TUGAS AKHIR - KI141502

IMPLEMENTASI PEMILIHAN *FORWARDING NODE* PADA PROSES *ROUTE DISCOVERY AD-HOC ON DEMAND DISTANCE VECTOR*(AODV) BERDASARKAN *CONGESTION LEVEL* PADA VANETs

HUDA FAUZAN MURTADHO
NRP 05111540000022

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

Dosen Pembimbing II
Prof. Ir. Supeno Djanali, M.Sc, Ph.D.

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

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR – KI141502

IMPLEMENTASI PEMILIHAN *FORWARDING NODE* PADA PROSES *ROUTE DISCOVERY AD-HOC ON DEMAND DISTANCE VECTOR*(AODV) BERDASARKAN *CONGESTION LEVEL* PADA VANETs

HUDA FAUZAN MURTADHO
NRP 051115400000022

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

Dosen Pembimbing II
Prof. Ir. Supeno Djanali, M.Sc, Ph.D.

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

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – KI141502

IMPLEMENTATION OF FORWARDING *NODE* SELECTION FOR *ROUTE* DISCOVERY PROCESS IN AD-HOC ON DEMAND DISTANCE VECTOR (AODV) BASED ON *CONGESTION* LEVEL IN VANETS

HUDA FAUZAN MURTADHO
NRP 05111540000022

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

Second Advisor
Prof. Ir. Supeno Djanali, M.Sc, Ph.D.

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

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

IMPLEMENTASI PEMILIHAN *FORWARDING NODE* PADA PROSES *ROUTE DISCOVERY AD-HOC ON DEMAND DISTANCE VECTOR*(AODV) BERDASARKAN *CONGESTION LEVEL* PADA VANETs

TUGAS AKHIR

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

Oleh:
HUDA FAUZAN MURTADHO
NRP: 05111540000022

Disetujui oleh Pembimbing Tugas Akhir:

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

2. Prof. Ir. Supeno Djanali, M.Sc., Ph.D. .....
(NIP. 194806191973011001) (Pembimbing 2)

SURABAYA
JUNI, 2019

(Halaman ini sengaja dikosongkan)

IMPLEMENTASI PEMILIHAN *FORWARDING NODE* PADA PROSES *ROUTE DISCOVERY AD-HOC ON DEMAND DISTANCE VECTOR*(AODV) BERDASARKAN *CONGESTION LEVEL* PADA VANETs

Nama Mahasiswa	**: Huda Fauzan Murtadho**
NRP	**: 05111540000022**
Departemen	**: Informatika FTIK-ITS**
Dosen Pembimbing 1	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**
Dosen Pembimbing 2	**: Prof. Ir. Supeno Djanali, M.Sc, Ph.D.**

Abstrak

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

AODV merupakan salah satu *routing protocol* yang termasuk dalam klasifikasi *reactive routing protocol*, sebuah protokol yang hanya akan membuat rute ketika *node* sumber membutuhkannya. AODV memiliki dua fase, yaitu *route discovery* dan *route maintenance. Route discovery* digunakan untuk meminta dan meneruskan informasi rute yang terdiri dari proses pengiriman *Route Request* (RREQ) dan *Route Reply* (RREP), sedangkan *route maintenance* digunakan untuk mengetahui informasi adanya kesalahan pada rute. Pada fase ini terdapat proses pengiriman *Route Error* (RERR).Dalam lingkungan VANETs yang dinamis dan menggunakan *routing protocol* AODV, pencarian rute merupakan suatu mekanisme yang penting. Pemilihan rute yang tepat saat proses pencarian rute sangat diperlukan untuk memperpanjang waktu penggunaan rute. Salah satu cara yang dapat dilakukan adalah dengan memilih rute untuk menghindari terjadinya penurunan

kinerja karena arus traffic yang padat mengakibatkan delay packet yang tinggi sehingga packet yang dikirim akan berkurang.

Pada tugas akhir ini memodifikasi skema kontrol pada saat *Route* Discovery AODV berdasarkan *Congestion Level* pada VANETs.Untuk mengelola kondisi routing ini bisa dilakukan dengan memeriksa bagaimana padatnya *node* pada saat ini.Setelah paket RREP dibuat dan dikirim melalui *node* destinasi ,penghitungan *Congestion Counter* akan bertambah dalam sekali pengirim RREP kembali ke *node* sumber.Dan ketika *Congestion Counter* sudah mencapai *threshold* maka RREQ tidak akan melewati *node* yang sudah memiliki *Congestion Counter* yang padat.Lalu RREQ akan mencari jalur yang lain untuk mencapai *node* destinasi. Dari hasil uji coba, AODV yang dimodifikasi pada skenario adalah rata – rata kenaikan *Packet Delivery Ratio* (PDR) sebesar 0,7%,rata – rata penurunan *End to End Delay* (E2E Delay) sebesar 10,183%,rata-rata kenaikan *Routing Overhead* (RO) sebesar 3,6% dan rata – rata penurunan *Forwarded Route Request* (RREQ F) pada *node* 50 , 150 dan 200 sebesar 3,67%.Pada skenario *real* adalah rata – rata kenaikan *Packet Delivery Ratio* (PDR) pada *node* 100,150 dan 200 sebesar 0,72%, rata – rata penurunan *End to End Delay* (E2E Delay) sebesar 10,613%,rata-rata kenaikan *Routing Overhead* (RO) pada *node* 50,100,150 sebesar 2,2% dan penurunan pada *node* 200 1,41% dan rata – rata penurunan *Forwarded Route Request* (RREQ F) sebesar 5,89%.

Kata kunci: VANETs, AODV, Congestion Level

IMPLEMENTATION OF FORWARDING *NODE* SELECTION FOR *ROUTE* DISCOVERY PROCESS IN AD-HOC ON DEMAND DISTANCE VECTOR (AODV) BASED ON *CONGESTION* LEVEL IN VANETS

Student's Name	**: Huda Fauzan Murtadho**
Student's ID	**: 05111540000022**
Department	**: Informatics – FTIK ITS**
First Advisor	**: Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.**
Second Advisor	**: Prof. Ir. Supeno Djanali, M.Sc, Ph.D.**

Abstract

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

AODV is one of the routing protocols included in the reactive routing protocol classification, a protocol that will only create routes when the source node needs it. AODV has two phases, namely route discovery and route maintenance. Route discovery is used to request and forward route information which consists of the process of sending Route Request (RREQ) and Route Reply (RREP), while maintenance routes are used to find out information about errors in the route. In this phase there is a Route Error (RERR) sending process. In a VANETs environment that is dynamic and uses the AODV routing protocol, route search is an important mechanism. The selection of the right route when the route search process is very necessary to extend the route usage time. One way that can be done is to choose a route to avoid a decrease in performance because of the heavy traffic flow resulting in high packet delay so that the packet sent will decrease.

In this final assignment modify the control scheme when Route Discovery AODV is based on Congestion Level on VANETs. To manage routing conditions. This can be done by checking how dense the node is at this time, and after the RREP packet is created and sent through nodes, the calculation of Congestion counter will increase once the RREP sender returns to the source node. And when the Congestion counter has reached the threshold, the RREQ will not pass through the node that already has a Congestion counter that is solid. Then RREQ will look for another path to reach the destination node. The evaluation shows that, the modified AODV in the grid scenario succeeded in the average increase in Packet Delivery Ratio (PDR) at nodes 100,150 and 200 by 1.78%, the average decrease in End to End Delay (E2E Delay) was 10.183% and average - the average decrease in RREQ F at nodes 50, 150 and 200 is 3.67% while in the real scenario the average increase in Packet Delivery Ratio (PDR) on nodes 100,150 and 200 is 0.72%, average decreases in End to End Delay (E2E Delay) is 10.613% and the average decrease in RREQ F is 5.89%

Keyword: VANETs, AODV, Congestion Level

KATA PENGANTAR

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan Tugas Akhir yang berjudul **"Implementasi Pemilihan *Forwarding Node* Pada Proses *Route Discovery Ad-Hoc On Demand Distance Vector*(AODV) Berdasarkan *Congestion Level* Pada Vanets".**

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 Bambang Agus Sukaca dan Ibu Titik Purwanti selaku kedua orangtua penulis atas segala dukungan berupa motivasi serta doa sehingga penulis dapat mengerjakan Tugas Akhir ini.
3. Bachtiar Rivai dan Annisa Nur Rahmawati selaku kakak dan adik penulis atas segala dukungan yang telah diberikan sehingga penulis tetap semangat dalam mengerjakan Tugas Akhir ini.
4. Bapak Dr. Eng. Radityo Anggoro, S.Kom., M.Sc., dan Bapak Prof. Ir. Supeno Djanali, M.Sc, Ph.D. selaku dosen pembimbing, atas arahan dan bantuannya dalam pengerjaan Tugas Akhir ini.
5. Sahabat Kami Kembali ,Sahabat Roemah Perdjoeangan, dan teman – teman angkatan 2015 yang selama ini sudah membantu penulis dan memotivasi dalam menyelesaikan Tugas Akhir ini
6. Teman-teman HMTC Dagri Kreasi yang selama ini sudah memberikan pengalaman dan ilmu-ilmu kebersamaan serta

kekeluargaan sehingga membuat penulis bersemangat menjalani perkuliahan hingga pengerjaan Tugas Akhir.

7. TC 15, TC 16 dan TC 17 yang sudah membuat penulis cukup pusing, sedih dan senang akan tingkah lakunya.
8. Teman – teman KMKS 2015 dan ASTOR yang selama ini sudah menyemangati Penulis dalam menjalankan kuliah di ITS.
9. Sahabat AJK dan Admin serta Penghuni MI yang sudah membantu dan menyediakan fasilitas selama penulis mengerjakan Tugas Akhir ini.
10. 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, Juni 2019

Huda Fauzan Murtadho

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Saat ini perkembangan teknologi informasi dan komunikasi menjadi salah satu indikator kemajuan peradaban manusia. Salah satu teknologi yang membantu manusia dalam berkomunikasi dengan mudah adalah *Vehicular Ad hoc Networks* (VANETs). VANETs merupakan suatu mekanisme yang dapat menghubungkan kendaraan satu dengan yang lainnya menggunakan jaringan nirkabel. VANETs dapat berguna pada banyak hal, seperti mengemudi secara otomatis, navigasi, pencegahan kecelakaan yang dapat meningkatkan keamanan berkendara, serta dapat mengurangi kemacetan lalu lintas [1].

Routing protocol dalam VANET dibedakan menjadi dua model, yaitu *proactive* dan *reactive routing*. *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. Contoh *proactive routing protocol* adalah *Destination-Sequenced Distance Vector* (DSDV), dan *Optimized Link State Routing protocol* (OLSR) sedangkan contoh *reactive routing protocol* adalah *Adhoc On-Demand Distance Vector* (AODV), dan *Dynamic Source Routing* (DSR).

Salah satu contoh pengimplementasian protokol AODV adalah diimplementasikannya protokol tersebut kedalam jaringan sensor nirkabel. *Adhoc On-Demand Distance Vector* (AODV) merupakan salah satu *Routing protocol* reaktif yang paling terkenal saat ini. Dalam lingkungan VANETs yang dinamis dan menggunakan *routing protocol* AODV, pencarian rute merupakan suatu mekanisme yang penting. Pemilihan rute yang tepat saat proses pencarian rute sangat diperlukan untuk memperpanjang

waktu penggunaan rute. Salah satu cara yang dapat dilakukan adalah dengan memilih node yang tepat(*forwarding node*) untuk menghindari terjadinya penurunan kinerja karena arus *Congestion* yang padat mengakibatkan delay packet yang tinggi serta terjadinya *Congestion* pada jalur pengiriman sehingga packet yang dikirim akan berkurang.

Pada Tugas Akhir ini mengimplementasikan control skema yang bernama "*CC-AODV: An effective multiple paths Congestion control AODV* " untuk mengelola kondisi routing.Dengan entri table ini,tingkat pengiriman paket meningkat secara signifikan sementara drop packet menurun,namun implementasinya akan menyebabkan overhead pada paket.Serta untuk menurunkan penurunan kinerja yang disebabkan oleh *Congestion* pada paket ketika data dikirimkan menggunakan AODV.Selanjutnya CC-AODV menentukan sebuah jalur untuk data dengan menggunakan label *Congestion Counter* . Ini bisa dilakukan dengan memeriksa bagaimana padatnya *node* pada saat ini,dan setelah paket RREP dibuat dan dikirim melalui *node* ,penghitungan *Congestion Counter* akan bertambah dalam sekali pengirim RREP kembali ke source *node*.Dan ketika *Congestion Counter* sudah mencapai *threshold* maka RREQ tidak akan melewati *node* yang sudah memiliki *Congestion Counter* yang padat.Lalu RREQ akan mencari jalur yang lain untuk mencapai *node* destinasi.[4].

1.2 Rumusan Masalah

Tugas Akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana melakukan pemilihan rute yang efektif dalam proses *Route Discovery* AODV berdasarkan *Congestion Level* pada VANETs ?
2. Bagaimana dampak penambahan faktor *Congestion Level* terhadap performa *protocol* AODV secara keseluruhan diukur berdasarkan *Packet Delivery Ratio* (PDR),*Rouitng Overhead*(RO),*End to End Delay*(E2E) dan *Forwarded Route Request* (RREQ F)?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki batasan sebagai berikut:

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

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Membatasi *Rebroadcast Route Request* (RREQ) pada AODV berdasarkan *Congestion Level* pada VANETs.
2. Menganalisa performa AODV yangss dimodifikasi berdasarkan matriks *Packet Delivery Ratio* (PDR),*Rouitng Overhead*(RO),*End to End Delay*(E2E) dan *Forwarded Route Request* (RREQ F) .

1.5 Manfaat

Manfaat yang diperoleh dari pengerjaan Tugas Akhir ini adalah memberikan sebuah manfaat dalam mengetahui dampak performa AODV terhadap modifikasi yang dilakukan.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahapan awal dari Tugas Akhir ini adalah penyusunan Proposal Tugas Akhir. Proposal Tugas Akhir berisi pendahuluan, deskripsi dan gagasan metode-metode yang dibuat dalam Tugas Akhir ini. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya Tugas Akhir, rumusan masalah yang diangkat, batasan masalah untuk Tugas Akhir, dan manfaat dari hasil pembuatan Tugas Akhir ini. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan Tugas Akhir.

1.6.2 Studi Literatur

Pada tahap ini, dipelajari sejumlah referensi yang diperlukan dalam melakukan implementasi yaitu mengenai VANETs, AODV, NS2, OpenStreetMap, Java OpenStreetMap (JOSM), SUMO, dan AWK.

1.6.3 Analisis dan Desain Sistem

Pada tahap ini dilakukan analisis dari hasil percobaan modifikasi AODV yang dibuat. Data yang dianalis berasal dari perhitungan *Packet Delivery Ratio (*PDR), *Routing Overhead* (RO), *Forwarded Route Request* (RREQ F), dan *End-to-End Delay* paket dari *node* ke *node* lainnya. Hal ini bertujuan untuk merumuskan solusi yang tepat untuk konfigurasi AODV yang dimodifikasi dalam lingkungan topologi MANET. Setelah selesai diaplikasikan pada MANET, dilakukan simulasi yang dilakukan pada VANETs dengan bantuan SUMO.

1.6.4 Implementasi Sistem

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

1.6.5 Pengujian dan Evaluasi

Pada tahap ini dilakukan pengujian menggunakan SUMO, sebuah *traffic generator* untuk membuat simulasi keadaan topologi yang diujikan. Hasil dari SUMO tersebut akan dijalankan pada NS-2 yang akan menghasilkan *trace file*. *Packet Delivery Ratio* (PDR), *Routing Overhead* (RO), *Forwarded Route Request*, dan *End-to-End Delay* akan dihitung dari *trace file* tersebut untuk menguji performa AODV yang telah dimodifikasi.

1.6.6 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku yang menjelaskan seluruh konsep, teori dasar dari metode yang digunakan, implementasi, serta hasil yang telah dikerjakan sebagai dokumentasi dari pelaksanaan Tugas Akhir.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

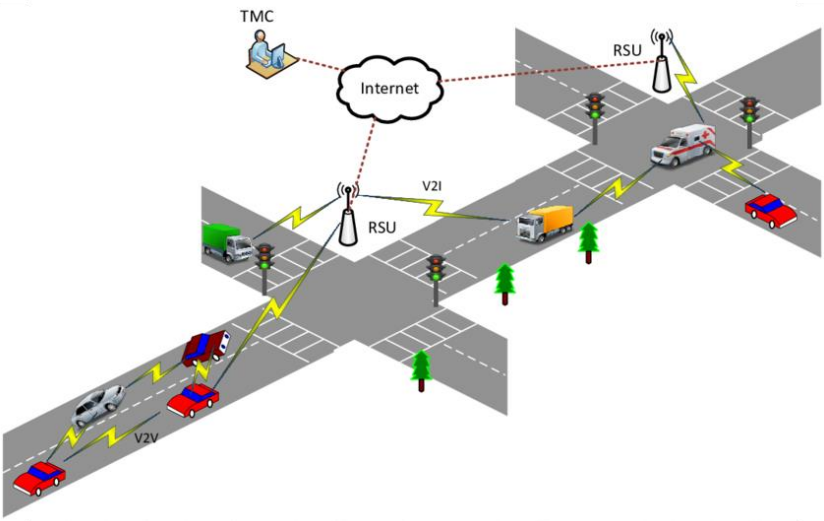
2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori atau penjelasan dari metode, algoritma, *library*, dan *tools* yang digunakan dalam penyusunan Tugas Akhir ini. Bab ini berisi tentang penjelasan singkat mengenai VANETs, AODV, NS2, OpenStreetMap, Java OpenStreetMap (JOSM), SUMO, dan AWK.
3. Bab III. Perancangan
   Bab ini berisi pembahasan mengenai perancangan skenario mobilitas *grid* dan *real*, perancangan simulasi pada NS2, perancangan modifikasi AODV, serta perancangan metrik analisis (*Packet Delivery Ratio, Routing Overhead, Forwarded Route Request, End-to-End Delay dan Forwarded Route Request*).
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 VANETs

Vehicular Ad Hoc Network (VANET) merupakan pengembangan dari *Mobile Ad Hoc Network* (MANET) yang memungkinkan terjadinya komunikasi antar kendaraan sebagai pengembangan *Intelligent Transport System* (ITS) dengan tujuan meningkatkan keselamatan dan kenyamanan berkendara. Komunikasi Wireless ini meliputi komunikasi *Inter-Vehicle Communication* (IVC), *Vehicle to Roadside* (V2R), atau *Roadside to Roadside* (R2R). Setiap *node* pada VANET berlaku baik sebagai partisipan ataupun *route*r pada jaringan, baik bagi *node* utama atau intermediate *node* yang berkomunikasi di dalam radius transmisinya. VANET merupakan jaringan yang self-organized, artinya jaringan ini tidak bergantung pada infrastruktur jaringan manapun. Ada beberapa *node* yang secara tetap berdiri sebagai Roadside Unit, yakni yang dapat memfasilitasi jaringan kendaraan dengan informasi data geografis ataupun akses internet [1]. Ilustrasi VANETs dapat dilihat pada Gambar 2.1.



Gambar 2.1 Ilustrasi VANETs *[12]*

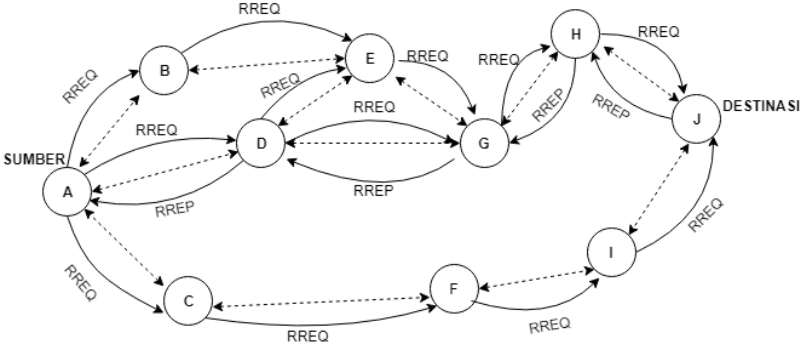
Dalam Tugas Akhir ini, penulis akan mengimplementasikan *routing protocol* AODV yang dimodifikasi dan menguji performa protokol tersebut pada lingkungan VANETs.

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

Ad hoc On Demand Distance Vector (AODV) algoritma routing adalah protokol routing yang dirancang untuk jaringan ad hoc mobile. AODV mampu baik unicast dan multicast routing. Ini adalah algoritma permintaan, artinya membangun rute antara *node* yang dibutuhkan oleh sumber *node*. Ini mempertahankan rute ini selama mereka dibutuhkan oleh sumber. Selain itu, bentuk-bentuk AODV yang menghubungkan anggota grup *multicast*. Percabangan yang terdiri dari anggota kelompok dan *node* yang diperlukan untuk menghubungkan ke *node* lainnya. AODV menggunakan nomor urut untuk mengupdate rute. Itu adalah *loop-free* dan skala jumlah besar mobile *node*.

AODV menggunakan tabel routing dengan satu entry untuk setiap tujuan. Tanpa menggunakan routing sumber, AODV mempercayakan pada tabel routing untuk menyebarkan *RouteReply* (RREP) kembali ke sumber dan secara sekuensial akan mengarahkan paket data menuju ketujuan. AODV juga menggunakan sequence number untuk menjaga setiap tujuan agar didapat informasi routing yang terbaru dan untuk menghindari routing loops. Semua paket yang diarahkan membawa sequence number ini.

Penemuan jalur (*Path discovery*)atau *Route discovery* di-inisiasi dengan menyebarkan *RouteReply* (RREP). Ketika RREP menjelajahi *node*, ia akan secara otomatis men-setup path. Jika sebuah *node* menerima RREP, maka *node* tersebut akan mengirimkan RREP lagi ke *node* atau destination sequence number. Pada proses ini, *node* pertama kali akan mengecek destination sequence. [2]. Ilustrasi pencarian rute oleh AODV dapat dilihat pada Gambar 2.2.



Gambar 2.2 Ilustrasi pencarian rute *routing protocol* AODV *[13]*

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

- *Destination Address:* berisi alamat dari *node* tujuan.
- *Destination Sequence Number*: *sequence number* dari jalur komunikasi.
- *Next Hop*: alamat *node* yang akan meneruskan paket data.

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

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

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

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

Network Simulator 2 atau biasa disingkat NS-2 merupakan sebuah network simulator yang dibuat dengan tujuan riset dan pendidikan. Awalnya, NS dibangun sebagai varian dari *REAL Network Simulator* pada tahun 1989 di UCB (*University of California Berkeley*).

NS-2 merupakan salah satu perangkat lunak yang dapat menampilkan secara simulasi proses komunikasi dan bagaimana proses komunikasi tersebut berlangsung. NS-2 melayani simulasi untuk komunikasi dengan kabel dan nirkabel. NS-2 memiliki beberapa fitur kelebihan yang dapat dimanfaatkan dalam pemodelan dan pengujian VANET. NS-2 memiliki tools validasi yang berfungsi untuk menguji validitas pemodelan yang ada pada NS-2. NS-2 bersifat open source dibawah GPL (*GNU Public Licence)* [3].

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

2.3.1 Instalasi

NS-2 membutuhkan beberapa *package* yang harus sudah ter*install* sebelum memulai instalasi NS-2. Untuk meng*install dependency* yang dibutuhkan dapat dilakukan dengan *command* yang ditunjukkan pada Gambar 2.3

```
sudo apt-get install build-essential 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 2.4.

```
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.

2.3.2 *Trace File*

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

Tabel 2.1 Detail Penjelasan *Trace File* AODV

Kolom ke-	Penjelasan	Isi
1	*Event*	s : *sent* r : *received* f : *forwarded* D : *dropped*
2	*Time*	Waktu terjadinya *event*
3	ID *Node*	_x_ : dari 0 hingga banyak *node* pada topologi
4	*Layer*	AGT : *application* RTR : *routing* LL : *link layer* IFQ : *packet queue* MAC : *MAC* PHY : *physical*
5	*Flag*	--- : Tidak ada
6	*Sequence Number*	Nomor paket
7	*Packet Type*	AODV : paket *routing* AODV cbr : berkas paket CBR (*Constant Bit Rate*) RTS : *Request To Send* yang dihasilkan MAC 802.11 CTS : *Clear To Send* yang dihasilkan MAC 802.11 ACK : MAC ACK

		ARP : Paket *link layer address resolution protocol*
8	Ukuran	Ukuran paket pada *layer* saat itu
9	Detail MAC	[a b c d] a : perkiraan waktu paket b : alamat penerima c : alamat penerima d : IP header
10	*Flag*	-------- : Tidak ada
11	*Detail* IP *source, destination,* dan *nexthop*	[a:b c:d e f] a : IP *source node* b : *port source node* c : IP *destination node* (jika -1 berarti *broadcast*) d : *port destination node* e : IP *header* ttl f : IP *nexthop* (jika 0 berarti *node* 0 atau *broadcast*)

2.4 OpenStreetMap

OpenStreetMap (OSM) adalah sebuah proyek berbasis web untuk membuat peta dunia yang gratis dan terbuka, dibangun sepenuhnya oleh sukarelawan dengan melakukan survei menggunakan GPS, mendigitalisasi citra satelit dan 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 [4].

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

2.5 *Java OpenStreetMap Editor* (JOSM)

Java OpenStreetMap Editor atau biasa disingkat JOSM merupakan sebuah aplikasi desktop berbasis Java dan dapat dioperasikan pada sistem operasi seperti Windows, Mac OS, dan Linux. JOSM adalah alat penyunting bagi data OpenStreetMap. JOSM pertama kali dikembangkan oleh *Immanuel Scholz* pada tahun 2005.Aplikasi ini tidak membutuhkan koneksi internet kala menyunting data OSM, sedangkan situs untuk mengunduhnya dapat diakses di josm.openstreetmap.de untuk mendapatkan versi terbaru dari aplikasi ini [3] .

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

2.6 *Simulation of Urban Mobility* (SUMO)

Simulation of Urban Mobility (SUMO) merupakan paket simulasi lalu lintas yang bersifat *open-source* dimana pengembangannya dimulai pada tahun 2001. Dan semenjak itu

SUMO telah berubah menjadi sebuah simulasi lalu lintas dengan kelengkapan fitur dan pemodelannya termasuk kemampuan jalannya jaringan untuk membaca *format* yang berbeda.

SUMO juga memungkinkan untuk mendefinisikan kendaraan dengan sifat tertentu seperti panjang kendaraan, kecepatan maksimum, percepatan dan perlambatannya. SUMO juga menyediakan pilihan bagi pengguna menentukan rute acak untuk kendaraan. Ada juga pilihan yang tersedia untuk model sistem transportasi umum, dimana setiap kendaraan datang dan berangkat sesuai dengan jadwal [5].

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

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

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

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

2.7 AWK

AWK adalah bahasa pemrograman yang digunakan untuk melakukan *text processing* dan ekstraksi data [6]. AWK merupakan sebuah program filter untuk teks, seperti halnya perintah grep pada terminal linux. AWK dapat digunakan untuk mencari bentuk / model dalam sebuah berkas teks ke dalam bentuk teks lain. AWK dapat juga digunakan untuk melakukan proses aritmatika seperti yang dilakukan oleh perintah expr. AWK sama halnya seperti bahasa shell atau C yang memiliki karakteristik yaitu sebagai *tool* yang cocok untuk *jobs* juga sebagai pelengkap untuk *filter* standar.

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

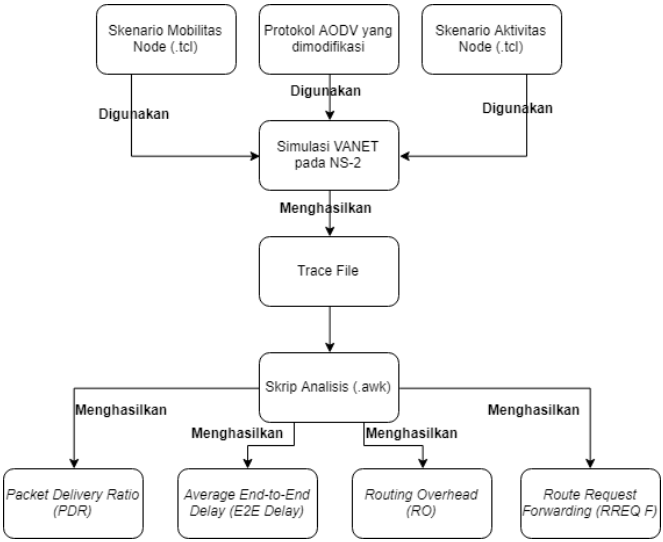
(Halaman ini sengaja dikosongkan)

BAB III
PERANCANGAN

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

3.1 Deskripsi Umum

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



Gambar 3.1 Rancangan Simulasi

Modifikasi akan diawali dengan pencarian jumlah tetangga setiap *node* perantara (*one-hop node)*. Jumlah tetangga tiap *node* akan didapatkan dengan menggunakan HELLO *messages* yang terdapat pada AODV. Setelah jumlah tetangga setiap *node* didapatkan, maka modifikasi dilanjutkan untuk melakukan perhitungan nilai *threshold* yang dilakukan setiap *i* interval waktu dan penyeleksian *forwarding node* yang bertugas untuk *rebroadcast* paket *Route Request* (RREQ) yang akan diteruskan. Hal tersebut dilakukan dengan cara menyeleksi *node* – *node* mana saja yang mempunyai jumlah tetangga lebih dari nilai *threshold* yang sudah ditentukan. Jika *node* tersebut mempunyai jumlah tetangga lebih dari *threshold*, maka *node* tersebut akan meneruskan paket RREQ, namun jika sebaliknya, maka *node* tersebut tidak akan meneruskan paket RREQ atau paket akan di-*drop*.

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

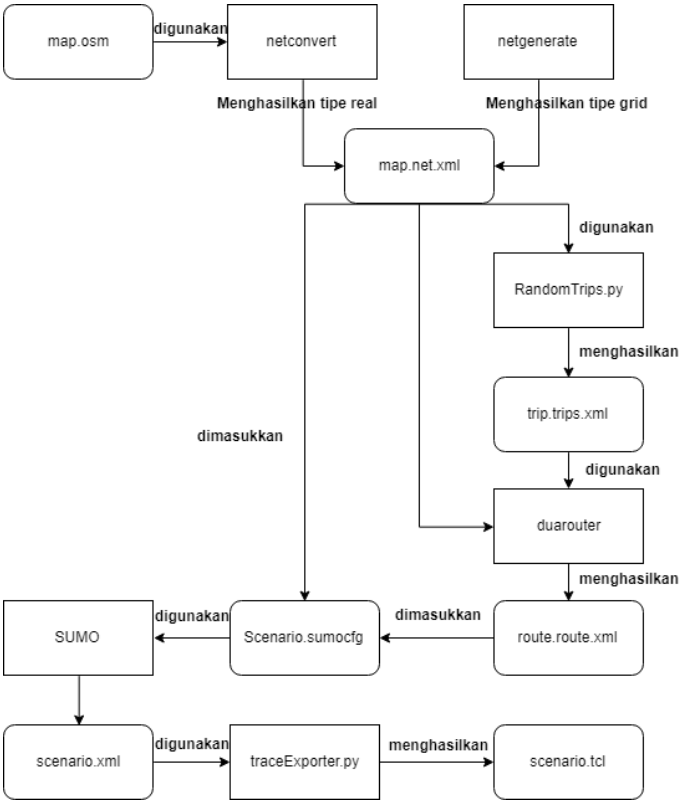
Tabel 3.1 Daftar Istilah

No.	Istilah	Penjelasan
1	AODV	Singkatan dari *Ad hoc On-demand Distance Vector*. Protokol yang digunakan pada Tugas Akhir ini.
2	PDR	*Packet Delivery Ratio*. Salah satu metrik analisis yang diukur. Berupa

No.	Istilah	Penjelasan
		rasio jumlah pengiriman paket yang terkirim.
3	E2E	*Average End-to-End Delay*. Jeda waktu yang diukur saat paket terkirim.
4	RO	*Routing Overhead*. Jumlah *control packet* yang terkirim
5	RREQ	*Route Request*. Paket *request* pada AODV yang dikirim untuk mendapatkan rute
6	RREP	*Route Reply*. Paket *reply* pada AODV yang dikirim ke *node* sumber melalui rute yang sudah terbuat.
7	*Threshold*	Batas nilai yang dijadikan sebagai acuan

3.2 Perancangan Skenario Mobilitas

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



Gambar 3.2 Rancangan Pembuatan Skenario Mobilitas

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 900 m x 900 m. Dengan 9 titip persimpangan, maka akan didapatkan 16 petak dan panjang tiap peta adalah 200m.

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 dua*route*r.

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

3.2.2 Perancangan Skenario *Real*

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

Setelah mendapatkan peta area yang akan dijadikan simulasi, peta tersebut dikonversi ke dalam bentuk *file* dengan ekstensi .net.xml menggunakan *tools* SUMO yaitu netconvert. Tahap berikutnya memiliki tahapan yang sama seperti merancang skenario *grid*, yaitu membuat pergerakan *node* menggunakan randomTrips dan dua*route*r. 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

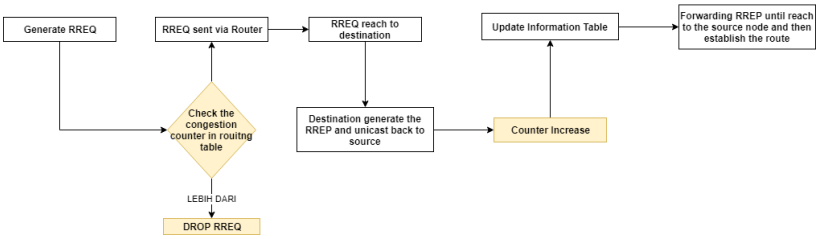
dihasilkan tersebut dikonversi ke dalam bentuk *file* dengan ekstensi .tcl agar dapat diterapkan pada NS-2.

3.3 Perancangan Modifikasi *Routing protocol* AODV

Protokol AODV yang diajukan pada Tugas Akhir ini merupakan modifikasi dari protokol AODV yang mengubah mekanisme *route discovery* pada protokol tersebut. Pada protokol AODV,mekanisme pada pengiriman paket yang sudah mendapatkan rute dari source melewati *node* intermediate hingga sampai destinasi akan dilakukan secara terus menerus hingga seluruh paket terkirim atau adanya pemutusan rute yang dipakai dan bisa jadi membebani *node* yang dilalui,maka pada AODV modifikasi ini akan ada perubahan rute lain jika *node* tersebut sudah dianggap macet dan kemacetan tersebut ditentukan dengan *Congestion counter* yang akan ditambahkan disetiap *node*nya.Perubahan rute ini akan dilakukan ketika pengiriman paket RREQ diawal yang mem*broadcast* ke semua *node* untuk mencapai *node* destinasi,lalu disetiap melewati *node intermediate* akan dilakukan pengecekan disetiap *node*nya apakah sudah mencapai *threshold* yang ditentukan untuk menghindari *Congestion*. Bila *node* yang dilalui sudah mencapai *threshold* maka paket RREQ akan didrop dan akan mencari rute lain. Dan jika *node* yang dilalui masih belum mencapai *threshold* maka paket RREQ akan diteruskan hingga mencapai *node* destinasi. Setelah mendapatkan rute yang dipilih maka *node* destinasi akan mengirimkan balik RREP kepada *node* sumber dan pada proses ini pada pengiriman paket RREP untuk setiap *node* yang dilalui akan bertambah *counter*-nya yang nanti kita akan sebut sebagai *Congestion counter* . Karena beberapa paket RREQ di-drop, maka kemacetan dan tabrakan paket pada jaringan akan lebih rendah sehingga bisa menaikkan PDR.Struktur *routing table* yang ditambahkan parameter baru seperti berikut Table 3.2.Dan untuk flow chart dari modifikasi yang saya jelaskan diatas terdapat pada Gambar 3.3

Table 3.2 Struktur Routing Table

Dest_address	Sequence_No	Hop_Count	Next_Hop	Flags	*Congestion* Counter



Gambar 3.3 Flowchart modifikasi pada AODV menggunakan CC-AODV

3.3.1 Perancangan Penghitungan Jumlah *Congestion Counter* untuk Setiap *Node* yang akan dilalui

Pada Tugas Akhir ini perhitungan jumlah *Congestion Counter* dilakukan dengan menggunakan packet RREP yang ada pada *protocol* AODV.RREP akan dikirimkan ketika packet RREQ yang diterima oleh *node* destinasi.Pada hal ini setiap packet RREQ yang dikirim untuk menuju *node* destinasi akan melewati *node* intermediate.Dengan *protocol* AODV akan melewati semua *node* intermediate dan akan mencari rute dengan packet RREQ yang sampai terlebih dahulu di *node* destinasi.Kemudian packet RREP akan dikirimkan kembali menuju *node* asal melalui rute yang sudah dipakai untuk mengirim packet RREQ pada awal tadi.Modifikasi akan dilakukan dengan menambahkan parameter baru *Congestion Counter* di dalam header Routing Table dengan nama `Congestion_counter`.Pada saat pengiriman packet RREP jika packet RREP melalui *node* intermediate maka

*Congestion*_counter akan bertambah yang diubah pada fungsi recvReply yang terdapat pada file aodv.cc didalam folder ns-2.35/aodv. *Pseudocode* untuk perhitungan *Congestion counter* dapat dilihat pada Gambar 3.3.

```
if reply == index
      drop RREP packet
else
  rt_reverse*route* = rt_lookup(destination)
  if(reverse*route* && rt_hops != UNREACHABLE)
     if(*Congestion*counter > threshold)
      drop RREP Packet
      return
     else
      forward
      *Congestion*counter++
  else
    drop RREP packet
```

Gambar 3.4 Pseudocode Perhitungan Jumlah *Congestion_Counter*

3.3.2 Perancangan Penentuan *Threshold*

Threshold akan ditentukan melalui perhitungan berapa pasang *node* yang dibuat untuk setipa proses pengiriman packet dalam hal ini ada 4 pasang *node*.*Threshold* yang dimaksud adalah batas *Congestion* yang dibuat karena dalam hal ini untuk menentukan batas *Congestion* pada *protocol* AODV masih belum bisa diketahui maka di ambil dengan setiap *node* diberi batasan *Congestion* dengan maksimal *node* intermediate dilalui oleh jalur pengiriman packet yaitu 4 dan nilai yang akan menjadi *threshold* akan selalu mengikuti dari jumlah pasangan *node* yang dibuat pada scenario nantinya.Lalu *threshold* akan dicek terlebih dahulu dalam setiap pengiriman packet RREQ. Pada modifikasi ini threshold akan diberi nama variable *Congestion*_threshold yang akan di define pada file aodv.cc dalam folder ns-2.35/aodv pada line 9 terdapat pada lampiran.

3.3.3 Perancangan Pemilihan *Forwarding Node*

Setelah melakukan penghitungan *Congestion counter* yang ada pada setiap *node*, akan dilakukan pemilihan *forwarding node*, yaitu *node* yang berhak untuk melakukan *rebroadcast* paket RREQ. Penentuan *forwarding node* dilakukan dengan cara membandingkan jumlah *Congestion counter* dengan *threshold* atau batas nilai yang sudah ditentukan. Apabila jumlah tetangga yang dimiliki *node* tersebut melebihi *threshold*, maka *node* akan di-*drop* bila tidak maka packet RREQ akan melakukan *rebroadcast*. Langkah tersebut dilakukan untuk mengurangi jumlah paket RREQ yang dikirim sehingga dapat mengurangi kemacetan yang terjadi pada jaringan.Dengan memodifikasi pada *SendRequest* dan *RecvRequest*. *Pseudocode* untuk pemilihan *forwarding node* dapat dilihat pada Gambar 3.4.

```
if Congestioncounter > threshold
      drop RREQ packet
end if
```

Gambar 3.5 Pesudocode Pemilihan *Forwarding Node*

3.4 Perancangan Simulasi pada NS-2

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

Kode yang diubah diantaranya adalah pencarian jumlah *node* tetangga pada *file node*.cc dan perbandingan dengan *threshold* pada *file* aodv.cc. Pada saat simulasi NS-2 dijalankan, maka *routing protocol* AODV akan menyeleksi *node* mana saja yang berhak melakukan *rebroadcast* paket RREQ.

3.5 Perancangan Metrik Analisis

Berikut ini merupakan parameter – parameter yang akan dianalisis pada Tugas Akhir ini untuk dapat membandingkan performa dari *routing protocol* AODV yang asli dengan AODV yang telah dimodifikasi:

3.5.1 *Packet Delivery Ratio* (PDR)

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

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

Keterangan:

PDR = *Packet Delivery Ratio*

received = banyak paket data yang diterima

sent = banyak paket data yang dikirimkan

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

Average End-to-End Delay dihitung berdasarkan rata-rata *delay* antara waktu paket data diterima dan waktu paket dikirimkan dalam satuan detik. Delay tiap paket didapat dari rentang waktu antara *node* asal saat mengirimkan paket dan *node* tujuan menerima paket. Delay tiap paket tersebut semua dijumlahkan dan dibagi dengan

jumlah paket yang berhasil diterima, maka akan didapatkan rata – rata E2E, yang dapat dihitung dengan persamaan 3.2.

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

Keterangan:

E2E = *End-to-End Delay*

CBRRecvTime = Waktu *node* asal mengirimkan paket

CBRSentTime = Waktu *node* tujuan menerima paket

recvnum = Jumlah paket yang berhasil diterima

3.5.3 *Routing Overhead* (RO)

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

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

Keterangan:

sentnum = Jumlah paket yang berhasil dikirim

3.5.4 *Forwarded Route Request* (RREQ F)

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

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

Keterangan:

rreqsent = Paket *route request* yang dikirim

BAB IV
IMPLEMENTASI

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

4.1 Implementasi Skenario Mobilitas

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

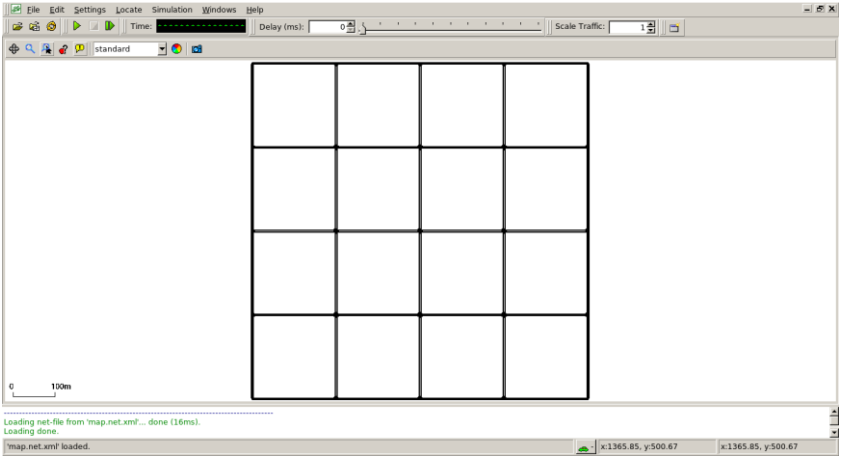
4.1.1 Skenario *Grid*

Dalam mengimplementasikan skenario *grid*, SUMO menyediakan *tools* untuk membuat peta *grid* yaitu netgenerate. Pada Tugas Akhir ini, penulis membuat peta *grid* dengan luas 900 m x 900 m yang terdiri dari titik persimpangan antara jalan vertikal dan jalan horisontal sebanyak 5 titik x 5 titik. Dengan jumlah titik persimpangan sebanyak 5 titik tersebut, maka terbentuk 16 buah petak. Sehingga untuk mencapai luas area sebesar 900 m x 900 m dibutuhkan luas per petak sebesar 200 m x 200 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=5 --
grid.length=200 --default.speed=20 --
tls.guess=1 --output-file=map.net.xml
```

Gambar 4.1 Perintah netgenerate

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



Gambar 4.2 Hasil *Generate* Peta *Grid*

Setelah peta terbentuk, maka dilakukan pembuatan *node* dan pergerakan *node* dengan menentukan titik awal dan titik akhir setiap *node* 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 42 -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* dua*route*r. Perintah penggunaan *tools* dua*route*r 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 dua*route*r

Ketika menggunakan *tools* dua*route*r, 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

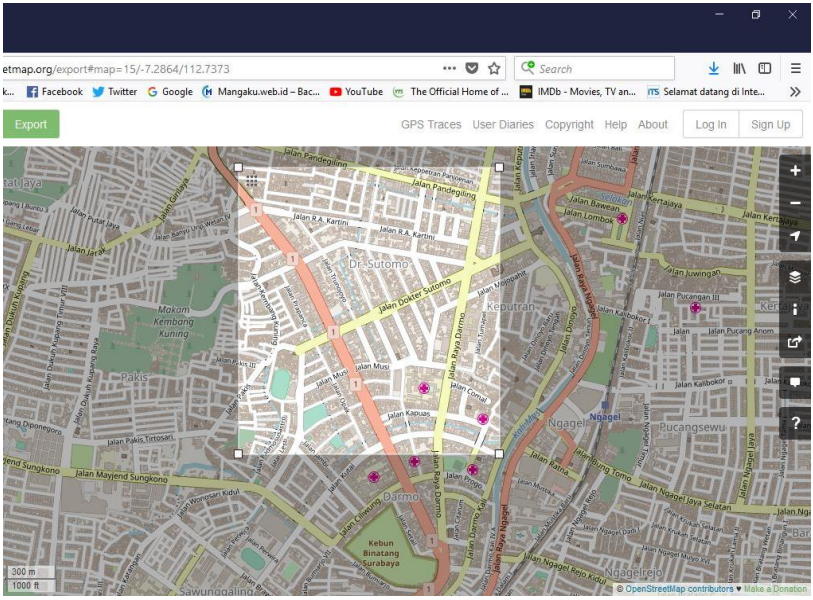
ini adalah traceExporter. Perintah untuk menggunakan traceExporter dapat dilihat pada Gambar 4.7.

```
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 penulis mengambil area jalan sekitar Jl. Dr. Soetomo Surabaya. Setelah menentukan area simulasi, ekspor data peta dari OpenStreetMap seperti yang ditunjukkan pada Gambar 4.8.



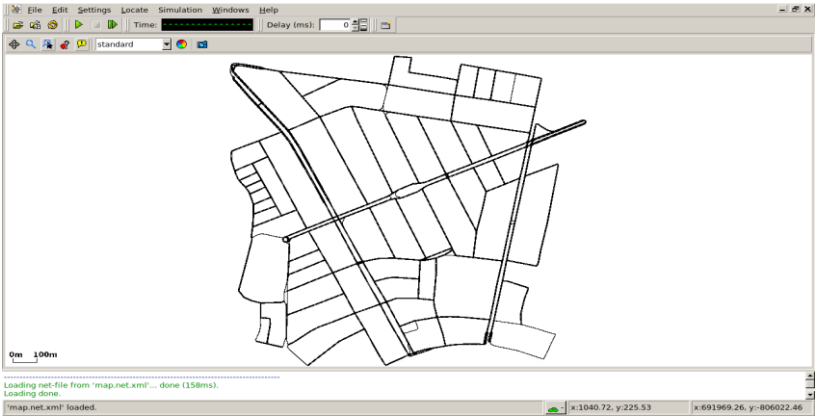
Gambar 4.8 Ekspor Peta dari OpenStreetMap

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* dua*router*. Kemudian membuat *file* skenario berekstensi .xml menggunakan *tool* SUMO dengan bantuan *file* skrip berekstensi .sumocfg. Selanjutnya dilakukan konversi *file* skenario berekstensi .tcl untuk dapat disimulasikan pada

NS-2 menggunakan *tool* traceExporter. Perintah untuk menggunakan *tools* tersebut sama dengan ketika membuat skenario *grid* di atas.

4.2 Implementasi Modifikasi pada *Routing protocol* AODV untuk Menentukan *Forwarding Node*

Pada Tugas Akhir ini dilakukan modifikasi pada *routing protocol* AODV agar dapat mengurangi jumlah *forwarding node*, yaitu *node* yang bertugas untuk melakukan *rebroadcast* paket RREQ. Hal tersebut dilakukan dengan cara memilih *forwarding node* berdasarkan *Congestion counter* tersebut dengan *threshold* yang dihitung berdasarkan jumlah pasangan *node* yang ditentukan pada scenario yang membuat *threshold* bersifat statis, sehingga dapat dilihat peningkatan performa pada *routing* AODV yang telah dimodifikasi.

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

- Implementasi Penghitungan Jumlah *Congestion Counter* untuk Setiap *Node* yang akan dilalui
- Implementasi Penentuan *Threshold*
- Implementasi Pemilihan *Forwarding Node*

Kode implementasi dari *routing protocol* AODV pada NS-2 versi 2.35 berada pada direktori ns-2.35/aodv. Pada direktori tersebut terdapat beberapa file diantaranya seperti aodv.cc, aodv.h dan sebagainya. Pada Tugas Akhir ini, penulis memodifikasi *file* aodv.cc yang terdapat dalam *folder* ns-2.35/aodv untuk menghitung jumlah *Congestion counter*, menentukan *threshold* dan menentukan forwarding *node* dan *file* aodv.h yang ada di dalam *folder* ns-2.35/aodv untuk mendaftarkan header baru. Pada bagian ini penulis akan menjelaskan langkah – langkah dalam mengimplementasikan modifikasi *routing protocol* AODV untuk mengurangi jumlah *forwarding node* yang melakukan *rebroadcast*.

4.2.1 Implementasi Penghitungan Jumlah *Congestion Counter* untuk Setiap *Node* yang akan dilalui

Langkah awal yang dilakukan untuk menghitung jumlah tetangga seperti yang telah dirancang pada subbab 3.3.1 adalah dengan cara menerima packet RREQ yang dikirim oleh *node* source untuk menuju *node* destinasi.

Node yang mengirimkan packet RREQ sudah dapat dipastikan berada di sekitar *node* tersebut dan berada di *range* transmisi dari *node* yang sedang dieksekusi. Secara *default*, AODV Didalam proses *route* discovery, *node* asal membroadcast *route* request (RREQ) `AODV::sendRequest`.RREQ akan di broadcast kesemua tetangga terdekat dari *node* source.Ketika *node* tujuan atau *node* yang memiliki *route* ke tujuan menerima packet RREQ. *node* tujuan membalas paket RREQ dengan *route* reply (RREP) packet. RREP dibuat dan dikirim kembail ke *node* asal hanya. Ketika menerima RREP dengan fungsi `AODV::recvReply`, setiap *node* diantara asal dan tujuan mengupdate routing table next-hop dengan RREP ke tujuannya. *Node* asal kemudian memilih *route* dengan jumlah hop paling sedikit untuk mengirimkan packet tujuannya.. Pada tugas akhir ini, penulis memanfaatkan fungsi paket tersebut dengan tambahan modifikasi untuk menambahkan counter ketika menerima packet RREP dan dengan menambahkan parameter pada header Routing Table yaitu *Congestion*_counter.Fungsi tersebut terdapat pada kode sumber aodv.cc dan aodv_rtable.h yang terdapat dalam folder ns2.3.5/aodv. Untuk potongan kode tersebut bisa dilihat pada Gambar 4.11.

```
class aodv_rtable {
 public:
      aodv_rtable() {
     Congestion_counter = 0;
    }
        int   Congestion_counter;
}
```

```
if(ih->daddr() == index || suppress_reply)
Packet::free(p);
  else {
    aodv_rt_entry *rt0 =
rtable.rt_lookup(ih->daddr());

 if(rt0 && (rt0->rt_hops != INFINITY2)) {
      assert (rt0->rt_flags == RTF_UP);
      rp->rp_hop_count += 1;
      rp->rp_src = index;
      if (rtable.Congestion_counter >
CONGESTION_THRESHOLD){
        drop(p, DROP_RTR_NO_ROUTE);
        return;
      }
      else{
        forward(rt0, p, NO_DELAY);
        rtable.Congestion_counter++;
      rt->pc_insert(rt0->rt_nexthop); //
nexthop to RREQ source
      }
    }
    else {
      drop(p, DROP_RTR_NO_ROUTE);
    }
  }
```

Gambar 4.11 Potongan modifikasi header table dan fungsi recvReply

4.2.2 Implementasi Perhitungan Threshold

Pada tahap selanjutnya setelah dapat mengetahui jumlah tetangga, langkah yang harus dilakukan selanjutnya adalah menentukan *threshold* .

Penentuan ini dilakukan pada dengan mendefine variable yang terletak pada kode sumber aodv.cc yang terletak pada ns2.35/aodv. Variabel ini diisikan dengan berapa jumlah kemacetan yang menjadi batas kemacetan. Variabel ini digunakan pada beberapa modifikasi untuk menentukan *Congestion Counter* yang nantinya akan men-*drop* packet RREQ. Potongan kode untuk menentukan berapa batas *Congestion Threshold* ada pada lampiran file AODV.CC pada line nomor 9.

4.2.3 Implementasi Pemilihan *Forwarding Node*

Pada tahap selanjutnya setelah dapat mengetahui jumlah *Congestion Counter* dan *Threshold* , langkah yang harus dilakukan selanjutnya adalah pemilihan *forwarding node* untuk melanjutkan paket RREQ.

Penghitungan ini dilakukan pada fungsi `recvRequest()` yang terletak pada kode sumber aodv.cc yang terletak pada ns2.35/aodv. Apabila jumlah *Congestion Counter* dari *threshold* yang ditentukan , maka paket RREQ akan di*drop*. Potongan kode untuk proses seleksi *forwarding node* dapat dilihat pada file AODV.CC fungsi method *sendRequest* pada lampiran dengan line dari kode nomor 18-27

Dengan proses penyeleksian *node* mana saja yang dapat melanjutkan paket RREQ sudah dapat mengurangi jumlah paket *routing overhead* untuk paket RREQ.

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)  50
set val(nn)      50
set val(seed)    1.0
set val(adhocRouting) AODV
set val(stop)    200
set val(cp)      "cbr50node.tcl"
set val(sc)      "scenario.tcl"
```

Gambar 4.12 Implementasi Simulasi NS-2

Pada konfigurasi dilakukan pemanggilan terhadap *file traffic* yang berisikan konfigurasi *node* asal, *node* tujuan, pengiriman paket, serta *file* skenario yang berisi pergerakan *node* yang telah di*generate* oleh SUMO. Kode implementasi pada NS-2 dapat dilihat pada lampiran A.4 Kode Skenario NS-2.

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

```
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(42) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(43) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 1
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 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.

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 Routing Overhead.

4.4.1 Implementasi *Packet Delivery Ratio* (PDR)

Pada subbab 2.3.2 telah ditunjukkan contoh struktur data *event* yang dicatat dalam *trace file* oleh NS-2. Kemudian, pada persamaan 3.1 telah dijelaskan bagaimana menghitung PDR. Skrip awk untuk menghitung PDR berdasarkan kedua informasi tersebut dapat dilihat pada lampiran A.3 Kode Skrip AWK *Packet Delivery Ratio*.

PDR didapatkan dengan cara menghitung setiap baris terjadinya *event* pengiriman dan penerimaan paket data yang dikirim melalui agen pada *trace file*. Skrip menyaring setiap baris yang mengandung *string* AGT karena kata kunci tersebut 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.4 Kode Skrip AWK Rata-Rata *End-to-End Delay*.

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

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

```
sum_delay = 0
counter = 0

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

Gambar 4.15 Pseudocode untuk Perhitungan 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.5 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.6 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 untuk 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
UJICOBA DAN EVALUASI

Pada bab ini akan dilakukan tahap ujicoba dan evaluasi sesuai dengan rancangan dan implementasi. Dari hasil yang didapatkan setelah melakukan uji coba, akan dilakukan evaluasi sehingga dapat ditarik kesimpulan pada bab selanjutnya.

5.1 Lingkungan Uji Coba

Uji coba dilakukan pada perangkat dengan spesifikasi seperti yang tertera pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat yang Digunakan

Komponen	**Spesifikasi**
CPU	Intel(R) Core(TM) i5-7400 CPU @ 3.00Ghz
Sistem Operasi	Ubuntu 18.04 LTS
Linux Kernel	Linux kernel 4.4
Memori	8.0 GB

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

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

Parameter lingkungan uji coba yang digunakan pada NS-2 dapat 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.3 Kode Skrip AWK *Packet Delivery Ratio*, A.4 Kode Skrip AWK Rata-Rata

End-to-End Delay, A.5 Kode Skrip AWK *Routing Overhead*, dan A.9 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 & CC-AODV
3	Waktu simulasi	200 detik
4	Area simulasi	900 m x 900 m *GRID* 1500 m x 1500 m *REAL*
5	Jumlah *Node*	50,100, 150, 200
6	Pasangan *Node*	4
7	Radius transmisi	250m
8	Kecepatan maksimum	20 m/s
9	Protokol MAC	IEEE 802.11p
10	Model Propagasi	*Two-ray ground*

5.2 Hasil Uji Coba

5.2.1 Hasil Uji Coba Skenario *Grid*

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

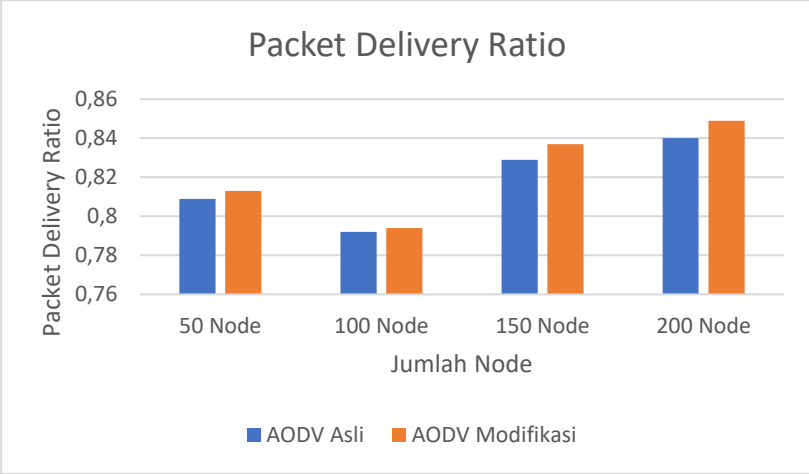
Pengambilan data uji PDR, RO, dan E2E pada skenario *grid* dilakukan sebanyak 10 kali dengan skenario mobilitas *random* pada peta *grid* dengan luas area 900 m x 900 m dan *node* sebanyak 50 untuk lingkungan yang jarang,100 dan 150 *node* untuk lingkungan yang sedang, dan 200 *node* untuk lingkungan yang padat dilakukan pada kecepatan standar yaitu 20 m/s. Untuk uji coba setiap lingkungan menggunakan *threshold* yang berbeda-beda karena satu *threshold* tidak bisa disamakan untuk semua lingkungan disebabkan jumlah tetangga yang dimiliki oleh setiap *node* pada setiap lingkungan

berbeda-beda, tergantung pada jumlah *node* pada lingkungan simulasi tersebut.

Hasil pengambilan data *packet delivery ratio* pada skenario *grid* 50, 100,150, dan 200 *node* dengan menggunakan AODV asli dan AODV yang telah dimodifikasi yang ditunjukkan pada Gambar 5.13, Gambar 5.24, Gambar 5.35 dan grafik yang merepresentasikan hasil perhitungan PDR, RO, dan E2E yang ditunjukkan pada Gambar 5.1, Gambar 5.2, Gambar 5.3, dan Gambar 5.4

Tabel 5.3 Hasil Rata - Rata PDR Skenario *Grid*

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	0,809	0,813	+0,004
100	0,792	0,794	+0.002
150	0,829	0,837	+0.008
200	0.84	0.849	+0.009



Gambar 5.1 Grafik PDR Skenario *Grid*

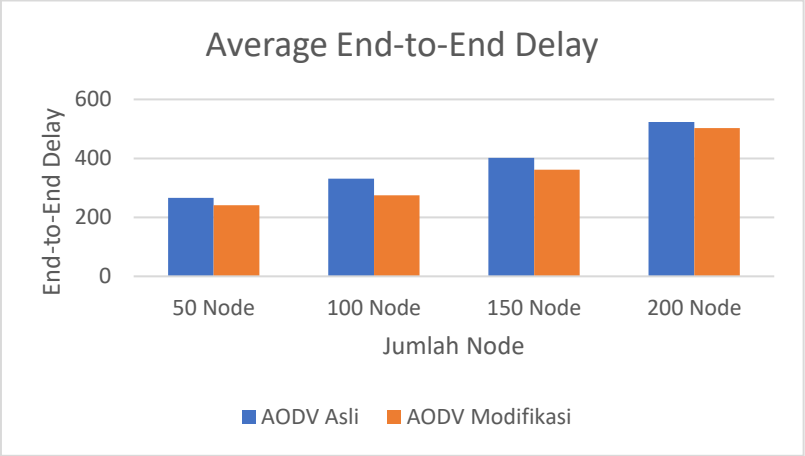
Berdasarkan grafik pada Gambar 5.1 dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi dan juga *routing protocol* AODV asli mengalami penurunan yang signifikan pada packet delivery ratio. Pada lingkungan yang jarang dengan jumlah 50 *node*, menghasilkan perbedaan selisih PDR sebesar 0,004, atau naik menjadi sekitar 0,5% dimana *routing protocol* AODV yang telah dimodifikasi kenaikan dalam hal PDR tersebut. Pada lingkungan yang sedang dengan jumlah 100 *node*, menghasilkan perbedaan selisih PDR sebesar 0,002, atau naik menjadi sekitar 0,25% dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal PDR tersebut dari AODV asli. Pada lingkungan yang sedang dengan jumlah 150 *node*, menghasilkan perbedaan selisih PDR sebesar 0.008, atau naik menjadi sekitar 0,97% dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal PDR tersebut dari AODV asli.Pada lingkungan yang padat dengan jumlah 200 *node*, menghasilkan perbedaan selisih PDR sebesar 0,009, atau naik menjadi sekitar 1,07% dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal PDR tersebut.

Grafik PDR yang dimana pada *node* jarang yaitu 50 *node*, *node* sedang 100,150 dan *node* padat pada *node* 200 mengalami kenaikan dengan menggunakan modifikasi. Jika keempat lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang lebih sedikit atau pada lingkungan dengan *node* yang jarang, menghasilkan PDR yang kurang daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan PDR yang lebih bagus daripada AODV asli dengan jumlah selisih PDR yang cukup signifikan.

Hasil pengambilan data rata-rata untuk *end-to-end delay* (E2E) pada skenario *grid* dengan jumlah *node* 50,100, 150, dan 200 dapat dilihat pada Gambar 5.2.

Tabel 5.4 Hasil Rata - Rata E2E Skenario *Grid*

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	266.651ms	240.923ms	-25.728ms
100	331.464ms	275.014ms	-56.45ms
150	402.322ms	361.807ms	-40.515ms
200	523.897ms	503.004ms	-20.894ms



Gambar 5.2 Grafik E2E Skenario *Grid*

Berdasarkan grafik pada Gambar 5.2 dapat dilihat bahwa rata-rata *end-to-end delay* antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami penurunan yang signifikan. Pada lingkungan yang jarang dengan jumlah 50 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 25.728 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 9.65%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal

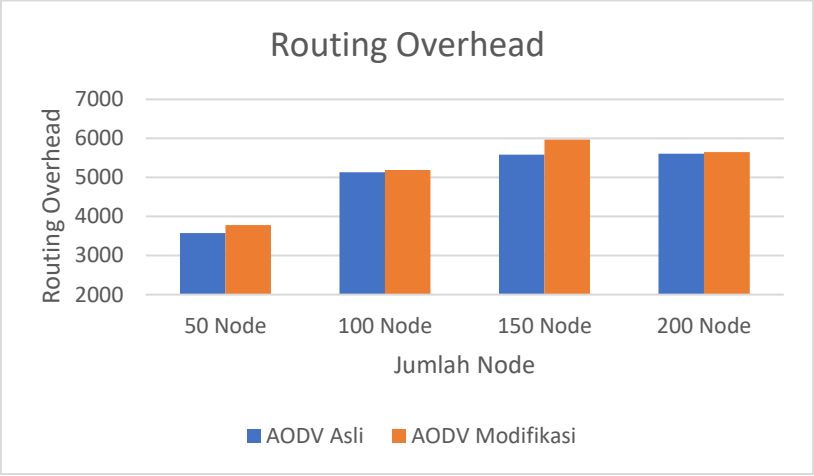
end-to-end delay tersebut. Sedangkan pada lingkungan sedang dengan jumlah 100 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 56.45 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami kenaikan sebesar 17.03%, dimana *routing protocol* AODV yang asli lebih unggul dalam hal *end-to-end delay* tersebut.Pada pada lingkungan sedang dengan jumlah 150 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 40.515 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami kenaikan sebesar 10.07%, dimana *routing protocol* AODV yang asli lebih unggul dalam hal *end-to-end delay* tersebut.Pada lingkungan yang padat dengan jumlah 200 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 20.894 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 3.98%, dimana *routing protocol* AODV yang telah dimodifikasi jauh lebih unggul dalam hal *end-to-end delay* tersebut.

Jika keempat lingkungan tersebut dibandingkan, memang pada lingkungan yang jarang, sedang, maupun padat *routing protocol* AODV yang telah dimodifikasi selalu lebih unggul dalam hal *end-to-end delay*. Dapat dilihat bahwa dengan jumlah *node* jarang,sedang dan padat menghasilkan *end-to-end delay* yang lebih baik dan dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan *end-to end delay* yang lebih baik daripada AODV asli dengan jumlah selisih *end-to-end delay* yang cukup signifikan.Hal ini bisa terjadi karena dengan AODV modifikasi kali ini menghindari adanya *Congestion* sehingga packet-packet akan lebih sedikit mengalami adanya delay dalam pengiriman.

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

Tabel 5.5 Hasil Rata - Rata RO Skenario *Grid*

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	3580	3780	+200
100	5133	5195	+62
150	5582	5971	+389
200	5607	5653	+45



Gambar 5.3 Grafik Routing Overhead Skenario *Grid*

Berdasarkan grafik pada Gambar 5.3 dapat dilihat bahwa rata-rata *routing overhead* antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami perubahan yang agak kurang bagus pada routing overhead. Pada lingkungan yang jarang dengan jumlah 50 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 200 atau mengalami penurunan sebesar 5.59%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *routing overhead* tersebut karena menghasilkan *routing*

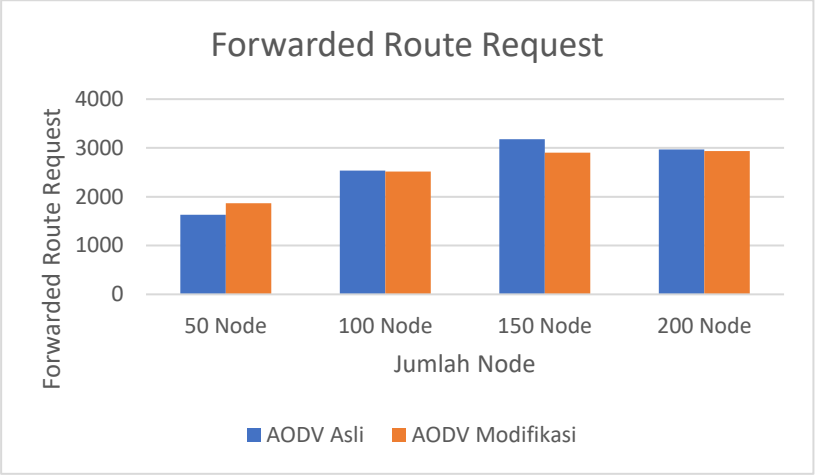
overhead yang lebih rendah dari routing AODV asli. Pada lingkungan yang sedang dengan jumlah 100 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 62 atau mengalami kenaikan sebesar 1.21%, dimana *routing protocol* AODV yang telah dimodifikasi mengalami kekalahan dalam hal routing overhead tersebut dari AODV asli. Pada lingkungan yang sedang dengan jumlah 150 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 389 atau mengalami kenaikan sebesar 6.97%, dimana *routing protocol* AODV yang telah dimodifikasi mengalami kekalahan dalam hal routing overhead tersebut dari AODV asli. Pada lingkungan yang padat dengan jumlah 200 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 45 atau mengalami kenaikan sebesar 0.82%, dimana *routing protocol* AODV yang telah dimodifikasi mengalami kekalahan dalam hal *routing overhead* tersebut.

Jika keempat lingkungan tersebut dibandingkan dapat dilihat bahwa pada *node* jarang nilai routing overhead mengalami hasil yang kurang dari *protocol* asli AODV,sedangkan pada *node* yang sedang yaitu pada *node* 100 dan 150 *node* mengalami hasil yang kurang dari *protocol* AODV asli dan pada *node* padat yaitu *node* 200 hasil routing overhead dengan AODV modifikasi mengalami hasil kurang dari daripada *protocol* AODV asli. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan routing overhead yang kurang dari atau dalam hal ini lebih tinggi daripada AODV asli dengan jumlah selisih routing overhead yang sangat tipis ini mungkin bisa terjadi karena adanya pemutusan link atau rute sehingga perlu adanya pencarian rute yang lebih sering ketimbang AODV asli sehingga nilai routing overhead modifikasi bisa lebih dari nilai asli.

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

Tabel 5.6 Hasil Rata - Rata RREQ F Skenario *Grid*

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	1633	1870	+237
100	2539	2514	-25
150	3181	2901	-280
200	2968	2938	-30



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 fluktuatif. Pada lingkungan yang jarang dengan jumlah 50 *node*, menghasilkan perbedaan selisih *forwarded route* request sebesar 237 atau mengalami kenaikan sebesar 14.51%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal

forwarded route request tersebut karena menghasilkan *forwarded route* request yang lebih rendah dari *routing protocol* AODV asli. Pada lingkungan yang sedang dengan jumlah 100 *node*, menghasilkan perbedaan selisih *forwarded route* request sebesar 25 atau mengalami penurunan sebesar 0.98%, dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal *forwarded route* request tersebut dari *forwarded route* request AODV asli.Pada lingkungan yang sedang dengan jumlah 150 *node*, menghasilkan perbedaan selisih *forwarded route* request sebesar 280 atau mengalami penurunan sebesar 8.8%, dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal *forwarded route* request tersebut dari *forwarded route* request AODV asli. Pada lingkungan yang padat dengan jumlah 200 *node*, menghasilkan perbedaan selisih *forwarded route* request sebesar 30 atau mengalami penurunan sebesar 1.01%, dimana *routing protocol* DSR yang telah dimodifikasi juga unggul dalam hal *forwarded route* request tersebut.

Jika keempat lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang sedang, menghasilkan *forwarded route* request yang lebih bagus atau lebih sedikit daripada di lingkungan dengan jumlah *node* yang jarang maupun yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan *forwarded route* request yang lebih bagus atau dalam hal ini lebih rendah daripada AODV asli dengan jumlah selisih *forwarded route* request yang cukup signifikan.

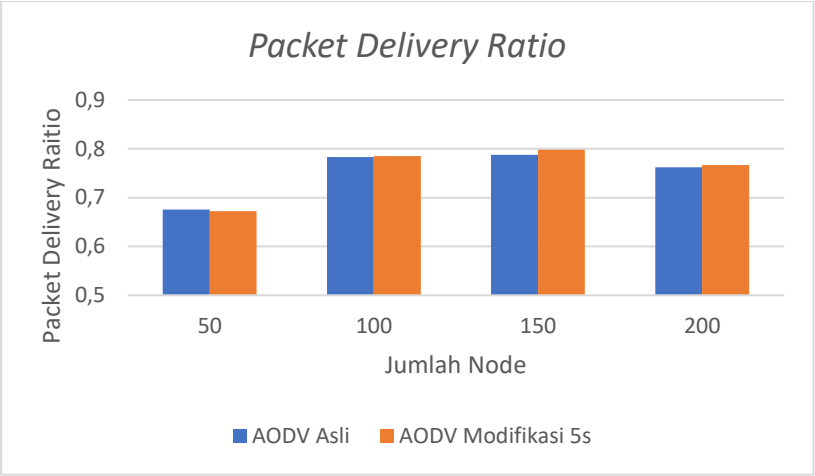
5.2.2 Hasil Uji Coba Skenario *Real*

Pengujian pada skenario *real* digunakan untuk melihat perbandingan *packet delivery ratio*, *rata-rata end-to-end delay*, *routing overhead* antara *routing protocol* AODV asli dan *routing protocol* AODV yang telah dimodifikasi dalam pemilihan *node* yang dapat menerima paket *route* request. Pengujian dilakukan sesuai dengan perancangan parameter lingkungan simulasi yang dapat dilihat pada Tabel 3.2.

Pengambilan data uji PDR, rata-rata *end-to-end delay*, *routing overhead*, dengan luas area 1500 m x 1500 m dan *node* sebanyak 50 untuk lingkungan yang jarang,100 dan 150 untuk lingkungan yang sedang, dan 200 untuk lingkungan yang padat. Seperti pada uji coba skenario *grid* untuk uji coba setiap lingkungan menggunakan threshold yang berbeda-beda karena satu threshold tidak bisa disamakan untuk semua lingkungan disebabkan jumlah tetangga yang dimiliki oleh setiap *node* pada setiap lingkungan berbeda-beda, tergantung pada jumlah *node* pada lingkungan simulasi tersebut.. Hasil analisis dapat dilihat pada Tabel 5.7, Tabel 5.8, Tabel 5.9, dan Tabel 5.10. dan grafik yang merepresentasikan hasil perhitungan PDR, E2E, RO, dan RREQ F yang ditunjukkan pada Gambar 5.5, Gambar 5.6, Gambar 5.7, dan Gambar 5.8

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

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	0.676	0.672	-0.0037
100	0.783	0.785	+0.0015
150	0.788	0.798	+0.0108
200	0.762	0.767	+0.0047



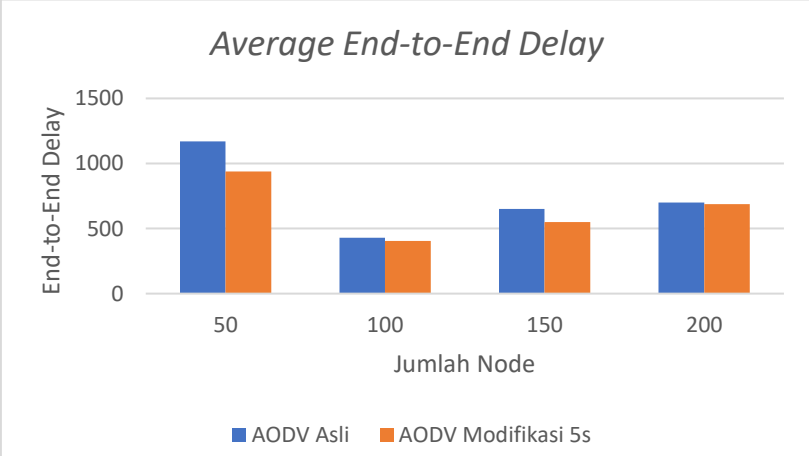
Gambar 5.5 Grafik Rata - Rata PDR Skenario *Real*

Berdasarkan grafik pada Gambar 5.5 dapat dilihat bahwa *routing protocol* AODV yang telah dimodifikasi dan juga *routing protocol* AODV asli mengalami penurunan yang signifikan pada packet delivery ratio. Pada lingkungan yang jarang dengan jumlah 50 *node*, menghasilkan perbedaan selisih PDR sebesar 0.0037, atau turun menjadi sekitar 0.5% dimana *routing protocol* AODV yang telah dimodifikasi mengalami penurunan dalam hal PDR tersebut. Pada lingkungan yang sedang dengan jumlah 100 *node*, menghasilkan perbedaan selisih PDR sebesar 0.0015 , atau naik menjadi sekitar 0.2% dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal PDR tersebut dari AODV asli. Pada lingkungan yang sedang dengan jumlah 150 *node*, menghasilkan perbedaan selisih PDR sebesar 0.0108, atau naik menjadi sekitar 1.37% dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal PDR tersebut dari AODV asli.Pada lingkungan yang padat dengan jumlah 200 *node*, menghasilkan perbedaan selisih PDR sebesar 0.0047, atau naik menjadi sekitar 0.6% dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal PDR tersebut.

Grafik PDR yang dimana pada *node* jarang yaitu 50 *node* lebih rendah dari AODV asli, *node* sedang 100,150 dan *node* padat pada *node* 200 mengalami kenaikan dengan menggunakan modifikasi. Jika keempat lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang lebih sedikit atau pada lingkungan dengan *node* yang jarang, menghasilkan PDR yang kurang dari daripada di lingkungan dengan jumlah *node* yang sedang maupun yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan PDR yang lebih bagus daripada AODV asli dengan jumlah selisih PDR yang cukup signifikan..

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

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	1170.82ms	938.68ms	-232.14ms
100	430.54ms	406.08ms	-24.46ms
150	651.05ms	550.26ms	-100.79ms
200	698.88ms	688.59ms	-10.29ms



Gambar 5.6 Grafik E2E pada Skenario *Real*

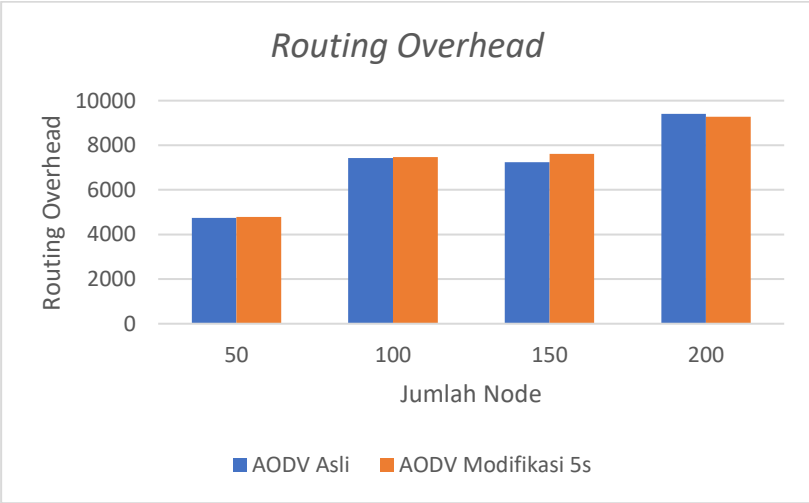
Berdasarkan grafik pada Gambar 5.6 dapat dilihat bahwa rata-rata *end-to-end delay* antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami penurunan yang signifikan. Pada lingkungan yang jarang dengan jumlah 50 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 232.14 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 19.82%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *end-to-end delay* tersebut. Sedangkan pada lingkungan sedang dengan jumlah 100 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 24.46 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami kenaikan sebesar 5.68%, dimana *routing protocol* AODV yang asli lebih unggul dalam hal *end-to-end delay* tersebut.Pada pada lingkungan sedang dengan jumlah 150 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 100.79 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami kenaikan sebesar 15.48%, dimana *routing protocol* AODV yang asli lebih unggul dalam hal *end-to-end delay* tersebut.Pada lingkungan yang padat dengan jumlah 200 *node*, terjadi perbedaan selisih *end-to-end delay* sebesar 20.89366667 ms antara *routing protocol* AODV asli dengan *routing protocol* AODV yang telah dimodifikasi atau mengalami penurunan sebesar 1.47%, dimana *routing protocol* AODV yang telah dimodifikasi jauh lebih unggul dalam hal *end-to-end delay* tersebut.

Jika keempat lingkungan tersebut dibandingkan, memang pada lingkungan yang jarang, sedang, maupun padat *routing protocol* AODV yang telah dimodifikasi selalu lebih unggul dalam hal *end-to-end delay*. Dapat dilihat bahwa dengan jumlah *node* jarang,sedang dan padat menghasilkan *end-to-end delay* yang lebih baik dan dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan *end-to end delay* yang lebih baik daripada AODV asli dengan jumlah selisih *end-to-end delay* yang cukup signifikan.Hal ini bisa terjadi karena dengan AODV modifikasi kali ini menghindari adanya

Congestion sehingga packet-packet akan lebih sedikit mengalami adanya delay dalam pengiriman.

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

Jumlah *Node*	**AODV Asli**	**AODV Modifikasi**	**Perbedaan**
50	4745	4784	+39
100	7422	7470	+48
150	7241	7617	+376
200	9409	9276	-133



Gambar 5.7 Grafik Rata - Rata RO Skenario *Real*

Berdasarkan grafik pada Gambar 5.7 dapat dilihat bahwa rata-rata *routing overhead* antara *routing protocol* AODV asli dan AODV yang telah dimodifikasi mengalami perubahan yang fluktuatif. pada *routing overhead*. Pada lingkungan yang jarang dengan jumlah 50 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 39

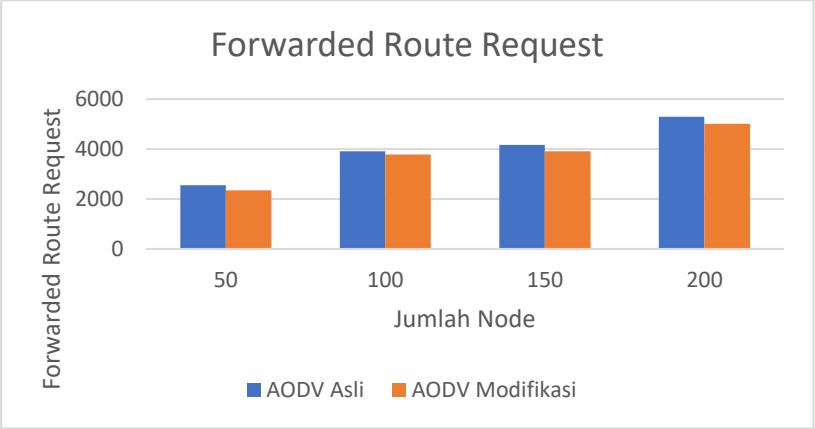
atau mengalami kenaikan sebesar 0.83%, dimana *routing protocol* AODV yang telah dimodifikasi kalah dalam hal routing overhead tersebut karena menghasilkan *routing overhead* yang lebih tinggi dari routing AODV asli. Pada lingkungan yang sedang dengan jumlah 100 *node*, menghasilkan perbedaan selisih routing overhead sebesar 48 atau mengalami penurunan sebesar 0.64%, dimana *routing protocol* AODV yang telah dimodifikasi mengalami kekalahan dalam hal *routing overhead* tersebut dari AODV asli. Pada lingkungan yang sedang dengan jumlah 150 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 376 atau mengalami kenaikan sebesar 5.19%, dimana *routing protocol* AODV yang telah dimodifikasi mengalami kekalahan dalam hal *routing overhead* tersebut dari AODV asli. Pada lingkungan yang padat dengan jumlah 200 *node*, menghasilkan perbedaan selisih *routing overhead* sebesar 133 atau mengalami penurunan sebesar 1.41%, dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal *routing overhead* tersebut.

Jika keempat lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* jarang yaitu *node* 50 menghasilkan nilai routing overhead yang lebih banyak daripada *protocol* AODV asli sehingga bisa dikatakan kurang baik dan dengan jumlah *node* sedang yaitu *node* 100 dan 150 juga mendapatkan hasil yang serupa dan dalam hal ini pada *node* padat yaitu 200 *node* menghasilkan hasilnya yang lebih baik daripada *protocol* AODV asli. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan routing overhead yang kurang dari atau dalam hal ini lebih tinggi daripada AODV asli dengan jumlah selisih routing overhead yang sangat tipis ini mungkin bisa terjadi karena adanya pemutusan link atau rute sehingga perlu adanya pencarian rute yang lebih sering ketimbang AODV asli sehingga nilai routing overhead modifikasi bisa lebih dari nilai asli.

Untuk hasil pengambilan data *forwarded route request* (RREQ F) pada skenario *grid* 150 *node*, 200 *node*, dan 300 *node* dapat dilihat pada Gambar 5.8.

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

Jumlah *Node*	AODV Asli	AODV Modifikasi	Perbedaan
50	2556	2349	-207
100	3906	3783	-123
150	4171	3908	-263
200	5299	5017	-282

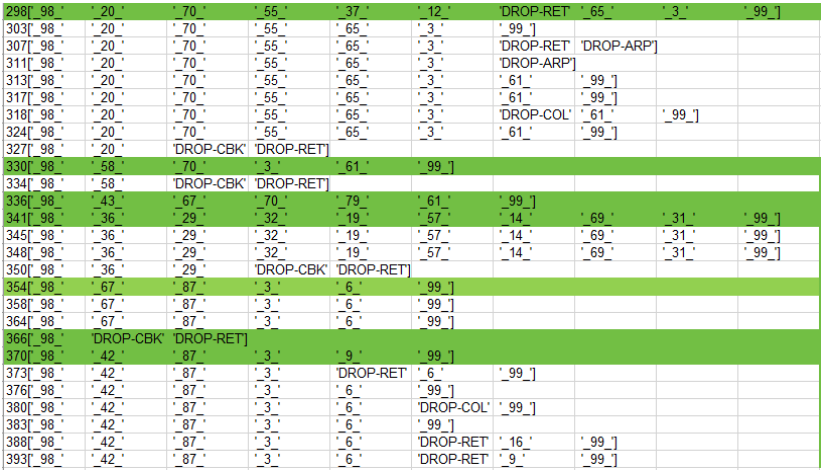


Gambar 5.8 Grafik Rata - Rata RREQ F Skenario *Real*

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 fluktuatif. Pada lingkungan yang jarang dengan jumlah 50 *node*, menghasilkan perbedaan selisih *forwarded route* request sebesar 207 atau mengalami penurunan sebesar 8.8%, dimana *routing protocol* AODV yang telah dimodifikasi unggul dalam hal *forwarded route* request tersebut karena menghasilkan *forwarded route* request yang lebih rendah dari *routing protocol* AODV asli. Pada lingkungan yang sedang dengan jumlah 100 *node*, menghasilkan perbedaan selisih *forwarded route* request sebesar 123 atau mengalami penurunan sebesar 3.16%, dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal *forwarded route* request tersebut

dari *forwarded route* request AODV asli.Pada lingkungan yang sedang dengan jumlah 150 *node*, menghasilkan perbedaan selisih *forwarded route* request sebesar 263 atau mengalami penurunan sebesar 6.3%, dimana *routing protocol* AODV yang telah dimodifikasi juga unggul dalam hal *forwarded route* request tersebut dari *forwarded route* request AODV asli. Pada lingkungan yang padat dengan jumlah 200 *node*, menghasilkan perbedaan selisih *forwarded route* request sebesar 282 atau mengalami penurunan sebesar 5.31%, dimana *routing protocol* DSR yang telah dimodifikasi juga unggul dalam hal *forwarded route* request tersebut.

Dalam hal ini terdapat pula bukti untuk pembatasan forwarding node yang bertugas untuk rebroadcast paket dengan membandingkan hasil *route* dari AODV asli dengan hasil pemilihan *route* AODV Modifikasi sebagai gambar berikut Gambar 5.9. dan Gambar 5.10.



298['_98_' '_20_' '_70_' '_55_' '_37_' '_12_' 'DROP-RET' '_65_' '_3_' '_99_']
303['_98_' '_20_' '_70_' '_55_' '_65_' '_3_' '_99_']
307['_98_' '_20_' '_70_' '_55_' '_65_' '_3_' 'DROP-RET' 'DROP-ARP']
311['_98_' '_20_' '_70_' '_55_' '_65_' '_3_' 'DROP-ARP']
313['_98_' '_20_' '_70_' '_55_' '_65_' '_3_' '_61_' '_99_']
317['_98_' '_20_' '_70_' '_55_' '_65_' '_3_' '_61_' '_99_']
318['_98_' '_20_' '_70_' '_55_' '_65_' '_3_' 'DROP-COL' '_61_' '_99_']
324['_98_' '_20_' '_70_' '_55_' '_65_' '_3_' '_61_' '_99_']
327['_98_' '_20_' 'DROP-CBK' 'DROP-RET']
330['_98_' '_58_' '_70_' '_3_' '_61_' '_99_']
334['_98_' '_58_' 'DROP-CBK' 'DROP-RET']
336['_98_' '_43_' '_67_' '_70_' '_79_' '_61_' '_99_']
341['_98_' '_36_' '_29_' '_32_' '_19_' '_57_' '_14_' '_69_' '_31_' '_99_']
345['_98_' '_36_' '_29_' '_32_' '_19_' '_57_' '_14_' '_69_' '_31_' '_99_']
348['_98_' '_36_' '_29_' '_32_' '_19_' '_57_' '_14_' '_69_' '_31_' '_99_']
350['_98_' '_36_' '_29_' 'DROP-CBK' 'DROP-RET']
354['_98_' '_67_' '_87_' '_3_' '_6_' '_99_']
358['_98_' '_67_' '_87_' '_3_' '_6_' '_99_']
364['_98_' '_67_' '_87_' '_3_' '_6_' '_99_']
366['_98_' 'DROP-CBK' 'DROP-RET']
370['_98_' '_42_' '_87_' '_3_' '_9_' '_99_']
373['_98_' '_42_' '_87_' '_3_' 'DROP-RET' '_6_' '_99_']
376['_98_' '_42_' '_87_' '_3_' '_6_' '_99_']
380['_98_' '_42_' '_87_' '_3_' '_6_' 'DROP-COL' '_99_']
383['_98_' '_42_' '_87_' '_3_' '_6_' '_99_']
388['_98_' '_42_' '_87_' '_3_' '_6_' 'DROP-RET' '_16_' '_99_']
393['_98_' '_42_' '_87_' '_3_' '_6_' 'DROP-RET' '_9_' '_99_']

Gambar 5.9 *Trace Route* pada AODV Asli

299['_98_'	'_76_'	'_80_'	'_72_'	'_10_'	'_74_'	'_92_'	'_69_'	'_99_']
302['_98_'	'_76_'	'_80_'	'_72_'	'_10_'	'_74_'	'_92_'	'_69_'	'_99_']
308['_98_'	'_76_'	'_80_'	'_72_'	'_10_'	'_74_'	'_92_'	'_69_'	'_99_']
311['_98_'	'_76_'	'_80_'	'_72_'	'DROP-CBK'	'DROP-RET']			
316['_98_'	'_67_'	'_87_'	'_9_'	'_79_'	'_99_']			
322['_98_'	'_67_'	'_87_'	'_9_'	'_79_'	'DROP-RET'	'_16_'	'_99_']	
325['_98_'	'_67_'	'_87_'	'_9_'	'_79_'	'_16_'	'_99_']		
330['_98_'	'_67_'	'_87_'	'_9_'	'_79_'	'_16_'	'_99_']		
334['_98_'	'_67_'	'_87_'	'_9_'	'_79_'	'_16_'	'_99_']		
338['_98_'	'_67_'	'_87_'	'DROP-CBK'	'DROP-RET']				
342['_98_'	'_67_'	'_70_'	'DROP-COL'	'_79_'	'_16_'	'_99_']		
347['_98_'	'_67_'	'_70_'	'_9_'	'_99_']				
351['_98_'	'_67_'	'_70_'	'_9_'	'_99_']				
357['_98_'	'_67_'	'_70_'	'_9_'	'_99_']				
361['_98_'	'_67_'	'_70_'	'_9_'	'_99_']				
363['_98_'	'DROP-CBK'	'DROP-RET']						
368['_98_'	'_42_'	'_67_'	'_79_'	'DROP-COL'	'_52_'	'_99_']		
373['_98_'	'_42_'	'_67_'	'_79_'	'_52_'	'_99_']			
377['_98_'	'_42_'	'_67_'	'_79_'	'_52_'	'_99_']			
382['_98_'	'_42_'	'_67_'	'_79_'	'_52_'	'_99_']			
387['_98_'	'_42_'	'_67_'	'_79_'	'_52_'	'_99_']			
391['_98_'	'_42_'	'_67_'	'_79_'	'_52_'	'_99_']			

Gambar 5.10 *Trace Route* pada AODV Modifikasi

Dan dengan gambar diatas dapat dilihat bahwa pada Gambar 5.9 merupakan *trace route* dari AODV asli menggunakan jalur *route* awal node yaitu node 20,58,43,36,67,42 dan terdapat 7 jalur *route* untuk mencapai node destinasi berbeda dengan halnya pada Gambar 5.10 yang merupakan *trace route* dari AODV modifikasi menggunakan jalur *route* awal node yaitu node 76,67,42 dan terdapat hanya 3 jalur *route* yang dipilih untuk mencapai destinasi.Dalam hal ini dapat disimpulkan bahwa pemilihan *route* yang tepat dengan modifikasi *Congestion* berhasil membatasi jumlah *forwarding node* yang dilakukan dan menghindari beberapa *route* yang berpotensi terjadi lebih banyak *drop packet*.

Dapat dilihat rata-rata penurunan yang terjadi untuk *forwarded route* request adalah sebesar 5.89%. Jika keempat lingkungan tersebut dibandingkan dapat dilihat bahwa dengan jumlah *node* yang sedang, menghasilkan *forwarded route* request yang lebih bagus atau lebih sedikit daripada di lingkungan dengan jumlah *node* yang jarang maupun yang padat baik untuk *routing protocol* AODV asli maupun *routing protocol* AODV yang telah dimodifikasi. Dapat dilihat pula bahwa AODV yang telah dimodifikasi menghasilkan *forwarded route* request yang lebih bagus atau dalam hal ini lebih rendah daripada AODV asli dengan jumlah selisih *forwarded route* request yang cukup signifikan.

(Halaman ini sengaja dikosongkan)

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. AODV yang dimodifikasi sudah berhasil membatasi jumlah *forwarding node* yang bertugas untuk *rebroadcast* paket RREQ dengan jumlah dari *Forwarded Route* Request pada modifikasi yaitu dalam sample *route* trace yang ditampilkan yaitu hanya 3 node dan pada asli menggunakan 7 node.
2. Dampak pembatasan *forwarding node* terhadap performa protokol AODV pada skenario *grid* adalah rata – rata kenaikan *Packet Delivery Ratio* (PDR) sebesar 0,7%,rata – rata penurunan *End to End Delay* (E2E Delay) sebesar 10,183%,rata-rata kenaikan *Routing Overhead* (RO) sebesar 3,6% dan rata – rata penurunan *Forwarded Route Request* (RREQ F) pada *node* 50 , 150 dan 200 sebesar 3,67%.
3. Dampak pembatasan *forwarding node* terhadap performa protokol AODV pada skenario *real* adalah rata – rata kenaikan *Packet Delivery Ratio* (PDR) pada *node* 100,150 dan 200 sebesar 0,72%, rata – rata penurunan *End to End Delay* (E2E Delay) sebesar 10,613%,rata-rata kenaikan *Routing Overhead* (RO) pada *node* 50,100,150 sebesar 2,2% dan penurunan pada *node* 200 1,41% dan rata – rata penurunan *Forwarded Route Request* (RREQ F) sebesar 5,89%.

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. Menambahkan aspek lain untuk melakukan pembatasan *forwarding node* yang meneruskan paket RREQ seperti arah, kecepatan, dan energi.
3. Bisa untuk mengembangkan nilai *Congestion* pada suatu *node* yang dinamis dan heterogen sesuai dengan kondisi *node* agar hasil bisa lebih maksimal.
4. Diharapkan bisa menghandle kondisi extreme dengan menandai *node* yang pernah mengalami *Congestion.*

DAFTAR PUSTAKA

[1] R. Brendha dan V. S. J. Prakash, "A Survey on Routing Protocols for Vehicular Ad hoc Networks," IEEE, Coimbatore, 2017.

[2] R. F. Sari dan A. Syarif, "Analisis Kinerja Protokol Routing Ad Hoc On-Demand Distance Vector (AODV) pada Jaringan Ad Hoc," p. 22, October 2010.

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

[4] "OpenStreetMap," [Online]. Available: https://www.openstreetmap.org/. [Diakses 10 06 2019].

[5] D. Krajzewics, J. Erdmann, M. Behrisch dan L. Bieker, "Recent Development and Application of SUMO," *International Journal On Advances in Systems and Measurements,* p. 128, December 2012.

[6] "AWK," [Online]. Available: http://tldp.org/LDP/abs/html/awk.html. [Diakses 10 06 2019].

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

[8] A. Rhim dan Z. Dziong, "Routing Based on Link Expiration Time for MANET Performance Improvement," IEEE, Kuala Lumpur, 2009.

[9] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum dan L. Viennot, "Optimized Link State Routing Protocol for Ad hoc Networks," IEEE, Lahore, 2001.

[10] S. N. Ferdous dan M. S. Hossain, "Randomized Energy-Based AODV Protocol for Wireless Ad-Hoc Network," IEEE, Dhaka, 2016.

[11] “JOSM,” [Online]. Available: https://josm.openstreetmap.de/. [Diakses 10 06 2019].

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

[13] M. Iqbal, M. Shafiq, H. Attaullah, J.-G. Choi, K. Akram dan X. Wang, “Design and Analysis of a Novel Hybrid Wireless Mesh Network Routing Protocol,” p. 22, January 2014.

[14] N. Wang, Y. Mai dan F. M. Rodriguez, “CC-AODV: An effective multiple paths *Congestion* control AODV,” dalam *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, USA, 2018.

LAMPIRAN

A.1 Kode Skenario NS-2

```
set val(chan)   Channel/WirelessChannel;
set val(prop)   Propagation/TwoRayGround;
set val(netif)  Phy/WirelessPhy;
set val(mac)    Mac/802_11;
set val(ifq)    Queue/DropTail/PriQueue;
set val(ll)     LL;
set val(ant)    Antenna/OmniAntenna;
set opt(x)      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.2 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.3 Kode Skrip AWK *Packet Delivery Ratio*

```
BEGIN {
        sendLine = 0;
        recvLine = 0;
        fowardLine = 0;
}

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

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

$0 ~/^f.* RTR/ {
        fowardLine ++ ;
}

END {
        printf "cbr s:%d r:%d, r/s
Ratio:%.4f, f:%d \n", sendLine, recvLine,
(recvLine/sendLine),fowardLine;
}
```

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

```
BEGIN{
      sum_delay = 0;
      count = 0;
}
{
      if ($2 >= 101) {

            if($4 == "AGT" && $1 == "s" &&
seqno < $6) {
              seqno = $6;
            }

            if($4 == "AGT" && $1 == "s") {
                  start_time[$6] = $2;
            }

            else if(($7 == "cbr") && ($1 ==
"r")) {
                  end_time[$6] = $2;
            }

            else if($1 == "D" && $7 == "cbr")
{
                  end_time[$6] = -1;
            }
      }
}
END {

    for(i=0; i<=seqno; i++) {
            if(end_time[i] > 0) {
                  delay[i] = end_time[i] -
start_time[i];
                  count++;
        }
        else {
            delay[i] = -1;
        }
    }
```

```
    for(i=0; i<=seqno; i++) {
        if(delay[i] > 0) {
           n_to_n_delay = n_to_n_delay +
delay[i];
        }
    }
     n_to_n_delay = n_to_n_delay/count;
     printf "End-to-End Delay \t= "
n_to_n_delay * 1000 " ms \n";
}
```

A.5 Kode Skrip AWK *Routing Overhead*

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

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

A.6 Kode Skrip AWK *Forwarded Route Request*

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

A.7 Kode Skrip AODV.CC fungsi method sendRequest

```
1 void AODV::sendRequest(nsaddr_t dst) {
2  // Allocate a RREQ packet
3  Packet *p = Packet::alloc();
4  struct hdr_cmn *ch = HDR_CMN(p);
5  struct hdr_ip *ih = HDR_IP(p);
6  struct hdr_aodv_request *rq =
HDR_AODV_REQUEST(p);
7  aodv_rt_entry *rt = rtable.rt_lookup(dst);

8  assert(rt);

9  if (rt->rt_flags == RTF_UP) {
10    assert(rt->rt_hops != INFINITY2);
11    Packet::free((Packet *)p);
12    return;
13 }
14 if (rt->rt_req_timeout > CURRENT_TIME) {
15   Packet::free((Packet *)p);
16  return;
17  }

18  if (rtable.Congestion_counter >
CONGESTION_THRESHOLD)
19  {
20    #ifdef DEBUG
21    FILE *fp;
2    fp = fopen("debug.txt", "a");
22    fprintf(fp, "\n%f %s: sedang pada node : %d
Congestion counter : %d  di drop", CURRENT_TIME,
__FUNCTION__,index,rtable.Congestion_counter);
23    fclose(fp);
24    #endif // DEBUG
25    Packet::free((Packet *)p);
26    //drop(p, DROP_EXCEED_CONGESTION_LIMIT);
27    return;
  }

1 void AODV::sendRequest(nsaddr_t dst) {
```

A.8 Kode Skrip AODV.CC

```
1 #include <aodv/aodv.h>
2 #include <aodv/aodv_packet.h>
3 #include <random.h>
4 #include <cmu-trace.h>
5 #include <iostream>


6 #define max(a,b)        ( (a) > (b) ?
(a) : (b) )
7 #define CURRENT_TIME
Scheduler::instance().clock()

8 #define DROP_EXCEED_CONGESTION_LIMIT
"DECL"
9 #define CONGESTION_THRESHOLD 4


// count neighbour
10 int count_neighbour[1000];
11 int nodes_count=0;

// init treshold
12 int th = -1;
13 int old_th;

#ifdef DEBUG
14 static int route_request = 0;
#endif

1 #include <aodv/aodv.h>
2 #include <aodv/aodv_packet.h>
```

A.10 Kode Skrip AODV_RTABLE.H

```
1 class aodv_rtable {
2 public:
3     aodv_rtable() {
4      LIST_INIT(&rthead);
5      Congestion_counter = 0;
6     }

7         aodv_rt_entry*        head() { return
rthead.lh_first; }

8    aodv_rt_entry*      rt_add(nsaddr_t id);
9    void                rt_delete(nsaddr_t id);
10   aodv_rt_entry*      rt_lookup(nsaddr_t id);
11   int                   Congestion_counter;

11 private:
         LIST_HEAD(aodv_rthead, aodv_rt_entry)
rthead;
};

#endif /* _aodv__rtable_h__ */
```

A.11 Kode Skrip AODV.CC Fungsi Method Send Reply

```
void AODV::sendReply(nsaddr_t ipdst, u_int32_t
hop_count, nsaddr_t rpdst, u_int32_t rpseq,
u_int32_t lifetime, double timestamp) {
  Packet *p = Packet::alloc();
  struct hdr_cmn *ch = HDR_CMN(p);
  struct hdr_ip *ih = HDR_IP(p);
  struct hdr_aodv_reply *rp =
HDR_AODV_REPLY(p);
  aodv_rt_entry *rt = rtable.rt_lookup(ipdst);

  #ifdef DEBUG
    FILE *fp;
    fp = fopen("debug.txt", "a");
    fprintf(fp, "\n%f %s: sedang pada node :
%d lifetime : %d ", CURRENT_TIME,
__FUNCTION__,index,lifetime);
    fclose(fp);
  #endif // DEBUG

    if (lifetime == 0 )
    {
      #ifdef DEBUG
    fp = fopen("debug.txt", "a");
    fprintf(fp, "\n%f %s: sedang pada node :
%d Congestion counter : %d  lifetime habis",
CURRENT_TIME,
__FUNCTION__,index,rtable.Congestion_counter);
    fclose(fp);
  #endif // DEBUG
      rtable.Congestion_counter--;
    }

  #ifdef DEBUG
    fprintf(stderr, "sending Reply from %d at
%.2f\n", index,
Scheduler::instance().clock());
```

A.12 Kode Skrip AODV.CC Fungsi Method RecvReply

```
if(ih->daddr() == index || suppress_reply)
Packet::free(p);
  else {
    // Find the rt entry
    aodv_rt_entry *rt0 = rtable.rt_lookup(ih-
>daddr());
    // If the rt is up, forward
    if(rt0 && (rt0->rt_hops != INFINITY2)) {
      assert (rt0->rt_flags == RTF_UP);
      rp->rp_hop_count += 1;
      rp->rp_src = index;
      if (rtable.Congestion_counter >
CONGESTION_THRESHOLD){
        #ifdef DEBUG
        fp = fopen("debug.txt", "a");
        fprintf(fp, "\n%f %s: dropping Route
Reply waktu forward ada pada node : %d \n",
CURRENT_TIME,__FUNCTION__,index);
        fclose(fp);
      #endif // DEBUG
        drop(p, DROP_RTR_NO_ROUTE);
        return;
      }
      else{
        forward(rt0, p, NO_DELAY);
        rtable.Congestion_counter++;
      rt->pc_insert(rt0->rt_nexthop); //
nexthop to RREQ source
      }
    }
    else {

      #ifdef DEBUG
        fp = fopen("debug.txt", "a");
        fprintf(fp, "\n%f %s: dropping Route
Reply ada pada node : %d \n",
CURRENT_TIME,__FUNCTION__,index);
```

BIODATA PENULIS

Huda Fauzan Murtadho, lahir di Tasikmalaya, 23 Agustus 1996. Penulis adalah anak kedua dari tiga bersaudara. Penulis menempuh pendidikan sekolah dasar di SDN 1 Klaten lalu melanjutkan pendidikan sekolah menengah pertama di SMPN 1 Klaten dan penulis menempuh pendidikan menengah atas di SMA Negeri 1 Klaten. Selanjutnya penulis melanjutkan pendidikan sarjana di Departemen Informatika, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya. Selama kuliah, penulis aktif dalam berbagai organisasi baik tingkat jurusan maupun universitas.Dalam menyelesaikan pendidikan S1, penulis mengambil bidang minat Arsitektur dan Jaringan Komputer (AJK). Sebagai mahasiswa, penulis berperan aktif dalam beberapa organisasi kampus seperti staf Dalam Negeri Himpunan Mahasiswa Teknik-Computer (HMTC) ITS, Kepala Departemen Dalam Negeri Himpunan Mahasiswa Teknik-Computer (HMTC) ITS , staf Internal Affair Badan Eksekutif Mahasiswa FTIf ITS. Selain itu, penulis juga menjadi staf REEVA SCHEMATICS 2016, staf ahli REEVA SCHEMATICS 2017 dan sebagai Konseptor Internal pada acara FTIf Festival 2017. Penulis pernah melakukan kerja praktik di PT. PLN(Persero) Desember 2017 – Januari 2018 dan membuat aplikasi berbasis web Monitoring Mutasi Rotasi Promosi pada bidang Talenta. Penulis dapat dihubungi melalui nomor *handphone*: 082138020890 atau *email*: hudafauzanm@gmail.com.