas Akhir ini penulis akan mengimplementasikan *routing protocol* AODV yang dimodifikasi dengan menambahkan jalur alternatif. *Routing protocol* yang diajukan pada tugas akhir ini merupakan modifikasi dari *routing protocol* AODV dengan menambahkan jalur alternatif. Setiap *node* akan memelihara dua jalur yang berbeda dari *node* sumber ke *node* tujuan. Penerusan RREQ juga dibatasi dengan variabel *broadcast node*. Hal ini dilakukan untuk mendapatkan jalur alternatif yang unik.

Modifikasi *routing protocol* AODV ini akan disimulasikan menggunakan NS-2 dengan skenario lingkungan VANET yang dibuat menggunakan *tools* SUMO. Simulasi yang dilakukan menghasilkan *trace file*, kemudian dianalisis menggunakan AWK untuk mendapatkan performa *routing protocol* yang telah dimodifikasi. Performa dinilai berdasarkan: *packet delivery ratio* (PDR), *packet loss*, *average end-to-end delay* (E2E), *routing overhead* (RO) dan *forwarded route request* (RREQ F). Lalu dibandingkan dengan performa *routing protocol* AODV asli. Diagram rancangan simulasi dengan *routing protocol* AODV asli dan AODV yang telah dimodifikasi dapat dilihat Gambar 3.1 dan Gambar 3.2.



Gambar 3.1 Diagram Rancangan Simulasi dengan Routing Protocol AODV Murni



Gambar 3.2 Diagram Rancangan Simulasi dengan Routing Protocol AODV Modifikasi

Daftar istilah yang sering digunakan pada buku Tugas Akhir ini dapat dilihat pada Tabel 3.1.

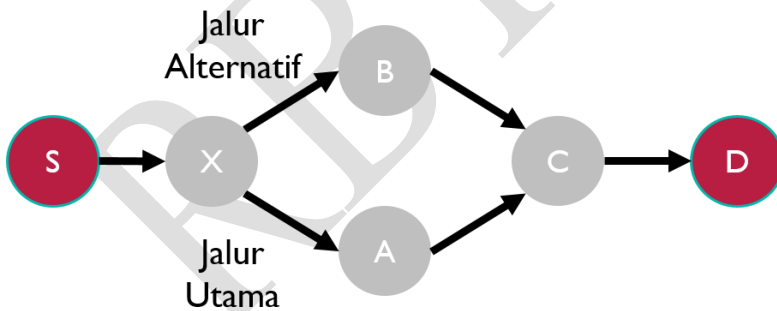
Tabel 3.1 Daftar Istilah

No.	Istilah	Penjelasan
1	AODV	<i>Ad hoc On demand Distance Vector</i>
2	PDR	<i>Packet Delivery Ratio</i>
3	E2E	<i>Average End-to-End Delay</i>
4	RO	<i>Routing Overhead</i>
5	RREQ	<i>Route Request</i>
6	RREP	<i>Route Reply</i>
7	AODV-AP	<i>AODV with Alternative Path</i>

No.	Istilah	Penjelasan
8	<i>Broadcast Node</i>	Field baru RREQ yang berisi informasi <i>list id node</i> yang meneruskan RREQ dan tentangnya
9	<i>Flag</i>	Variabel yang disimpan di <i>node</i> untuk membatasi penerusan RREQ sebanyak dua kali

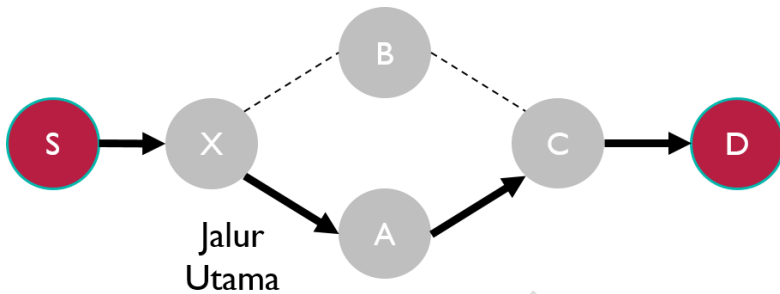
3.2 Perancangan Mekanisme Alternative Path

Alternative path atau jalur alternatif merupakan jalur cadangan yang digunakan ketika jalur utama rusak. Setiap *node* akan memelihara dua jalur yang berbeda dari *node* sumber ke *node* tujuan. Seperti pada Gambar 3.3 pada kasus *node X*, *Node X* memiliki jalur utama untuk menuju ke *node D* melalui A, dan juga memiliki jalur alternatif untuk menuju ke *node D* melalui B.



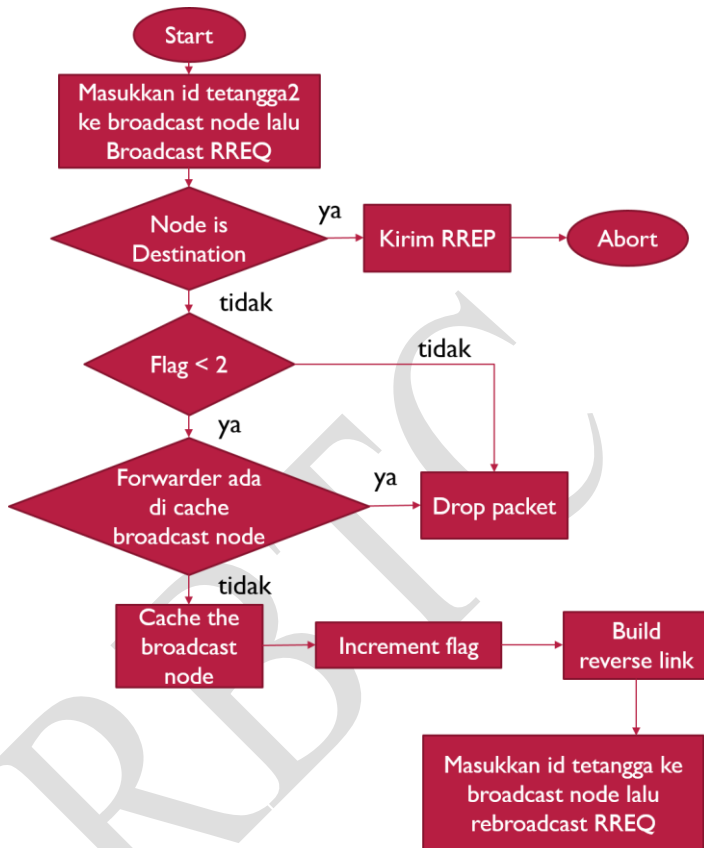
Gambar 3.3 AODV dengan Alternative Path

Berbeda dengan AODV murni yang hanya memiliki 1 rute saja. Seperti pada Gambar 3.4.



Gambar 3.4 AODV tanpa Alternative Path

Jalur alternatif dibuat dengan mengimplementasikan flag RREQ dan *broadcast node*, mekanismenya ditunjukkan pada diagram alur yang ditunjukkan pada Gambar 3.5.



Gambar 3.5 Diagram Alir Alternative Path

3.2.1 Perancangan Mekanisme Flag RREQ

Ketika mekanisme *route discovery* dilakukan, *node* sumber akan *mbroadcast* route request (RREQ) pada jaringan untuk mencari *node* tujuan. Setiap *node* yang menerima RREQ akan meneruskan kepada tetangganya. *Node* akan mengeblok bila sudah dua kali meneruskan RREQ yang sama. Sehingga *node* mempunyai dua rute untuk menuju ke *node* tujuan. Ketika *node*

mendapatkan sebuah RREQ baru, maka flag akan direset. *Pseudocode* untuk mekanisme flag pada setiap *node* dapat dilihat pada Gambar 3.6.

```

If (old RREQ)
    If flag_RREQ < 2
        Build reverse link
        Rebroadcast
        Increment flag
    Else
        Reset flag

```

Gambar 3.6 *Pseudocode Mekanisme Flag RREQ pada Node*

3.2.2 Perancangan Mekanisme Broadcast Node

Penerusan RREQ juga dibatasi dengan variabel *broadcast node*. Hal ini dilakukan untuk mendapatkan rute alternatif yang unik. Maksudnya, rute berbeda dan tidak bertumpu pada satu *node*, sehingga ketika rute rusak pada satu *node* tersebut rute alternatif tidak percuma. Variabel *broadcast node* merupakan field baru yang ditambahkan pada RREQ. *Broadcast node* berisi informasi list id tetangga *node* yang meneruskan RREQ. Jadi, ketika *node* akan meneruskan RREQ kepada tetangganya dia akan menyertakan list id *node* tetangganya. Ketika sebuah *node* pertama kali menerima RREQ maka *node* tersebut akan menyimpan sementara (cache) *broadcast node* yang ada didalam RREQ. Ketika menerima lagi RREQ yang sama, dia akan mengecek apakah *node* yang meneruskan RREQ ada di list *broadcast node* atau tidak. Jika iya, RREQ akan di drop. Jika tidak ada, RREQ akan diteruskan dan dibuat reverse link/route ke *node* yang meneruskan RREQ. Mekanisme *broadcast node* ketika menerima RREQ. Perintah *pseudocode* dapat dilihat pada Gambar 3.7:

```

1. If (old RREQ)
2.   if (forwarder_RREQ is in
      broadcast_node)
3.     drop RREQ
4.   else
5.     append it's id neighbor to
      RREQ broadcast_node
6.     build reverse link
7.     rebroadcast RREQ
8. Else
9.   Clear cache broadcast_node
10.  Cache new broadcast_node

```

Gambar 3.7 Pseudocode Mekanisme Broadcast Node

3.3 Perancangan Skenario Mobilitas

Ada dua jenis skenario yang dibuat pada tugas akhir ini, yaitu skenario grid dan skenario real. Skenario grid merupakan skenario yang sederhana, karena jalanan berbentuk kotak-kotak. Skenario real merupakan skenario yang diambil dari peta asli.

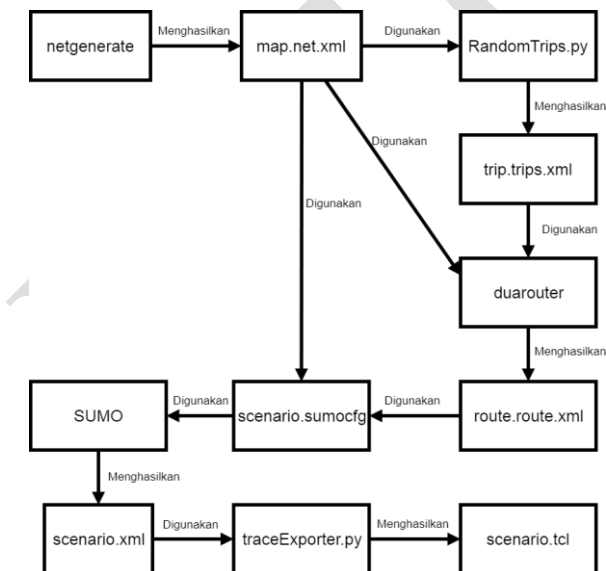
3.3.1 Perancangan Skenario Grid

Untuk membuat skenario grid perlu ditentukan beberapa parameter, yaitu berapa luas per kotak dan titik persimpangan. Pada tugas akhir ini luas per kotaknya 400 m² dan 4 titik persimpangan. Sehingga terdapat 9 petak dan luas topologi sebesar 1300 m x 1300 m.

Selain parameter yang telah ditentukan sebelumnya, perlu ditentukan juga kecepatan kendaraan. Skenario grid yang telah ditentukan kemudian dibuat dengan menggunakan fitur SUMO yaitu netgenerate menghasilkan *file .net.xml*. Lalu untuk membuat pergerakan *node* menggunakan fitur randomTrips dan duarouter.

Skenario *grid* yang telah ditentukan kemudian dibuat dengan menggunakan fitur SUMO yaitu *netgenerate*. Selain parameter yang telah ditentukan sebelumnya, dibutuhkan juga penentuan kecepatan kendaraan. Skenario *grid* yang dihasilkan oleh *netgenerate* memiliki ekstensi *.net.xml*. Peta *grid* ini kemudian digunakan untuk membuat pergerakan *node* dengan fitur SUMO yaitu *randomTrips* dan *duarouter*.

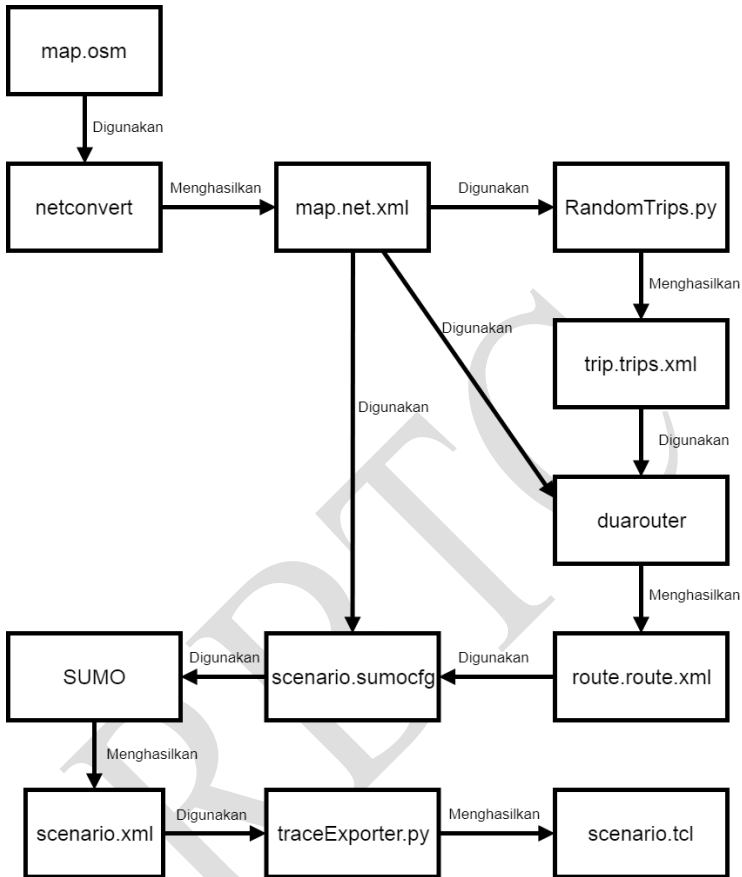
Skenario mobilitas *grid* dihasilkan dengan menggabungkan *file* peta *grid* *map.net.xml* dan *file* pergerakan *node* yang telah degenerate *route.route.xml*, yang akan menghasilkan *file* dengan ekstensi *.xml*. Lalu *file* *.xml* dikonversi dengan fitur *traceExporter* menjadi *file* *.tcl* agar dapat dijalankan di NS-2. Alur pembuatan skenario *grid* dapat dilihat pada Gambar 3.8.



Gambar 3.8 Alur Pembuatan Skenario Mobilitas Grid

3.3.2 Perancangan Skenario Real

Langkah awal untuk membuat skenario mobilitas real adalah memilih area peta untuk dijadikan lingkungan skenario. Pada tugas akhir ini, penulis menggunakan OpenStreetMap untuk mengambil area peta. Area yang diambil diexport menjadi *file* .osm. Lalu *file* tersebut konversi menggunakan fitur netconvert pada SUMO sehingga menghasilkan *file* .net.xml. generate seperti pada skenario grid. Tahap berikutnya memiliki tahapan yang sama seperti merancang skenario mobilitas *grid*. Alur perancangan skenario mobilitas *real* dapat dilihat pada Gambar 3.9.



Gambar 3.9 Alur Pembuatan Skenario Mobilitas Real

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 .tcl yang berisi konfigurasi lingkungan simulasi. Konfigurasi lingkungan simulasi VANETs pada NS-2 dapat dilihat pada Tabel 3.2.

Tabel 3.2 Parameter Lingkungan Simulasi dengan Skenario

No.	Parameter	Spesifikasi
1.	<i>Network Simulator</i>	NS-2 versi 2.35
2.	<i>Routing Protocol</i>	AODV, AODV-AP
3.	Waktu Simulasi	200, 300, 500
4.	Area Simulasi	1300 m x 1300 m dan 1500 m x 1500 m
5.	Banyak Kendaran	60, 100, 150, 200, 300
6.	Agen Pengirim	<i>Constant Bit Rate (CBR)</i>
7.	<i>Protocol MAC</i>	IEEE 802.11
8.	Propagasi sinyal	<i>Two-ray ground</i>
9.	<i>Source/Destination</i>	Statis
10.	Tipe Kanal	<i>Wireless Channel</i>

3.5 Perancangan Metrik Analisis

Berikut ini adalah parameter-parameter yang akan dianalisis pada tugas akhir ini untuk dapat membandingkan performa dari *routing protocol* AODV yang telah dilakukan modifikasi dan *routing protocol* AODV yang asli:

3.5.1 *Packet Delivery Ratio* (PDR)

Packet Delivery Ratio (PDR) merupakan persentase antara jumlah paket yang berhasil diterima oleh *node* tujuan dengan jumlah paket yang dikirimkan oleh *node* sumber. PDR dihitung dengan persamaan 3.1.

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

PDR menunjukkan seberapa besar keberhasilan paket yang dikirimkan sampai ke tujuan. Semakin tinggi PDR, artinya semakin banyak pengiriman paket yang berhasil sampai ke tujuan.

3.5.2 Average End-to-End Delay (E2E)

E2E merupakan rata-rata dari *delay* atau waktu yang dibutuhkan setiap paket untuk sampai ke *node* tujuan dalam satuan milidetik. *Delay* tiap paket didapatkan dari rentang waktu antara *node* asal mengirimkan paket (CBRSentTime) dan *node* tujuan menerima paket (CBRRecvTime). E2E dihitung dengan menjumlahkan semua *delay* paket lalu dibagi jumlah paket yang berhasil diterima (recvnum), seperti yang ditunjukkan pada persamaan 3.2.

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

3.5.3 Routing Overhead (RO)

Routing Overhead adalah jumlah *routing packet* yang diperlukan pada jaringan. RO didapatkan dengan menjumlahkan *routing packet* yang diperlukan saat simulasi, terdiri dari: RREQ, RREP, RRER, dan *hello message*. Perhitungan *Routing Overhead* dapat dilihat dengan persamaan 3.3.

$$RO = \sum_{m=1}^{sent \text{ and forwarded num}} packet \text{ sent} \quad (3.3)$$

3.5.4 Forwarded Route Request (RREQ F)

RREQ F adalah jumlah paket kontrol *route request* yang diforward selama simulasi. Jadi RREQ F dapat dihitung dengan menjumlahkan semua paket kontrol route request pada saat itu. Perhitungan *Forwarded Route Request* dapat dilihat dengan persamaan 3.4

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

3.5.5 Packet Loss

Packet loss adalah jumlah paket yang telah dikirim oleh *source node* tetapi gagal mencapai *destination node*. *Packet loss* bisa disebabkan rute yang rusak, sehingga *packet* tidak dapat sampai. Perhitungan *Forwarded Route Request* dapat dilihat dengan persamaan 3.5

$$Packet\ Loss = received - sent \quad (3.5)$$

BAB IV IMPLEMENTASI

Pada bab ini membahas tentang implementasi dari perancangan yang sudah dilakukan pada bab perancangan. Implementasi berupa kode sumber untuk membangun program.

4.1 Implementasi Modifikasi pada *Routing Protocol AODV* untuk Memilih *Forwarding Node*

Routing protocol AODV merupakan salah satu *routing protocol* yang umum digunakan dalam simulasi VANETs. Tetapi, *routing protocol* ini belum cukup optimal. Karenanya penulis mengajukan modifikasi ini.

Routing protocol yang diajukan pada tugas akhir ini merupakan modifikasi dari *routing protocol* AODV dengan menambahkan rute alternatif. Setiap *node* akan memelihara dua rute yang berbeda dari *node* sumber ke *node* tujuan. Penerusan RREQ juga dibatasi dengan variabel *broadcast node*. Hal ini dilakukan untuk mendapatkan rute alternatif yang unik.

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

- Implementasi Mekanisme *Flag_RREQ*
- Implementasi Mekanisme *Broadcast_Node*

Kode implementasi dari *routing protocol* AODV pada NS-2 versi 2.35 berada pada direktori ns-2.35/aodv. Di dalam direktori tersebut terdapat beberapa *file* diantaranya seperti *aodv.h*, *aodv.cc*, *aodv_packet.h*, dan sebagainya. Pada Tugas Akhir ini penulis hanya memodifikasi *file* *aodv.cc*. Pada bagian ini penulis akan menjelaskan langkah-langkah dalam mengimplementasikan modifikasi *routing protocol* AODV.

4.1.1 Implementasi Mekanisme *Flag RREQ*

Implementasi yang mekanisme *flag RREQ* seperti yang telah dirancang pada subbab 3.3.1 adalah sebagai berikut:

1. Deklarasi variable

Variable flag RREQ digunakan sebagai penghitung sebuah *node* sudah melakukan *rebroadcast* berapa kali. Sehingga bisa dideklarasikan sebagai integer. Lalu untuk dapat membuat flag setiap *node*. Flag RREQ berupa array. Deklarasi variable ini dilakukan di *file* *aodv.cc*. Kode sumbernya bisa dilihat pada Gambar 4.1:

```
int flag_RREQ[302];
```

Gambar 4.1 Potongan kode untuk deklarasi variable flag

2. Pengubahan kode sumber untuk mekanisme *flag RREQ*

Lalu dilakukan penambahan kode sumber bagaimana flag RREQ berjalan. Pengubahan dilakukan di *file* *aodv.cc* dan pada fungsi *AODV::recvRequest()*. Kode sumbernya bisa dilihat pada Gambar 4.2.

```
if (id_lookup(rq->rq_src, rq->rq_bcast_id)) {
    if(flag_RREQ[index]<2){
        flag_RREQ[index]++;
    }
    else{
        Packet::free(p);
        return;
    }
}
else{
    flag_RREQ[index]=1;
}
```

Gambar 4.2 Potongan kode untuk mekanisme flag

Kode implementasi ini diletakkan pada fungsi `AODV::recvRequest()` dan dapat dilihat pada lampiran A.1 Kode Fungsi `AODV::recvRequest ()`.

4.1.2 Implementasi Mekanisme *Broadcast Node*

Implementasi yang mekanisme *broadcast node* seperti yang telah dirancang pada subbab 3.3.2 adalah sebagai berikut:

1. Deklarasi variable

Variable *broadcast node* digunakan sebagai pengecekan sebuah rute apakah dia unik atau tidak. Pengecekan dilakukan dengan membandingkan forwarder dengan isi *broadcast node* yang tersimpan pada suatu *node*. Sehingga bisa dideklarasikan sebagai integer vector. Lalu untuk dapat membuat *broadcast node* setiap *node*. *Broadcast node* berupa array vector. Dan juga dibutuhkan deklarasi variable `list_neighbor` untuk mendapatkan list neighbor saat mengirim RREQ. Deklarasi variable ini dilakukan di file `aodv.cc`. Kode sumbernya bisa dilihat pada Gambar 4.3.

```
std::vector < std::vector<int> >
list_neighbor(302, std::vector<int>(0));
std::vector < std::vector<int> >
broadcast_node(302, std::vector<int>(0));
```

Gambar 4.3 Potongan kode deklarasi variable *broadcast node*

2. Penambahan kode sumber untuk mekanisme *flag RREQ*

Lalu dilakukan penambahan kode sumber bagaimana *broadcast node* berjalan. Pengubahan dilakukan di file `aodv.cc` dan pada fungsi `AODV::recvRequest()`. Kode sumbernya bisa dilihat pada Gambar 4.4.

```

if (id_lookup(rq->rq_src, rq-
>rq_bcast_id)) {
    for (unsigned int i = 0; i <
broadcast_node[index].size(); i++)
    {
        if(broadcast_node[index][i]==ih-
>saddr()){
            Packet::free(p);
            return;
        }
    }
}
else{
    broadcast_node[index].clear();
    broadcast_node[index].push_back(ih-
>saddr());
    for (unsigned int i = 0; i <
list_neighbor[ih->saddr()].size(); i++){

        broadcast_node[index].push_back(list_neig
hbor[ih->saddr()][i]);
    }
}
}

```

Gambar 4.4 Potongan kode untuk mekanisme broadcast node

3. Implementasi update list neighbor

Update list neighbor dilakukan pada saat menambah atau menghapus daftar tetangga. Hal ini dilakukan untuk ketika *node* akan mengappend list tetangganya ke *broadcast node*. Penambahan kode dilakukan di fungsi AODV::nb_delete() dan AODV::nb_insert(). Kode sumbernya bisa dilihat pada Gambar 4.5.


```

AODV::nb_insert(nsaddr_t id) {
    list_neighbor[index].push_back(id);
}

AODV::nb_delete(nsaddr_t id) {
    for (unsigned int i = 0; i <
list_neighbor[index].size(); i++)
    {
        if(list_neighbor[index][i]==id)
        {
            list_neighbor[index].erase
(list_neighbor[index].begin()+i);
            break;
        }
    }
}

```

Gambar 4.5 Potongan kode update neighbor

4.2 Implementasi Skenario Mobilitas

Ada dua jenis skenario yang diimplementasikan pada tugas akhir ini, yaitu skenario grid dan skenario real. Skenario grid merupakan skenario yang sederhana, karena jalanan berbentuk kotak-kotak. Skenario real merupakan skenario yang diambil dari peta asli.

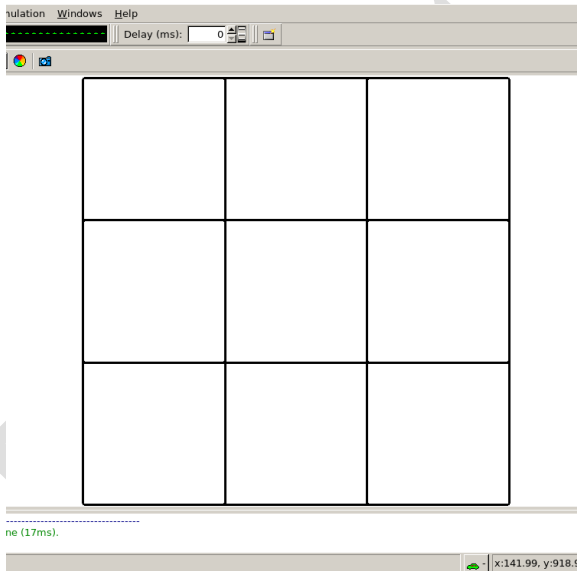
4.2.1 Skenario *Grid*

Implementasi dilakukan seperti pada rancangan pada sub bab 3.3.1. Kode sumber netgenerate dapat dilihat pada Gambar 4.6.

```
netgenerate --grid --grid.number=4 --
grid.length=400 --default.speed=20 --
tls.guess=1 --output-file=map.net.xml
```

Gambar 4.6 Kode sumber netgenerate

Hasilnya berupa *file* map bernama map.net.xml dapat dilihat pada Gambar 4.7.



Gambar 4.7 Hasil Generate Peta Grid

Setelah peta terbentuk, langkah selanjutnya adalah membuat *node* dan pergerakannya. Implementasi dilakukan dengan menentukan titik awal dan titik akhir *node* secara random menggunakan fitur randomTrips. Contoh perintah untuk menggunakan randomTrips dengan *node* sebanyak 58 dengan output *file* bernama trip.trips.xml dapat dilihat pada Gambar 4.8.

```
python $SUMO_HOME/tools/randomTrips.py -n
map.net.xml -e 58 -l --trip-
attributes="departLane=\"best\"
departSpeed=\"max\"
departPos=\"random_free\"" -o
trip.trips.xml
```

Gambar 4.8 Perintah randomTrips

Langkah selanjutnya adalah membuat rute *node* untuk bergerak mencapai tujuan mengikuti jalan raya yang telah dibuat pada map.net.xml. Hal ini dapat dilakukan dengan menggunakan fitur duarouter. Langkah ini membutuhkan *file* hasil sebelumnya, yaitu map.net.xml dan trip.trips.xml. Hasil dari langkah ini adalah *file* bernama route.route.xml. Penggunaan fitur dapat dilihat pada Gambar 4.9.

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

Gambar 4.9 Perintah duarouter

Langkah selanjutnya adalah menjadikan peta (map.net.xml) dan pergerakan *node* (route.route.xml) menjadi sebuah skenario dalam bentuk *file* berekstensi .xml. Untuk melakukannya dibutuhkan sebuah *file* skrip dengan ekstensi .sumocfg untuk menggabungkan *file* peta dan rute pergerakan *node*. Isi dari *file* skrip .sumocfg dapat dilihat pada Gambar 4.10.

```
<?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.route.xml"/>
  </input>
  <time>
    <begin value="0"/>
    <end value="200"/>
  </time>
  ...
</configuration>
```

Gambar 4.10 File Skrip .sumocfg

File .sumocfg disimpan pada level direktori yang sama dengan *file* peta (*map.net.xml*) dan *file* rute pergerakan *node* (*routes.route.xml*). Sebelum dikonversi, *file .sumocfg* dibuka dengan menggunakan *sumo-gui*. Kemudian skenario dapat dibuat dengan perintah pada Gambar 4.11. *File* output dari perintah tersebut bernama *scenario.xml*.

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

Gambar 4.11 Perintah SUMO

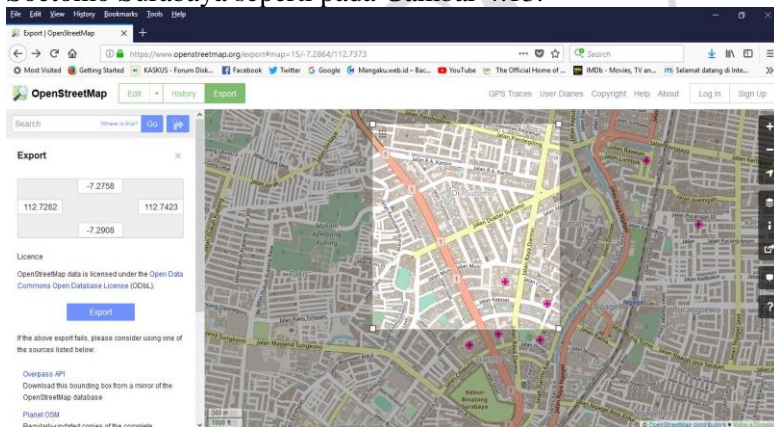
Langkah selanjutnya, *file* *scenario scenario.xml* dikonversi ke menjadi *scenario.tcl* sehingga dapat disimulasikan menggunakan NS-2. Fitur yang digunakan pada proses ini adalah *traceExporter*. Perintah *traceExporter* dapat dilihat pada Gambar 4.12.

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

Gambar 4.12 Perintah traceExporter

4.2.2 Skenario Real

Implementasi skenario real seperti pada rancangan sub bab 3.2.2 langkah awal untuk membuat skenario mobilitas real adalah memilih area peta untuk dijadikan lingkungan simulasi. Pada Tugas Akhir ini penulis mengambil area jalan sekitar Jl. Dr. Soetomo Surabaya seperti pada Gambar 4.13.



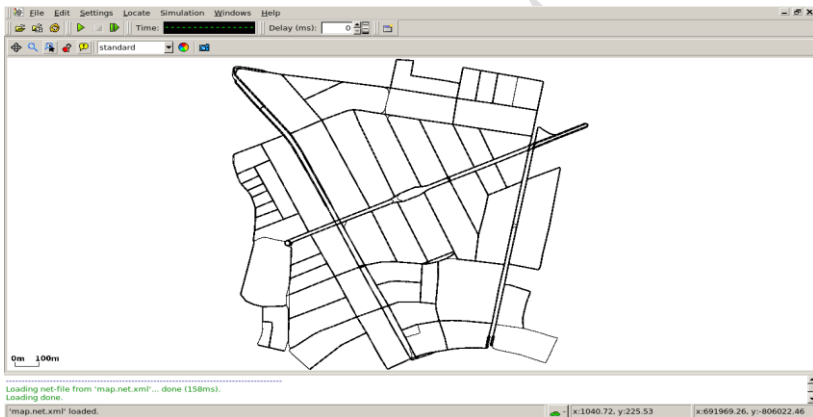
Gambar 4.13 Ekspor Peta dari OpenStreetMap

Area yang dipilih kemudian diekspor menjadi *file* peta dengan ekstensi .osm. Lalu *file* tersebut dikonversi menjadi peta dalam *file* berekstensi .xml menggunakan fitur netconvert dari SUMO. Perintah netconvert untuk mengkonversi map.osm menjadi map.net.xml dapat dilihat pada Gambar 4.14.

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

Gambar 4.14 Perintah netconvert

File hasil konversi map.net.xml dapat dilihat menggunakan sumo-gui seperti yang ditunjukkan pada Gambar 4.15.



Gambar 4.15 Hasil Konversi Peta Real

Langkah selanjutnya sama dengan ketika membuat skenario mobilitas *grid*.

4.3 Implementasi Simulasi pada NS-2

Langkah pertama dalam implementasi simulasi VANETs adalah pendeskripsian lingkungan simulasi pada sebuah *file* berekstensi .tcl. *File* ini berisikan konfigurasi perangkat dan topologi. Potongan konfigurasi lingkungan simulasi dapat dilihat pada Gambar 4.16.

```

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)         1300;
set opt(y)         1300;
set val(ifqlen)    1000;
set val(nn)        60;
set val(seed)      1.0;
set val(adhocRouting) AODV;
set val(stop)      200;
set val(cp)        "cbr60.txt";
set val(sc)        "scenario1.txt";

```

Gambar 4.16 Konfigurasi Lingkungan Simulasi

Pada konfigurasi juga dilakukan pemanggilan *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.2 Kode Skenario NS-2. Konfigurasi *traffic* dilakukan dengan membuat *file* berekstensi .txt. Potongan konfigurasi *file* traffic dapat dilihat pada Gambar 4.17.

```

set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(98) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(99) $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_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 2.5568388786897245 "$cbr_(0)
start"

```

Gambar 4.17 Konfigurasi Traffic

Penentuan *node* sumber dan *node* tujuan pengiriman paket terdapat pada konfigurasi tersebut. Pengiriman dimulai pada detik ke- 2.55. Implementasi konfigurasi *file* traffic untuk simulasi pada NS-2 dapat dilihat pada lampiran A.3 Kode Konfigurasi Traffic

4.4 Implementasi Metrik Analisis

Dari hasil *trace file* yang dihasilkan setelah simulasi dapat dianalisis performa dari *routing protocol* yang dipakai dengan menghitung beberapa metrik. Pada Tugas Akhir ini, metrik yang akan dianalisis adalah *Packet Delivery Ratio* (PDR), *packet loss*, *Average End-to-End Delay* (E2E), dan *Routing Overhead* (RO), dan RREQ F. Alat yang digunakan untuk menganalisis metrik adalah awk. Untuk menjalankan *file* awk dapat dilakukan dengan memasukkan perintah awk -f *file*.awk scenario.tr. Pada sub bab berikut akan dijelaskan setiap metrik.

4.4.1 Implementasi *Packet Delivery Ratio*

Untuk menghitung PDR telah dijelaskan pada sub bab 3.5.1. Untuk mendapatkan informasi yang diperlukan, diperlukan pencocokan kata kunci tertentu. Untuk menghitung PDR kata yang dicocokkan adalah AGT, karena layer AGT merupakan layer untuk komunikasi data. Kemudian dihitung paket terkirim dan diterima. Lalu dihitung menggunakan persamaan 3.1. Skrip awk untuk menghitung PDR dapat dilihat pada lampiran A.4 Skrip AWK *Packet Delivery*.

4.4.2 Implementasi Rata-Rata *End-to-End Delay*

Perhitungan *Average End to End Delay* telah dijelaskan pada sub bab 3.5.2. Lalu untuk skrip awk untuk menghitung E2E dapat dilihat pada lampiran A.5 Kode Skrip AWK Rata-Rata *End-to-End Delay*.

Contoh perintah untuk menjalankan skrip awk untuk menganalisis *trace file* adalah awk -f e2e.awk scenario.tr.

4.4.3 Implementasi *Routing Overhead*

Perhitungan *Routing Overhead* telah dijelaskan pada sub bab 3.5.3. Lalu untuk skrip awk untuk menghitung RO dapat dilihat pada lampiran A.6 Kode Skrip AWK *Routing Overhead*.

Contoh perintah untuk menjalankan skrip awk untuk menganalisis *trace file* adalah awk -f ro.awk scenario.tr.

4.4.4 Implementasi *Forwarded Route Request*

Perhitungan *Forwarded Route Request* telah dijelaskan pada sub bab 3.5.4. Lalu untuk skrip awk untuk menghitung RREQ F dapat dilihat pada lampiran A.7 Kode Skrip AWK *Forwarded Route Request*.

Contoh perintah untuk menjalankan skrip awk untuk menganalisis *trace file* adalah `awk -f rreqf.awk scenario.tr`.

4.4.5 Implementasi *Packet Loss*

Perhitungan *Forwarded Route Request* telah dijelaskan pada sub bab 3.5.5. Lalu untuk skrip awk untuk menghitung *packet loss* dapat dilihat pada lampiran A.8 Kode Skrip AWK *Packet Loss*.

Contoh perintah untuk menjalankan skrip awk untuk menganalisis *trace file* adalah `awk -f packetloss.awk scenario.tr`.

BAB V

UJI COBA DAN EVALUASI

Bab ini membahas tentang uji coba dan evaluasi dari implementasi yang telah dijelaskan pada bab implementasi. Hasil uji coba kemudian dievaluasi sehingga menghasilkan kesimpulan pada bab selanjutnya.

5.1 Lingkungan Uji Coba

Uji coba dilakukan pada perangkat komputer *notebook* dengan spesifikasi yang terdapat pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat yang Digunakan

Komponen	Spesifikasi
CPU	AMD Quad Core A8-7410 2.2 GHz (2M Cache, up to 2.5 GHz)
Sistem Operasi	Linux Mint 18.2 "Sonya" - Cinnamon (64-bit)
Linux Kernel	Linux kernel 4.4
Memori	6.0 GB
Penyimpanan	512 GB

Pengujian dilakukan dengan menjalankan simulasi skenario yang telah dirancang dan diimplementasikan pada bab sebelumnya. Simulasi tersebut menghasilkan *trace file* dengan ekstensi *.tr*. Dari *file* tersebut dianalisis menggunakan skrip *awk* sehingga mendapatkan performanya.

5.2 Hasil Uji Coba

Hasil uji coba terdiri dari hasil uji coba skenario *grid* dan uji coba skenario *real*.

5.2.1 Hasil Uji Coba Skenario *Grid*

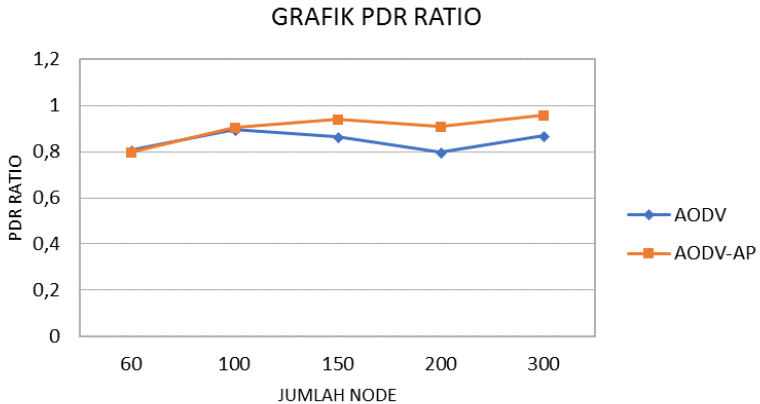
Pengujian pada skenario *grid* dilakukan untuk melihat perbandingan performa *routing protocol* AODV asli dan *routing protocol* AODV yang telah dimodifikasi meliputi *packet delivery ratio (PDR)*, *packet loss*, *average end-to-end delay (E2E)*, *routing overhead*, dan *forwarded route request* antara *node* sumber dan *node* tujuan. Pengujian dilakukan sesuai dengan perancangan *parameter* lingkungan simulasi yang dapat dilihat pada Tabel 3.2.

Pengambilan data uji PDR, E2E, *packet loss*, *routing overhead*, dan *forwarded route request* dengan luas area 1300 m x 1300 m dan 1500 m x 1500 m dengan *node* sebanyak 60, 100, 150, 200, dan 300 untuk lingkungan yang padat dilakukan pada kecepatan standar yaitu 20 m/s.

Hasil pengambilan data *packet delivery ratio* pada skenario *grid* 60, 100, 150, 200, dan 300 *node* dengan menggunakan AODV asli dan AODV yang telah dimodifikasi dapat dilihat pada Tabel 5.2 dan Gambar 5.1.

Tabel 5.2 PDR Skenario *Grid*

Routing Protocol	Jumlah Node				
	60	100	150	200	300
AODV	0,80764	0,89576	0,86711	0,796	0,86734
AODV-AP	0,79623	0,90453	0,94124	0,91015	0,9565



Gambar 5.1 PDR Skenario Grid

Berdasarkan Tabel 5.2 dan Gambar 5.1 dapat dilihat bahwa *routing protocol* modifikasi memiliki kenaikan PDR yang signifikan. Hal ini dikarenakan penurunan *packet loss* yang signifikan akibat penambahan *alternative path* sehingga PDR menjadi naik. Berikut perbandingan kenaikan PDR rasionya:

- 60 node: PDR ratio masih dibawah AODV murni turun sebesar 0,01 atau -1% dari AODV murni
- 100 node: PDR ratio naik sebesar 0,008 atau 1% dari AODV murni
- 150 node: PDR ratio naik sebesar 0,07 atau 9% dari AODV murni
- 200 node: PDR ratio naik sebesar 0,11 atau 14% dari AODV murni
- 300 node: PDR ratio naik sebesar 0,09 atau 10% dari AODV murni

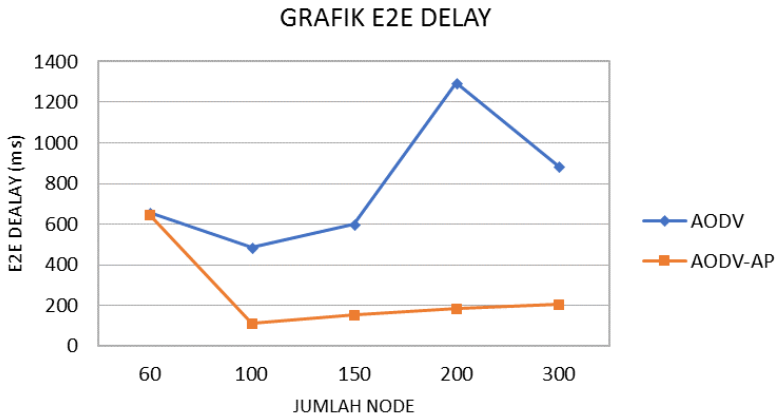
Dapat dilihat rata-rata kenaikan yang terjadi untuk PDR adalah 0,05 atau 7% dari AODV murni. Dapat dilihat juga bahwa dengan jumlah *node* yang lebih banyak atau pada lingkungan dengan *node* yang padat, AODV modifikasi menghasilkan PDR yang lebih bagus daripada di lingkungan dengan jumlah *node* yang

jarang, jika dibandingkan dengan AODV murni. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan PDR yang lebih bagus daripada AODV asli dengan jumlah selisih PDR yang cukup signifikan.

Hasil pengambilan data *average end-to-end delay* (E2E) pada skenario *grid* dengan jumlah *node* 60, 100, 150, 200, dan 300 dapat dilihat pada Tabel 5.3 dan Gambar 5.2.

Tabel 5.3 E2E Delay Skenario Grid

Routing Protocol	Jumlah Node				
	60	100	150	200	300
AODV	658,77	487,15	658,77	1297,73	887,46
AODV-AP	649,38	112,38	151,59	187,54	203,84



Gambar 5.2 E2E Delay Skenario Grid

Berdasarkan Tabel 5.3 dan Gambar 5.2 dapat dilihat bahwa *routing protocol* modifikasi memiliki penurunan *delay* yang signifikan. Hal ini dikarenakan perbedaan tindakan ketika menemui rute yang rusak antara AODV dengan AODV-AP.

AODV akan mencari ulang rute sehingga *delay* akan bertambah. Sedangkan AODV-AP tidak, dia akan menggunakan *alternative path* sehingga tidak terjadi *delay* pencarian rute. Berikut perbandingan penurunan *delay*nya:

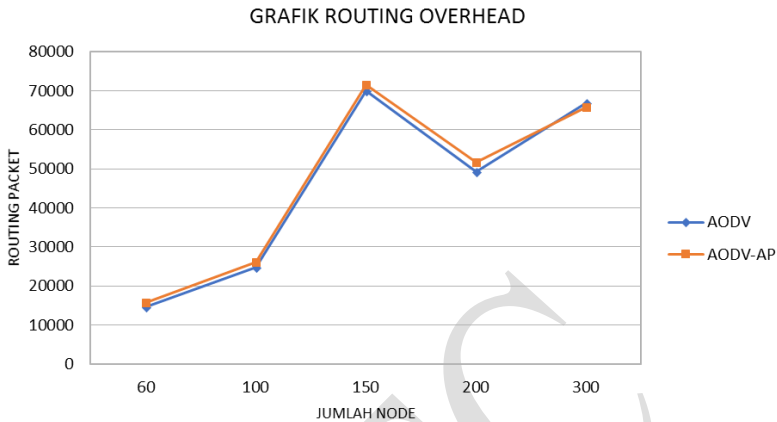
- 60 *node*: *delay* turun sebanyak 9,39 ms atau -1% dari AODV murni
- 100 *node*: *delay* turun sebanyak 374,77 ms atau -77% dari AODV murni
- 150 *node*: *delay* turun sebanyak 447,81 ms atau -75% dari AODV murni
- 200 *node*: *delay* turun sebanyak 1110,19 ms atau -86% dari AODV murni
- 300 *node*: *delay* turun sebanyak 683,62 ms atau -77% dari AODV murni

Dapat dilihat rata-rata penurunan E2E sebanyak 525,16 atau 63% dari AODV murni. Dari hasil uji coba tersebut, AODV modifikasi lebih unggul dalam hal E2E daripada AODV murni.

Untuk hasil pengambilan data *routing overhead* pada skenario *grid* 60, 100, 150, 200, dan 300 *node* dapat dilihat pada Tabel 5.4 dan Gambar 5.3.

Tabel 5.4 Routing Overhead Skenario Grid

Routing Protocol	Jumlah Node				
	60	100	150	200	300
AODV	14698	24886	69988	49228	66981
AODV-AP	15778	26200	71526	51729	65781



Gambar 5.3 Hasil Routing Overhead Skenario Grid

Berdasarkan Tabel 5.4 dan Gambar 5.3 dapat dilihat bahwa *routing protocol* modifikasi cenderung memiliki routing overhead yang lebih besar dibanding AODV murni tetapi cenderung sama. Hal ini dikarenakan hanya RREQ F-nya yang meningkat akibat penambahan *alternative path*. RREP juga bertambah, tetapi tidak sebanyak RREQnya sehingga tidak terlalu mempengaruhi. Berikut perbandingan *routing overhead*nya:

- 60 node: bertambah sebanyak 1080,40 atau 7% dari AODV murni
- 100 node: bertambah sebanyak 1313,80 atau 5% dari AODV murni
- 150 node: bertambah sebanyak 1538,10 atau 2% dari AODV murni
- 200 node: bertambah sebanyak 2501,30 atau 5% dari AODV murni
- 300 node: berkurang sebanyak 1199,80 atau -2% dari AODV murni

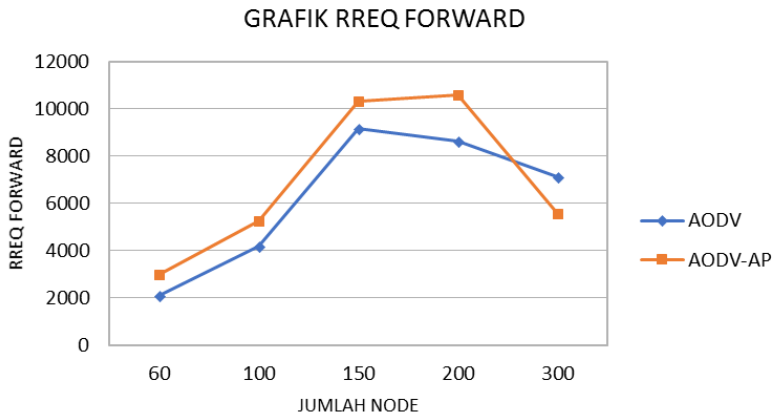
Dapat dilihat rata-rata pertambahan *routing overhead* sebanyak 1046,76 atau 4% dari AODV murni. *Routing overhead* bertambah dikarenakan AODV modifikasi memberi kesempatan

setiap *node* untuk *membroadcast* dua kali. Dari hasil uji coba tersebut, dapat disimpulkan AODV modifikasi tidak lebih unggul dalam hal routing overhead daripada AODV murni.

Untuk hasil pengambilan data *forwarded route request* (RREQ F) pada skenario *grid* 60, 100, 150, 200, dan 300 *node* dapat dilihat pada Tabel 5.5 dan Gambar 5.4.

Tabel 5.5 Hasil RREQ F Skenario Grid

Routing Protocol	Jumlah Node				
	60	100	150	200	300
AODV	2099,5	4180,3	9179,3	8624,3	7107
AODV-AP	2987,3	5239,7	10326	10574,8	5580,7



Gambar 5.4 Hasil RREQ F Skenario Grid

Berdasarkan Tabel 5.5 dan Gambar 5.4 dapat dilihat bahwa *routing protocol* modifikasi cenderung memiliki RREQ F yang lebih besar dibanding AODV murni. Hal ini dikarenakan pada AODV-AP setiap *node* network diberi kesempatan dua kali rebroadcast RREQ, sehingga RREQ F-nya menjadi tinggi. Sedangkan AODV

murni tidak, setiap node hanya diberi kesempatan sekali saja. Berikut perbandingan RREQ F-nya:

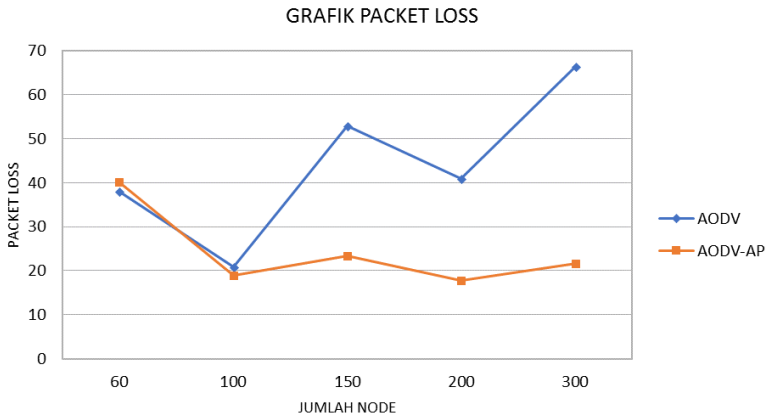
- 60 *node*: bertambah sebanyak 887,80 atau 42% dari AODV murni
- 100 *node*: bertambah sebanyak 1059,40 atau 25% dari AODV murni
- 150 *node*: bertambah sebanyak 1146,70 atau 12% dari AODV murni
- 200 *node*: bertambah sebanyak 1950,50 atau 23% dari AODV murni
- 300 *node*: berkurang sebanyak 1526,30 atau 16% dari AODV murni

Dapat dilihat rata-rata pertambahan RREQ F sebanyak 1046,76 atau 4% dari AODV murni. RREQ F bertambah dikarenakan AODV modifikasi memberi kesempatan setiap *node* untuk mem**roadcast** dua kali. Dari hasil uji coba tersebut, dapat disimpulkan AODV modifikasi tidak lebih unggul dalam hal routing overhead daripada AODV murni.

Untuk hasil pengambilan data *packet loss* pada skenario grid 60, 100, 150, 200, dan 300 *node* dapat dilihat pada Tabel 5.6 dan Gambar 5.5.

Tabel 5.6 Hasil Packet Loss Skenario Grid

Routing Protocol	Jumlah Node				
	60	100	150	200	300
AODV	52,9	20,9	52,9	41	66,4
AODV-AP	40,1	18,9	23,4	17,8	21,7



Gambar 5.5 Hasil Packet Loss Skenario Grid

Berdasarkan Tabel 5.6 dan Gambar 5.5 dapat dilihat bahwa *routing protocol* modifikasi memiliki penurunan *packet loss* yang signifikan. Hal ini terjadi karena perbedaan tindakan ketika pengiriman paket menemui rute yang rusak. Pada AODV murni, paket didrop sehingga *packet loss* naik. Sedangkan pada AODV-AP tidak, dia bisa menggunakan jalur alternatif sehingga *packet loss* menurun. Berikut perbandingan penurunan *packet loss*nya:

- 60 node: naik sebanyak 2 paket atau 6% dari AODV murni
- 100 node: turun sebanyak 2 paket atau -10% dari AODV murni
- 150 node: turun sebanyak 29,5 paket atau -56% dari AODV murni
- 200 node: turun sebanyak 23,2 paket atau -57% dari AODV murni
- 300 node: turun sebanyak 44,7 paket atau -67% dari AODV murni

Dapat dilihat rata-rata penurunan *packet loss* sebanyak 19,5 atau 63% dari AODV murni. Dari hasil uji coba tersebut, dapat disimpulkan bahwa AODV modifikasi lebih unggul dalam hal *packet loss* daripada AODV murni.

5.2.2 Hasil Uji Coba Skenario Real

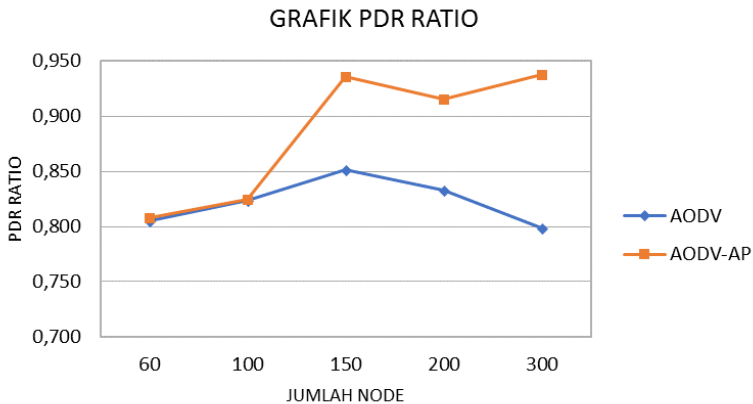
Pengujian pada skenario *real* dilakukan untuk melihat perbandingan performa *routing protocol* AODV asli dan *routing protocol* AODV yang telah dimodifikasi meliputi *packet delivery ratio* (PDR), *packet loss*, *average end-to-end delay* (E2E), *routing overhead*, dan *forwarded route request* antara *node* sumber dan *node* tujuan. Pengujian dilakukan sesuai dengan perancangan *parameter* lingkungan simulasi yang dapat dilihat pada Tabel 3.2.

Pengambilan data uji PDR, E2E, *packet loss*, *routing overhead*, dan *forwarded route request* dengan luas area 1300 m x 1300 m dan 1500 m x 1500 m dengan *node* sebanyak 60, 100, 150, 200, dan 300 untuk lingkungan yang padat dilakukan pada kecepatan standar yaitu 20 m/s.

Hasil pengambilan data *packet delivery ratio* pada skenario *real* 60, 100, 150, 200, dan 300 *node* dengan menggunakan AODV asli dan AODV yang telah dimodifikasi dapat dilihat pada Tabel 5.7 Gambar 5.6.

Tabel 5.7 Hasil PDR Skenario Real

Routing Protocol	Jumlah Node				
	60	100	150	200	300
AODV	0,805	0,824	0,852	0,833	0,799
AODV-AP	0,808	0,825	0,936	0,916	0,938



Gambar 5.6 Hasil PDR Skenario Real

Berdasarkan Tabel 5.7 dan Gambar 5.6 dapat dilihat bahwa *routing protocol* modifikasi memiliki kenaikan PDR yang signifikan. Hal ini dikarenakan penurunan *packet loss* yang signifikan akibat penambahan *alternative path* sehingga PDR menjadi naik. Berikut perbandingan kenaikan PDR rasionya:

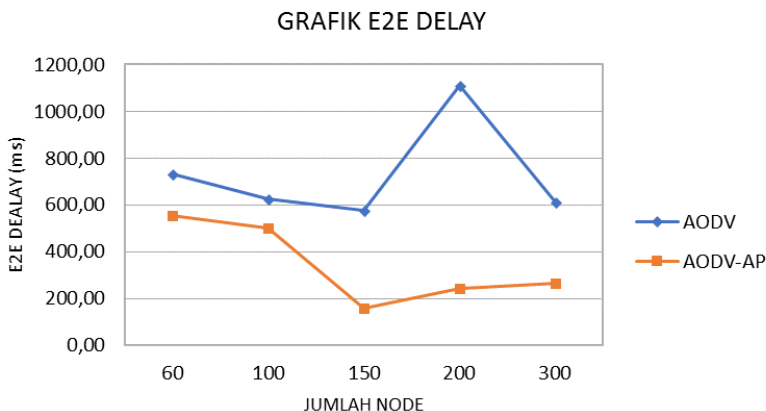
- 60 node: naik sebesar 0,003 atau 0,3% dari AODV murni
- 100 node: naik sebesar 0,001 atau 0,1% dari AODV murni
- 150 node: naik sebesar 0,08 atau 9% dari AODV murni
- 200 node: sebesar 0,13 atau 17% dari AODV murni
- 300 node: PDR ratio naik sebesar 0,06 atau 7% dari AODV murni

Dapat dilihat rata-rata kenaikan yang terjadi untuk PDR adalah 0,06 atau 7,5% dari AODV murni. Dapat dilihat juga bahwa dengan jumlah *node* yang lebih banyak atau pada lingkungan dengan *node* yang padat, AODV modifikasi menghasilkan PDR yang lebih bagus daripada di lingkungan dengan jumlah *node* yang jarang, jika dibandingkan dengan AODV murni. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan PDR yang lebih bagus daripada AODV asli dengan jumlah selisih PDR yang cukup signifikan.

Hasil pengambilan data *average end-to-end delay* (E2E) pada skenario *real* dengan jumlah *node* 60, 100, 150, 200, dan 300 dapat dilihat pada Tabel 5.8 dan Gambar 5.7.

Tabel 5.8 E2E Delay Skenario Real

Routing Protocol	Jumlah Node				
	60	100	150	200	300
AODV	733,13	623,79	578,34	1109,84	614,03
AODV-AP	552,69	500,25	160,76	244,45	265,85



Gambar 5.7 E2E Delay Skenario Real

Berdasarkan Tabel 5.8 dan Gambar 5.7 dapat dilihat bahwa *routing protocol* modifikasi memiliki penurunan *delay* yang signifikan. Hal ini dikarenakan perbedaan tindakan ketika menemui rute yang rusak antara AODV dengan AODV-AP. AODV akan mencari ulang rute sehingga *delay* akan bertambah. Sedangkan AODV-AP tidak, dia akan menggunakan *alternative path* sehingga tidak terjadi *delay* pencarian rute. Berikut perbandingan penurunan *delay*nya:

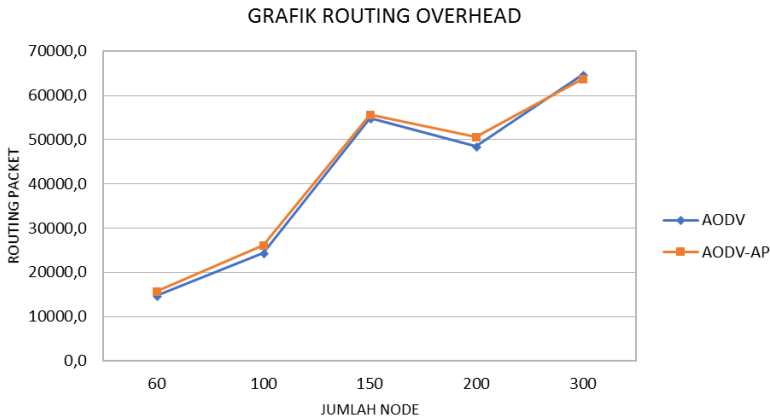
- 60 *node*: *delay* turun sebanyak 180,44 ms atau -25% dari AODV murni
- 100 *node*: *delay* turun sebanyak 123,54 ms atau -20% dari AODV murni
- 150 *node*: *delay* turun sebanyak 417,58 ms atau -72% dari AODV murni
- 200 *node*: *delay* turun sebanyak 865,39 ms atau -78% dari AODV murni
- 300 *node*: *delay* turun sebanyak 348,18 ms atau -57% dari AODV murni

Dapat dilihat rata-rata penurunan E2E sebanyak 387,02 atau 50% dari AODV murni. Dari hasil uji coba tersebut, AODV modifikasi lebih unggul dalam hal E2E daripada AODV murni.

Untuk hasil pengambilan data *routing overhead* pada skenario *real* 60, 100, 150, 200, dan 300 *node* dapat dilihat pada Tabel 5.9 dan Gambar 5.8.

Tabel 5.9 Routing Overhead Skenario Real

Routing Protocol	Jumlah Node				
	60	100	150	200	300
AODV	14687,5	24459,8	54841,3	48565,6	64669,6
AODV-AP	15719,0	26191,8	55743,2	50680,4	63722,0



Gambar 5.8 Hasil Routing Overhead Skenario Real

Berdasarkan Tabel 5.9 dan Gambar 5.8 dapat dilihat bahwa *routing protocol* modifikasi cenderung memiliki routing overhead yang lebih besar dibanding AODV murni tetapi cenderung sama. Hal ini dikarenakan hanya RREQ F-nya yang meningkat akibat penambahan *alternative path*. RREP juga bertambah, tetapi tidak sebanyak RREQnya sehingga tidak terlalu mempengaruhi. Berikut perbandingan *routing overhead*nya:

- 60 *node*: bertambah sebanyak 1031,45 atau 7% dari AODV murni
- 100 *node*: bertambah sebanyak 1732,00 atau 7% dari AODV murni
- 150 *node*: bertambah sebanyak 901,90 atau 2% dari AODV murni
- 200 *node*: bertambah sebanyak 2114,80 atau 4% dari AODV murni
- 300 *node*: berkurang sebanyak 947,60 atau -1% dari AODV murni

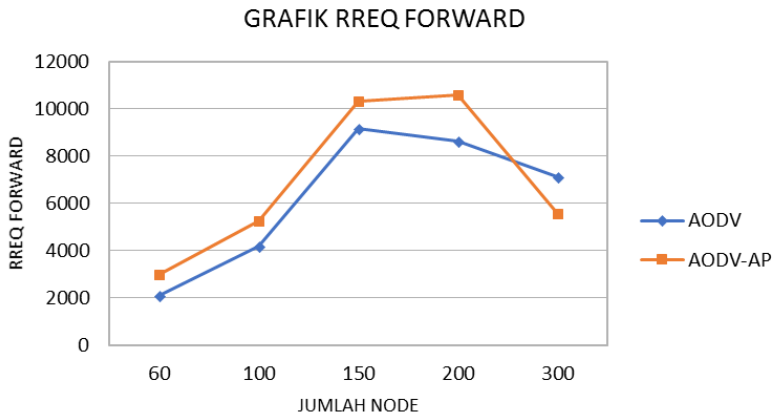
Dapat dilihat rata-rata pertambahan *routing overhead* sebanyak 966,51 atau 4% dari AODV murni. *Routing overhead* bertambah dikarenakan AODV modifikasi memberi kesempatan

setiap *node* untuk *membroadcast* dua kali. Dari hasil uji coba tersebut, dapat disimpulkan AODV modifikasi tidak lebih unggul dalam hal routing overhead daripada AODV murni.

Untuk hasil pengambilan data *forwarded route request* (RREQ F) pada skenario *real* 60, 100, 150, 200, dan 300 *node* dapat dilihat pada Tabel 5.10 dan Gambar 5.9.

Tabel 5.10 Hasil RREQ F Skenario Real

Routing Protocol	Jumlah Node				
	60	100	150	200	300
AODV	2178,3	3898,1	5670,1	8037,7	5181,2
AODV-AP	3103,7	5449,0	6379,7	9753,1	4331,4



Gambar 5.9 Hasil RREQ F Skenario Real

Berdasarkan Tabel 5.10 dan Gambar 5.9 dapat dilihat bahwa *routing protocol* modifikasi cenderung memiliki RREQ F yang lebih besar dibanding AODV murni. Hal ini dikarenakan pada AODV-AP setiap *node* network diberi kesempatan dua kali rebroadcast RREQ, sehingga RREQ F-nya menjadi tinggi.

Sedangkan AODV murni tidak, setiap node hanya diberi kesempatan sekali saja. Berikut perbandingan RREQ F-nya:

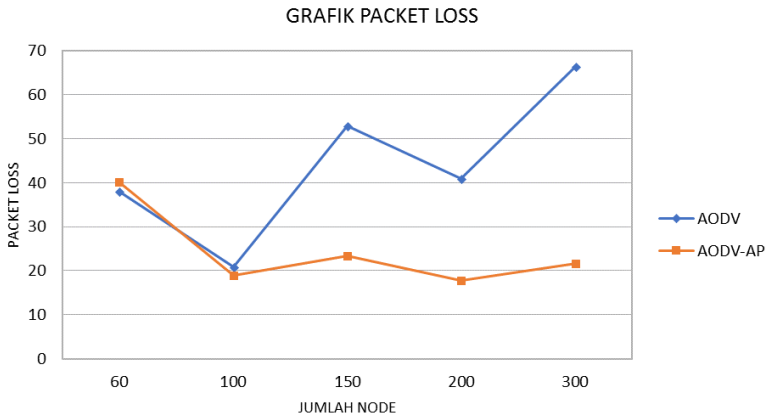
- 60 *node*: bertambah sebanyak 922,10 atau 42% dari AODV murni
- 100 *node*: bertambah sebanyak 1566,80 atau 40% dari AODV murni
- 150 *node*: bertambah sebanyak 698,55 atau 12% dari AODV murni
- 200 *node*: bertambah sebanyak 1724,30 atau 21% dari AODV murni
- 300 *node*: berkurang sebanyak 855,19 atau 20% dari AODV murni

Dapat dilihat rata-rata pertambahan RREQ F sebanyak 811,31 atau 20% dari AODV murni. RREQ F bertambah dikarenakan AODV modifikasi memberi kesempatan setiap *node* untuk mem*broadcast* dua kali. Dari hasil uji coba tersebut, dapat disimpulkan AODV modifikasi tidak lebih unggul dalam hal routing overhead daripada AODV murni.

Untuk hasil pengambilan data *packet loss* pada skenario real 60, 100, 150, 200, dan 300 *node* dapat dilihat pada Tabel 5.11 dan Gambar 5.10.

Tabel 5.11 Hasil Packet Loss Skenario Real

Routing Protocol	Jumlah Node				
	60	100	150	200	300
AODV	38,45	35,2	58,85	33,5	100,3
AODV-AP	37,95	34,35	25,35	16,7	31,05



Gambar 5.10 Hasil Packet Loss Skenario Real

Berdasarkan tabel 5.11 dan grafik 5.10 dapat dilihat bahwa *routing protocol* modifikasi memiliki penurunan *packet loss* yang signifikan. Hal ini terjadi karena perbedaan tindakan ketika pengiriman paket menemui rute yang rusak. Pada AODV murni, paket didrop sehingga *packet loss* naik. Sedangkan pada AODV-AP tidak, dia bisa menggunakan jalur alternatif sehingga *packet loss* menurun. Berikut perbandingan penurunan *packet loss*nya:

- 60 node: turun sebanyak 0,5 paket atau -1% dari AODV murni
- 100 node: turun sebanyak 0,85 paket atau -2% dari AODV murni
- 150 node: turun sebanyak 33,5 paket atau -57% dari AODV murni
- 200 node: turun sebanyak 16,80 paket atau -50% dari AODV murni
- 300 node: turun sebanyak 69,25 paket atau -69% dari AODV murni

Dapat dilihat rata-rata penurunan *packet loss* sebanyak 24,18 atau -36% dari AODV murni. Dari hasil uji coba tersebut,

AODV modifikasi lebih unggul dalam hal *packet loss* daripada AODV murni.

BAB VI

KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang diperoleh dari tugas akhir yang telah dikerjakan dan saran untuk pengembangan tugas akhir ini di masa depan.

6.1 Kesimpulan

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

1. Implementasi alternative path dilakukan dengan ditambahkannya mekanisme RREQ *flag* dan *broadcast node*.
2. Dampak dari modifikasi ini adalah meningkatkan performa *routing protocol* dengan detail: PDR meningkat sebanyak 7,25%, E2E berkurang sebanyak 56,5%, dan *packet loss* berkurang sebanyak 49,5% daripada AODV murni. Tetapi masih terdapat kelemahan pada *routing overhead* yang bertambah sebanyak 4% dan RREQ F yang bertambah sebanyak 12% daripada AODV murni.

6.2 Saran

Saran yang diberikan dari hasil uji coba dan evaluasi Tugas Akhir ini adalah sebagai berikut:

1. Untuk mendapatkan hasil uji coba yang lebih akurat, dapat dilakukan uji coba yang lebih banyak, misalnya lebih dari 10 kali percobaan untuk setiap skenario. Karena pengerjaan tugas akhir ini cukup singkat sehingga hanya sempat melakukan percobaan 10 kali setiap skenario.
2. Ide implementasi *broadcast node* untuk mendapatkan rute alternatif yang unik sudah bagus, kedepannya bisa ditingkatkan lagi dengan membatasi *broadcast* dengan *broadcast node* untuk mengurangi *rebroadcast route request*

yang tidak penting. Sehingga dapat mengurangi routing overhead dan RREQ F.

RBTC

DAFTAR PUSTAKA

- [1] L. Khan, N. Ayub dan A. Saeed, “Anycast Based Routing in Vehicular Adhoc Networks (VANETS) using Vanetmobisim,” COMSATS IIT Abbottabad, Pakistan, 2009.
- [2] A. V. Aho, B. W. Kerninghan dan P. J. Weinberger, *The AWK Programming Language*, Boston: Addison-Wesley, 1988.
- [3] S. Mittal, R. Kaur dan K. C. Purohit, “Enhancing the Data Transfer rate by Creating Alternative Path for AODV Routing Protocol in VANET,” dalam *IEEE*, 2016.
- [4] M. K. Marina dan S. R. Das, “On-demand Multipath Distance Vector Routing in Ad Hoc Networks,” dalam *IEEE*, 2001.
- [5] S. Ajmal, A. Rasheed, A. Qayyum dan A. Hasan, “Classification of VANET MAC, Routing and Approaches A Detailed Survey,” *Journal of Universal Computer Science*, vol. 20, 2014.
- [6] D. Sutariya dan S. Pradhan, “An Improved AODV Routing Protocol for VANETs in City Scenarios,” dalam *ICAESM*, 2012.
- [7] “JOSM,” JOSM, [Online]. Available: <http://wiki.openstreetmap.org/wiki/JOSM>. [Diakses 6 June 2018].
- [8] “SUMO,” SUMO, [Online]. Available: http://www.sumo.dlr.de/userdoc/Sumo_at_a_Glance.html. [Diakses 6 June 2018].
- [9] “OpenStreetMap,” OpenStreetMap, [Online]. Available: <https://www.openstreetmap.org/about>. [Diakses 6 June 2018].

(Halaman ini sengaja dikosongkan)

RBTC

LAMPIRAN

A.1 Kode Fungsi AODV::recvRequest ()

```
void
AODV::recvRequest(Packet *p) {
    struct hdr_ip *ih = HDR_IP(p);
    struct hdr_aodv_request *rq =
    HDR_AODV_REQUEST(p);
    aodv_rt_entry *rt;
    if(rq->rq_dst != index) {
        if(rq->rq_src == index) {
            Packet::free(p);
            return;
        }
    }
    if (id_lookup(rq->rq_src, rq-
    >rq_bcast_id)) {

        for (unsigned int i = 0; i <
        broadcast_node[index].size(); i++)
        {
            if(broadcast_node[index][i]==ih-
            >saddr()){
                Packet::free(p);
                return;
            }
        }
        if(flag_RREQ[index]<2){
            flag_RREQ[index]++;
        }
        else{
            Packet::free(p);
            return;
        }
    }
}
```

```

else{
    flag_RREQ[index]=1;
    broadcast_node[index].clear();
    broadcast_node[index].push_back(ih-
>saddr());
    for (unsigned int i = 0; i <
list_neighbor[ih->saddr()].size(); i++){

        broadcast_node[index].push_back(list_neighb
or[ih->saddr()][i]);
    }
}
id_insert(rq->rq_src, rq->rq_bcast_id);
aodv_rt_entry *rt0;

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

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

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

```

```

rt0->rt_req_timeout = 0.0;
rt0->rt_req_last_ttl = rq->rq_hop_count;
rt0->rt_expire = CURRENT_TIME +
ACTIVE_ROUTE_TIMEOUT;
}

assert (rt0->rt_flags == RTF_UP);
Packet *buffered_pkt;
while ((buffered_pkt = rqueue.dequeue(rt0-
>rt_dst))) {
if (rt0 && (rt0->rt_flags == RTF_UP)) {
assert(rt0->rt_hops != INFINITY2);
forward(rt0, buffered_pkt, NO_DELAY);
}
}
}

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

if(rq->rq_dst == index) {
seqno = max(seqno, rq->rq_dst_seqno)+1;
if (seqno%2) seqno++;

sendReply(rq->rq_src,
1,
index,
seqno,
MY_ROUTE_TIMEOUT,
rq->rq_timestamp);

Packet::free(p);
}

```

```

else if (rt && (rt->rt_hops != INFINITY2)
&&
(rt->rt_seqno >= rq->rq_dst_seqno) ) {
assert(rq->rq_dst == rt->rt_dst);
sendReply(rq->rq_src,
rt->rt_hops + 1,
rq->rq_dst,
rt->rt_seqno,
(u_int32_t) (rt->rt_expire -
CURRENT_TIME),
rq->rq_timestamp);
rt->pc_insert(rt0->rt_nexthop);
rt0->pc_insert(rt->rt_nexthop);
#ifdef RREQ_GRAT_RREP

sendReply(rq->rq_dst,
rq->rq_hop_count,
rq->rq_src,
rq->rq_src_seqno,
(u_int32_t) (rt->rt_expire -
CURRENT_TIME),
rq->rq_timestamp);
#endif
Packet::free(p);
}
else {
ih->saddr() = index;
ih->daddr() = IP_BROADCAST;
rq->rq_hop_count += 1;
if (rt) rq->rq_dst_seqno = max(rt-
>rt_seqno, rq->rq_dst_seqno);
forward((aodv_rt_entry*) 0, p, DELAY);
}
}

```

A.2 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)         1300;
set opt(y)         1300;
set val(ifqlen)    1000;
set val(nn)        60;
set val(seed)      1.0;
set val(adhocRouting) AODV;
set val(stop)      200;
set val(cp)        "cbr60.txt";
set val(sc)        "scenario1.txt";
set ns_            [new Simulator]
set topo [new Topography]
set tracefd [open scenario1.tr w]
set namtrace [open scenario1.nam w]
set topo [new Topography]
$topo load_flatgrid $opt(x) $opt(y)
set god_ [create-god $val(nn)]
```

```

$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 $topo \
    -agentTrace ON \
    -routerTrace ON \
    -macTrace ON \
    -movementTrace ON \
Phy/WirelessPhy set RXThresh_ 5.57189e-11
; #400m
Phy/WirelessPhy set CSThresh_ 5.57189e-11
; #400m
for {set i 0} {$i < $val(nn)} {incr i} {
    set node_($i) [$ns_ node]
    $node_($i) random-motion 0 ;# disable
random motion
}
puts "Loading connection pattern..."
source $val(cp)
puts "Loading scenario file..."
source $val(sc)
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns_ initial_node_pos $node_($i) 20
}

```

```

for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop).0 "$node_($i)
reset";
}
$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.3 Kode Konfigurasi *Traffic*

```

set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(58) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(59) $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_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 2.5568388786897245 "$cbr_(0) start"

```

A.4 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.5 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.6 Kode Skrip AWK *Routing Overhead*

```

BEGIN {
    rt_pkts = 0;
}
{
    if ($2 >= 101) {
        if (($1 == "s" || $1 == "f")
&& ($4 == "RTR") && ($7 == "AODV")) {

            rt_pkts++;

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

```

A.7 Kode Skrip AWK *Forwarded Route Request*

```

BEGIN {
    rreqf = 0;
}
{
    if ($2 >= 101) {
        if (($1 == "f") && ($4 ==
"RTR") && ($7 == "AODV") ($19 == "[1"])) {

            rreqf++;

        }
    }
}
END {
    printf "Route Request F \t= %d \n",
rreqf;
}

```

A.8 Kode Skrip AWK *Packet Loss*

```

BEGIN {
    sendLine = 0;
    recvLine = 0;
}

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

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

END {
    printf "Packet loss \t= %d \n",
(sendLine-recvLine);
}

```

(Halaman ini sengaja dikosongkan)

RBTC

BIODATA PENULIS



Ilyas Bintang Prayogi lahir di Kediri pada 22 Agustus 1997. Penulis menempuh pendidikan formal RA Kusuma Mulia 1 Gadungan, Wates, Kab. Kediri (2001-2004), SDN Tawang 2, Wates, Kab. Kediri (2004-2006), SDN Ngronggo 5 Kediri (2006-2010), MTsN 2 Kediri (2010-2012), MAN 3 Kediri (sekarang MAN 2) (2012-2014), dan melanjutkan studi S1 di Departemen Informatika ITS (2014-2018). Bidang studi yang diambil oleh penulis pada saat berkuliah di Departemen Informatika ITS adalah Arsitektur Jaringan Komputer (AJK). Penulis aktif dalam organisasi Keluarga Muslim Informatika (2015-2017). Penulis juga aktif dalam organisasi Badan Eksekutif Mahasiswa FTIf (sekarang FTIK) (2015-2016). Penulis pernah kerja praktik di SMAN 6 Kediri periode Juli – Agustus 2017. Penulis dapat dihubungi melalui nomor *handphone* 085735835737 atau di email ilyasbintangpra@gmail.com.