TUGAS AKHIR – KI141502

ANALISIS KINERJA PERUBAHAN *HELLO INTERVAL* PADA AODV DI LINGKUNGAN VANET

MAGISTA BELLA PUSPITA
NRP 05111440000007

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

Dosen Pembimbing II
Henning Titi Ciptaningtyas, S.Kom., M.kom

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

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR – KI141502

ANALISIS KINERJA PERUBAHAN *HELLO INTERVAL* PADA AODV DI LINGKUNGAN VANET

MAGISTA BELLA PUSPITA
NRP 05111440000007

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

Dosen Pembimbing II
Henning Titi Ciptaningtyas, S.Kom., M.Kom

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

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – KI141502

PERFORMANCE ANALYSIS OF HELLO INTERVAL CHANGES ON AODV IN VANET ENVIRONMENT

MAGISTA BELLA PUSPITA
NRP 05111440000007

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

Second Advisor
Henning Titi Ciptaningtyas, S.Kom., M.Kom

Department of Informatics
Faculty of Information and Communication Technology
Sepuluh Nopember Institute of Technology
Surabaya 2018

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

ANALISIS KINERJA PERUBAHAN HELLO INTERVAL PADA AODV DI LINGKUNGAN VANET

TUGAS AKHIR

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

Oleh:
MAGISTA BELLA PUSPITA
NRP: 05111440000007

Disetujui oleh Pembimbing Tugas Akhir:

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

2. Henning Titi Ciptaningtyas, S.Kom., M.Kom .....
(NIP. 198407082010122004) (Pembimbing 2)

SURABAYA
JULI, 2018

(Halaman ini sengaja dikosongkan)

ANALISIS KINERJA PERUBAHAN *HELLO INTERVAL* PADA AODV DI LINGKUNGAN VANET

Nama Mahasiswa	**: Magista Bella Puspita**
NRP	**: 05111440000007**
Departemen	**: Informatika FTIK-ITS**
Dosen Pembimbing 1	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**
Dosen Pembimbing 2	**: Henning Titi Ciptaningtyas, S.Kom., M.Kom**

Abstrak

Vehicular Ad Hoc Networks (VANET) merupakan pengembangan dari *Mobile Ad Hoc Networks* (MANET). *Routing protocol* VANET dibagi menjadi dua model, yaitu *Topology-based Routing* dan *Geographic Routing.* Untuk *Topology-based Routing* masih dibagi menjadi dua, yaitu proaktif dan reaktif. Salah satu dari beberapa *routing protocol* yang reaktif yaitu AODV. *Reactive routing protocol* AODV memiliki *node* dengan mobilitas yang tinggi sehingga tidak jarang sebelum paket data tersampaikan, rute yang terjalin sudah berubah. Oleh karena itu, rawan terjadi kehilangan paket data di tengah pengiriman.

Melihat permasalahan di atas, pada Tugas Akhir ini akan dilakukan analisis kinerja pada *routing protocol* AODV dengan mengubah *Hello Interval*. Modifikasi yang dilakukan berfokus pada perubahan *Hello Interval* yang disimulasikan menggunakan skenario *grid* dan skenario *real*. Batasan modifikasi *Hello Interval* berkisar pada interval 5 detik, 10 detik, dan 15 detik.

Hasil uji coba telah menunjukkan bahwa protokol AODV modifikasi mampu menurunkan *Routing Overhead* sebesar 64.34 %. Namun pada performa metrik yang lain menunjukan hasil yang kompetitif dibandingkan dengan AODV asli.

Kata kunci: MANET, VANET, AODV, Hello Interval

(Halaman ini sengaja dikosongkan)

PERFORMANCE ANALYSIS OF HELLO INTERVAL CHANGES ON AODV IN VANET ENVIRONMENT

Student's Name	**:**	**Magista Bella Puspita**
Student's ID	**:**	**05111440000007**
Department	**:**	**Informatics – FTIK ITS**
First Advisor	**:**	**Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**
Second Advisor	**:**	**Henning Titi Ciptaningtyas, S.Kom., M.Kom**

Abstract

Vehicular Ad Hoc Networks (VANET) is a development of Mobile Ad Hoc Networks (MANET). VANET routing protocol is divided into two models, namely Topology-based Routing and Geographic Routing. For Topology-based Routing is still divided into two, namely proactive and reactive. One of several reactive routing protocols is AODV. Reactive routing protocol AODV has a node with high mobility so that not infrequently before data packets are delivered, the interwoven route has changed. Therefore, it is prone to lose the data packet in the middle of sending.

Seeing the problem above, this Final Project will perform performance analysis on AODV routing protocol by changing Hello Interval. The modifications made focus on simulated Hello interval changes using grid scenarios and real scenarios. The Hello Interval modification limits range from 5 second, 10 second, and 15 second intervals.

The test results have shown that the modified AODV protocol is able to decrease Routing Overhead by 64.34%. But on other performance metrics show competitive results compared with the original AODV.

Keywords: MANET, VANET, AODV, Hello Interval

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul **"Analisis Kinerja Perubahan *Hello Interval* pada AODV di Lingkungan VANET".**

Harapan Penulis semoga apa yang tertulis di dalam buku Tugas Akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini.

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

1. Allah SWT atas semua rahmat yang diberikan sehingga Penulis dapat menyelesaikan Tugas Akhir ini.
2. Bapak Imam Sujadi dan Ibu Dwi Wahyu Apriyanti selaku kedua orangtua Penulis atas segala dukungan berupa motivasi serta doa sehingga penulis dapat mengerjakan Tugas Akhir ini.
3. M. Bintang Persadha 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 Ibu Henning Titi Ciptaningtyas, S.Kom, M.Kom. selaku dosen pembimbing atas arahan dan bantuannya dalam pengerjaan Tugas Akhir ini.
5. M. Luqmanul Hakim Purnamawan selaku partner Penulis, yang membantu Penulis dalam mengerjakan Tugas Akhir dan juga memberi semangat kepada penulis untuk menyelesaikan Tugas Akhir.
6. Sahabat Penulis semasa kuliah Ilyas Bintang Prayogi a.k.a Pemuda Ceria dan Monica Indah Habsari a.k.a Ukhti Samyang yang telah menceriakan dan menemani hari-hari Penulis.

7. Rizky Fenaldo Maulana, S.Kom selaku ‘dosen pembimbing 3’ yang membimbing penulis dengan penuh kesabaran.
8. Geng Ululululululu yang mengajarkan penulis untuk tidak bertindak wacana.
9. Sahabat-sahabat AJK yang setia menemani di setiap malam-malam yang sunyi.
10. Teman-teman TC’14 yang sudah menyemangati Penulis selama menjalani masa perkuliahan di ITS.
11. Juga tidak lupa kepada semua pihak yang belum sempat disebutkan satu per satu yang telah membantu Penulis dalam menyelesaikan Tugas Akhir ini.

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, Juli 2018

Magista Bella Puspita

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi dan komunikasi pada zaman sekarang semakin pesat dan maju. Salah satu teknologi yang menunjang manusia untuk berkomunikasi dengan mudah yaitu *Vehicular Ad Hoc Networks* (VANET). *Vehicular Ad Hoc Networks* (VANET) merupakan pengembangan dari *Mobile Ad Hoc Networks* (MANET). Pada MANET node dalam jaringan memiliki mobilitas yang tinggi sehingga node tersebut bebas bergerak kemanapun, dan seringkali node mengubah rute dengan perangkat lain. Dalam konteks yang sama pada VANET kendaraan bertindak sebagai *mobile node* dan dapat berkomunikasi dengan kendaraan lain [1].

Routing protocol VANET dibagi menjadi dua model, yaitu *Topology-based Routing* dan *Geographic Routing.* Untuk *Topology-based Routing* masih dibagi menjadi dua, yaitu proaktif dan reaktif. *Proactive routing* adalah protokol yang bekerja dengan cara membentuk tabel routing dan melakukan *update* setiap saat pada selang waktu tertentu tanpa memperhatikan beban jaringan, *bandwidth* dan ukuran jaringan. Sedangkan *reactive routing* adalah merupakan mekanisme *routing* yang membentuk tabel *routing* jika ada permintaan pengiriman data atau terjadinya perubahan rute dalam setiap jaringan. Salah satu dari beberapa *routing protocol* yang reaktif yaitu AODV. AODV memiliki node dengan mobilitas yang tinggi sehingga tidak jarang sebelum paket data tersampaikan, rute yang terjalin sudah berubah. Oleh karena itu, rawan terjadi kehilangan paket data di tengah pengiriman.

Hello Interval merupakan interval waktu yang dibutuhkan *node* untuk mem-*broadcast Hello Message*. Dimana *Hello Message* bertugas untuk mengecek *node* tetangga mana saja yang tersedia sebagai kandidat *forwarding node* untuk menerima data paket ke *destination.* Oleh karena itu, apabila terdapat perubahan

pada *Hello Interval* maka akan sangat berpengaruh pada AODV secara keseluruhan.

Melihat permasalahan di atas, pada Tugas Akhir ini akan dilakukan analisis kinerja pada *routing protocol* AODV dengan mengubah *Hello Interval*. Hasil akhir yang diharapkan adalah mengetahui peran *Hello Interval* pada kinerja AODV yang diukur berdasarkan performansi *Packet Delivery Ratio* (PDR), *Routing Overhead*, *End-to-End Delay*, dan *Forwarded Route Request* (RREQ F). Sehingga di kemudian hari bisa dijadikan pedoman penelitian.

1.2 Rumusan Masalah

Berikut yang menjadi rumusan masalah pada Tugas Akhir ini adalah bagaimana pengaruh *Hello Interval* pada kinerja AODV di lingkungan VANET?

1.3 Batasan Permasalahan

Batasan masalah pada tugas akhir ini adalah sebagai berikut:

1. Jaringan yang digunakan adalah jaringan *Vehicular Ad hoc Networks* (VANET).
2. *Routing protocol* yang diujicobakan yaitu AODV.
3. Simulasi pengujian jaringan menggunakan *Network Simulator 2* (NS-2).
4. Pembuatan skenario uji coba menggunakan *Simulation of Urban Mobility* (SUMO).

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah mengetahui dampak *Hello Interval* pada kinerja AODV di lingkungan VANET.

1.5 Manfaat

Manfaat yang didapatkan dengan dibuatnya Tugas Akhir ini adalah sebagai pedoman penelitian yang berhubungan dengan pengembangan *Hello Interval* 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

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. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir. Tinjauan pustaka digunakan sebagai referensi yang mendukung pembuatan tugas akhir.

1.6.2 Studi Literatur

Literatur yang akan dipelajari untuk menunjang pembuatan sistem ini antara lain mengenai *Vehicular Ad Hoc Networks* (VANETs), *reactive routing protocol* AODV, dan *Network Simulator 2* (NS-2).

1.6.3 Analisis dan Desain Sistem

Pada tahap ini dilakukan analisis kinerja dari hasil percobaan pengubahan *Hello Interval* pada *reactive routing protocol* AODV. Data yang dianalisis berasal dari perhitungan *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), *End-to-End Delay*, dan *Forwarded Route Request* (RREQ F). Hal ini dimaksudkan untuk mengetahui dampak perubahan *Hello Interval* pada kinerja *reactive routing protocol* AODV dalam topologi VANET.

1.6.4 Implementasi Sistem

Pada tahap ini dilakukan perancangan model jaringan, implementasi yang dibuat berupa modifikasi protokol AODV untuk mengetahui kinerja *Hello Interval*.

1.6.5 Pengujian dan Evaluasi

Pengujian dilakukan dengan NS-2 *Network Simulator* dan akan menghasilkan *trace file*. Dari *trace file* tersebut akan dihitung *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), *End-to-End Delay*, dan *Forwarded Route Request* (RREQ F) untuk mengetahui performa *routing protocol* yang telah dimodifikasi.

1.6.6 Penyusunan Buku

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi sistem yang telah dibuat.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori atau penjelasan dari metode, algoritma, *library*, dan *tools* yang digunakan dalam penyusunan Tugas Akhir ini. Bab ini berisi tentang penjelasan singkat mengenai MANET, VANETs, AODV, NS2, SUMO, OpenStreetMap, Java OpenStreetMap (JOSM), AWK, dan *Hello Interval*.
3. Bab III. Perancangan
   Bab ini berisi pembahasan mengenai perancangan skenario mobilitas *grid* dan *real*, perancangan simulasi pada NS2, perancangan modifikasi AODV, serta perancangan metrik analisis (*Packet Delivery Ratio*, *Routing Overhead*, *End-to-End Delay*, dan *Forwarded Route Request* (RREQ F)).
4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi yang berbentuk kode sumber dari proses modifikasi protokol AODV, pembuatan simulasi pada NS2, SUMO, dan perhitungan metrik analisis.
5. Bab V. Uji Coba dan Evaluasi
   Bab ini berisikan hasil uji coba dan evaluasi dari implementasi yang telah dilakukan untuk menyelesaikan masalah yang dibahas pada Tugas Akhir.
6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam Tugas Akhir.
8. Lampiran
   Dalam lampiran terdapat tabel-tabel data hasil uji coba dan kode sumber program secara keseluruhan.

BAB II
TINJAUAN PUSTAKA

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

2.1 MANET

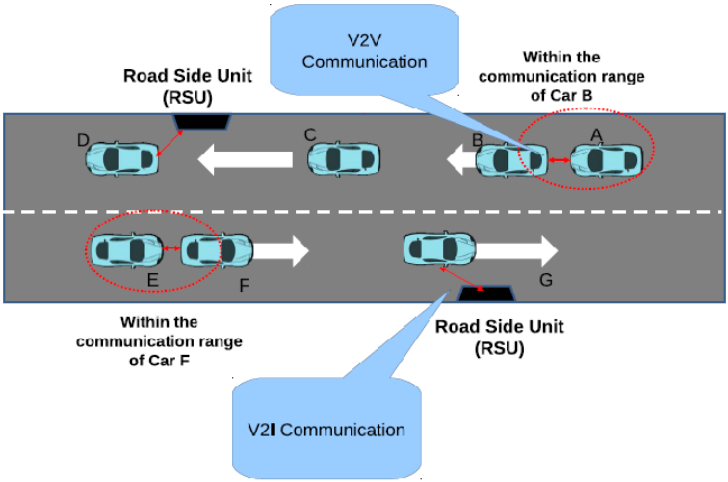
Mobile Ad-Hoc Network (MANET) adalah kumpulan *wireless mobile hosts* yang membentuk jaringan sementara tanpa bantuan infrastruktur yang berdiri sendiri atau administrasi terpusat. Dikarenakan mobilitas node pada jaringan, node ini mengorganisir dan mengonfigurasi diri sendiri. Tidak hanya bertindak sebagai *host*, node juga bisa berfungsi sebagai *router*. Node mengarahkan data menuju atau dari node lain dalam jaringan. Dalam MANET, protokol *routing* diperlukan untuk menemukan jalur spesifik antara sumber dan tujuan. Tujuan utama dari setiap protokol routing jaringan *ad-hoc* adalah untuk memenuhi tantangan topologi yang berubah secara dinamis. Oleh karena itu, rute yang efisien antara dua node dengan *routing overhead* dan konsumsi *bandwidth* seminimal mungkin harus dibangun. Perancangan protokol *routing* ini sangat menantang karena mobilitas dan sifat dinamis jaringan *mobile ad-hoc*. Protokol *routing* MANET dikategorikan menjadi dua jenis yaitu proaktif dan reaktif [2].

2.2 VANET

VANET adalah teknologi yang menjanjikan untuk komunikasi antar kendaraan di jalan. Mereka adalah bentuk khusus dari jaringan *mobile ad hoc* (MANET), yang memiliki fitur menarik dan unik yang membuatnya berbeda dari jenis MANET lainnya. Daya transmisi lebih tinggi, kemampuan komputasi yang lebih tinggi dan beberapa variasi mobilitas yang dapat diprediksi dibandingkan dengan MANET secara umum. VANET menggabungkan sejumlah besar aplikasi, seperti keselamatan lalu lintas jalan raya, pemantauan lalu

lintas yang kooperatif, pencegahan tabrakan, pengiriman konten dan perhitungan jalan secara *real-time*, sementara aplikasi tersebut memiliki tantangan untuk dicapai karena mobilitas yang tinggi dan perubahan topologi jaringan yang sering terjadi. Begitu kecepatan atau jalur kendaraan berubah, topologi jaringan akan bervariasi dalam VANET [3]**.**

Pada VANET mereka mengikuti Wi-Fi 802.11p dan WiMax 802.16 untuk komunikasi yang efisien antar kendaraan saat dalam perjalanan. Komunikasi dalam VANET termasuk dalam dua kategori yaitu komunikasi antar kendaraan dengan kendaraan yaitu komunikasi V2V dan yang kedua adalah komunikasi antar kendaraan dengan infrastruktur yaitu komunikasi V2I. Dalam VANET, RSU (*Road Side Unit*) dapat memberikan bantuan dalam menemukan fasilitas seperti restoran dan pom bensin serta mem-*broadcast* pesan yang terkait seperti kecepatan kendaraan kepada pengendara lain. Sebagai contoh sebuah kendaraan terhubung dengan lampu lalu lintas melalui komunikasi V2I dan lampu lalu lintas dapat memberikan informasi ke kendaraan ketika dalam keadaan lampu kuning atau merah. Ini dapat berfungsi sebagai tanda pemberitahuan kepada pengemudi dan akan sangat membantu para pengendara [4]. Namun, VANET juga menghadapi cukup banyak karakteristik sulit, seperti mobilitas dan skala besar. Node di VANET lebih dinamis karena sebagian besar kendaraan mampu melakukan perjalanan dengan kecepatan tinggi dan mengubah lokasi yang sering mengarah ke topologi jaringan yang dinamis, sementara hubungan antar node dapat sangat sering terhubung dan terputus [1]. Ilustrasi VANET dapat dilihat pada Gambar 2.1



Gambar 2.1 Ilustrasi VANET

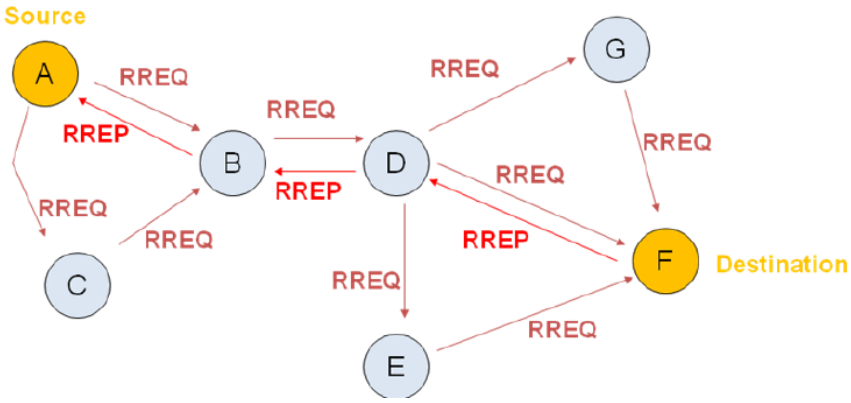
Dalam Tugas Akhir ini, akan diimplementasikan *routing protocol* AODV yang dimodifikasi dengan beberapa interval dan performa protokol tersebut akan diuji pada lingkungan VANET.

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

AODV adalah protokol *routing on-demand* yang terdiri dari metode *unicast* dan *multicast*. Umumnya di AODV, cara korespondensi antar node sumber dan tujuan dihitung saat ada kebutuhan transfer data, kapanpun sumber ingin mengirim paket data ke tujuan, maka perlu dicari jalur *routing* antara *source* dan *destination*. AODV menemukan jalur terpendek dengan bantuan mekanisme perutean dan jalur perutean ini harus dipelihara sampai *routing link* terputus. Jalur dibangun dan dipertahankan dengan: *RREQs Control Packet (Route REQuests), RREP Control Packet (Route Reply), RERRs Control Packet (Route Errors) dan RREP-Back Control Packet (Route Reply Acknowledgement).*

Control packet RREQ pertama kali dimulai oleh *source node* untuk menghitung jalur terpendek dalam multicast sort. *Control*

packet RREP dikirimkan oleh *destination node* dan dijawab ke node terdekat saat menerima *control packet* RREQ untuk menemukan tipe jalur *unicast*. Umumnya dalam protokol *routing* AODV *Hello message* pertama kali digunakan untuk menjaga keseragaman jalur rute yang telah dikenal sebelumnya. Setiap node dalam jaringan nirkabel mempertahankan tabel *routing*nya sendiri yang membantu untuk menemukan jalur *routing*, setiap tabel routing berisi informasi seperti: alamat tujuan, informasi lengkap node tetangga terdekat, jumlah *hop* antara *source node* dan *destination node*, periode penghentian setelah bagian paket dibuang, protokol *routing* AODV menggunakan konsep *sequence number* yang ditugaskan ke semua paket, yang membatasi penyiaran *control packet* awal yang tidak perlu; *sequence number* ini memungkinkan penggunaan rute baru untuk perpindahan node [5]. Ilustrasi pencarian rute oleh AODV dapat dilihat pada Gambar 2.2



Gambar 2.2 Mekanisme *route discovery* AODV

Pada setiap *node* yang menggunakan protokol AODV pasti memiliki sebuah *routing table* dengan *field* sebagai berikut:

- *Destination Address:* berisi alamat dari *node* tujuan.
- *Destination Sequence Number*: *sequence number* dari jalur komunikasi.

- *Next Hop*: alamat *node* yang akan meneruskan paket data.
- *Hop Count*: jumlah *hop* yang harus dilakukan agar paket dapat mencapai *node* tujuan.
- *Lifetime*: waktu dalam milidetik yang diperlukan *node* untuk menerima RREP.
- *Routing Flags*: status jalur. Terdapat tiga tipe status, yaitu *up* (valid), *down* (tidak valid) atau sedang diperbaiki.

Sebagai contoh proses *route discovery* dalam AODV, ilustrasi pada Gambar 2.2 menggambarkan bagaimana *source node*, yaitu *node* A mencari rute untuk menuju *destination node* yaitu *node* F. *Node* A akan membuat paket RREQ dan melakukan *broadcast* kepada semua *node* tetangganya (*neighbor node*). Jika *destination sequence number* yang terdapat pada paket RREQ sama atau lebih kecil dari yang ada pada *routing table* dan rute menuju *node* tujuan belum ditemukan, maka paket tersebut tidak akan dilanjutkan (*drop*). Jika *destination sequence number* pada RREQ lebih besar dibandingkan dengan yang terdapat pada *routing table*, maka *entry* pada *routing table* akan diperbarui dan paket tersebut akan diteruskan oleh *neighbor node* sekaligus membuat *reverse path* menuju *source node.* Paket RREQ akan diteruskan hingga mencapai *node* F. Kemudian, jika rute menuju *node* F sudah terbentuk di dalam *routing table* dan memiliki *routing flags* "*up*", maka *node* F akan mengirimkan paket RREP melalui rute tersebut menuju *node* .

Pada Tugas Akhir ini, digunakan *routing protocol* AODV yang akan diimplementasikan pada lingkungan VANET dengan beberapa skenario.

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

NS-2 adalah sebuah simulator jaringan yang dikembangkan di UC Berkely dan ditulis dalam bahasa C ++ dan OTcl yang berorientasi objek. NS-2 adalah alat yang sangat umum digunakan untuk mensimulasikan jaringan area lokal dan interlokal. Ini mengimplementasikan protokol jaringan seperti TCP dan UPD;

perilaku sumber lalu lintas seperti FTP, Telnet, Web, CBR dan VBR; Mekanisme manajemen antrian *router* seperti Drop Tail, RED dan CBQ; algoritma *routing* seperti Dijkstra, dan banyak lagi. NS-2 juga mengimplementasikan *multicasting* dan beberapa lapisan protokol MAC untuk simulasi LAN. Simulator ini merupakan *open source*, sehingga memungkinkan setiap orang untuk melakukan perubahan pada kode yang ada, selain menambahkan protokol dan fungsionalitas baru untuk itu. Hal ini yang membuat NS-2 lebih populer dibandingkan dengan aplikasi lainnya yang mana memudahkan pengguna untuk mengevaluasi fungsionalitas dan desain pada penelitian sebuah jaringan. Simulator dikembangkan dalam dua bahasa: C++ dan OTcl[1]. C++ digunakan untuk implementasi protokol yang terperinci seperti TCP. *TCL scripting* adalah aplikasi antarmuka untuk NS-2 yang digunakan untuk membuat pengaturan perintah dan antarmuka [6].

Pada Tugas Akhir ini, NS-2 digunakan untuk melakukan simulasi lingkungan VANET menggunakan protokol AODV yang sudah dimodifikasi. *Trace file* yang dihasilkan oleh NS-2 juga digunakan untuk mengukur peforma *routing* protokol AODV yang sudah dimodifikasi. NS-2 yang digunakan pada tugas akhir ini adalah NS-2 versi 2.35.

2.4.1 Instalasi

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

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

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

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

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

Gambar 2.4 Baris kode yang diubah pada file ls.h

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

2.4.2 *Trace File*

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

Tabel 2.1 Detail Penjelasan *Trace File* AODV

Kolom ke-	Penjelasan	Isi
1	*Event*	s : *sent* r : *received* f : *forwarded* D : *dropped*
2	*Time*	Waktu terjadinya *event*
3	ID *Node*	_x_ : identifier dari node yang melakukan *event*
4	*Layer*	Network layer tempat terjadi event AGT : *application* RTR : *routing* LL : *link layer* IFQ : *packet queue* MAC : *MAC*

		PHY : *physical*
5	*Flag*	--- : Tidak ada
6	*Sequence Number*	Nomor urut paket
7	*Packet Type*	AODV : paket *routing* AODV cbr : berkas paket CBR (*Constant Bit Rate*) RTS : *Request To Send* yang dihasilkan MAC 802.11 CTS : *Clear To Send* yang dihasilkan MAC 802.11 ACK : MAC ACK ARP : Paket *link layer address resolution protocol*
8	Ukuran	Ukuran paket pada *layer* saat itu
9	Detail MAC	[a b c d] a : perkiraan waktu paket b : alamat penerima c : alamat asal d : IP header
10	*Flag*	-------- : Tidak ada
11	*Detail* IP *source, destination,* dan *nexthop*	[a:b c:d e f] a : IP *source node* b : *port source node* c : IP *destination node* (jika -1 berarti *broadcast*) d : *port destination node* e : IP *header* ttl f : IP *nexthop* (jika 0 berarti *node* 0 atau *broadcast*)

2.5 *Simulation of Urban Mobility* (SUMO)

Simulation of Urban Mobility atau biasa dikenal dengan SUMO merupakan sebuah program *open source* simulator lalu lintas jalan yang memungkinkan pengguna untuk membangun simulasi pergerakan kendaraan pada topologi jaringan VANET yang disesuaikan. Pengimplementasian SUMO dimulai pada tahun 2001, dengan sebuah rilis *open source* tahun 2002, dikembangkan oleh Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker.

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 [4]:

- 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 netconvert digunakan untuk mengonversi peta dari OpenStreetMap.
- randomTrips.py
  Tool dalam SUMO untuk membuat rute acak yang akan dilalui oleh kendaraan dalam simulasi.
- duarouter
  Tool dalam SUMO untuk melakukan perhitungan rute berdasarkan definisi yang diberikan dan memperbaiki kerusakan rute.

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

Pada Tugas Akhir ini, SUMO digunakan untuk menghasilkan skenario VANET dan pergerakan *node* sehingga menyerupai keadaan lalu lintas yang sebenarnya. Untuk setiap skenario VANET yang dibuat menggunakan SUMO, akan dihasilkan pergerakan *node* yang acak sehingga setiap skenario memiliki pergerakan yang berbeda. SUMO yang digunakan pada Tugas Akhir ini adalah SUMO versi 0.25.0.

2.6 OpenStreetMap (OSM)

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 mengumpulan serta membebaskan data geografis yang tersedia di publik. Melalui *Open Data Commons Open Database License 1.0*, kontributor OSM dapat memiliki, memodifikasi, dan membagikan data peta secara luas. Terdapat beragam jenis peta digital yang tersedia di internet, namun sebagian besar memiliki keterbatasan secara legal maupun teknis. Hal ini membuat masyarakat, pemerintah, peneliti dan akademisi,

inovator, dan banyak pihak lainnya tidak dapat menggunakan data yang tersedia di dalam peta tersebut secara luas. Di sisi lain, baik peta dasar OSM maupun data yang tersedia di dalamnya dapat diunduh secara gratis dan terbuka, untuk kemudian digunakan untuk diredistribusikan kembali.

Di banyak tempat di dunia ini, terutama di daerah terpencil dan terbelakang secara ekonomi, tidak terdapat insentif komersil sama sekali bagi perusahaan pemetaan untuk mengembangkan data di tempat ini. OSM dapat menjadi jawaban di banyak tempat seperti ini, baik itu pengembangan ekonomi, tata kota, kontinjensi bencana, maupun untuk berbagai tujuan lainnya [7].

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

2.7 *Java OpenStreetMap Editor* (JOSM)

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

Pada Tugas Akhir ini, aplikasi ini digunakan 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. JOSM yang digunakan dalam Tugas Akhir ini yaitu JOSM versi 10301.

2.8 AWK

AWK merupakan sebuah command pada Linux yang didesain untuk *text processing* dan berfungsi sebagai alat penyaringan (*filtering tools*) yang fungsinya hampir sama dengan perintah *grep.* Perintah AWK juga biasanya dipakai untuk mengolah dan menganalisis *log file* yang isinya sangat panjang. AWK dapat juga

digunakan untuk melakukan proses aritmatika seperti yang dilakukan oleh perintah expr.

Pada Tugas Akhir ini, AWK digunakan untuk memproses data yang dihasilkan dari simulasi NS-2 dan mendapatkan analisis mengenai *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), *End-to-End Delay*, dan *Forwarded Route Request* (RREQ F).

2.9 *Hello Interval*

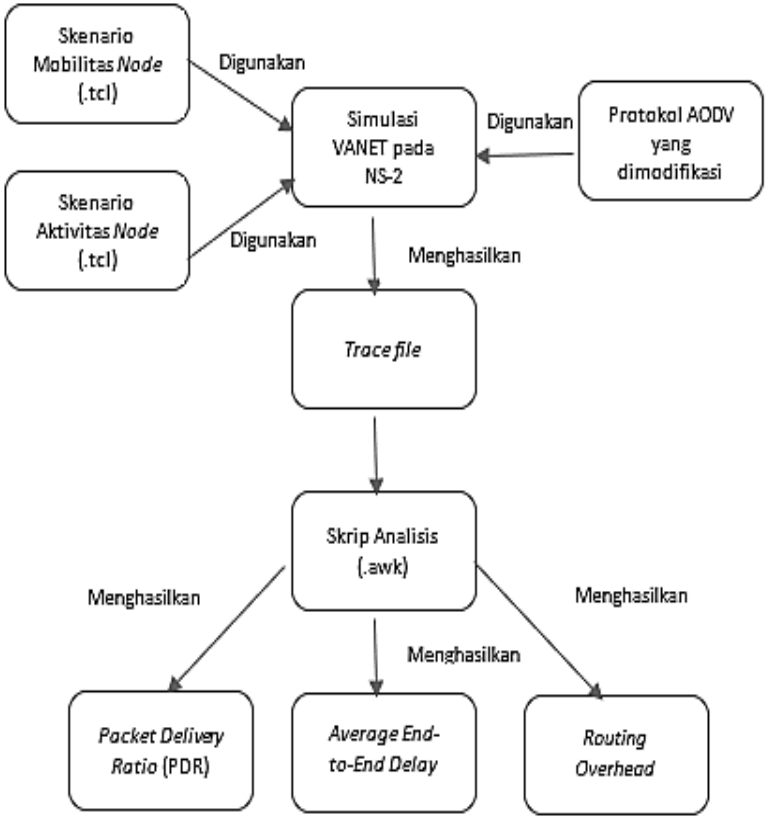
Hello Interval adalah jeda waktu yang diperlukan fungsi *sendHello*() pada AODV untuk mengecek node tetangga. Waktu yang dihasilkan dari rumus asli *Hello Interval* adalah antara 0.2 sampai 0.7 detik. Pada Tugas Akhir ini *Hello Interval* diubah menjadi statis dengan angka 5, 10, dan 15 detik.

BAB III
PERANCANGAN

Perancangan merupakan bagian penting dari pembuatan sistem secara teknis sehingga bab ini secara khusus menjelaskan perancangan sistem yang dibuat dalam Tugas Akhir. Berawal dari deskripsi umum sistem hingga perancangan skenario, alur dan implementasinya.

3.1 Deskripsi Umum

Pada Tugas Akhir ini akan dilakukan implementasi dan analisis dari *Hello Interval* pada *routing protocol* AODV yang dijalankan pada NS-2. Dimana terdapat 2 (dua) jenis skenario yang digunakan sebagai perbandingan pengukuran lalu lintas kota Surabaya, yaitu skenario *grid* dan *real*. Pada skenario *grid* peta dibuat dengan bantuan aplikasi SUMO. Sedangkan pada skenario *real* peta didapatkan dengan mengambil daerah yang diinginkan sebagai area simulasi menggunakan OpenStreetMap dengan bantuan *editor* JOSM. Setelah peta yang diinginkan terbentuk, dilakukan simulasi lalu lintas dengan menggunakan SUMO. Hasil simulasi SUMO (*trace file*) digunakan untuk mencari kinerja dari *Hello Interval* pada simulasi *routing protocol* AODV yang dijalankan pada NS-2. Kemudian hasil simulasi dari NS-2 dianalisis dengan menggunakan skrip AWK untuk mendapatkan *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), *End-to-End Delay* (E2E), dan *Forwarded Route Request* (RREQ F). Analisis tersebut bertujuan untuk mengetahui dampak perubahan *Hello Interval* pada kinerja protokol AODV. Diagram rancangan simulasi dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Alur Rancangan Simulasi

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

Tabel 3.1 Daftar Istilah

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

No.	Istilah	Penjelasan
2	PDR	*Packet Delivery Ratio*. Salah satu metrik analisis yang diukur. Berupa rasio jumlah pengiriman paket yang terkirim.
3	E2E	*Average End-to-End Delay*. Jeda waktu yang diukur saat paket terkirim.
4	RO	*Routing Overhead*. Jumlah *control packet* yang terkirim
5	*Hello Interval*	*Hello Interval* adalah jeda waktu yang diperlukan fungsi *sendHello*() pada AODVuntuk mengecek node tetangga.

3.2 Perancangan Skenario Mobilitas

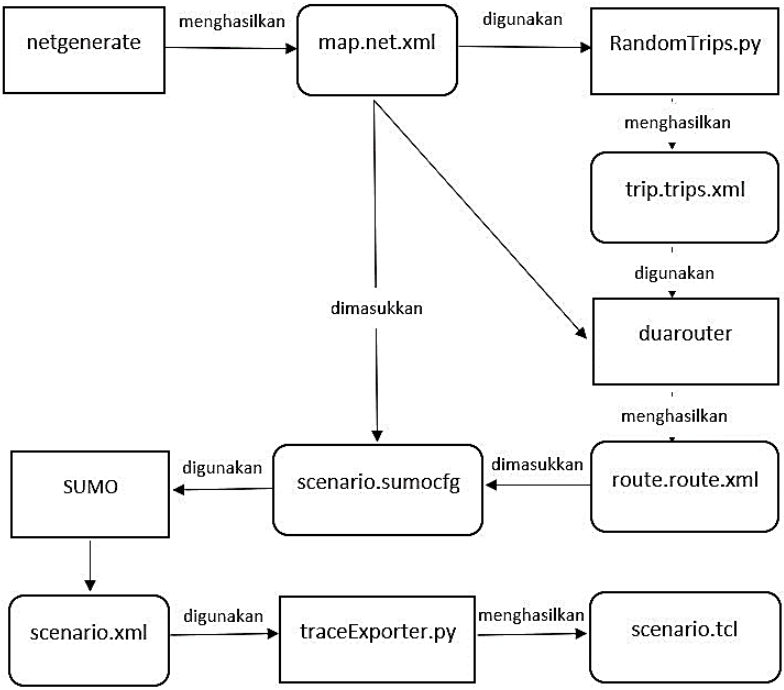
Perancangan skenario mobilitas dimulai dengan membuat area simulasi, pergerakan *node*, dan implementasi pergerakan *node*. Dalam Tugas Akhir ini, terdapat dua macam area simulasi yang akan digunakan yaitu peta *grid* dan *real*. Peta *grid* yang dimaksud adalah bentuk jalan berpetak – petak sebagai 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 adalah salah satu area di Surabaya.

3.2.1 Perancangan Skenario *Grid*

Perancangan skenario mobilitas *grid* diawali dengan merancang luas area peta *grid* yang dibutuhkan. Luas area tersebut bisa didapatkan dengan cara menentukan terlebih dahulu jumlah titik persimpangan yang diinginkan, sehingga dari jumlah persimpangan

tersebut dapat diketahui berapa banyak peta yang dibutuhkan. Dengan mengetahui jumlah petak yang dibutuhkan, dapat ditentukan panjang tiap petak sehingga mendapatkan luas area yang dibutuhkan yaitu 1300 m x 1300 m. Dengan 4 titik persimpangan, maka akan didapatkan 9 petak dan panjang tiap peta adalah 400m.

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.



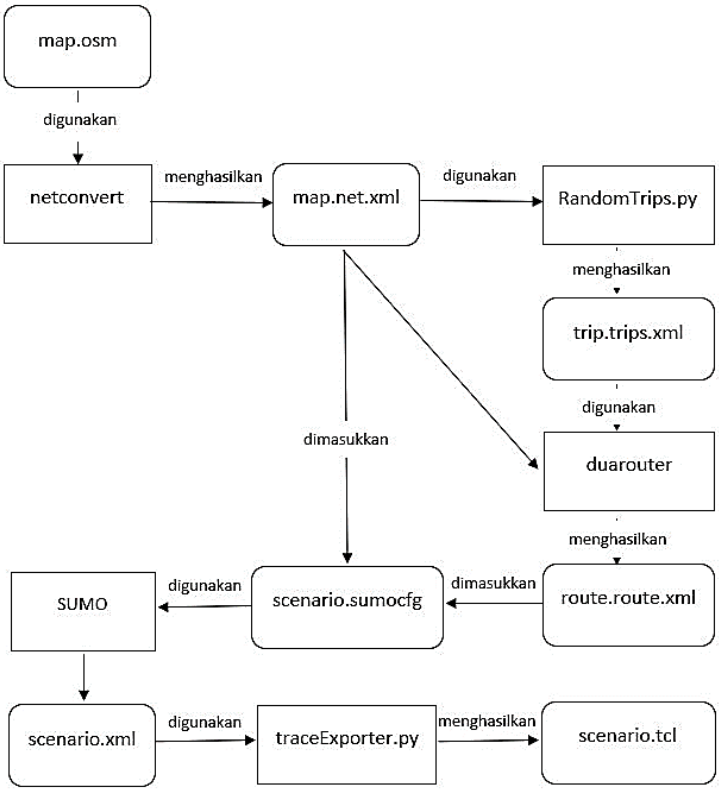
Gambar 3.2 Alur Pembuatan Skenario *Grid*

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. Alur pembuatan skenario *grid* dapat dilihat pada Gambar 3.2.

3.2.2 Perancangan Skenario *Real*

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

Setelah mendapatkan peta area yang akan dijadikan simulasi, peta tersebut dikonversi ke dalam bentuk *file* dengan ekstensi .net.xml menggunakan *tools* SUMO yaitu netconvert. Tahap berikutnya memiliki tahapan yang sama seperti merancang skenario *grid*, yaitu membuat pergerakan *node* menggunakan randomTrips dan duarouter. Kemudian dilakukan penggabungan *file* peta *real* yang sudah dikonversi ke dalam *file* dengan ekstensi .net.xml dan *file* pergerakan *node* yang sudah dibuat sebelumnya. Hasil dari penggabungan tersebut merupakan *file* skenario berkekstensi .xml. *File* yang dihasilkan tersebut dikonversi ke dalam bentuk *file* dengan ekstensi .tcl agar dapat diterapkan pada NS-2. Alur pembuatan skenario *real* dapat dilihat pada Gambar 3.3.



Gambar 3.3 Alur Pembuatan Skenario *Real*

3.3 Perancangan Simulasi pada NS-2

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

Kode yang diubah diantaranya adalah interval pada *Hello Interval* pada fungsi *sendHello*() AODV. Pada saat simulasi NS-2 dijalankan, maka akan diketahui performa dari *Hello Interval*.

3.4 Perancangan Metrik Analisis

Berikut ini merupakan parameter – parameter yang akan dianalisis pada Tugas Akhir ini untuk dapat mengetahui performa dari *Hello Interval* pada AODV:

3.4.1 *Packet Delivery Ratio* (PDR)

Packet delivery ratio merupakan perbandingan dari jumlah paket data yang dikirim dengan paket data yang diterima. PDR dapat menunjukkan keberhasilan paket yang dikirimkan. Semakin tinggi PDR artinya semakin berhasil pengiriman paket yang dilakukan Rumus untuk menghitung PDR dapat dilihat pada persamaan 3.1.

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

Keterangan:

PDR = *Packet Delivery Ratio*

received = banyak paket data yang diterima

sent = banyak paket data yang dikirimkan

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

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

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

Keterangan:

E2E = *End-to-End Delay*

CBRRecvTime = Waktu *node* asal mengirimkan paket

CBRSentTime = Waktu *node* tujuan menerima paket

recvnum = Jumlah paket yang berhasil diterima

3.4.3 *Routing Overhead* (RO)

Routing Overhead adalah jumlah paket kontrol *routing* yang ditransmisikan per data paket ke *node* tujuan selama simulasi terjadi. *Routing Overhead* didapatkan dengan menjumlahkan semua paket kontrol *routing* yang ditransmisikan, baik itu paket *route request* (RREQ), *route reply* (RREP), maupun *route error* (RERR). Perhitungan *Routing Overhead* dapat dilihat dengan persamaan 3.3.

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

3.4.4 *Forwarded Route Request* (RREQ F)

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

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

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

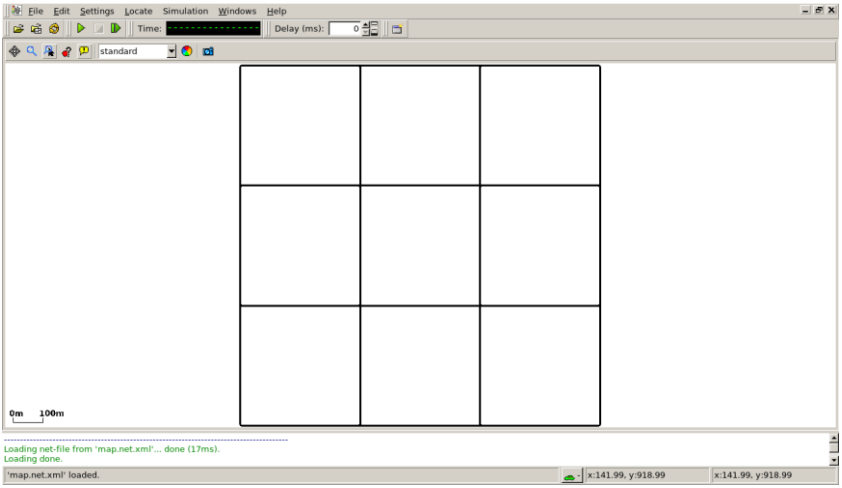
4.1.1 Skenario *Grid*

Dalam mengimplementasikan skenario *grid*, SUMO menyediakan *tools* untuk membuat peta *grid* yaitu netgenerate. Pada Tugas Akhir ini, peta *grid* dibuat dengan luas 1300 m x 1300 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 1300 m x 1300 m dibutuhkan luas per petak sebesar 400 m x 400 m. Berikut perintah netgenerate untuk membuat peta tersebut dengan kecepatan *default* kendaraan sebesar 20m/s dapat dilihat pada Gambar 4.1.

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

Gambar 4.1 Perintah netgenerate

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



Gambar 4.2 Hasil *Generate* Peta *Grid*

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

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

Gambar 4.3 Perintah randomTrips

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

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

Gambar 4.4 Perintah duarouter

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

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

Gambar 4.5 File Skrip .sumocfg

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

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

Gambar 4.6 Perintah SUMO untuk membuat skenario .xml

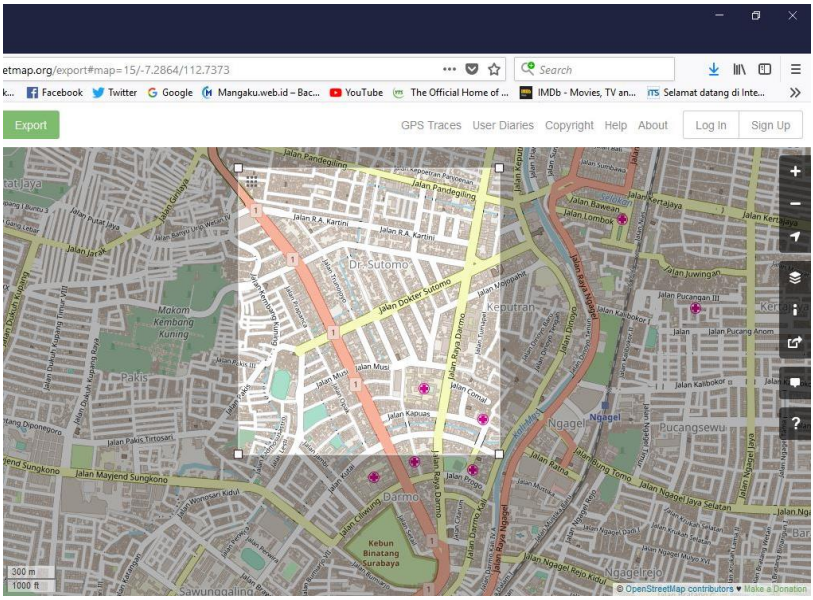
File skenario berekstensi .xml selanjutnya dikonversi ke dalam bentuk *file* berekstensi .tcl agar dapat disimulasikan menggunakan NS-2. *Tools* yang digunakn untuk melakukan konversi ini adalah traceExporter. Perintah untuk menggunakan traceExporter dapat dilihat pada Gambar 4.7.

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

Gambar 4.7 Perintah traceExporter

4.1.2 Skenario *Real*

Dalam mengimplementasikan skenario *real*, langkah pertama adalah menentukan area yang akan dijadikan area simulasi. Pada Tugas Akhir ini area jalan yang diambil yaitu sekitar Jl. Dr. Soetomo Surabaya. Area yang diambil seluas 1300m x 1300m. Setelah menentukan area simulasi, ekspor data peta dari OpenStreetMap seperti yang ditunjukkan pada Gambar 4.8.



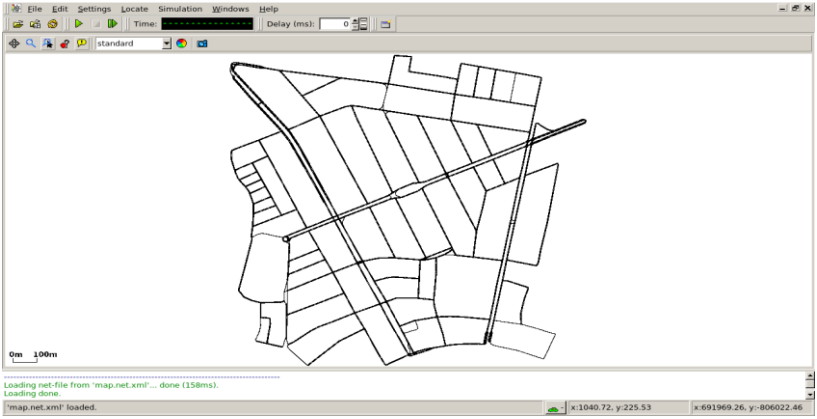
Gambar 4.8 Ekspor Peta dari OpenStreetMap

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 Fungsi *sendHello* untuk Menentukan *HelloInterval*

Pada Tugas Akhir ini dilakukan modifikasi pada *routing protocol* AODV dengan mengubah fungsi *sendHello*().Hal tersebut

dilakukan dengan cara menentukan *HelloInterval* dari yang semula rumus asli pada AODV menjadi angka statis 5, 10, dan 15.

Kode implementasi dari routing protocol AODV pada NS-2 versi 2.35 berada pada direktori ns-2.35/aodv. Pada direktori tersebut terdapat beberapa file diantaranya seperti aodv.cc, aodv.h dan sebagainya. Pada Tugas Akhir ini, *file* aodv.cc yang terdapat dalam *folder* ns-2.35/aodv dimodifikasi untuk mengetahui performa dari *Hello Interval*. Pada bagian ini akan dijelaskan langkah – langkah dalam mengimplementasikan modifikasi *routing protocol* AODV.

```
agent->sendHello();
   double interval = 5;
   assert(interval >= 0);
   Scheduler::instance().schedule(this,
&intr, interval);
```

Gambar 4.11 Potongan Kode Fungsi *sendHello*()

4.3 Implementasi Simulasi pada NS-2

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

```
set val(chan)    Channel/WirelessChannel
set val(prop)    Propagation/TwoRayGround
set val(netif)   Phy/WirelessPhy
set val(mac)     Mac/802_11
set val(ifq)     Queue/DropTail/PriQueue
set val(ll)      LL
set val(ant)     Antenna/OmniAntenna
set opt(x)       1500
set opt(y)       1500
set val(ifqlen)  1000
set val(nn)      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 4.12 Implementasi Simulasi NS-2

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

Konfigurasi untuk *file traffic* bisa dilakukan dengan membuat *file* berekstensi .txt untuk menyimpan konfigurasi tersebut. Pada *file* konfigurasi lingkungan simulasi, *file traffic* tersebut dimasukkan agar dibaca sebagai *file traffic*. Potongan konfigurasi *file traffic* dapat dilihat pada Gambar 4.13.

```
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(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"
```

Gambar 4.13 Implementasi Simulasi File Traffic

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

4.4 Implementasi Metrik Analisis

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

4.4.1 Implementasi *Packet Delivery Ratio* (PDR)

Pada subbab 2.4.2 telah ditunjukkan contoh struktur data *event* yang dicatat dalam *trace file* oleh NS-2. Kemudian, pada persamaan 3.1 telah dijelaskan bagaimana menghitung PDR. Skrip awk untuk menghitung PDR berdasarkan kedua informasi tersebut

dapat dilihat pada lampiran A.4 Kode Skrip AWK *Packet Delivery Ratio*.

PDR didapatkan dengan cara menghitung setiap baris terjadinya *event* pengiriman dan penerimaan paket data yang dikirim melalui agen pada *trace file*. Skrip menyaring setiap baris yang mengandung *string* AGT karena kata kunci tersebut menunjukan *event* yang berhubungan dengan paket komunikasi data. Penghitungan dilakukan dengan menjumlahkan paket yang dikirimkan dan paket yang diterima dengan menggunakan karakter pada kolom pertama sebagai *filter*. Kolom pertama menunjukan event yang terjadi dari sebuah paket. Setelah itu nilai PDR dihitung dengan cara persamaan 3.1. Pseudocode untuk menghitung PDR dapat dilihat pada Gambar 4.14.

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

Gambar 4.14 Pseudocode untuk Menghitung PDR

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

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

Skrip awk untuk menghitung E2E dapat dilihat pada lampiran A.5 Kode Skrip AWK Rata-Rata *End-to-End Delay*.

Dalam perhitungan E2E, langkah yang digunakan untuk mendapatkan E2E hampir sama dengan ketika mencari PDR, hanya saja yang perlu diperhatikan adalah waktu dari sebuah *event* yang tercatat pada kolom ke-2 dengan *filter event* pada kolom ke-4 adalah layer AGT dan *event* pada kolom pertama guna membedakan paket dikirim atau diterima. Setelah seluruh baris yang memenuhi didapatkan, akan dihitung *delay* dari paket dengan mengurangi waktu dari paket diterima dengan waktu dari paket dikirim dengan syarat memiliki *id* paket yang sama.

Setelah mendapatkan *delay* paket, langkah selanjutnya adalah dengan mencari rata-rata dari *delay* tersebut dengan menjumlahkan semua *delay* paket dan membaginya dengan jumlah paket. *Pseudocode* untuk menghitung rata-rata E2E dapat dilihat pada Gambar 4.15.

```
sum_delay = 0
counter = 0
for i = 1 to the number of rows
     counter++
     if layer == AGT and event == s then
      start_time[packet_id] = time
     else if layer == AGT and event == r then
      end_time[packet_id] = time
     end if
     delay[packet_id] = end_time[packet_id] –
start_time[packet_id]
     sum_delay += delay[packet_id]
e2e = sum_delay / counter
```

Gambar 4.15 Pseudocode untuk Perhitungan Rata-Rata E2E

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

4.4.3 Implementasi *Routing Overhead* (RO)

Seperti yang telah dijelaskan sebelumnya, *routing overhead* merupakan jumlah dari paket kontrol *routing* baik itu RREQ, RREP, maupun RERR. Dengan begitu, untuk mendapatkan RO yang perlu dilakukan adalah menjumlahkan tiap paket dengan *filter event sent* pada kolom pertama dan *event layer* RTR pada kolom ke-4. Perhitungan RO telah dijelaskan pada persamaan 3.3. Skrip AWK untuk menghitung RO dapat dilihat pada lampiran A.6 Kode Skrip AWK *Routing Overhead*. *Pseudocode* untuk menghitung RO dapat dilihat pada Gambar 4.16.

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

Gambar 4.16 Pseudocode untuk Perhitungan *Routing Overhead*

Contoh perintah pengeksekusian skrip awk untuk menganalisis *trace file* adalah `awk -f ro.awk scenario1.tr`.

4.4.4 Implementasi *Forwarded Route Request* (RREQ F)

Forwarded Route Request (RREQ F) adalah jumlah paket *control route request* yang diteruskan per-data paket ke *node* tujuan selama simulasi terjadi. Dengan begitu, untuk mendapatkan RREQ F yang perlu dilakukan adalah menjumlahkan tiap paket dengan *filter event sent* pada kolom pertama, *event layer* RTR pada kolom ke-4, dan *event layer route request* pada kolom-25. Penyaringan juga dilakukan pada kolom ke-3 yang menunjukan *node* id. Selama *node* bukan *node* sumber, maka akan terus dilakukan penjumlahan baris

yang terdapat pada *file* dengan ekstensi .tr. Perhitungan RREQ F telah dijelaskan pada persamaan 3.4. Skrip awk untuk menghitung RREQ F dapat dilihat pada lampiran A.7 Kode Skrip AWK *Forwarded Route Request* . Pseudocode untuk menghitung RREQ F dapat dilihat pada Gambar 4.17.

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

Gambar 4.17 Pseudocode Perhitungan *Forwarded Route Request*

Contoh perintah pengeksekusian skrip awk untuk menganalisis *trace file* adalah `awk -f rreqf.awk scenario1.tr.`

(Halaman ini sengaja dikosongkan)

BAB V
UJI COBA 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 (TM) i3-2120 CPU @ 3.30GHz
Sistem Operasi	Ubuntu 16.04 LTS
Linux Kernel	Linux kernel 4.4
Memori	4.0 GB

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

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

Parameter lingkungan uji coba yang digunakan pada NS-2 dapat diihat pada Tabel 5.2. Pengujian dilakukan dengan menjalankan skenario yang disimulasikan pada NS-2. Dari simulasi tersebut dihasilkan sebuah *trace file* dengan ekstensi .tr yang akan dianalisis dengan bantuan skrip awk untuk mendapatkan PDR, E2E, RO, dan RREQ F menggunakan kode yang terdapat pada lampiran A.4 Kode Skrip AWK *Packet Delivery Ratio*, A.5 Kode Skrip AWK Rata-Rata *End-to-End Delay*, A.6 Kode Skrip AWK *Routing Overhead*, dan A.7 Kode Skrip AWK *Forwarded Route Request.*

Tabel 5.2 Lingkungan Uji Coba

No.	Parameter	Spesifikasi
1	Network simulator	NS-2.35
2	Routing protocol	AODV
3	Waktu simulasi	200 detik
4	Area simulasi	1500 m x 1500 m
5	Jumlah *Node*	60, 150, 300
6	Radius transmisi	400m
7	Kecepatan maksimum	20 m/s
8	Protokol MAC	IEEE 802.11p
9	Model Propagasi	*Two-ray ground*

5.2 Hasil Uji Coba

Hasil uji coba dari skenario *grid* dan skenario *real* dari masing-masing analisis metric, yaitu *Packet Delivery Ratio* (PDR), *Average End-to-End Delay*, *Routing Overhead* (RO), dan *Forwarded Route Request* (RREQ F) dapat dilihat sebagai berikut :

5.2.1 Hasil Uji Coba Skenario *Grid*

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

Pengambilan data uji PDR, RO, RREQ F, dan E2E pada skenario *grid* dilakukan sebanyak 10 kali dengan skenario mobilitas *random* pada peta *grid* dengan luas area 1300 m x 1300 m dan *node* sebanyak 60 untuk lingkungan yang jarang, 150 *node* untuk lingkungan yang sedang, dan 300 *node* untuk lingkungan yang padat dilakukan pada kecepatan standar yaitu 20 m/s. Untuk uji coba setiap lingkungan menggunakan interval yang berbeda-beda untuk mencari nilai interval yang terbaik dari hasil skenario. Interval waktu yang digunakan adalah 5 detik, 10 detik dan 15 detik.

Pada Tugas Akhir yang terdahulu sudah dilakukan penelitian terhadap pencarian *node* tetangga. Maka pada Tugas Akhir ini berfokus pada waktu yang diperlukan untuk pencarian *node* tetangga. Pada kasus ini maka digunakan waktu yang sudah ditentukan intervalnya. Hasil simulasi dapat dilihat pada Tabel 5.3, Tabel 5.4, Tabel 5.5, dan Tabel 5.6.

Tabel 5.3 Hasil Rata - Rata PDR Skenario Grid

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**			**Perbedaan**		
		5s	**10s**	**15s**	**5s**	**10s**	**15s**
60	0.808	0.809	0.759	0.789	0.001	0.048	0.019
150	0.867	0.832	0.895	0.899	0.035	0.028	0.032
300	0.873	0.705	0.754	0.816	0.168	0.118	0.057

Tabel 5.4 Hasil Rata - Rata E2E Skenario Grid

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi (ms)**			**Perbedaan**		
		5s	**10s**	**15s**	**5s**	**10s**	**15s**
60	658.770	745.595	800.695	917.613	86.825	141.925	258.843
150	599.404	1,003.873	680.685	720.978	404.469	81.281	121.574
300	1,075.968	1,501.456	2,913.859	1,185.089	425.488	1,837.890	109.121

Tabel 5.5 Hasil Rata - Rata RO Skenario Grid

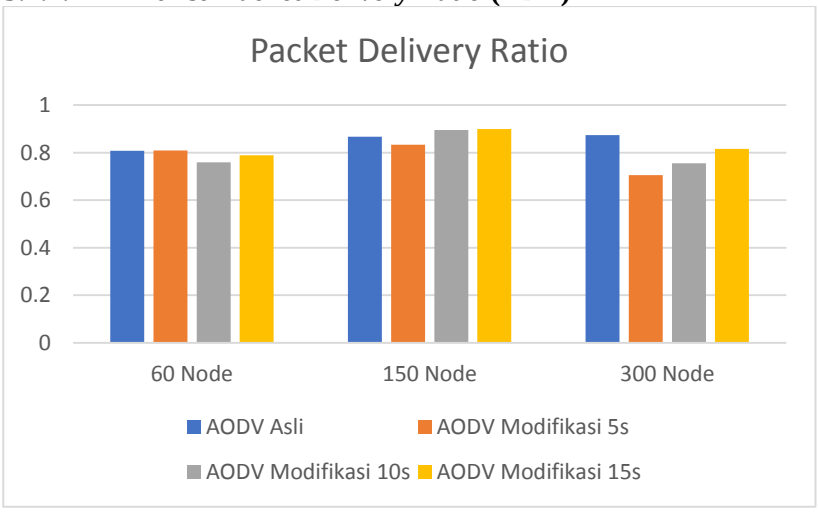
Jumlah *Node*	**AODV Asli**	**AODV Modifikasi (packet)**			**Perbedaan**		
		5s	**10s**	**15s**	**5s**	**10s**	**15s**
60	14,697.500	7,861.300	5,257.200	4,547.200	6,836.200	9,440.300	10,150.300

150	69,98 7.500	38,09 9.100	25,14 0.900	19,52 1.900	31,88 8.400	44,84 6.600	50,46 5.600
300	63,13 1.300	35,21 8.700	20,59 8.800	16,54 4.100	27,91 2.600	42,53 2.500	46,58 7.200

Tabel 5.6 Hasil Rata - Rata RREQ F Skenario Grid

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi (packet)**			**Perbedaan**		
		5s	**10s**	**15s**	**5s**	**10s**	**15s**
60	2,099 .500	3,573. 600	2,929. 200	2,726. 300	1,474. 100	829.7 00	626.8 00
150	9,179 .300	20,90 4.000	16,42 9.100	13,80 1.000	11,72 4.700	7,249. 800	4,621 .700
300	3,284 .200	19,69 6.200	13,42 2.200	11,82 3.300	16,41 2.000	10,13 8.000	8,539 .100

5.2.1.1 Analisa *Packet Delivery Ratio* (PDR)



Gambar 5.1 Grafik PDR Skenario *Grid*

Berdasarkan grafik pada Gambar 5.1, dapat dilihat bahwa *routing protocol* AODV asli dan AODV yang telah dimodifikasi dengan interval 5 detik, 10 detik dan 15 detik perubahan yang fluktuaktif. Pada lingkungan yang jarang dengan *node* berjumlah 60 dan dibandingkan dengan hasil PDR AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih PDR sebesar 0.001, dimana terjadi kenaikan sebesar 0.12 %, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih PDR sebesar 0.048, dimana terjadi penurunan sebesar 6 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih PDR sebesar 0.019, dimana terjadi penurunan sebesar 2.35 %. Berdasarkan hal tersebut, routing protocol AODV modifikasi dengan interval 5 detik berhasil mengungguli *routing protocol* AODV asli, *routing protocol* AODV modifikasi dengan interval 10 detik dan *routing protocol* AODV dengan interval 15 detik.

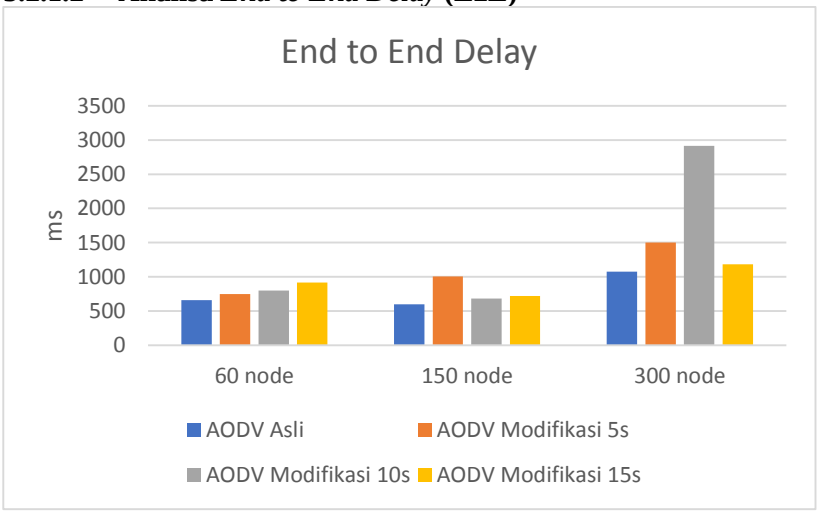
Pada lingkungan yang sedang dengan *node* berjumlah 150 dan dibandingkan dengan hasil PDR AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih PDR sebesar 0.035, dimana terjadi penurunan sebesar 4%, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih PDR sebesar 0.028, dimana terjadi kenaikan sebesar 3.27 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih PDR sebesar 0.032, dimana terjadi kenaikan sebesar 3.7%. Berdasarkan hal tersebut, routing protocol AODV modifikasi dengan interval 15 detik berhasil mengungguli *routing protocol* AODV asli, *routing protocol* AODV modifikasi dengan interval 5 detik *dan routing protocol* AODV dengan interval 10 detik.

Pada lingkungan yang padat dengan *node* berjumlah 300 dan dibandingkan dengan hasil PDR AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih PDR sebesar 0.168, dimana terjadi penurunan sebesar 19.2 %, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih PDR sebesar 0.118, dimana terjadi penurunan sebesar 13.56 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan

perbedaan selisih PDR sebesar 0.057, dimana terjadi penurunan sebesar 6.5 %. Berdasarkan hal tersebut, *routing protocol* AODV asli berhasil mengungguli *routing protocol* AODV modifikasi 5 detik, *routing protocol* AODV modifikasi dengan interval 10 detik, dan *routing protocol* AODV dengan interval 15 detik.

Pada lingkungan *node* dengan tingkat kepadatan jarang, AODV modifikasi dengan interval 5 detik menghasilkan *Packet Delivery Ratio* (PDR) yang paling tinggi dibanding AODV yang lain yaitu sebesar 0.809. Pada lingkungan *node* dengan tingkat kepadatan sedang, AODV modifikasi dengan interval 15 detik menghasilkan *Packet Delivery Ratio* (PDR) yang paling tinggi dibanding AODV yang lain yaitu sebesar 0.899. Pada lingkungan *node* dengan tingkat kepadatan padat, AODV asli menghasilkan *Packet Delivery Ratio* (PDR) yang paling tinggi dibanding AODV yang lain yaitu sebesar 0.873. Berdasarkan hasil simulasi pada ketiga lingkungan tersebut, dapat dilihat bahwa tidak dapat disimpulkan mana yang lebih unggul karena pada setiap percobaan dihasilkan *Packet Delivery Ratio* (PDR) yang berbeda di lingkungan *node* dengan tingkat kepadatan jarang, sedang, maupun padat.

5.2.1.2 Analisa *End to End Delay* (E2E)



Gambar 5.2 Grafik E2E Skenario *Grid*

Berdasarkan grafik pada Gambar 5.2 dapat dilihat bahwa rata-rata E2E antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi dengan interval 5 detik, 10 detik dan 15 detik mengalami perubahan yang fluktuatif. Pada lingkungan yang jarang dengan jumlah 60 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 86.825 ms dimana AODV asli unggul dalam hal ini, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 141.925 ms dimana AODV asli unggul dalam hal ini dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 258.843 ms dimana AODV asli unggul dalam hal ini.

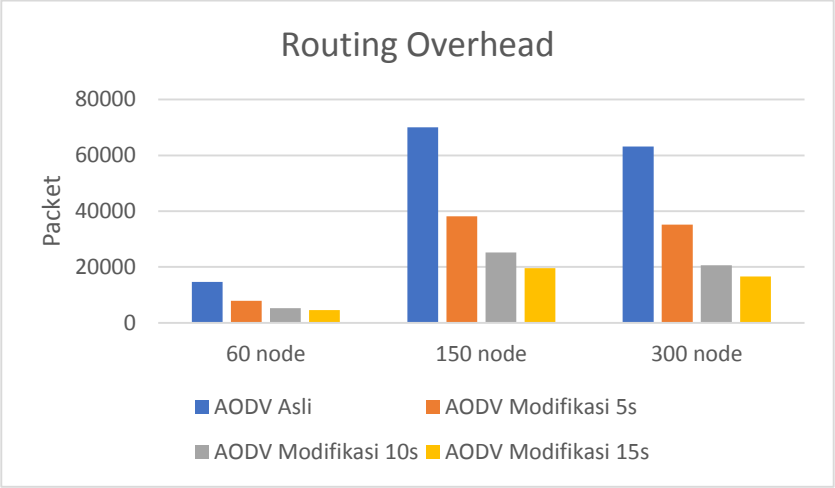
Pada lingkungan yang sedang dengan jumlah 150 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 404.469 ms dimana AODV asli unggul dalam hal ini, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 81.281 ms dimana AODV asli unggul dalam hal ini dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 121.574 ms dimana AODV asli unggul dalam hal ini.

Pada lingkungan yang padat dengan jumlah 300 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 425.488 ms dimana AODV asli unggul dalam hal ini, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 1,837.890 ms dimana AODV asli unggul dalam hal ini dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 109.121 ms dimana AODV asli unggul dalam hal ini.

Berdasarkan hasil simulasi pada ketiga lingkungan tersebut, dapat dilihat bahwa jumlah *node* yang sedikit atau lingkungan jarang belum tentu menghasilkan rata rata *End to End Delay* yang lebih bagus atau lebih sedikit daripada di lingkungan dengan jumlah *node*

yang sedang maupun yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. AODV asli masih mengungguli AODV modifikasi dengan interval 5 detik, AODV modifikasi dengan interval 10 detik dan AODV modifikasi dengan interval 15 detik. Dapat dilihat pula bahwa AODV asli menghasilkan *End to End Delay* yang lebih bagus atau dalam hal ini lebih rendah daripada AODV modifikasi dengan jumlah selisih *End to End Delay* yang cukup signifikan. Hasil rata – rata E2E pada AODV asli yaitu 778.047 ms. Hal ini dikarenakan waktu *delay* tergantung dari rata – rata waktu paket yang terkirim. Semakin banyak paket yang terkirim, maka semakin beragam *delay*nya.

5.2.1.3 Analisa *Routing Overhead* (RO)



Gambar 5.3 Grafik Routing Overhead Skenario *Grid*

Berdasarkan grafik pada Gambar 5.3, dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi dengan interval 5 detik, 10 detik dan 15 detik dan juga *routing protocol* AODV asli mengalami perubahan yang signifikan. Pada lingkungan yang jarang dengan jumlah 60 *node* dan dibandingkan dengan AODV asli, AODV

modifikasi dengan interval 5 detik menghasilkan perbedaan selisih RO sebesar 6,836.2, dimana terjadi penurunan RO sebesar 46.51 %, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih RO sebesar 9,440.3, dimana terjadi penurunan RO sebesar 64.23 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih RO sebesar 10,150.3 dimana terjadi penurunan RO sebesar 69.06 %. Berdasarkan hasil tersebut, maka *routing protocol* AODV yang telah dimodifikasi dengan interval 15 detik unggul dalam hal RO tersebut karena menghasilkan RO yang lebih rendah dari AODV asli.

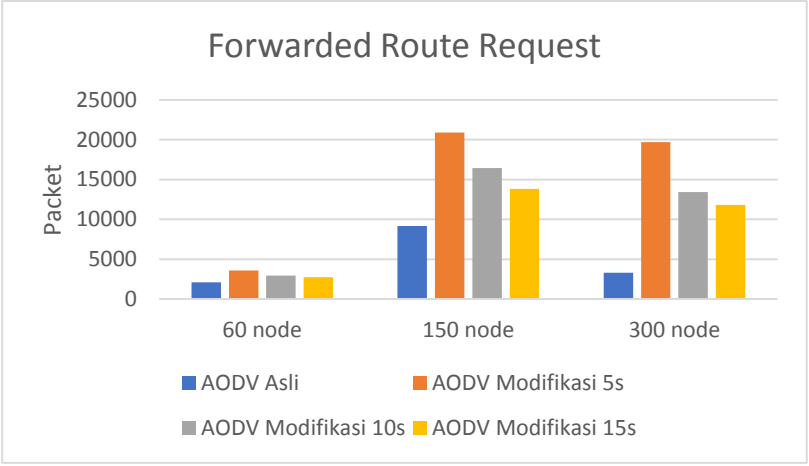
Pada lingkungan yang sedang dengan jumlah 150 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih RO sebesar 31,888.4, dimana terjadi penurunan RO sebesar 45.56 %, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih RO sebesar 44,846.6, dimana terjadi penurunan RO sebesar 64.08 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih RO sebesar 50,465.6, dimana terjadi penurunan RO sebesar 72.11 %. Berdasarkan hasil tersebut, maka *routing protocol* AODV yang telah dimodifikasi dengan interval 15 detik unggul dalam hal RO tersebut karena menghasilkan RO yang lebih rendah dari AODV asli.

Pada lingkungan yang padat dengan jumlah 300 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih RO sebesar 27,912.6, dimana terjadi penurunan RO sebesar 44.21 %, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih RO sebesar 42,532.5, dimana terjadi penurunan RO sebesar 67.37 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih RO sebesar 46,587.2, dimana terjadi penurunan RO sebesar 73.79 %. Berdasarkan hasil tersebut, maka *routing protocol* AODV yang telah dimodifikasi dengan interval 15 detik unggul dalam hal RO tersebut karena menghasilkan RO yang lebih rendah dari AODV asli.

Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang lebih sedikit atau pada lingkungan dengan *node* yang jarang, menghasilkan RO yang lebih sedikit daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. AODV modifikasi dengan interval 15 detik berhasil mengungguli AODV asli, AODV modifikasi dengan interval 5 detik dan AODV modifikasi dengan interval 10 detik dengan nilai rata – rata penurunan RO pada AODV yang dimodifikasi adalah sebesar 60.77 %. Dapat dilihat pula bahwa AODV yang telah dimodifikasi dengan interval 15 detik menghasilkan RO yang lebih bagus atau dalam hal ini lebih rendah daripada AODV asli.

5.2.1.4 Analisa *Forwarded Route Request* (RREQ F)

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



Gambar 5.4 Grafik *Forwarded Route Request* Skenario *Grid*

Berdasarkan grafik pada Gambar 5.4, dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi dan juga *routing protocol* asli mengalami perubahan *forwarded route request* (RREQ F) yang signifikan. Pada lingkungan yang jarang dengan jumlah 60 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 1,474.1, dimana terjadi kenaikan RREQ F sebesar 70.21 %, sedangkan AODV modifikasi dengan 10 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 829.7, dimana terjadi kenaikan RREQ F sebesar 39.52 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 626.8, dimana terjadi kenaikan RREQ F sebesar 29.85 %. Dari hasil tersebut *routing protocol* AODV asli lebih unggul dalam hal RREQ F tersebut karena menghasilkan RREQ F yang lebih rendah dari *routing protocol* AODV modifikasi dengan interval 5 detik, AODV modifikasi dengan interval 10 detik dan AODV modifikasi dengan interval 15 detik.

Pada lingkungan yang sedang dengan jumlah 150 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 11,724.7, dimana terjadi kenaikan RREQ F sebesar 127.73 %, sedangkan AODV modifikasi dengan 10 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 7,249.8, dimana terjadi kenaikan RREQ F sebesar 79.98 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 4,621.7, dimana terjadi kenaikan RREQ F sebesar 50.35 %. Dari hasil tersebut *routing protocol* AODV asli lebih unggul dalam hal RREQ F tersebut karena menghasilkan RREQ F yang lebih rendah dari *routing protocol* AODV modifikasi dengan interval 5 detik, AODV modifikasi dengan interval 10 detik dan AODV modifikasi dengan interval 15 detik.

Pada lingkungan yang padat dengan jumlah 300 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 16,412, dimana terjadi kenaikan RREQ F sebesar 499.73 %,

sedangkan AODV modifikasi dengan 10 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 10,138, dimana terjadi kenaikan RREQ F sebesar 308.69 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 8,539.1, dimana terjadi kenaikan RREQ F sebesar 260%. Dari hasil tersebut *routing protocol* AODV asli lebih unggul dalam hal RREQ F tersebut karena menghasilkan RREQ F yang lebih rendah dari *routing protocol* AODV modifikasi dengan interval 5 detik, AODV modifikasi dengan interval 10 detik dan AODV modifikasi dengan interval 15 detik.

Berdasarkan hasil simulasi pada ketiga lingkungan tersebut, dapat dilihat bahwa dengan jumlah *node* yang sedikit atau lingkungan jarang menghasilkan RREQ F yang lebih bagus atau lebih sedikit daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. AODV asli masih mengungguli AODV modifikasi dengan interval 5 detik, AODV modifikasi dengan interval 10 detik dan AODV modifikasi dengan interval 15 detik. Dapat dilihat pula bahwa AODV asli menghasilkan RREQ F yang lebih bagus atau dalam hal ini lebih rendah daripada AODV modifikasi dengan jumlah selisih RREQ F yang cukup signifikan.

5.2.2 Hasil Uji Coba Skenario *Real*

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

Pengambilan data uji PDR, E2E, RO, dan RREQ F pada skenario *real* dilakukan sebanyak 10 kali dengan skenario mobilitas random pada peta *real* dengan luas area 1300 m x 1300 m dan *node* sebanyak 60 untuk lingkungan yang jarang, 150 *node* untuk lingkungan yang sedang, dan 300 *node* untuk lingkungan yang padat dilakukan pada kecepatan standar yaitu 20 m/s. Untuk uji coba setiap

lingkungan menggunakan interval yang berbeda-beda untuk mencari nilai interval yang terbaik dari hasil skenario. Interval waktu yang digunakan adalah 5 detik, 10 detik dan 15 detik. Hasil analisis dapat dilihat pada Tabel 5.7, Tabel 5.8, Tabel 5.9 dan Tabel 5.10.

Tabel 5.7 Hasil Rata - Rata Perhitungan PDR pada Skenario Real

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**			**Perbedaan**		
		5s	**10s**	**15s**	**5s**	**10s**	**15s**
60	0.803	0.798	0.853	0.821	0.005	0.050	0.018
150	0.837	0.852	0.887	0.892	0.015	0.050	0.055
300	0.730	0.683	0.738	0.752	0.047	0.008	0.022

Tabel 5.8 Hasil Rata -Rata Perhitungan E2E pada Skenario Real

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi (ms)**			**Perbedaan**		
		5s	**10s**	**15s**	**5s**	**10s**	**15s**
60	807.494	676.238	603.025	692.312	131.256	204.469	115.182
150	557.277	1,726.532	1,580.677	1,283.649	1,169.255	1,023.400	726.372
300	340.595	2,040.447	7,378.128	5,819.546	1,699.852	7,037.533	5,478.951

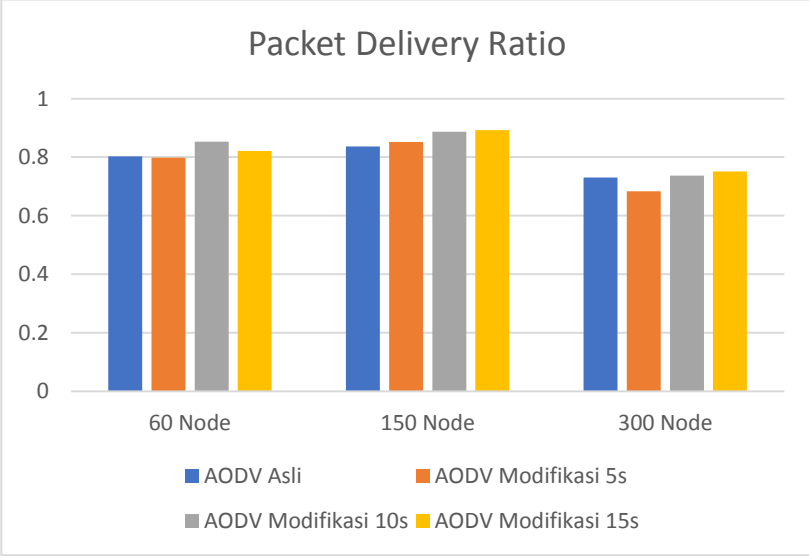
Tabel 5.9 Hasil Rata - Rata Perhitungan RO pada Skenario Real

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi (packet)**			**Perbedaan**		
		5s	**10s**	**15s**	**5s**	**10s**	**15s**
60	14,677.5	7,626.7	5350	4,347.7	7,050.8	9,327.5	10,329.8
150	39,695	20,026.4	11,757.9	8,009.3	19,668.6	27,937.1	31,685.7
300	62,358.5	31,773.6	18,116.2	14,112.9	30,584.9	44,242.3	48,245.6

Tabel 5.10 Hasil Rata - Rata Perhitungan RREQ F pada Skenario Real

Jumlah *Node*	AODV Asli	AODV Modifikasi (packet)			Perbedaan		
		5s	10s	15s	5s	10s	15s
60	2,257.1	3,614.3	3,127.7	2,833.2	1,357.2	870.6	576.1
150	2,160.9	9,849.8	7,169.4	5,124.5	7,688.9	5,008.5	2,963.6
300	2,604.3	16,796.5	11,197.9	9,686.7	14,192.2	8,593.6	7,082.4

5.2.2.1 Analisa *Packet Delivery Ratio* (PDR)



Gambar 5.5 Grafik Rata - Rata PDR Skenario *Real*

Berdasarkan grafik pada Gambar 5.5, *routing protocol* AODV asli dan AODV yang telah dimodifikasi baik dengan interval 5 detik, 10 detik dan 15 detik perubahan yang fluktuaktif. Pada lingkungan yang jarang dengan *node* berjumlah 60 dan dibandingkan

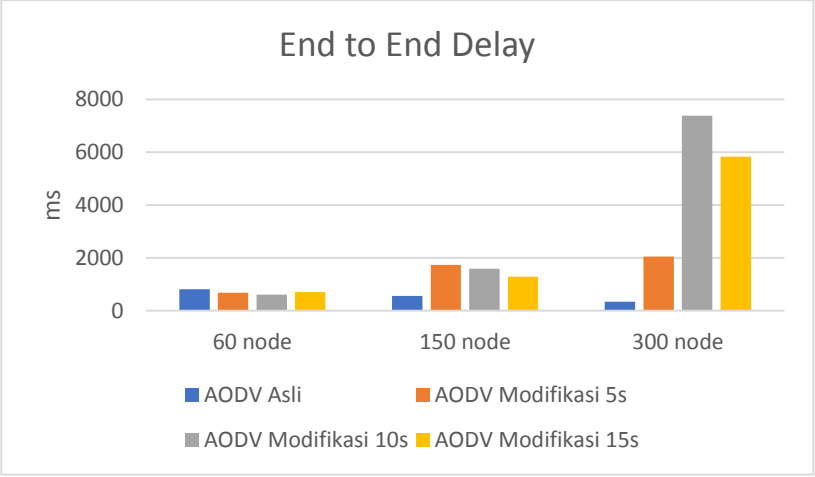
dengan hasil PDR AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih PDR sebesar 0.005, dimana terjadi penurunan sebesar 0.58 %, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih PDR sebesar 0.050, dimana terjadi kenaikan sebesar 6.28 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih PDR sebesar 0.018, dimana terjadi kenaikan sebesar 2.27%. Berdasarkan hal tersebut, routing protocol AODV modifikasi dengan interval 10 detik berhasil mengungguli *routing protocol* AODV asli, *routing protocol* AODV modifikasi dengan interval 5 detik dan *routing protocol* AODV dengan interval 15 detik.

Pada lingkungan yang sedang dengan *node* berjumlah 150 dan dibandingkan dengan hasil PDR AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih PDR sebesar 0.015, dimana terjadi kenaikan sebesar 1.85%, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih PDR sebesar 0.050, dimana terjadi kenaikan sebesar 5.96% dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih PDR sebesar 0.055, dimana terjadi kenaikan sebesar 6.63%. Berdasarkan hal tersebut, routing protocol AODV modifikasi dengan interval 15 detik berhasil mengungguli *routing protocol* AODV asli, *routing protocol* AODV modifikasi dengan interval 5 detik *dan routing protocol* AODV dengan interval 10 detik.

Pada lingkungan yang padat dengan *node* berjumlah 300 dan dibandingkan dengan hasil PDR AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih PDR sebesar 0.047, dimana terjadi penurunan sebesar 6.41%, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih PDR sebesar 0.008, dimana terjadi kenaikan sebesar 1.04% dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih PDR sebesar 0.022, dimana terjadi kenaikan sebesar 2.96%. Berdasarkan hal tersebut, routing protocol AODV modifikasi dengan interval 15 detik berhasil mengungguli *routing protocol* AODV asli, *routing protocol* AODV modifikasi dengan interval 5 detik dan *routing protocol* AODV dengan interval 10 detik.

Pada lingkungan *node* dengan tingkat kepadatan jarang, AODV modifikasi dengan interval 10 detik menghasilkan *Packet Delivery Ratio* (PDR) yang paling tinggi dibanding AODV yang lain yaitu sebesar 0.803. Pada lingkungan *node* dengan tingkat kepadatan sedang, AODV modifikasi dengan interval 15 detik menghasilkan *Packet Delivery Ratio* (PDR) yang paling tinggi dibanding AODV yang lain yaitu sebesar 0.892. Pada lingkungan *node* dengan tingkat kepadatan padat, AODV modifikasi dengan interval 15 detik menghasilkan *Packet Delivery Ratio* (PDR) yang paling tinggi dibanding AODV yang lain yaitu sebesar 0.752. Berdasarkan hasil simulasi pada ketiga lingkungan tersebut, dapat dilihat AODV modifikasi lebih unggul daripada AODV asli.

5.2.2.2 Analisa *End to End Delay* (E2E)



Gambar 5.6 Grafik E2E pada Skenario *Real*

Berdasarkan grafik pada Gambar 5.6, dapat dilihat bahwa rata-rata E2E antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi dengan interval 5 detik, 10 detik dan 15 detik mengalami perubahan yang fluktuatif. Pada lingkungan yang jarang

dengan jumlah 60 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 131.256 ms dimana AODV modifikasi dengan interval 5 detik unggul dalam hal ini, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 204.469 ms dimana AODV modifikasi dengan interval 10 detik unggul dalam hal ini dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 115.182 ms dimana AODV modifikasi dengan interval 15 detik unggul dalam hal ini.

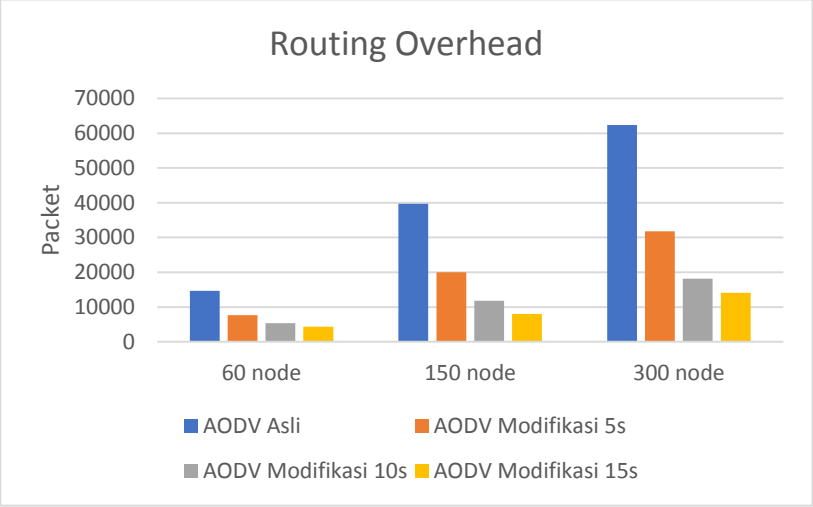
Pada lingkungan yang sedang dengan jumlah 150 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 1,169.255 ms dimana AODV asli unggul dalam hal ini, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 1,023.400 ms dimana AODV asli unggul dalam hal ini dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 726.372 ms dimana AODV asli unggul dalam hal ini.

Pada lingkungan yang padat dengan jumlah 300 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 1,699.852 ms dimana AODV asli unggul dalam hal ini, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 7,037.533 ms dimana AODV asli unggul dalam hal ini dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih *end-to-end delay* sebesar 5,478.951 ms dimana AODV asli unggul dalam hal ini.

Jika ketiga lingkungan tersebut dibandingkan, memang pada lingkungan yang jarang, sedang, maupun padat tidak selalu lebih unggul dalam hal *End to End Delay*. Routing protocol AODV modifikasi dengan interval 10 detik lebih unggul daripada *routing protocol* AODV asli, AODV modifikasi dengan interval 5 detik dan AODV modifikasi dengan interval 15 detik pada lingkungan jarang dengan jumlah 60 node. Namun pada lingkungan yang sedang dan

padat dengan jumlah 150 node dan 300 node AODV asli lebih unggul daripada AODV modifikasi dengan interval 5 detik, AODV modifikasi dengan interval 10 detik, dan AODV modifikasi dengan interval 15 detik. Hasil rata – rata E2E tidak dapat dianalisis karena terjadi fluktuasi dan tidak stabil. Hal ini dikarenakan waktu *delay* tergantung dari rata – rata waktu paket yang terkirim. Semakin banyak paket yang terkirim, maka semakin beragam *delay*nya.

5.2.2.3 Analisa *Routing Overhead* (RO)



Gambar 5.7 Grafik Rata - Rata RO Skenario *Real*

Berdasarkan pada grafik Gambar 5.8, dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi dengan interval 5 detik, 10 detik dan 15 detik dan juga *routing protocol* AODV asli mengalami perubahan yang fluktuatif. Pada lingkungan yang jarang dengan jumlah 60 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih RO sebesar 7,050.8, dimana terjadi penurunan RO sebesar 48.04%, sedangkan pada AODV modifikasi dengan interval 10 detik

menghasilkan perbedaan selisih RO sebesar 9,327.5, dimana terjadi penurunan RO sebesar 63.55% dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih RO sebesar 10,329.8, dimana terjadi penurunan RO sebesar 70.38%. Berdasarkan hasil tersebut, maka *routing protocol* AODV yang telah dimodifikasi dengan interval 15 detik unggul dalam hal RO tersebut karena menghasilkan RO yang lebih rendah dari AODV asli.

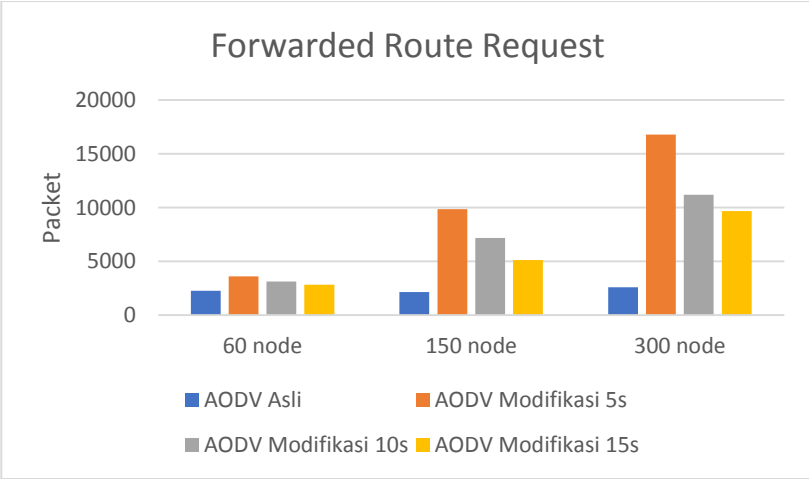
Pada lingkungan yang sedang dengan jumlah 150 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih RO sebesar 19,668.6, dimana terjadi penurunan RO sebesar 49.55 %, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih RO sebesar 27,937.1, dimana terjadi penurunan RO sebesar 70.38 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih RO sebesar 31,685.7, dimana terjadi penurunan RO sebesar 79.82 %. Berdasarkan hasil tersebut, maka *routing protocol* AODV yang telah dimodifikasi dengan interval 15 detik unggul dalam hal RO tersebut karena menghasilkan RO yang lebih rendah dari AODV asli.

Pada lingkungan yang padat dengan jumlah 300 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih RO sebesar 30,584.9, dimana terjadi penurunan RO sebesar 49.05 %, sedangkan pada AODV modifikasi dengan interval 10 detik menghasilkan perbedaan selisih RO sebesar 44,242.3, dimana terjadi penurunan RO sebesar 70.95 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih RO sebesar 48,245.6, dimana terjadi penurunan RO sebesar 77.37 %. Berdasarkan hasil tersebut, maka *routing protocol* AODV yang telah dimodifikasi dengan interval 15 detik unggul dalam hal RO tersebut karena menghasilkan RO yang lebih rendah dari AODV asli.

Jika ketiga lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang lebih sedikit atau pada lingkungan dengan *node* yang jarang, menghasilkan RO yang lebih sedikit daripada di lingkungan dengan jumlah *node* yang sedang maupun

yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. AODV modifikasi dengan interval 15 detik berhasil mengungguli AODV asli, AODV modifikasi dengan interval 5 detik, dan AODV modifikasi dengan interval 10 detik dengan nilai rata – rata penurunan RO pada AODV yang dimodifikasi adalah sebesar 64.34 %. Dapat dilihat pula bahwa AODV yang telah dimodifikasi dengan interval 15 detik menghasilkan RO yang lebih bagus atau dalam hal ini lebih rendah daripada AODV asli.

5.2.2.4 Analisa *Forwarded Route Request* (RREQ F)



Gambar 5.9 Grafik Rata - Rata RREQ F Skenario *Real*

Berdasarkan grafik pada Gambar 5.10, dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi dan juga *routing protocol* asli mengalami perubahan *forwarded route request* (RREQ F) yang signifikan. Pada lingkungan yang jarang dengan jumlah 60 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 1,357.2, dimana terjadi kenaikan RREQ F

sebesar 60.13 %, sedangkan AODV modifikasi dengan 10 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 870.6, dimana terjadi kenaikan RREQ F sebesar 38.57 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 576.1, dimana terjadi kenaikan RREQ F sebesar 25.52 %. Dari hasil tersebut *routing protocol* AODV yang telah dimodifikasi dengan beberapa interval tidak ada yang unggul dalam hal RREQ F tersebut karena menghasilkan RREQ F yang lebih tinggi dari *routing protocol* AODV asli.

Pada lingkungan yang sedang dengan jumlah 150 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 7,688.9, dimana terjadi kenaikan RREQ F sebesar 355.82 %, sedangkan AODV modifikasi dengan 10 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 5,008.5, dimana terjadi kenaikan RREQ F sebesar 231.78 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 2,963.6, dimana terjadi kenaikan RREQ F sebesar 137.15 %. Dari hasil tersebut *routing protocol* AODV yang telah dimodifikasi dengan beberapa interval tidak ada yang unggul dalam hal RREQ F tersebut karena menghasilkan RREQ F yang lebih tinggi dari *routing protocol* AODV asli.

Pada lingkungan yang padat dengan jumlah 300 *node* dan dibandingkan dengan AODV asli, AODV modifikasi dengan interval 5 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 14,192.2, dimana terjadi kenaikan RREQ F sebesar 544.95 %, sedangkan AODV modifikasi dengan 10 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 8,593.6, dimana terjadi kenaikan RREQ F sebesar 329.98 % dan pada AODV modifikasi dengan interval 15 detik menghasilkan perbedaan selisih *forwarded route request* sebesar 7,082.4, dimana terjadi kenaikan RREQ F sebesar 271.95 %. Dari hasil tersebut *routing protocol* AODV yang telah dimodifikasi dengan beberapa interval tidak ada

yang unggul dalam hal RREQ F tersebut karena menghasilkan RREQ F yang lebih tinggi dari *routing protocol* AODV asli.

Berdasarkan hasil simulasi pada ketiga lingkungan tersebut, dapat dilihat bahwa dengan jumlah *node* yang sedikit atau lingkungan jarang menghasilkan RREQ F yang lebih bagus atau lebih sedikit daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. AODV asli masih mengungguli AODV modifikasi dengan interval 5 detik, AODV modifikasi dengan interval 10 detik dan AODV modifikasi dengan interval 15 detik. Dapat dilihat pula bahwa AODV asli menghasilkan RREQ F yang lebih bagus atau dalam hal ini lebih rendah daripada RREQ F asli dengan jumlah selisih RREQ F yang cukup signifikan.

BAB VI
KESIMPULAN DAN SARAN

Pada Bab ini akan diberikan kesimpulan yang diperoleh dari Tugas Akhir yang telah dikerjakan dan saran tentang pengembangan dari Tugas Akhir ini yang dapat dilakukan di masa yang akan datang.

6.1 Kesimpulan

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

1. Perubahan *Hello Interval* mempengaruhi kinerja AODV sebagai berikut :
   - Dampak perubahan *Hello Interval* terhadap performa protokol AODV pada skenario *grid* adalah rata – rata penurunan *Routing Overhead* (RO) sebesar 60.77 %, rata – rata kenaikan *End to End Delay* sebesar 43.98 %, dan rata - rata kenaikan *Forwarded Route Request* (RREQ F) sebesar 162.79 %.
   - Dampak perubahan *Hello Interval* terhadap performa protokol AODV pada skenario *real* adalah rata – rata penurunan *Routing Overhead* (RO) sebesar 64.34 % dan rata - rata kenaikan *Forwarded Route Request* (RREQ F) sebesar 221.76 %.
   - AODV modifikasi dengan hasil *Routing Overhead* (RO) terbaik pada skenario *grid* adalah AODV modifikasi dengan interval 15 detik dengan hasil penurunan *Routing Overhead* (RO) sebesar 71.65 %.
   - AODV modifikasi dengan hasil *Routing Overhead* (RO) terbaik pada skenario *real* adalah AODV modifikasi dengan interval 15 detik dengan hasil penurunan *Routing Overhead* (RO) sebesar 75.86 %.
   - Aspek yang menjadi lebih baik ketika *Hello Interval* diubah adalah *Routing Overhead.*

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.
2. Menambah parameter dan skenario uji coba sehingga membuat hasil semakin variatif untuk dianalisis.
3. Menambahkan interval waktu yang berbeda untuk memberikan hasil yang semakin bervariasi.

DAFTAR PUSTAKA

[1] B. Ayyappan and D. M. Kumar, "Vehicular Ad Hoc Networks (VANET) : Architectures, Methodologies, And Design Issues," *Second International Conference on Science Technology Engineering and Management (ICONSTEM)*, p. 4, 2016.

[2] M. N. Alslaim, H. A. Alaqel and S. S. Zaghloul, "A Comparative Study of MANET Routing Protocols," *IEEE*, p. 5, 2014.

[3] Y. Wu, X. Peng, Z. Xu and X. Lin, "AODV-MR: AODV with multi RREP for VANET," *IEEE*, 2014.

[4] F. Nutrihadi, R. Anggoro and R. M. Ijtihadie, "Studi Kinerja VANET Scenario Generators: SUMO dan VanetMobisim untuk Implementasi Routing Protocol AODV menggunakan Network Simulator (NS-2)," vol. 5, p. 6, 2016.

[5] S. Biradar and P. Kulkarni, "An Improved Quality of Service Using R-AODV Protocol in MANETs," *Third International Conference 3rd on Artificial Intelligence Modelling and Simulation*, p. 6, 2015.

[6] M. Ali, M. Welzl, A. Adnan and F. Nadeem, "Using the NS-2 Network Simulator for Evaluating Network on Chips (NoC)," *2nd International Conference on Emerging Technologies*, p. 7, 2006.

[7] "OpenStreetMap," [Online]. Available: https://www.openstreetmap.org/. [Accessed 17 Juni 2018].

[8] "JOSM," [Online]. Available: https://josm.openstreetmap.de/. [Accessed 17 Juni 2018].

(Halaman ini sengaja dikosongkan)

LAMPIRAN

A.1 Kode Fungsi sendHello()

```
void
HelloTimer::handle(Event*) {

  #ifdef DEBUG
  FILE *fp = fopen("trace.log","a+");
  fprintf(fp, "HelloTimer::handle\n");
  fclose(fp);
  #endif

   agent->sendHello();
   double interval = 5;
   assert(interval >= 0);
   Scheduler::instance().schedule(this,
&intr, interval);
}
```

```
void
HelloTimer::handle(Event*) {

#ifdef DEBUG
FILE *fp = fopen("trace.log","a+");
fprintf(fp, "HelloTimer::handle\n");
fclose(fp);
#endif

agent->sendHello();
double interval = 10;
assert(interval >= 0);
Scheduler::instance().schedule(this,
&intr, interval);
}
```

```
void
HelloTimer::handle(Event*) {

  #ifdef DEBUG
  FILE *fp = fopen("trace.log","a+");
  fprintf(fp, "HelloTimer::handle\n");
  fclose(fp);
  #endif

   agent->sendHello();
   double interval = 15;
   assert(interval >= 0);
   Scheduler::instance().schedule(this,
&intr, interval);
}
```

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)      "cbr1.txt";
set val(sc)      "scen1modif.txt";

set ns_          [new Simulator]

# setup topography object

set topo   [new Topography]

# create trace object for ns and nam

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

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

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

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

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

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

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

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

# Define node initial position in nam

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

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

    $ns_ initial_node_pos $node_($i) 20
}

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

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

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

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

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

BIODATA PENULIS

Magista Bella Puspita, lahir di Magelang, 19 September 1996. Penulis adalah anak pertama dari dua bersaudara. Penulis menempuh pendidikan sekolah dasar di SD Negeri Secang 1 lalu melanjutkan pendidikan sekolah menengah pertama di SMP Negeri 2 Magelang dan penulis menempuh pendidikan menengah atas di SMA Negeri 3 Magelang. Selanjutnya penulis melanjutkan pendidikan sarjana di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Arsitektur dan Jaringan Komputer (AJK). Sebagai mahasiswa, penulis berperan aktif dalam beberapa organisasi kampus seperti staf Hubungan Luar Himpunan Mahasiswa Teknik-Computer (HMTC) ITS dan Sekretaris Departemen Kewirausahaan Himpunan Mahasiswa Teknik-Computer (HMTC) ITS. Selain itu, penulis juga menjadi staf Web dan Kesekretariatan SCHEMATICS 2015 dan staf National Seminar of Technology SCHEMATICS 2016. Penulis pernah melakukan kerja praktik di Dinas Perpustakaan dan Kearsipan Jawa Timur periode Juli – Agustus 2017 dan membuat aplikasi berbasis web Buku Tamu Digital. Penulis dapat dihubungi melalui nomor *handphone*: 085641889903 atau *email*: bellapuspita.mbp@gmail.com