TUGAS AKHIR – KI141502

MODIFIKASI FUNGSI PERIODE PERBARUAN *NODE* PADA *ROUTING PROTOCOL* DSDV DI DALAM LINGKUP JARINGAN VANETS

MONICA INDAH HABSARI
NRP 05111440000060

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

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

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

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR – KI141502

MODIFIKASI FUNGSI PERIODE PERBARUAN *NODE* PADA *ROUTING PROTOCOL* DSDV DI DALAM LINGKUP JARINGAN VANETS

MONICA INDAH HABSARI
NRP 05111440000060

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

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

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

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESES – KI141502

FUNCTION MODIFICATION OF NODE UPDATE PERIOD ON DSDV ROUTING PROTOCOL IN VANETS SCOPE

MONICA INDAH HABSARI
NRP 05111440000060

First Advisor
Ir. Muchammad Husni, M.Kom.

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

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

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

MODIFIKASI FUNGSI PERIODE PERBARUAN *NODE* PADA *ROUTING PROTOCOL* DSDV DI DALAM LINGKUP JARINGAN VANETS

TUGAS AKHIR

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

Oleh:
MONICA INDAH HABSARI
NRP 05111440000060

Disetujui oleh Pembimbing Tugas Akhir:

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

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

SURABAYA
JANUARI, 2019

(Halaman ini sengaja dikosongkan)

MODIFIKASI FUNGSI PERIODE PERBARUAN *NODE* PADA *ROUTING PROTOCOL* DSDV DI DALAM LINGKUP JARINGAN VANETS

Nama Mahasiswa : Monica Indah Habsari
NRP : 05111440000060
Departemen : Informatika FTIK-ITS
Dosen Pembimbing 1 : Ir. Muchammad Husni, M.Kom.
Dosen Pembimbing 2 : Dr.Eng. Radityo Anggoro, S.Kom., M.Sc.

Abstrak

Vehicular Ad-hoc Networks (VANET) adalah kondisi spesial dari *Mobile Ad-hoc Networks* (MANET), yang mana memiliki istilah "jaringan di atas roda". VANET adalah jaringan Ad-hoc yang dibentuk diantara kendaraan sebagaimana kebutuhan komunikasi mereka. Setiap kendaraan yang terlibat di dalam lingkup pengembangan jaringan VANET harus memiliki kemampuan transmisi dan penerimaan sinyal dengan kecepatan yang tinggi. VANET menjadi lebih menantang jika dibandingkan dengan MANET, karena VANET memiliki tingkat mobilitas *node* yang tinggi dan perubahan topologi yang cepat.

Destination Sequenced Distance Vector (DSDV) merupakan protokol perutean yang menggunakan algoritma Bellman-Ford untuk menghitung lintasannya. *Metrik cost* yang digunakan adalah *hop count*, yang merupakan jumlah *hop* yang dibutuhkan paket untuk mencapai destinasinya. DSDV merupakan protokol proaktif, sehingga mampu menjaga *routing table* dengan entri-entri untuk semua *node* di dalam jaringan, tidak hanya untuk *node* tetangga. Setiap perubahan akan disebarluaskan melalui mekanisme perbaruan berkala dan dengan trigger yang digunakan pada DSDV. *Routing update* pada DSDV dilakukan dengan dua cara, yakni *full dump* dan *incremental*.

Routing update full dump dilakukan secara berkala setiap 15 detik.

Tugas akhir ini memodifikasi fungsi yang berkenaan dengan periode perbaruan node pada protokol perutean DSDV di dalam lingkup jaringan VANETs. Modifikasi fungsi ini menyesuaikan periode perbaruan *node* dalam *routing table* dengan kecepatan pengiriman paket antar *node* pada kendaraan. Sehingga dengan adanya modifikasi ini diharapkan performa dari protokol perutean DSDV akan menjadi lebih representatif dan mengurangi tingkat *packet loss* dalam tranmisi dan penerimaan paket.

Dari hasil uji coba, terbukti bahwa modifikasi yang dilakukan dapat meningkatkan nilai *Packet Delivery Ratio* (PDR). Rata-rata peningkatan nilai PDR pada skenario *grid* adalah 6,83%. Rata-rata peningkatan nilai PDR pada skenario *real* adalah 5,12%.

Kata kunci: **MANETs, VANETs, DSDV, Periode Perbaruan, *Network Simulator* NS2**

FUNCTION MODIFICATION OF NODE UPDATE PERIOD ON DSDV ROUTING PROTOCOL IN VANETS SCOPE

Student's Name	**: Monica Indah Habsari**
Student's ID	**: 05111440000060**
Department	**: Informatics FTIK-ITS**
First Advisor	**: Ir. Muchammad Husni, M.Kom.**
Second Advisor	**: Dr.Eng. Radityo Anggoro, S.Kom, M.Sc.**

Abstract

Vehicular Ad-hoc Networks (VANET) is a special condition of Mobile Ad-hoc Networks (MANET), which has the term "network on wheels". VANET is an Ad-hoc network that is formed between vehicles as their communication needs. Every vehicle involved in the scope of VANET network development must have the ability to transmit and receive signals at high speeds. VANET is more challenging compared to MANET, because VANET has high level of node mobility and rapid topology changes.

Destination Sequenced Distance Vector (DSDV) is a routing protocol that uses the Bellman-Ford algorithm to calculate its trajectory. The cost metric used is hop count, which is the number of hops needed for the package to reach its destination. DSDV is a proactive protocol, so it is able to maintain the routing table with entriesfor all nodes in the network, not only for the neighboring nodes. Each change will be disseminated through a periodic update mechanism and with a trigger used on DSDV. Routing updates on DSDV are done in two ways, called full dump and incremental. Full dump update routing is done regularly every 15 seconds.

This final assignment modifies the function with respect to the update period of the DSDV routing protocol node within

the VANET network. This function modification adjust the node update period in the routing table with the speed of sending packets between nodes on the vehicle. So that with this modification it is expected that the performance of the DSDV routing protocol will become more representative and reduce packet loss rates in packet transmission and reception.

From the results of the experiments, it was proven that the modifications made could increase the value of the Packet Delivery Ratio (PDR). The average increase in PDR value in the grid scenario is 6.83%. The average increase in PDR value in the real scenario is 5.12%.

Keywords**: MANETs, VANETs, DSDV, Update Period, Network Simulator NS2

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

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

"MODIFIKASI FUNGSI PERIODE PERBARUAN *NODE* PADA *ROUTING PROTOCOL* DSDV DI DALAM LINGKUP JARINGAN VANETS"

Melalui lembar ini, penulis ingin menyampaikan ucapan terimakasih dan penghormatan yang sebesar-besarnya kepada:

1. Orang tua dan segenap keluarga besar yang selalu memberikan dukungan penuh untuk menyelesaikan Tugas Akhir ini.
2. Bapak Radityo Anggoro dan Bapak Muchammad Husni selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberikan petunjuk selama proses pengerjaan Tugas Akhir ini.
3. Ibu Dini Adni Navastara dan Bapak Rizky Januar Akbar yang pernah menjadi dosen wali penulis yang telah memberi semangat dan mengarahkan penulis agar dapat menyelesaikan perkuliahan dengan baik.
4. Sahabat-sahabat dekat penulis selama perkuliahan di Teknik Informatika ITS, terutama anggota grup KanMakan.
5. Para admin lab AJK yang selalu penulis repotkan semasa pengerjaan TA.
6. Seluruh teman-teman seangkatan TC14 yang selalu bersama dan memberikan banyak sekali kenangan.
7. Teman-teman Keputih Gg 3C/4, terutama penghuni lantai 2 yang tidak pernah berhenti mengingatkan penulis untuk segera menyelesaikan TA.

8. Teman-teman komunitas Surabaya Osoji Club, NihoNime, Marine Dream yang menjadi keluarga baru tempat dimana penulis bisa sejenak melepas kepenatan perkuliahan serta memberi semangat menyelesaikan perkuliahan.
9. Teman-teman yang dulu satu kepengurusan bersama penulis baik di BIMITS, UKM TDC, UKM IFLS, maupun KWU HMTC ITS. Penulis bangga dan bahagia pernah menjadi bagian dari kalian.
10. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satu per satu.

Penulis telah berusaha sebaik mungkin dalam penyusunan Tugas Akhir ini, namun tak lupa penulis mohon maaf apabila di kemudian hari ditemukan kekurangan atau kesalahan pada Tugas Akhir ini. Guna perbaikan di masa mendatang, penulis dengan besar hati akan menerima segala kritik dan saran yang membangun dari pembaca sekalian.

Surabaya, Januari 2019

Monica Indah Habsari

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

Pada bab ini akan dijelaskan mengenai garis besar Tugas Akhir yang meliputi latar belakang, rumusan permasalahan, batasan permasalahan, tujuan, manfaat, metodologi, dan sistematika penulisan.

1.1 Latar Belakang

Teknologi komunikasi terus dikembangkan dengan tujuan memudahkan manusia dalam melakukan komunikasi. Para ahli terdorong untuk mengembangkan teknik komunikasi jarak jauh yang lebih efisien dengan metode telekomunikasi yang memanfaatkan teknologi elektronika, yang disebut teknik komunikasi data [12]. Jaringan *wireless* (nirkabel) merupakan salah satu teknik komunikasi data yang terus mengalami perbaruan. Sistem jaringan *wireless* (nirkabel) yang kini mendapat perhatian lebih untuk sektor industri, sains, medis, dan militer adalah jaringan ad-hoc, dikarenakan jaringan ini merupakan jaringan yang mandiri atau berdiri sendiri sehingga tidak diperlukan adanya fungsionalitas *access point* sebagai pendamping di dalam jaringan ad-hoc untuk menghubungkan antar perangkat, atau disebut juga sebagai *Independent Basic Service Set* (IBSS). Beberapa jenis jaringan ad-hoc yang telah dikembangkan, antara lain: *Wireless Ad-hoc Network* (WANETs), *Mobile Ad-hoc Network* (MANETs), *Vehicular Ad-hoc Network* (VANETs), *Smart Phone Ad-hoc Network* (SPANs), *Internet Based Mobile Ad-hoc Network* (iMANETs), Military / Tactical MANETs, *Self Powered Ad-hoc Network* (SPAN), dan lain-lain perkembangannya.

Vehicular Ad-hoc Networks (VANET) adalah kondisi spesial dari *Mobile Ad-hoc Networks* (MANET), yang mana memiliki istilah “jaringan di atas roda”. VANETs adalah jaringan ad-hoc yang dibentuk diantara kendaraan sebagaimana kebutuhan

komunikasi mereka [2]. Setiap kendaraan yang terlibat di dalam lingkup pengembangan jaringan VANETs harus memiliki kemampuan transmisi dan penerimaan sinyal dengan kecepatan yang tinggi.

Beberapa jenis protokol perutean yang dapat digunakan di dalam lingkup jaringan VANETs, yakni: proaktif, reaktif, dan hibrida. Protokol perutean jenis proaktif adalah protokol yang akan memperbarui *routing table* secara berkala, contohnya *Destination Sequenced Distance Vector* (DSDV). Protokol perutean jenis reaktif adalah protokol yang akan membentuk rute jika suatu *node* memintanya, contohnya *Ad hoc On Demand Distance Vector* (AODV) dan *Dynamic Source Routing* (DSR). Protokol perutean hibrida adalah protokol kombinasi yang mempertimbangkan keuntungan dari proaktif dan reaktif, contohnya *Zone Routing Protocol* (ZRP).

Tugas Akhir ini dibuat dengan maksud untuk melakukan uji simulasi jaringan guna meningkatkan performa jaringan VANET, khususnya pada protokol perutean *Destination Sequenced Distance Vector* (DSDV). Faktor yang akan menentukan kebaikan performa jaringan modifikasi adalah *Packet Delivery Ratio* (PDR), *End to End Delay / One Way Delay* (E2E), dan *Routing Overhead* (RO).

1.2 Rumusan Permasalahan

Berikut beberapa rumusan permasalahan yang akan diselesaikan pada tugas akhir ini:

1. Bagaimana memodifikasi fungsi periode perbaruan *node* berdasarkan faktor kecepatan kendaraan pada protokol perutean DSDV di dalam lingkup jaringan VANETs?
2. Bagaimana pengaruh modifikasi fungsi periode perbaruan *node* berdasarkan faktor kecepatan kendaraan terhadap performa protokol perutean DSDV di dalam lingkup jaringan VANETs?

1.3 Batasan Permasalahan

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

1. Lingkup jaringan yang digunakan adalah *Vehicle Ad-hoc Networks* (VANETs).
2. Protokol perutean yang digunakan adalah *Destination Sequenced Distance Vector* (DSDV).
3. Modifikasi jaringan hanya akan diuji cobakan menggunakan aplikasi *Network Simulator 2* (NS-2).
4. Skenario uji coba akan dibuat menggunakan *Simulation of Urban Mobility* (SUMO).

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah sebagai berikut:

1. Untuk menerapkan modifikasi fungsi periode perbaruan *node* berdasarkan faktor kecepatan kendaraan pada protokol perutean DSDV di dalam lingkup jaringan VANETs.
2. Untuk menganalisis pengaruh modifikasi fungsi periode perbaruan *node* berdasarkan faktor kecepatan pada protokol perutean DSDV di dalam lingkup jaringan VANETs dengan skenario *grid* dan *real*.

1.5 Manfaat

Manfaat dari pembuatan tugas akhir ini adalah sebagai berikut:

1. Memberikan manfaat di dalam lingkup jaringan VANETs dengan mendapatkan rata-rata performa yang lebih baik.
2. Memberikan informasi tentang pengaruh modifikasi fungsi periode perbaruan *node* berdasarkan faktor kecepatan kendaraan terhadap kinerja protokol perutean DSDV di dalam lingkup jaringan VANETs.

3. Menerapkan salah satu ilmu di bidang jaringan yang telah dipelajari selama perkuliahan di Departemen Informatika agar berguna bagi masyarakat.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Subbab metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula subbab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir. Dan disertai dengan daftar pustaka terkait dengan penyelesaian tugas akhir.

1.6.2 Studi Literatur

Studi literatur mencakup pengumpulan informasi mengenai apa saja yang bisa dijadikan referensi dalam pengerjaan tugas akhir. Mengumpulkan informasi dan studi literatur mengenai topologi jaringan VANETs, protokol perutean DSDV, simulator jaringan NS2, dan juga tentang SUMO, OpenStreetMap, JOSM dan AWK. Informasi didapatkan dari jurnal, buku, dan materi kuliah yang berhubungan dengan topik tugas akhir.

1.6.3 Analisis dan Desain Sistem

Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari protokol perutean DSDV di dalam lingkup jaringan VANETs. Tahap ini mendefinisikan alur dari modifikasi fungsi pada protokol perutean DSDV. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dihasilkan analisis dan hasil percobaan konfigurasi protokol perutean DSDV modifikasi berdasarkan faktor kecepatan kendaraan di dalam lingkup jaringan VANETs.

1.6.4 Implementasi Sistem

Pada tahap ini dilakukan implementasi model jaringan dalam penerapan protokol perutean DSDV hasil modifikasi dengan generator skenario SUMO dan pembuatan peta *real* menggunakan OSM dan JOSM. Skenario-skenario dari generator SUMO dan JOSM akan disimulasikan ke dalam simulator jaringan NS2. Selain itu juga terdapat implementasi program untuk metode analisis menggunakan AWK.

1.6.5 Pengujian dan Evaluasi

Pada tahapan ini dilakukan pengujian skenario secara bertahap dengan simulator VANETs dan dari tiap tahapan akan dilakukan evaluasi. Pengujian akan dilakukan dengan melihat kesesuaian hasil modifikasi dengan perencanaan awal. Dan juga dilakukan evaluasi kebaikan performa dari protokol DSDV yang telah dimodifikasi. Kebaikan performa diukur dengan menghitung *Packet Delivery Ratio* (PDR), *End to End Delay / One Way Delay* (E2E), dan *Routing Overhead* (RO). Hasil evaluasi dari protokol perutean DSDV modifikasi akan dibandingkan dengan DSDV murni.

1.6.6 Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir, hasil implementasi sistem yang telah dibuat, serta hasil pengujian dan evaluasi sistem. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
    a. Latar Belakang
    b. Rumusan Permasalahan
    c. Batasan Permasalahan
    d. Tujuan
    e. Manfaat
    f. Metodologi
    g. Sistematika Penulisan Laporan
2. Tinjauan Pustaka
3. Perancangan Sistem
4. Implementasi
5. Pengujian dan Evaluasi
6. Kesimpulan dan Saran
7. Daftar Pustaka

1.7 Sistematika Penulisan Laporan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini.

1. Bab I. Pendahuluan
   Bab ini memuat latar belakang pembuatan Tugas Akhir, rumusan permasalahan, batasan permasalahan, tujuan,

manfaat, metodologi yang digunakan, dan sistematika penulisan Tugas Akhir.

2. Bab II. Tinjauan Pustaka
   Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini. Adapun beberapa teori yang dibahas disini adalah tentang MANETs, VANETs, DSDV, NS2, SUMO, OpenStreetMap, JOSM, dan AWK.
3. Bab III. Perancangan Sistem
   Bab ini membahas mengenai perancangan sistem yang digunkan di dalam Tugas Akhir ini. Di dalam bab ini akan dibahas alur pembuatan skenario mobilitas *grid* dan *real* untuk uji coba sistem. Dalam bab ini akan dibahas pula perancangan alur modifikasi fungsi periode perbaruan node pada protokol perutean DSDV di dalam lingkup jaringan VANETs. Perancangan simulasi jaringan pada NS2 dan metode analisis juga dibahas dalam bab ini.
4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi dari perancangan sistem yang telah dibuat pada bab sebelumnya. Bab ini akan memuat implementasi pembuatan skenario mobilitas *grid* dan *real* menggunakan SUMO, OpenStreetMap, dan JOSM untuk simulasi lingkup jaringan VANETs, implementasi modifikasi fungsi periode perbaruan *node* pada protokol perutean DSDV, implementasi simulasi jaringan pada NS2, serta implementasi metode analisis untuk evaluasi sistem.
5. Bab V. Uji Coba dan Evaluasi
   Bab ini memuat hasil uji coba dari implementasi modifikasi fungsi periode perbaruan *node* pada protokol perutean DSDV di dalam lingkup jaringan VANETs terhadap skenario mobilitas *grid* dan *real* yang telah dibuat dengan menggunakan simulator jaringan NS2. Hasil uji coba berupa *trace file* dengan ekstensi .tr akan dievaluasi dengan parameter analisis *Packet Delivery Ratio* (PDR), *End to End Delay / One Way Delay* (E2E) dan *Routing Overhead* (RO)

menggunakan kode *script* AWK yang telah dibuat dan akan dibandingkan dengan hasil evaluasi dari protokol perutean DSDV murni.

6. Bab VI. Kesimpulan dan Saran
   Bab ini menjelaskan kesimpulan dari hasil uji coba dari implementasi sistem dan saran-saran terkait pengembangan sistem lebih lanjut.
7. Daftar Pustaka
   Daftar Pustaka memuat daftar referensi yang digunakan untuk mengembangkan Tugas Akhir ini.
8. Lampiran
   Lampiran memuat kode sumber program secara keseluruhan dari Tugas Akhir ini.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai teori-teori dasar dan alat bantu dalam pembuatan tugas akhir. Pembahasan ini bertujuan untuk menjelaskan gambaran umum dari MANETs, VANETs, protokol perutean DSDV, simulator jaringan NS2, SUMO, OpenStreetMap, JOSM, dan AWK.

2.1 *Mobile Ad hoc Network* (MANET)

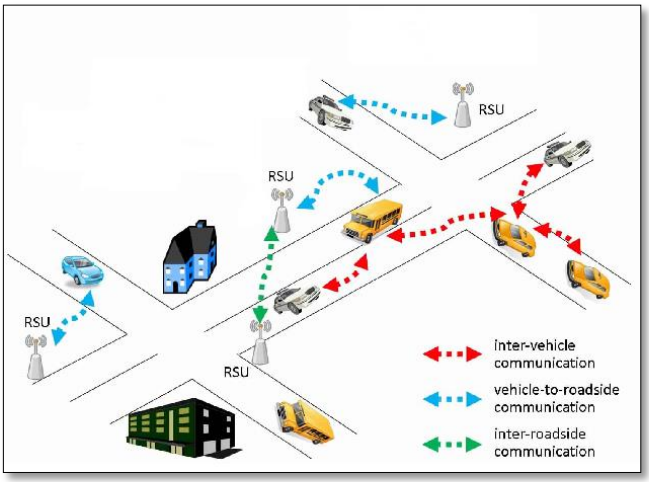
MANET adalah jaringan yang dapat secara dinamis dibentuk untuk bertukar informasi oleh *node* atau *router* nirkabel. *Node* jaringan mungkin dapat berupa laptop, ponsel, *Personal Digital Assistant* (PDA), pemutar MP3, kamera digital, dan lain-lain. Pada MANET tidak akan ditemukan infrastruktur jaringan dan administrasi terpusat yang telah ada sebelumnya. Dengan kata lain, *router* bebas bergerak secara acak dan mengatur diri mereka sendiri. Hal ini menyebabkan topologi nirkabel pada MANET sering sewaktu-waktu berubah dan tidak terduga. Jadi, jaringan semacam ini mungkin beroperasi dalam sistem yang berdiri sendiri maupun yang tersambung dengan jaringan internet. Sebuah MANET dapat dikatakan sebagai jaringan yang memiliki banyak node yang bebas atau otonom, dan biasanya terdiri dari perangkat *mobile* atau jaringan *mobile* lainnya yang dapat beroperasi tanpa administrasi jaringan *top-down* yang rumit [3].

2.2 *Vehicular Ad hoc Network* (VANET)

Jaringan MANET yang bergerak pada kendaraan digolongkan sebagai jaringan VANET. *Node* jaringan yang digunakan untuk membuat jaringan seluler berada pada kendaraan yang bergerak. Setiap kendaraan dalam jaringan VANET berperan sebagai *router*, menghubungkan antar kendaraan untuk saling terhubung dan berkomunikasi, membuat jaringan dengan

jangkauan yang cukup luas [3]. Komukasi ini bertujuan untuk menyediakan transportasi yang efisien dan aman. VANET menjadi lebih menantang jika dibandingkan dengan MANET, karena VANET memiliki tingkat mobilitas *node* yang tinggi dan perubahan topologi yang cepat.

Setiap *node* kendaraan dalam jaringan VANET dilengkapi dengan sebuah unit antarmuka jaringan yang disebut *on-board unit* (OBU), yang tersusun dari modul komunikasi, modul pemrosesan, dan unit *display*. Dengan bantuan modul komunikasi pada OBU, setiap *node* kendaraan mampu berkomunikasi dengan node kendaraan (OBU) disekitarnya. Sistem kendaraan mungkin tersusun dari *road side unit* (RSU) sebagai tambahan dari OBU, merupakan alat antarmuka jaringan yang terletak pada kendaraan yang bergerak dan terhubung dengan keduanya baik dalam mode ad-hoc dan mode infrastruktur. Jaringan kendaraan memiliki 2 tipe utama komunikasi, yakni komunikasi antar kendaraan (OBU ke OBU) dan komunikasi antara kendaraan dengan RSU (OBU ke RSU) [13].



Gambar 2.1 Tipe-Tipe Komunikasi di dalam Lingkup Jaringan VANETs [7]

2.3 *Destination Sequenced Distance Vector* (DSDV)

DSDV merupakan protokol perutean yang menggunakan algoritma Bellman-Ford untuk menghitung lintasannya. *Metric cost* yang digunakan adalah *hop count*, yang merupakan jumlah *hop* yang dibutuhkan paket untuk mencapai destinasinya. DSDV merupakan protokol proaktif, sehingga mampu menjaga *routing table* dengan entri-entri untuk semua *node* di dalam jaringan, tidak hanya untuk *node* tetangga. Setiap perubahan akan disebarluaskan melalui mekanisme perbaruan berkala dan dengan *trigger* yang digunakan pada DSDV. Berdasar pada perbaruan-perbaruan ini, ada sebuah peluang bahwa dalam jaringan akan mengalami *routing loop*. Untuk menghindari *routing loop* setiap perbaruan dari node akan disertakan *sequence number*.

Sequence number dari setiap *node* dipilih secara independen tetapi tetap harus bertambah setiap kali ada perbaruan berkala dari sebuah *node*. Untuk perbaruan normal, *sequence number* yang digunakan adalah bilangan genap, karena setiap perbaruan berkala maka *sequence number* pada *node* akan bertambah 2 dan meneruskan perbaruan ke *routing message* [14].

Routing update dilakukan dengan dua cara, yakni *full dump dan incremental*. *Full dump* akan mengharuskan seluruh tabel perutean dikirim ke tetangga, jarang digunakan ketika tidak terjadi pergerakan dari *node*, cocok digunakan ketika lebih sering terjadi perubahan jaringan. *Incremental* mengirimkan entri-entri yang memerlukan perubahan, lebih sering ditransmisikan, cocok digunakan ketika jaringan relatif stabil [15].

2.4 *Network Simulator Version 2* (NS2)

Network Simulator Version 2, atau lebih akrab disebut dengan NS2, telah ada sejak 1989 dan menjadi populer di dalam lingkup studi jaringan karena sifatnya yang fleksibel dan modular. NS2 merupakan alat simulasi yang telah terbukti berguna dalam studi mengenai sifat-sifat dinamis dari jaringan-jaringan

komunikasi. NS2 dapat digunakan sebagai simulasi fungsi dan protokol (seperti algoritma perutean, TCP, UDP) dari jaringan kabel maupun nirkabel. Jadi secara umum, NS2 memberikan cara untuk menentukan suatu protokol jaringan dan menirukan perilaku yang sesuai aslinya [4].

NS2 menggunakan bahasa pemrograman C/C++ dan *Object-oriented Tool Command Language* (OTCL). C/CC++ digunakan pada mekanisme internal dan obyek simulasi dalam NS2, misalnya pada protokol perutean. Sedangkan OTCL digunakan sebagai definisi lingkungan atau topologi [4]. Simulator jaringan NS2 akan menghasilkan 2 keluaran berupa 1 *file* .nam dan 1 *file* .tr. *File* .nam merupakan *file* yang dapat menampilkan jejak simulasi jaringan. *File* .tr merupakan *file* simulasi jaringan yang dapat digunakan untuk mengukur performa protokol perutean DSDV murni dan DSDV hasil modifikasi.

2.4.1 Instalasi NS2

Pada dasarnya instalasi dan penggunakan NS2 juga dapat dilakukan pada sistem operasi Windows, namun untuk Tugas Akhir ini NS2 digunakan pada lingkup sistem operasi Linux Ubuntu. Langkah-langkah instalasi NS2 pada sistem operasi Linux Ubuntu adalah sebagai berikut:

1. Memasang beberapa *dependency packages* yang dibutuhkan sebelum memulai instalasi NS2. Berikut ini adalah perintah untuk memasang *dependency packages*:

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

   Salah satu dependency yang disebutkan adalah *compiler* GCC-4.3 yang mana sudah tidak tersedia saat ini, maka dari itu pastikan GCC-4.4 telah terpasang agar NS2 dapat digunakan. Versi GCC-4.4 merupakan versi paling lama

yang dapat digunakan. Berikut ini adalah perintah untuk memasang *compiler* GCC-4.4:

```
sudo apt-get install gcc-4.4
```

2. Mengunduh dan mengekstrak simulator jaringan NS 2.35. Berikut ini adalah perintah untuk mengunduh NS 2.35:

```
wget sourceforge.net/projects/nsnam/files/
latest/download -O ns.tar.gz
```

Berikut ini adalah perintah untuk mengekstrak NS 2.35:

```
tar -xvzf ns.tar.gz
```

3. Mengubah baris kode ke-137 pada *file* ls.h yang ada di dalam direktori linkstate NS 2.35.
Berikut ini adalah perintah untuk masuk ke dalam direktori linkstate NS 2.35:

```
cd ~ns-allinone-2.35/ns-2.35/linkstate
```

Pada *file* ls.h, ubah baris ke-137 menjadi sebagai berikut:

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

4. Memulai instalasi NS2.
Berikut ini adalah perintah untuk memulai instalasi NS2:

```
sudo su cd ~/ns-allinone-2.35/ ./install
```

2.4.2 *Trace File*

Trace file merupakan *file* simulasi jaringan yang dihasilkan dari simulator jaringan NS2. *Trace file* memberikan informasi yang lengkap mengenai alur pengiriman paket data. Pada Tugas Akhir ini, *trace file* digunakan untuk menganalisis performa protokol perutean DSDV yang disimulasikan.

Tabel 2.1 Penjelasan Isi Trace File NS2

Kolom	Penjelasan	Isi
1	*Event*	s: *sent* r: *received* f: *forwarded* D: *dropped*
2	*Time*	Waktu terjadinya *event*
3	*ID Node*	_x_: *identifier* dari *node* yang melakukan *event* (mulai 0 hingga banyak *node*)
4	*Layer*	*Network* layer tempat terjadinya *event* AGT: *application* RTR: *routing* LL: *link layer* IFQ: *packet queue* MAC: MAC PHY: *physical*
5	*Flag*	---: Tidak ada
6	*Sequence Number*	*Sequence number* atau nomor urut paket
7	Tipe Paket	*message*: paket *routing* DSDV cbr: berkas paket CBR (*Constant Bit Rate*) RTS: *Request To Send* MAC 802.11 CTS: *Clear To Send* 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 paket c: Alamat asal paket 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)*

Detail penjelasan isi dari *trace file* ditunjukkan pada Tabel 2.1. Berikut ini adalah beberapa contoh isi dari *trace file*:

```
s 0.556838879 _28_ AGT --- 11 cbr 512 [0 0 0
0] ------- [28:0 29:0 32 0] [0] 0 0

r 72.042486122 _29_ AGT --- 633 cbr 532 [13a
1d 4 800] ------- [28:0 29:0 27 29] [71] 6 0

s 0.522406133 _17_ RTR --- 10 message 32 [0
ffffffff 16 800] ------- [22:255 -1:255 32 0]
```

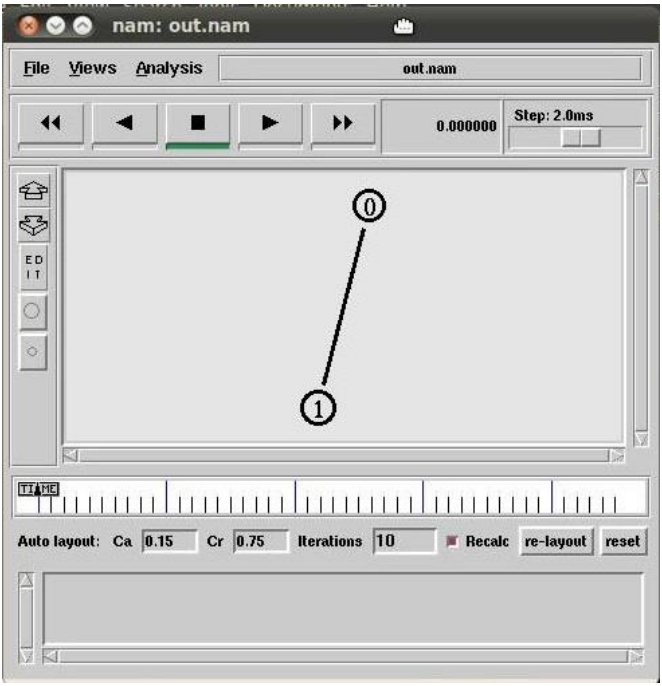
Gambar 2.2 Contoh Isi Dari *Trace File*

2.4.3 *Network Animator* (NAM) *File*

NAM *file* merupakan alat animasi berbasis bahasa Tcl/TK untuk memvisualisasikan jejak setiap *event* dalam suatu simulasi jaringan dan data jejak paket secara nyata. Teori perancangan di balik pembuatan NAM *file* adalah untuk membuat sebuah animator yang mampu membaca *dataset* animasi yang besar dan *extensible* sehingga dapat digunakan pada situasi visualisasi jaringan yang berbeda. Dengan keharusan tersebut, NAM *file* dirancang untuk membaca perintah-perintah *event* animasi sederhana dari suatu *trace file* yang besar. Untuk mengatasi *dataset* animasi yang besar, memori akan menyimpan jumlah informasi yang minimum. Perintah-perintah *event* disimpan di dalam *file* dan dibaca ulang dari *file* ketika dibutuhkan [16].
Berikut ini adalah perintah untuk menjalankan NAM *file*:

```
nam out.nam
```

Setelah perintah di atas dijalankan maka akan terbuka jendela animasi simulasi jaringan seperti pada Gambar 2.2.



Gambar 2.3 Contoh Tampilan *File* NAM

2.5 *Simulation of Urban Mobility* (SUMO)

SUMO, bekerja dibawah lisensi *Eclipse Public Lincense V2*, merupakan program *open source* yang bersifat mikroskopik, artinya setiap kendaraan dimodelkan secara eksplisit, memiliki rutenya sendiri, dan bergerak secara individu dalam jaringan, serta bersifat berkelanjutan untuk membuat simulasi lalu lintas jalan dan dirancang untuk menangani jaringan jalan yang besar [8]. Dengan berbagai fitur yang tersedia, SUMO memungkinkan pengguna untuk dapat membangun simulasi kendaraan di dunia nyata yang bergerak di dalam jaringan VANET.

Mobilitas data lalu lintas dihasilkan dari SUMO *trace exporter* yang akan diekspor ke simulator NS2 sebagai simulator

jaringan untuk menganalisis kinerja VANET. Dengan menggunakan *commands* pada SUMO seperti *netconvert* dan *polyconvert*, data Open Street Map (OSM) dapat dikonversikan menjadi *configuration file* (cfg). File konfigurasi ini digunakan sebagai tempat penyimpanan informasi dan pengaturan. *Trace files* juga dapat dihasilkan dari SUMO, yang mana menjadi langkah vital dalam pembuatan skenario jaringan untuk simulator NS2.

Gambar 2.4 Logo SUMO [8]

Langkah-langkah untuk melakukan instalasi SUMO pada Linux Ubuntu adalah sebagai berikut [8]:

1. Memasang beberapa *tools* dan *libraries* yang dibutuhkan terlebih dahulu sebelum melakukan instalasi SUMO. Seperti clang++, cmake, Xerces-C, Fox Toolkit, Proj, GDAL, serta beberapa *library* opsional.

2. Mengunduh dan mengekstrak *source code* SUMO
   Berikut ini adalah perintah untuk mengunduh SUMO dan menyimpannya pada direktori home:

```
wget  sourceforge.net/projects/sumo/files/
sumo/version1.1.0/sumo-src1.1.0.tar.gz/
download -O sumo.tar.gz
```

Berikut ini adalah perintah untuk mengekstrak SUMO:

```
tar -xvzf sumo.tar.gz
```

3. Melakukan *repository checkout* dengan github SUMO *Eclipse*.
   Berikut ini adalah perintah untuk *repository checkout*:

```
cd sumo-src-1.1.0/
git clone –recursive
github.com/eclipse/sumo
cd sumo
git fetch origin
refs/replace/*:refs/replace/*
```

4. Mendefinisikan variabel SUMO_HOME sesuai dengan path penyimpanan *source code* SUMO.
   Berikut ini perintah untuk mendefinisikan variabel SUMO_HOME:

```
export SUMO_HOME=”/home/<user>/sumo-src-
1.1.0/”
```

5. Membangun SUMO *binaries* dengan `cmake`. Perlu diketahui, untuk membangun SUMO *binaries* dapat dilakukan dengan cara lain, yakni menggunakan `autotools` (legacy) atau `clang`. Tugas Akhir ini me *build* SUMO *binaries* dengan perintah cmake.
   Berikut ini adalah perintah untuk membangun SUMO *binaries* dengan `cmake`, lakukan di direktori `root` SUMO:

```
mkdir build/cmake-build
cd build/cmake-build
```

Menjalankan perintah `cmake` untuk *build* SUMO versi *full*.

Berikut ini adalah perintah untuk *run* `cmake`:

```
cmake ../..
```

Untuk menjadikan *build* menjadi lebih cepat lakukan perintah berikut:

```
make -j 8
```

6. Memulai instalasi SUMO *binaries*. Masuk ke direktori root sumo-src-1.1.0
   Berikut ini adalah perintah untuk memulai instalasi SUMO:

```
make
sudo make install
```

2.6 OpenStreetMap (OSM)

Gambar 2.5 Logo OpenStreetMap [9]

OpenStreetMap adalah proyek besar yang dibangun untuk membuat peta dunia berbasis web. OpenStreetMap sangat populer dikarenakan bersifat gratis dan terbuka. Siapapun akan memiliki

hak untuk menggunakan OpenStreetMap sesuai dengan kebutuhannya, misalnya penyelesaian isu-isu lingkungan, ekonomi, sosial, dan manajemen krisis, baik pada perangkat *mobile*, *website*, dan *hardware device* lainnya. OpenStreetMap sendiri dibangun oleh para sukarelawan dari komunitas pembuat peta yang berkolaborasi membuat dan mengelola data jalan raya dari seluruh dunia. Seluruh data dikumpulkan dan dipadukan dalam *website* OpenStreetMap yang dapat diakses kapanpun secara *online*. OpenStreetMap dapat diakses melalui https://www.openstreetmap.org/export.

Dibawah lisensi *Open Data Commons Open Database License* 1.0, seluruh kontributor OpenStreetMap dapat memiliki, memodifikasi, dan membagikan peta secara luas. Dengan sifat yang gratis dan terbuka ini, menjadikan OpenStreetMap sebagai pilihan masyarakat, pemerintah, peneliti, akademisi, inovator, maupun pihak lainnya. OpenStreetMap dapat digunakan secara bebas dan kemudian diredistribusikan kembali [9].

Tugas Akhir ini akan memanfaatkan OpenStreetMap sebagai acuan untuk mendapatkan peta sebenarnya guna simulasi lintasan jalan raya. Peta tersebut akan digunakan untuk pembuatan skenario *real.*

2.7 Java OpenStreetMap Editor (JOSM)

Java OpenStreetMap Editor (JOSM) adalah aplikasi yang dikembangkan oleh Immanuel Scholz pada akhir tahun 2005 dan saat ini tengah dikelola oleh Dirk Stocker. JOSM merupakan aplikasi *open source* dan berada dibawah lisensi GPL. Aplikasi JOSM digunakan untuk menyunting dan merapikan data yang didapatkan dari OpenStreetMap, cocok digunakan untuk yang sudah mahir menyunting data peta digital [10].

Aplikasi JOSM dapat diunduh pada alamat web https://josm.openstreetmap.de. Untuk dapat menggunakan JOSM pada Linux Ubuntu, pastikan bahwa versi dari Java pada perangkat yang digunakan minimal adalah Java versi 8. Sebelum

melakukan instalasi JOSM pastikan *library* yang dibutuhkan telah terpasang.

Gambar 2.6 Logo JOSM [10]

Berikut ini adalah perintah untuk mengecek versi Java:

```
java -version
```

Setelah mengunduh *file* `josm-tested.jar` dari halaman pengunduhan, *file* dapat dijalankan melalui perintah pada terminal. Berikut ini adalah perintah untuk menjalankan JOSM:

```
java –jar josm-tested.jar
```

Pada Tugas Akhir ini, penulis menggunakan data yang telah diambil dari OpenStreetMap lalu disunting. Penyuntingan dilakukan dengan menghilangkan dan juga menyambungkan jalan yang ada, serta menghilangkan bangunan-bangunan yang ada di peta. Penyuntingan dapat dilakukan secara *offline* tanpa memerlukan koneksi internet.

2.8 AWK

AWK merupakan sebuah bahasa pemrograman yang dirancang untuk pemrosesan teks dan biasanya digunakan untuk ekstraksi data dan alat pelaporan. AWK merupakan fitur standar dari kebanyakan sistem operasi Unix-like. Sejarah AWK bermula dari pengembangannya pada tahun 1970-an di *Bell's Telephone Laboratories*, oleh Alfred Aho, Peter Weinberger, dan Brian Kernighan. Penamaan AWK pun diambil dari huruf pertama nama belakang ketiga penciptanya [11].

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

Pada Tugas Akhir ini, AWK digunakan untuk mengekstrak data dari *trace file* keluaran simulator jaringan NS2 guna menghasilkan nilai performa *packet delivery* ratio (PDR), *end to end delay* (E2E), dan *routing overhead* (RO) dari protokol perutean DSDV murni dan DSDV hasil modifikasi.

BAB III
PERANCANGAN SISTEM

Pada bab ini akan dijelaskan mengenai perancangan implementasi sistem yang akan dibuat pada tugas akhir ini. Bagian yang dibahas pada bab ini berawal dari deskripsi umum, perancangan skenario, perancangan simulasi, dan perancangan metode analisis.

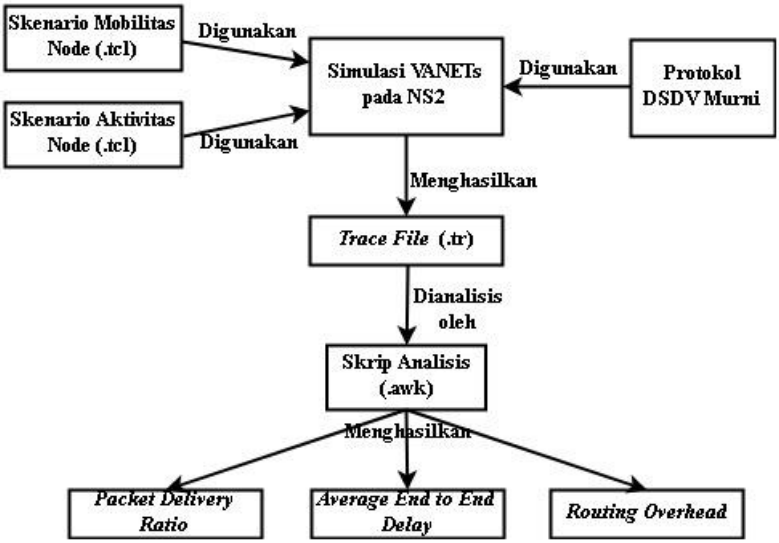
3.1 Deskripsi Umum

Tugas Akhir ini akan mengimplementasikan protokol perutean *Destinaation Sequence Distance Vector* (DSDV) yang dimodifikasi dengan menambahkan fungsi yang berkenaan dengan periode perbaruan setiap *node* berdasarkan faktor kecepatan kendaraan. Protokol perutean yang diajukan pada tugas akhir ini merupakan modifikasi dari protokol perutean DSDV dengan menambahkan batas/*threshold*. Modifikasi periode perbaruan *node* akan dilakukan jika dan hanya jika kendaraan memiliki kecepatan yang berada diatas threshold. Untuk kendaraan yang memiliki kecepatan dibawah *threshold* maka periode perbaruan *node* tidak akan berubah.

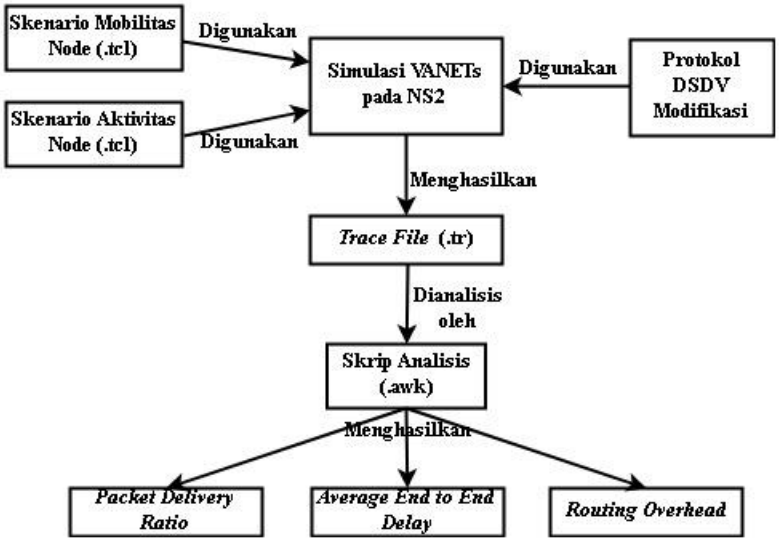
Sebelum melakukan modifikasi periode perbaruan setiap *node*, didefinisikan suatu variabel yang akan menyimpan nilai kecepatan setiap kendaraan. Nilai kecepatan tersebut akan digunakan sebagai penentu periode perbaruan rute setiap kendaraan. Semakin besar nilai kecepatan kendaraan, maka semakin kecil interval periode perbaruan *node*, sehingga semakin sering dilakukan perbaruan rute.

Modifikasi protokol perutean DSDV ini akan disimulasikan menggunakan simulator jaringan *Network Simulator Version 2* (NS2) dengan skenario yang berada di dalam lingkup jaringan *Vehicular Ad hoc Networks* (VANETs dan dibuat menggunakan *tools* pada generator skenario *Simulation of Urban Mobility* (SUMO). Simulasi yang dilakukan menghasilkan

trace file, kemudian dianalisis menggunakan AWK untuk mendapatkan performa protokol perutean DSDV yang telah dimodifikasi. Performa dinilai berdasarkan: *packet delivery ratio* (PDR), *end-to-end delay* (E2E), dan *routing overhead* (RO). Lalu dilakukan pembandingan dengan nilai performa protokol perutean DSDV murni. Diagram perancangan simulasi dengan protokol perutean DSDV murni dan DSDV hasil modifikasi dapat dilihat pada Gambar 3.1 dan Gambar 3.2.



Gambar 3.1 Diagram Perancangan Simulasi dengan *Routing Protocol* DSDV Murni



Gambar 3.2 Diagram Perancangan Simulasi dengan *Routing Protocol* DSDV Modifikasi

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

Tabel 3.1 Daftar Istilah

No.	Istilah	Penjelasan
1	DSDV	*Destination Sequence Distance Vector*
2	PDR	*Packet Delivery Ratio*
3	E2E	*Average End-to-End Delay*
4	RO	*Routing Overhead*
5	NS2	*Network Simulator Version 2*
6	SUMO	*Simulation of Urban Mobility*
7	Threshold	*Batas kecepatan kendaraan*
8	OSM	Peta keluaran OpenStreetMap (.osm)

3.2 Perancangan Skenario Mobilitas

Perancangan skenario mobilitas terdiri dari dua bagian, yakni perancangan skenario mobilitas *grid* dan perancangan skenario mobilitas *real*. Pada tahap perancangan ini mula-mula akan dibuat area simulasi dan pergerakan *node*. Perancangan skenario mobilitas *grid* menggunakan peta jalan yang saling berpotongan berbentuk petak-petak. Peta *grid* digunakan sebagai simulasi awal VANETs karena memberikan hasil yang lebih seimbang dan stabil. Peta *grid* dibuat langsung menggunakan generator skenario SUMO dengan menentukan panjang dan lebar area serta jumlah petak mendatar dan menurun. Peta *real* dibuat dengan menggunakan peta keluaran OSM sebagai area dan di*convert* sehingga dapat digunakan dalam generator skenario SUMO.

3.2.1 Perancangan Skenario Mobilitas Grid

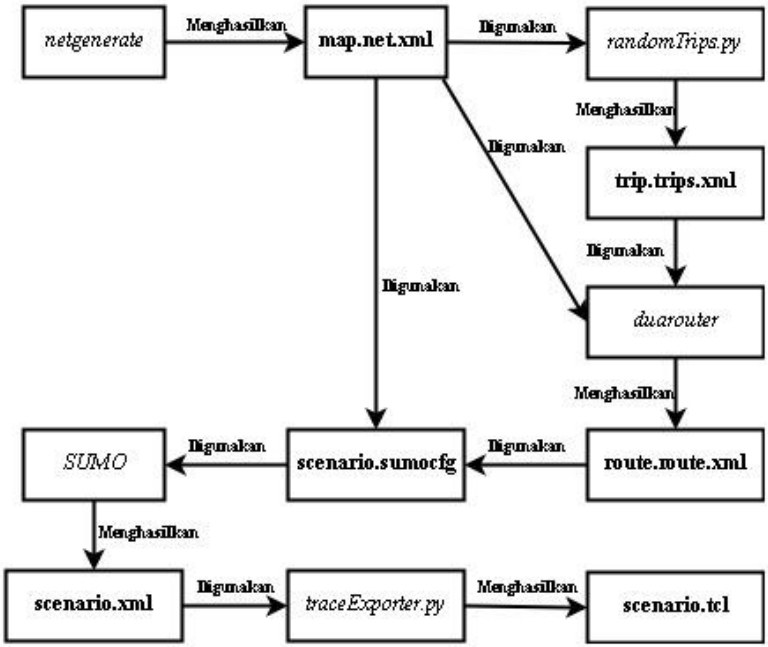
Untuk membuat perancangan skenario mobilitas grid perlu ditentukan beberapa parameter. Perancangan dimulai dengan menentukan luas area yang akan digunakan untuk simulasi. Kemudian menentukan panjang dan lebar area simulasi. Setelah itu untuk mendefinisikan panjang sisi setiap petak, menentukan jumlah titik persimpangan atau perpotongan jalan yang dibutuhkan, karena generator skenario SUMO akan membaca berapa petak yang terbentuk dengan definisi jumlah titik yang ditentukan. Misalkan luas area simulasi yang diinginkan adalah 1000000m^2, maka dapat ditentukan panjang area 1000m dan lebar area 1000m.Untuk dapat membuat 10 petak mendatar dan 10 petak menurun, makan jumlah titik persimpangan yang harus dibuat adalah 11 titik. Untuk luas area tersebut dengan banyak 10 petak mendatar dan 10 petak menurun, maka panjang sisi setiap petak adalah 100m.

Area simulasi *grid* yang telah ditentukan setiap parameternya dapat langsung dibuat dengan menggunakan generator skenario SUMO. *Tools* SUMO yang digunakan yaitu *netgenerate*. Dalam pembuatan skenario untuk Tugas Akhir ini, selain menentukan jumlah titik persimpangan, hal lain yang harus ditentukan adalah jumlah kendaraan dan kecepatan maksimal kendaraan dalam suatu skenario.

Parameter perancangan skenario mobilitas grid yang telah ditentukan kemudian dibuat dengan mengunakan fitur SUMO yaitu *netgenerate* menghasilkan *file* .net.xml. Untuk membuat pergerakan *node* di atas peta *grid* yang telah dibuat tersebut, dapat menggunakan *tools* dari generator SUMO yakni *randomTrips* dan *duarouter*.

Skenario mobilitas *grid* dihasilkan dengan menggabungkan *file* peta *grid* map.net.xml dan *file* pergerakan *node* sroute.route.xml, yang akan menghasilkan *file* dengan ekstensi .xml. Lalu *file* .xml dikonversi dengan fitur *trace*Exporter menjadi *file* .tcl agar dapat dijalankan di NS2. Diagram alur pembuatan skenario *grid* dapat dilihat pada *Gambar 3.3*.

Pada Tugas Akhir ini, peta *grid* yang akan digunakan adalah area dengan luas 1000000m^2, panjang 1000m, dan lebar 1000m. Jumlah petak yang dibuat adalah 10 x 10 petak, maka panajng sisi satu petak adalah 100m. Akan dibuat beberapa macam skenario *grid* berdasarkan jumlah kendaraan, yakni 30, 40, dan 50 *node*. Dan beberapa jenis kecepatan maksimal kendaraan, yakni 25, 30, 35, 40, 45, 50 m/s.



Gambar 3.3 Diagram Alur Pembuatan Skenario Mobilitas *Grid*

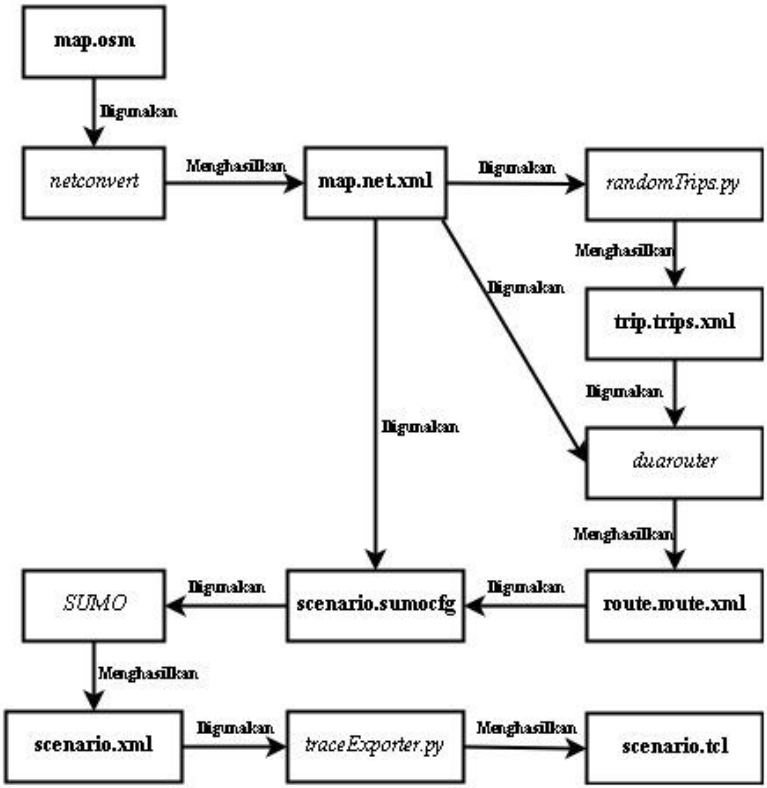
3.2.2 Perancangan Skenario Mobilitas Real

Untuk membuat perancangan skenario *real* perlu ditentukan terlebih dahulu area mana dari peta yang akan dijadikan area simulasi. Peta *real* untuk Tugas Akhir ini akan memanfaatkan peta dunia OpenStreetMap yang merupakan peta berbasis web yang mudah digunakan dan *open source*. Area akan mengambil suatu lokasi yang ada di Kota Surabaya. Area yang sudah terpilih dari web kemudian diunduh menggunakan fitur *export* dari OpenStreetMap. Peta hasil keluaran dari *export* tadi akan tersimpan dengan ekstensi .osm.

Peta *real* dengan ekstensi .osm tersebut belum dapat digunakan pada generator skenario SUMO, peta .osm akan

dikonversi dengan *tools netconvert* terlebih dahulu agar menjadi bentuk *file* peta dengan ekstensi .net.xml yang dapat dibaca oleh generator skenario SUMO. Langkah-langkah perancangan skenario mobilitas *real* selanjutnya sama dengan ketika membuat perancangan skenario mobilitas *grid*, yaitu menggunakan *tools randomTrips* dan *duarouter* untuk membuat pergerakan *node*.

Diagram alur perancangan skenario mobilitas *real* dapat dilihat pada *Gambar 3.4*.



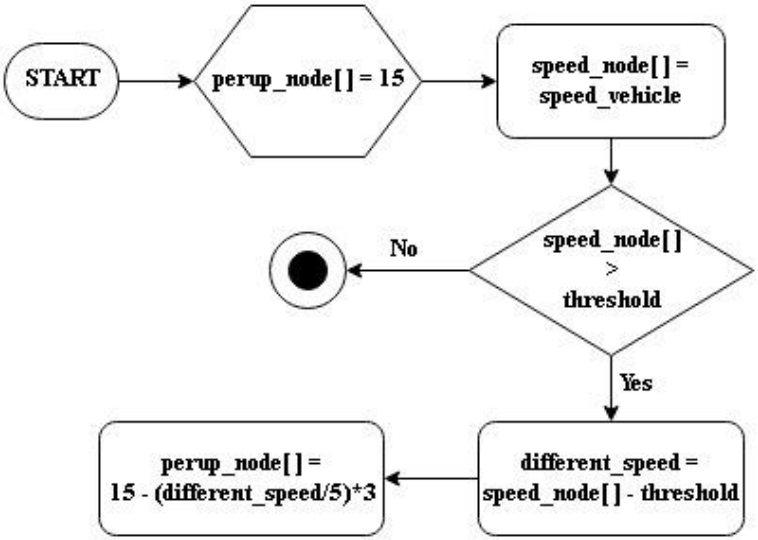
Gambar 3.4 Diagram Alur Pembuatan Skenario Mobilitas *Real*

3.3 Perancangan Modifikasi Periode Perbaruan *Node*

Pada dasarnya metode *update* yang terdapat pada protokol perutean DSDV terdiri dari 2 jenis, yaitu *full dump* dan *incremental*. Metode *update full dump* secara teratur melakukan perbaruan *node* setiap interval waktu (*update period*) 15 detik. Sedangkan metode *update incremental* hanya dilakukan pada kondisi waktu tertentu, misalnya ketika *node* tetangga tidak merespon dalam jangka waktu tertentu.

Modifikasi protokol perutean DSDV pada Tugas Akhir ini akan mengubah mekanisme metode *update full dump* dari DSDV murni. Dalam Tugas Akhir ini, periode perbaruan *node* ditentukan berdasarkan pada kecepatan *node*/kendaraan. Sehingga setiap *node* dengan kecepatan berbeda akan memiliki periode perbaruan *full dump* yang berbeda pula. Tahapan awal modifikasi adalah mendefinisikan suatu variabel yang akan menyimpan nilai kecepatan yang dimiliki oleh setiap *node*. Setelahnya akan dilakukan pengecekan perbandingan nilai kecepatan yang dimiliki suatu *node* dengan nilai *threshold* yang telah didefinisikan. Apabila nilai kecepatan suatu *node* melebihi *threshold*, maka periode perbaruan *full dump node* tersebut akan berubah menyesuaikan dengan besarnya nilai kecepatannya. Setiap kecepatan kendaraan bertambah 5 m/s dari *threshold* maka periode perbaruan *fulldump* akan berkurang 3 detik dari interval waktu *fulldump* DSDV murni. Apabila nilai kecepatan suatu *node* kurang dari atau sama dengan *threshold*, maka metode *update full dump* akan dilakukan setiap interval waktu 15 detik seperti pada DSDV murni.

Flowchart perancangan modifikasi periode perbaruan *node* protokol perutean DSDV pada Tugas Akhir ini dapat dilihat pada Gambar 3.5.



Gambar 3.5 Diagram Alur Perancangan Modifikasi DSDV

Nilai *threshold* pada Tugas Akhir ini dilakukan dengan melakukan *thresholding* untuk variabel peubah kecepatan kendaraan dalam protokol perutean DSDV. Berdasarkan literatur referensi pembuatan Tugas Akhir, *threshold* yang paling tepat adalah 25m/s [1]. Maka untuk membuktikan bahwa kenaikan kecepatan kendaraan akan menyebabkan penurunan performa pada protokol perutean DSDV murni. *Thresholding* dilakukan pada skenario 30 node sebanyak 10 skenario pada tiap-tiap angka kecepatan kendaraan yang akan digunakan sebagai *threshold* yaitu 15, 20, 35, 30, 35, 40 m/s. Langkah selanjutnya adalah mengevaluasi dan menghitung rata-rata nilai performa dari setiap jenis skenario. Hasil penghitungan rata-rata suatu skenario dibandingkan dengan rata-rata dari skenario dengan kecepatan berbeda. Hasil evaluasi *thresholding* yang telah dilakukan membuktikan bahwa pada protokol perutean DSDV murni, apabila kecepatan kendaraan semakin tinggi, akan menyebabkan

penurunan performa. Maka pada Tugas Akhir ini ditentukan nilai *threshold* adalah 25 m/s.
Hasil evaluasi *thresholding* kecepatan pada protokol perutean DSDV murni dapat dilihat pada *Tabel 3.2*.

Tabel 3.2 Evaluasi *Thresholding* Kecepatan *Node*

Kecepatan kendaraan (m/s)	PDR (%)	E2E (ms)	RO (jml paket)
15	61,07	40,13	1.500
20	57,24	31,17	1.499
25	56,71	108,94	1.512
30	55,70	65,74	1.510
35	53,57	32,40	1.511
40	52,55	48,59	1.516

3.3.1 Perancangan Modifikasi Untuk Mengetahui Kecepatan Setiap *Node*

Modifikasi protokol perutean DSDV pada Tugas Akhir ini dilakukan berdasarkan pada kecepatan setiap kendaraan/*node*. Maka diperlukan langkah untuk mengetahui besarnya nilai kecepatan yang dimiliki oleh sebuah *node*. Langkah pengecekan kecepatan sebuah *node* akan dilakukan setiap sebuah *node* melakukan aktivitas, demikian pula dengan perubahan periode perbaruan *node* yang akan dilakukan setiap sebuah *node* melakukan aktivitas. Sehingga data kecepatan dapat selanjutnya menjadi data yang akurat dan sesuai dengan kondisi terbaru. Hal-hal yang dimaksud sebagai aktivitas *node* adalah sebagai berikut: pengiriman paket, penerimaan paket, dan perbaruan rute.

Kecepatan *node* yang telah didapatkan kemudian akan digunakan untuk menentukan selisihnya dengan *threshold*.

Berikut ini adalah pseudocode untuk mengetahui kecepatan setiap *node*:

```
. . .

address = myaddr_
speed = node_->speed
perup_original = perup_

modif_perup(address, speed, perup_original)

. . .
```

Gambar 3.6 Pseudocode Untuk Mengetahui Kecepatan Setiap *Node*

3.3.2 Perancangan Modifikasi Untuk Menghitung Periode Perbaruan *Node*

Modifikasi ini akan mengakibatkan setiap *node* memiliki periode perbaruan *node* yang berbeda-beda. Data periode perbaruan *routing table* setiap *node* akan tersimpan dalam sebuah variabel *array*. Definisi awal periode perbaruan *node* adalah 15, dikarenakan metode *update full dump* pada protokol perutean DSDV murni adalah setiap interval waktu 15 detik. Perubahan nilai periode perbaruan *node* dilakukan apabila kecepatan suatu *node* lebih besar dari nilai *threshold*. Untuk suatu *node* dengan kecepatan di bawah *threshold*, nilai periode perbaruan *node* akan tetap seperti pada DSDV murni.
Berikut ini adalah pseudocode untuk menghitung periode perbaruan *node*:

```
. . .

if speed > threshold :
      speed_difference = speed - 25
      perup_node[address]      =      15      -
(floor(speed_difference/5)*3)

else :
      perup_node[address] = perup_original

. . .
```

Gambar 3.7 Pseudocode Untuk Menghitung Periode Perbaruan *Node*

3.4 Perancangan Simulasi pada NS2

Simulasi VANETs pada NS2 dilakukan dengan menggabungkan *file* skenario yang telah dibuat menggunakan SUMO dan *file* skrip .tcl yang berisi konfigurasi lingkungan simulasi. Konfigurasi lingkungan simulasi VANETs pada NS2 yang dilakukan dalam Tugas Akhir ini dapat dilihat pada *Tabel 3.3*.

Modifikasi dilakukan pada *file* dsdv.h dan dsdv.cc yang ada di dalam direktori dsdv pada NS2. Modifikasi berupa penambahan beberapa baris kode. Pada *file* dsdv.h dilakukan penambahan deklarasi variabel *array* untuk menyimpan data nilai periode perbaruan *node* dan kecepatan *node*. Pada *file* dsdv.cc ditambahkan sebuah fungsi untuk melakukan pengecekan dengan *threshold* dan untuk menghitung periode perbaruan *node* yang baru sesuai perancangan modifikasi.

Tabel 3.3 Parameter Lingkungan Simulasi dengan Skenario

No.	Parameter	Spesifikasi
1.	*Network Simulator*	NS2 versi 2.35
2.	*Routing Protocol*	DSDV
3.	Waktu Simulasi	200 detik
4.	Area Simulasi	1000m x 1000m
5.	Radius Transmisi	400m
6.	Jumlah Kendaraan	30, 40, 50
7.	Kecepatan Kendaraan	25, 30, 35, 40, 45, 50 m/s
8.	Tipe Koneksi	UDP
9.	Agen Pengirim	*Constant Bit Rate* (CBR)
10.	Ukuran Paket	512 bytes
11.	Kecepatan Generasi Paket	1 paket per detik
12.	*Protocol* MAC	IEEE 802.11
13.	*Source/Destination*	Statis
14.	Tipe Kanal	*Wireless Channel*
15.	Tipe Antenna	OmniAntenna
16.	Tipe *Interface Queue*	Droptail/PriQueue
17.	Tipe *Trace*	*Old Wireless Format Trace*

3.5 Perancangan Metode Analisis

Pada Tugas Akhir ini terdapat beberapa parameter yang akan dianalisis yaitu: *Packet Delivery Ratio* (PDR), *End-to-End Delay* (E2E), dan *Routing Overhead* (RO). Parameter-parameter tersebut digunakan untuk menentukan performa dari protokol perutean DSDV murni dan DSDV hasil modifikasi. Penjelasan untuk masing-masing parameter analisis adalah sebagai berikut:

3.5.1 *Packet Delivery Ratio* (PDR)

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

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

Keterangan:

received : Jumlah paket data yang diterima
sent : Jumlah paket data yang dikirim

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

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

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

$$E2E = \frac{\sum_{sent}^{i=0} (t_{received[i]} - t_{sent[i]}}{sent} \quad (3.2)$$

Keterangan:

i : Urutan/ id paket ke-i
$t_{received}$: Waktu paket diterima (ms)
t_{sent} : Waktu paket dikirim (ms)
sent : Jumlah paket data yang dikirim

3.5.3 *Routing Overhead* (RO)

Routing Overhead adalah jumlah paket kontrol *routing* yang ditransmisikan per data paket yang terkirim ke *node* tujuan selama simulasi berlangsung. RO dihitung berdasarkan jumlah paket *routing* yang ditransmisikan. Baris yang mengandng *routing overhead* pada *trace file* ditandai dengan paket yang bertipe *sent* (s) / *forwarded* (f) dan terdapat header paket dari protokol perutean DSDV. Persamaan untuk menghitung RO dapat dilihat pada persamaan 3.3.

$$RO = sentRTR + forwardedRTR \quad (3.3)$$

Keterangan:

sentRTR : Jumlah paket *routing* yang dikirim
forwardedRTR : Jumlah paket *routing* yang diteruskan

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari perancangan sistem yang telah dijelaskan pada bab sebelumnya. Bab ini meliputi implementasi skenario mobilitas, implementasi modifikasi fungsi periode perbaruan *node*, implementasi simulasi pada NS2, dan implementasi metode analisis.

4.1 Implementasi Skenario Mobilitas

Implementasi skenario mobilitas pada lingkungan VANETs dibagi menjadi dua, yaitu implementasi skenario mobilitas *grid* dan implementasi skenario mobilitas *real*. Skenario *grid* menggunakan peta jalan yang saling berpotongan berbentuk petak-petak. Peta *grid* dapat langsung digunakan pada SUMO. Skenario *real* menggunakan peta keluaran OSM sebagai area, peta tersebut di*convert* menjadi format yang dapat dibaca oleh generator skenario SUMO sehingga selanjutnya dapat digunakan.

4.1.1 Implementasi Skenario Mobilitas *Grid*

Pada Tugas Akhir ini, skenario akan dibuat berdasar variasi jumlah kendaraan/*node* dan variabel peubah kecepatan maksimal kendaraan. Mengenai penjabaran variasi jumlah *node* dan kecepatan maksimal kendaraan telah dijelaskan secara lengkap pada *heading* 3.2.1.

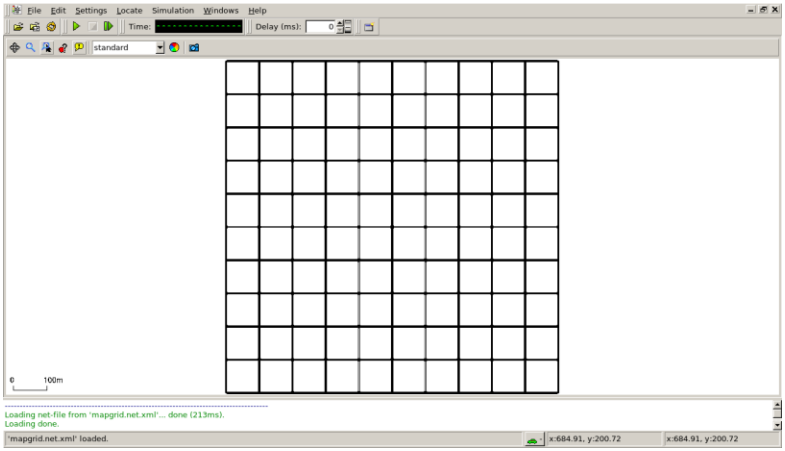
Dalam mengimplementasikan skenario *grid*, SUMO menyediakan *tools* untuk membuat peta *grid* yaitu *netgenerate*. Pada Tugas Akhir ini dibuat peta *grid* dengan luas 1000 m x 1000 m yang terdiri dari titik persimpangan antara jalan vertikal dan jalan horizontal sebanyak 11 titik x 11 titik. Dengan jumlah titik persimpangan sebanyak 11 titik tersebut, maka terbentuk 10 buah petak. Sehingga untuk mencapai luas area sebesar 1000 m x 1000 m dibutuhkan luas per petak sebesar 100 m x 100 m. Perintah

netgenerate untuk membuat peta tersebut dengan kecepatan *default* kendaraan sebesar 25 m/s dapat dilihat pada *Gambar 4.1*.

```
netgenerate    --grid    --grid.number=11    --
grid.length=100      --default.speed=25      --
tls.guess=1 --output-file=map.net.xml
```

Gambar 4.1 Perintah Untuk Menjalankan *Netgenerate*

Perintah *netgenerate* pada *Gambar 4.1* menghasilkan sebuah *file* keluaran dengan nama map.net.xml. *File* tersebut dapat dibuka menggunakan *tools sumo-gui*. Contoh peta area *grid* hasil keluaran *netgenerate* dapat dilihat pada Gambar 4.2.



Gambar 4.2 Contoh Hasil Keluaran Perintah *Netgenerate*

Setelah peta terbentuk, maka dilakukan pembuatan *node* dan pergerakan *node* dengan menentukan titik awal dan titik akhir setiap *node* secara acak menggunakan *tools randomTrips*. Perintah penggunaan *tools randomTrips* untuk membuat titik awal dan titik akhir sebanyak 28 *node* dapat dilihat pada *Gambar 4.3*. Hanya dibuat 28 *node* karena dibutuhkan 2 *node* untuk

menjadi *node* sumber dan *node* tujuan. *Node* sumber dan *node* tujuan merupakan *node* statis.

```
python /usr/share/sumo/tools/RandomTrips.py -n
mapgrid.net.xml -e 28 -l --trip-
attributes="departLane=\"best\"
departSpeed=\"max\" departPos=\"random_free\""
-o trips.trips.xml
```

Gambar 4.3 Perintah Untuk Menggunakan *Tools RandomTrips*

Selanjutnya dibuatkan rute yang digunakan kendaraan untuk mencapai tujuan dari *file* hasil sebelumnya menggunakan *tools duarouter*. *Tools duarouter* ini membaca *file* peta hasil generate dan *file* titik awal dan titik akhir *node*. *Tools* ini menghasilkan pergerakan setiap *node* dari titik awal menuju titik akhir yang telah didefinisikan. Secara *default* algoritma yang digunakan untuk membuat rute ini adalah algoritma dijkstra. Perintah penggunaan *tools duarouter* dapat dilihat pada *Gambar 4.4*.

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

Gambar 4.4 Perintah Untuk Menggunakan *Tools Duarouter*

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 guna menggabungkan *file* peta dan rute pergerakan *node*. Isi dari *file* skrip tersebut 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 Skrip dalam *File* .sumocfg

File .sumocfg disimpan di dalam direktori yang sama dengan *file* peta dan *file* pergerakan *node*. Untuk percobaan sebelum dikonversi, *file* .sumocfg dapat dibuka dengan menggunakan *tools sumo-gui*.

Kemudian dari *file* .sumocfg dibuat *file* skenario dalam bentuk *file* berekstensi .xml menggunakan *tools traceExporter* pada SUMO. Berikut ini adalah perintah untuk membuat *file* skenario dengan *traceExporter*:

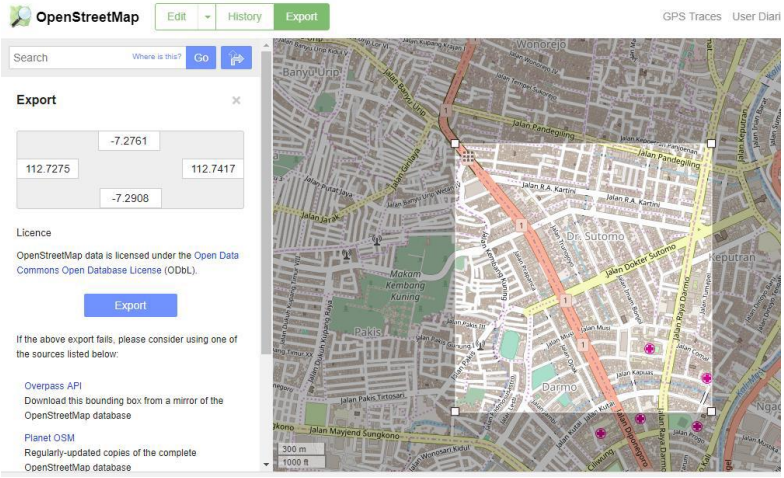
```
python sumo-0.27.1/tools/traceExporter.py --
fcd-input=scenario.xml          --ns2-mobility-
output=scenario.tcl
```

Gambar 4.6 Perintah Untuk Menggunakan *Tools TraceExporter*

4.1.2 Implementasi Skenario Mobilitas *Real*

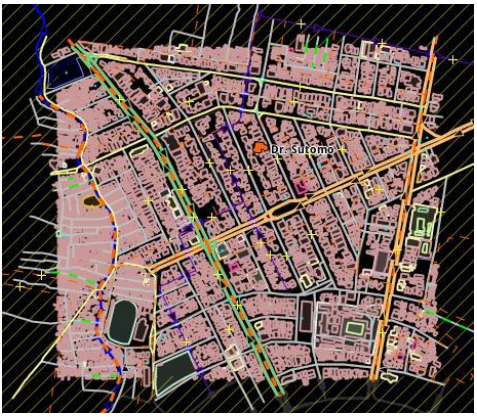
Dalam mengimplementasikan skenario *real*, langkah pertama adalah dengan menentukan area yang akan dijadikan area simulasi. Pada Tugas Akhir ini penulis mengambil area sekitar

Jalan Dr. Soetomo Surabaya. Setelah menentukan area simulasi, ekspor data peta tersebut dari OpenStreetMap seperti yang ditunjukkan pada .

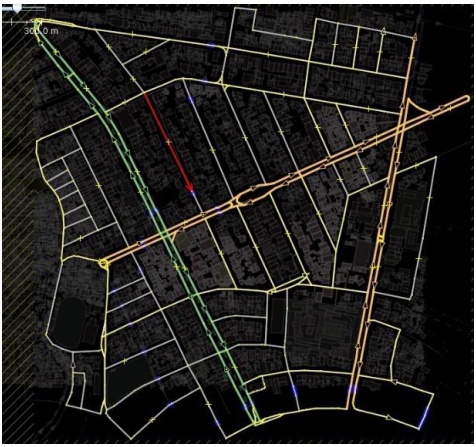


Gambar 4.7 Contoh Pemilihan Area OpenStreetMap

File hasil *export* dari OpenStreetMap tersebut adalah *file* peta dengan ekstensi .osm. *File* .osm tersebut perlu diubah menggunakan JOSM. Pengubahan dilakukan untuk menghilangkan bangunan-bangunan yang terdapat dalam peta. Selain itu juga untuk menghilangkan jalan yang terputus. Gambar *file* .osm sebelum diubah dan setelah diubah dapat dilihat pada .



Gambar 4.8 Peta .osm Sebelum Diedit



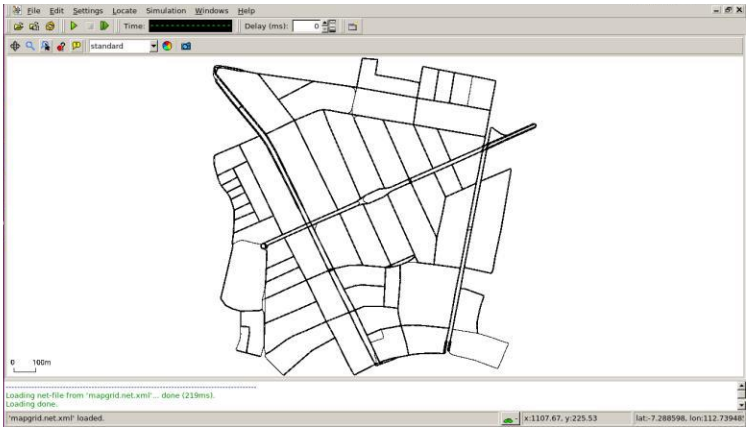
Gambar 4.9 Peta .osm Setelah Diedit

Selanjutnya *file* peta .osm tersebut dikonversi menjadi peta dalam bentuk *file* berekstensi .xml menggunakan *tools netconvert*. Perintah untuk menggunakan *netconvert* dapat dilihat pada *Gambar 4.10*.

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

Gambar 4.10 Perintah Untuk Menggunakan *Tools Netconvert*

Hasil konversi peta dari *file* berekstensi .osm menjadi *file* berekstensi .xml dapat dilihat menggunakan *tools sumo-gui* seperti yang ditunjukkan pada *Gambar 4.11*.



Gambar 4.11 Contoh Hasil Keluaran *Tools Netconvert*

Setelah berhasil menghasilkan peta *real* dengan ekstensi .net.xml, langkah selanjutnya yang harus dilakukan sama seperti ketika pembuatan skenario *grid*, yaitu membuat *node* dan pergerakannya. Hasil keluaran akhirnya juga sama dengan skenario *grid* yaitu *file* dengan ekstensi .tcl.

4.2 Implementasi Modifikasi Periode Perbaruan *Node*

Protokol perutean DSDV merupakan salah satu protokol yang sangat umum digunakan dalam komunikasi *ad hoc*. Protokol perutean DSDV termasuk jenik protokol proaktif. Yang artinya protokol perutean DSDV akan memperbarui *routing table* setiap

interval waktu tertentu. Namun performa protokol perutean DSDV murni ternyata belum optimal dalam pengiriman paket untuk kendaraan/*node* dengan kecepatan yang tinggi atau memiliki pergerakan yang cepat. Ini terjadi karena protokol perutean DSDV murni tidak memperhatikan faktor kecepatan *node* dalam penentuan periode perbaruan *routing table*.

Modifikasi pada Tugas Akhir ini akan mengganti metode perbaruan *full dump* yang ada pada protokol perutean DSDV murni. Periode perbaruan suatu *node* akan berubah tergantung pada kecepatan yang dimiliki *node* tersebut. *Node* dengan kecepatan di atas angka *threshold* akan mengalami perubahan periode perbaruan *node*, sedangkan *node* dengan kecepatan di bawah angka *threshold* tidak akan mengalami perubahan periode perbaruan *node*. Modifikasi ini dimaksudkan untuk meningkatkan performa protokol perutean DSDV.

Implementasi modifikasi pada Tugas akhir ini terbagi menjadi 2 bagian:

1. Implementasi Modifikasi Untuk Mengetahui Kecepatan *Node*.
2. Implementasi Modifikasi Untuk Menghitung Periode Perbaruan *Node*.

Implementasi pada protokol perutean DSDV akan memodifikasi *file* yang ada pada direktori dsdv di dalam NS2 versi 2.35. Setiap melakukan perubahan terhadap *file* didalam folder NS2, harus dilakukan *remake* agar perubahan terbaca oleh program dann dapat dijalankan sebagai simulasi. Langkah-langkah lebih lanjut akan dibahas secara rinci pada subbab di bawah.

4.2.1 Implementasi Modifikasi Untuk Mengetahui Kecepatan Setiap *Node*

Sesuai yang telah dijelaskan dalam subbab perancangan modifikasi untuk mengetahui kecepatan setiap *node* (subbab 0), langkah awal dalam modifikasi ini adalah untuk mendapatkan nilai kecepatan suatu *node* dan nilai periode perbaruan *node* pada

protokol perutean DSDV murni. Di dalam *file* dsdv.h, periode perbaruan sudah dideklarasi secara global dengan nama variabel `perup_` dengan nilai 15 (artinya perbaruan *routing table* secara *full dump* dilakukan setiap 15 detik). Sedangkan nilai kecepatan *node*, tidak terdapat deklarasi secara *default* pada DSDV murni. Nilai kecepatan *node* didapatkan dari *class node*. Di dalam *class node*, nilai kecepatan dapat diketahui dari fungsi `node_->speed()`. Nilai kecepatan tersebut dibutuhkan untuk menghitung nilai periode perbaruan setiap *node*. Nilai kecepatan dari *class node* didapatkan di dalam setiap fungsi DSDV pada *file* dsdv.cc dan di passing ke dalam fungsi untuk menghitung periode perbaruan *node* melalui parameter fungsi. Penjelasan tentang fungsi dijelaskan pada subbab berikutnya (subbab 4.2.2).
Potongan kode sumber untuk mengetahui kecepatan *node* yang dilakukan dalam fungsi `DSDV_Agent::recv` dapat dilihat pada *Gambar 4.12*.

```
void DSDV_Agent::recv (Packet * p, Handler *){
   . . .
   modif_perup(myaddr_,node_->speed(),
   perup_);
   . . .
}
```

Gambar 4.12 Kode Untuk Mengetahui Kecepatan *Node* Dalam Fungsi DSDV_Agent::recv

4.2.2 Implementasi Modifikasi Untuk Menghitung Periode Perbaruan *Node*

Sesuai perancangan yang telah dibuat sebelumnya, maka setelah mengetahui kecepatan setiap *node* dapat dilakukan implementasi modifikasi untuk menghitung periode perbaruan *node*. Penghitungan periode perbaruan *node* berdasarkan faktor kecepatan kendaraan/*node* ini akan dikemas dalam suatu fungsi `modif_perup()`.

Langkah awal dalam implementasi pada subbab ini adalah deklarasi fungsi untuk menghitung periode perbaruan *node*. Deklarasi global dilakukan di dalam *header file* dsdv.h. Fungsi `modif_perup()` dideklarasikan secara global agar dapat digunakan oleh *class* yang lain.
Berikut ini adalah potongan kode sumber untuk deklarasi global fungsi `modif_perup()` di dalam *header file* dsdv.h:

```
void modif_perup(
   int alamat,
   double speed,
   double perup_ori);
```

Gambar 4.13 Kode Untuk Mendeklarasikan Fungsi Modifikasi

Selain deklarasi fungsi, pada *file* dsdv.h juga dilakukan deklarasi variabel *array* yang digunakan untuk menyimpan nilai periode perbaruan setiap *node*. Berikut ini adalah potongan kode sumber deklarasi variabel *array* di dalam *header file* dsdv.h:

```
double *perup_node =
   (double*) malloc(200 * sizeof(double));
```

Gambar 4.14 Kode Untuk Mendeklarasikan Variabel *Array* Periode Perbaruan *Node*

Setelah deklarasi selesai dibuat, selanjutnya adalah penulisan fungsi `modif_perup()` di dalam *file* dsdv.cc. Dalam fungsi ini, dilakukan inisialisasi variabel pada parameter fungsi, deklarasi variabel untuk penghitungan selisih kecepatan dengan *threshold*, `double diff_speed = speed - 25`, serta penghitungan periode perbaruan *node* dalam sebuah variabel *temp*, `int temp_perup = 15 - (floor (diff_speed/5)*3)`. Selanjutnya dilakukan perbandingan nilai kecepatan dengan *threshold*. Jika nilai kecepatan lebih besar dari nilai *threshold* maka nilai `perup_node` akan menggunakan

nilai temp_perup, namun jika nilai kecepatan lebih kecil dari nilai *threshold* maka nilai perup_node akan menggunakan nilai perup_ori. Nilai temp_perup didapat dari penghitungan selisih kecepatan *node* dengan *threshold*, setiap kecepatan bertambah 5m/s dari *threshold*, maka periode perbaruan *node default* DSDV murni akan berkurang 3 detik, dan berlaku kelipatan. Ketika nilai temp_perup lebih kecil dari 3, maka nilai perup_node adalah 2. Dalam hal ini *node* dengan kecepatan setara atau melebihi 50m/s, maka nilai periode perbaruan *node* tersebut adalah 2 detik. Dengan penghitungan tersebut, semakin besar nilai kecepatan suatu *node*, maka *node* akan semakin sering melakukan perbaruan *routing table*.
Berikut ini adalah potongan kode sumber untuk menghitung nilai periode perbaruan *node* di dalam *file* dsdv.cc:

```
void modif_perup(int alamat, double speed,
double perup_ori){
  int temp_perup = 15 -
(floor(diff_speed/5)*3);
  if(speed>25){
    double diff_speed = speed - 25;
    if (temp_perup >= 3){
      perup_node[alamat]= temp_perup;
    }
    else if (temp_perup < 3){
      perup_node[alamat]= 2;
    }
  }
  else{
    perup_node[alamat]= perup_ori;
  }
}
```

Gambar 4.15 Kode Sumber Fungsi Untuk Menghitung Nilai Periode Perbaruan *Node*

Fungsi untuk penghitungan nilai periode perbaruan *node* tersebut dipanggil di beberapa fungsi pada *file* protokol DSDV diantaranya:

```
DSDVTriggerHandler::handle(),
DSDV_Agent::cancelTriggersBefore(),
DSDV_Agent::needTriggeredUpdate(),
DSDV_Agent::helper_callback(),
DSDV_Agent::updateRoute(),
DSDV_Agent::processUpdate(),
DSDV_Agent::forwardPacket(), DSDV_Agent::recv().
```

4.3 Implementasi Simulasi pada NS2

Langkah pertama dalam implementasi simulasi VANETs adalah pendeskripsian lingkungan simulasi pada sebuah *file* berekstensi .tcl. *File* ini berisikan konfigurasi perangkat dan topologi. Berikut ini adalah konfigurasi lingkungan simulasi:

```
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)         1000;
set opt(y)         1000;
set val(ifqlen)    1000;
set val(nn)        30;
set val(seed)            1.0;
set val(adhocRouting)    DSDV;
set val(stop)            200;
set val(cp)        "traffic";
set val(sc)        "scenario.tcl";
```

Gambar 4.16 Konfigurasi Lingkungan Simulasi

Pada konfigurasi juga dilakukan pemanggilan *file* traffic yang berisikan konfigurasi *node* asal, *node* tujuan, pengiriman paket, serta *file* skenario yang berisi pergerakan *node* yang telah digenerate oleh SUMO. Kode implementasi pada NS2 dapat dilihat pada lampiran A.1 Kode Skenario NS2. Konfigurasi *traffic* dilakukan dengan membuat *file* berekstensi .txt. Berikut ini adalah konfigurasi *file traffic*:

```
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(28) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(29) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 1
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 2.5568388786897245 "$cbr_(0) start"
```

Gambar 4.17 Konfigurasi Traffic

Penentuan *node* sumber dan *node* tujuan pengiriman paket terdapat pada konfigurasi tersebut. Pengiriman dimulai pada detik ke- 2.55. Implementasi konfigurasi *file* traffic untuk simulasi pada NS-2 dapat dilihat pada Gambar 4.17.

4.4 Implementasi Metode Analisis

Dari hasil *trace file* yang dihasilkan setelah simulasi dapat dianalisis performa dari protokol perutean yang dipakai dengan menghitung beberapa parameter. Pada Tugas Akhir ini, parameter yang akan dianalisis adalah *Packet Delivery Ratio* (PDR), *Average End-to-End Delay* (E2E), dan *Routing Overhead* (RO). Alat yang digunakan untuk menganalisis performa protokol perutean yang dipakai adalah AWK. Untuk menjalanlan *file* awk

dapat dilakukan dengan memasukkan perintah awk -f *file*.awk scenario.tr. Pada sub bab berikut akan dijelaskan setiap metrik.

4.4.1 Implementasi *Packet Delivery Ratio* (PDR)

Pada Tugas Akhir ini, penghitungan nilai PDR menggunakan skrip AWK. Persamaan untuk menghitung nilai PDR dapat dilihat pada persamaan 3.1. Skrip AWK menghitung nilai PDR berdasarkan pada *trace file* yang dihasilkan setelah menjalankan simulasi. Penjelasan mengenai *trace file* dijabarkan pada subbab 2.4.2. Kode skrip AWK untuk menghitung nilai PDR dapat dilihat pada lampiran A.2 Kode Skrip AWK *Packet delivery ratio*.

Proses perhitungan PDR dilakukan dengan menghitung jumlah paket data terkirim yang dilakukan oleh *node* pengirim dan jumlah paket data yang diterima oleh *node* penerima pada satu *trace file*. Penambahan jumlah paket terkirim dilakukan apabila pada baris *trace* yang bersangkutan mengandung semua kondisi dimana huruf "s" yang menandakan kolom pertama mengandung *send packet*, kolom ke-3 menunjukkan bahwa *node* yang melakukan pengiriman adalah *node* pengirim, kolom ke-4 dan kolom ke-7 mengandung huruf "AGT" dan "cbr" yang menandakan pengiriman paket yang dilakukan adalah pengiriman paket data. Pencatatan jumlah paket yang diterima dilakukan apabila pada baris *trace* yang bersangkutan mengandung semua kondisi dimana huruf "r" yang menandakan kolom pertama mengandung *received packet*, kolom ke-3 menunjukkan bahwa *node* yang menerima paket data adalah *node* penerima, kolom ke-4 dan kolom ke-7 mengandung huruf "AGT" dan "cbr" yang menandakan penerimaan paket yang diterima adalah paket data. Perhitungan dilakukan sampai baris terakhir *trace file*, dan hasilnya adalah hasil hitung nilai PDR simulasi skenario. Setelah itu nilai PDR dapat dihitung dengan persamaan yang telah dijelaskan.

Berikut ini adalah pseudocode penghitungan PDR:

```
ALGORITMA PDR( trace file)
//Input: trace file simulasi skenario
//Ouput: PDR
BEGIN (
sent ← 0
recv ← 0
recv_id ← 0
pdr ← 0
)
#count packet send
(if ($1 == "s" and $3 == "_28_" and $4 ==
"AGT" and $7 == "cbr" )
sent +1;
)
#count packet receive
if ($1 == "r" and $3 == "_29_" and $4 ==
"AGT" and $7 == "AGT")
recv +1;
)
END (
Pdr ←  ( recv / sent ) * 100
print pdr)
```

Gambar 4.18 Pseudocode Untuk Menghitung PDR

Berikut ini adalah perintah untuk mengeksekusi skrip awk guna menganalisis *trace file*:

```
awk –f pdr.awk 30node.tr
```

Hasil keluaran dari skrip yang telah dieksekusi tersebut adalah pdr: xx.xx %.

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

Pada Tugas Akhir ini, penghitungan nilai rata-rata E2E menggunakan skrip AWK. Persamaan untuk menghitung nilai

E2E dapat dilihat pada persamaan 3.2. Skrip AWK menghitung nilai E2E berdasarkan pada *trace file* yang dihasilkan setelah menjalankan simulasi. Kode skrip AWK untuk menghitung nilai rata-rata E2E dapat dilihat pada lampiran A.3 Kode Skrip AWK *End to End Delay* (E2E).

Perhitungan nilai rata-rata E2E dilakukan dengan menghitung selisih waktu paket data terkirim yang dilakukan oleh *node* pengirim dan waktu paket data diterima oleh *node* penerima di dalam satu *trace file*. Pencatatan waktu paket terkirim pada kolom ke-2 dilakukan apabila pada baris *trace* mengandung beberapa kondisi yaitu kolom ke-1 mengandung huruf "s" yang menandakan *send packet*, kolom ke-3 menunjukkan ID *node* pengirim, kolom ke-4 dan kolom ke-7 mengandung kata "AGT" dan "cbr" yang menunjukkan pengiriman paket data. Perhitungan waktu dan pencatatan ID serta jumlah paket yang diterima dilakukan apabila baris *trace* mengandung beberapa kondisi yaitu kolom ke-1 mengandung huruf "r" yang menandakan *received packet*, kolom ke-3 menunjukkan ID *node* penerima, kolom ke-4 dan kolom ke-7 mengandung kata "AGT" dan "cbr" yang menunjukkan pengiriman paket data. Perhitungan dilakukan sampai baris terakhir *trace file*. Selanjutnya dilakukan penghitungan nilai rata-rata E2E. Pseudocode E2E ditunjukkan pada *Gambar 4.19*.

Berikut ini adalah perintah untuk mengeksekusi skrip awk guna menganalisis *trace file*:

```
awk –f e2e.awk 30node.tr
```

Hasil keluaran dari skrip yang telah dieksekusi tersebut adalah E2E: xx.xx (*float number*).

```
ALGORITMA E2E (trace file)
//Input: trace file
//Ouput: E2E
BEGIN (
for i in pkt_id
      pkt_id[i] ← 0
for i in pkt_sent
      pkt_sent[i] ← 0
for i in pkt_recv
      pkt_recv[i] ← 0
delay = avg_delay ← 0
recv ← 0
recv_id ← 0
)
(if ($1 == "s" and $3 == "_28_" and $4 ==
"AGT" and $7 == "cbr)
      pkt_sent[$6] ←  $2
if ($1 == "r" and $3 == "_29_" and $4 ==
"AGT" and $7 == "cbr" and recv_id != $6 )
      recv +1
      recv_id ←  $6
      pkt_recv[$6]←  $2
)
END (
for i in pkt_recv
      delay += pkt_recv[i] - pkt_sent[i]
avg_delay ← delay/recv
print avg_delay)
```

Gambar 4.19 Pseudocode Untuk Menghitung E2E

4.4.3 Implementasi *Routing Overhead* (RO)

Pada Tugas Akhir ini, penghitungan nilai *Routing Overhead* (RO) menggunakan skrip AWK. Persamaan untuk menghitung nilai RO dapat dilihat pada persamaan 3.3. Skrip AWK msenghitung nilai RO berdasarkan pada *trace file* yang dihasilkan setelah menjalankan simulasi. Kode skrip AWK untuk

menghitung nilai RO dapat dilihat pada lampiran A.4 Kode Skrip AWK *Routing Overhead*.

Implementasi penghitungan *routing overhead* DSDV dihitung apabila beberapa kondisi terpenuhi yaitu kolom ke-1 diawali dengan huruf "s" yang berarti *send packet* atau huruf "f" yang berarti *forward packet*, kolom ke-4 mengandung kata "RTR" yang berarti *routing layer*, kolom ke-7 mengandung kata "*message*" yang menunjukkan paket *routing* DSDV. Pseudocode menghitung RO dapat dilihat pada Gambar 4.20.

```
ALGORITMA RO-DSDV (trace file)
//Input: trace file
//Ouput: jumlah routing overhead protocol
DSDV
BEGIN (
rt_pkts ← 0
)
(if (($1=="s" || $1=="f") && ($4 == "RTR") &&
($7 = "message"))
rt_pkts +1)
)
END (
print rt_pkts)
```

Gambar 4.20 Pseudocode Untuk Menghitung RO

Berikut ini adalah perintah untuk mengeksekusi skrip awk guna menganalisis *trace file*:

```
awk –f ro.awk 30node.tr
```

Hasil keluaran dari skrip yang telah dieksekusi tersebut adalah RO: xx.xx (*float number*).

BAB V
UJI COBA DAN EVALUASI

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

5.1 Lingkungan Uji Coba

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

Tabel 5.1 Spesifikasi Perangkat yang Digunakan

Komponen	Spesifikasi
CPU	Intel(R) Core(TM) i3-3217U CPU @ 1.80 GHz x 2
Sistem Operasi	Linux Ubuntu 16.04 (64-bit)
Linux Kernel	3.19.0-32-generic
Memori	2.0 GB
Penyimpanan	26.5 GB

Uji coba dilakukan dengan menjalankan skenario *grid* dan skenario real yang telah dibuat pada simulator jaringan NS2. Dari simulasi tersebut dihasilkan sebuah *trace file* dengan ekstensi .tr yang akan dianalisis dengan bantuan skrip AWK untuk mendapatkan nilai *Packet Delivery Ratio* (PDR), *Average End to End Delay* (E2E), dan *Routing Overhead* (RO) menggunakan kode pada lampiran masing-masing pada A.10 Kode Skrip AWK *Packet Delivery Ratio*, A.11 Kode Skrip AWK *End to End Delay*, A.12 Kode Skrip AWK *Routing Overhead*.

5.2 Hasil Uji Coba

Hasil uji coba memuat hasil analisis performa protokol perutean DSDV murni dan DSDV modifikasi. Hasil uji coba terdiri dari hasil uji coba skenario *grid* dan uji coba skenario *real*. Hasil uji coba akan ditampilkan dalam bentuk grafik dan tabel dengan rincian performa setiap kecepatan *node*.

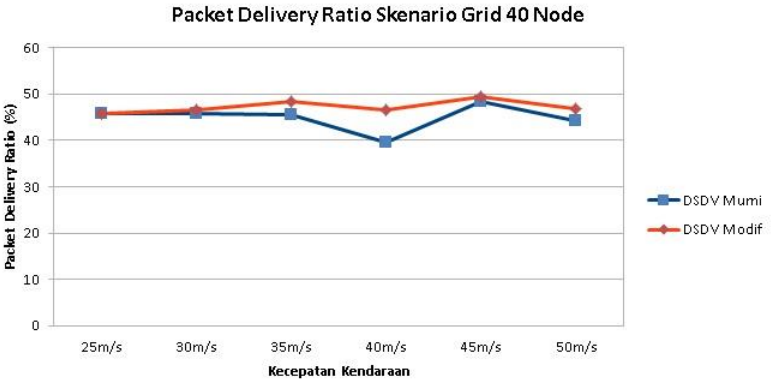
5.2.1 Hasil Uji Coba Skenario *Grid*

Uji coba pada skenario *grid* digunakan untuk melihat perbandingan *Packet Delivery Ratio* (PDR), *Average End to End Delay* (E2E), dan *Routing Overhead* (RO) antara protokol perutean DSDV murni dan protokol perutean DSDV yang telah dimodifikasi. Uji coba dilakukan sesuai dengan perancangan parameter lingkungan simulasi yang dapat dilihat pada Tabel 3.3.

Pengambilan data uji PDR, E2E, dan *RO* dengan luas area 1000m x 1000m, variasi *node* sebanyak 30, 40, dan 50, dan variasi kecepatan maksimal 25, 30, 35, 40, 45 dan 50 m/s. Dalam hal ini, variasi jumlah *node* digunakan untuk melihat pengaruh jumlah *node* terhadap performa. Sedangkan variasi kecepatan digunakan untuk melihat pengaruh kecepatan terhadap performa. Setiap variasi skenario tersebut di-*generate* 10 kali.

Hasil pengambilan data nilai *Packet Delivery Ratio* (PDR) pada skenario *grid* 40 node dapat dilihat pada Gambar 5.1 dan Tabel 5.2. Pada kecepatan maksimal *node* 25m/s nilai PDR tidak mengalami perubahan. Hal ini terjadi karena modifikasi periode perbaruan *node* hanya akan dilakukan jika kecepatan *node* lebih dari 25 m/s. Pada kecepatan maksimal *node* 30m/s nilai PDR mengalami kenaikan 1,57%. Pada kecepatan maksimal *node* 35m/s nilai PDR mengalami kenaikan 6,55%. Pada kecepatan maksimal *node* 40m/s nilai PDR mengalami kenaikan 17,68%. Pada kecepatan maksimal *node* 45m/s nilai PDR mengalami kenaikan 2,43%. Pada kecepatan maksimal *node* 50m/s nilai PDR mengalami kenaikan 5,95%. Kenaikan nilai PDR keenam variasi

kecepatan *node* tersebut apabila di rata-rata mencapai angka 6,83%.



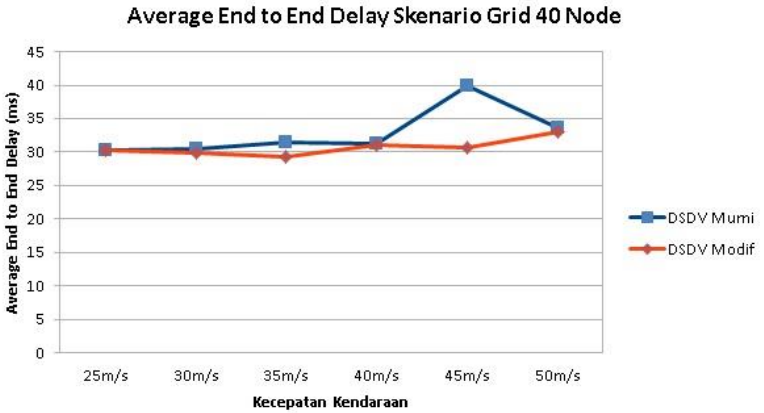
Gambar 5.1 Grafik PDR Skenario *Grid* 40 *Node*

Tabel 5.2 Tabel PDR Skenario *Grid* 40 Node

Kecepatan	DSDV Murni	DSDV Modif
25m/s	45,783	45,783
30m/s	45,844	46,565
35m/s	45,523	48,505
40m/s	39,664	46,680
45m/s	48,272	49,447
50m/s	44,155	46,783

Hasil pengambilan data nilai *average end-to-end delay* (E2E) pada skenario *grid* 40 *node* dapat dilihat pada Gambar 5.2 dan Tabel 5.3. Pada kecepatan maksimal *node* 25m/s tidak terjadi perubahan nilai E2E. Hal ini terjadi karena modifikasi periode perbaruan *node* hanya akan dilakukan jika kecepatan *node* lebih dari 25 m/s. Pada kecepatan maksimal *node* 30m/s nilai E2E mengalami penurunan 2,22%. Pada kecepatan maksimal *node*

35m/s nilai E2E mengalami penurunan 6,61%. Pada kecepatan maksimal *node* 40m/s nilai E2E mengalami penurunan 0,8%. Pada kecepatan maksimal *node* 45m/s nilai E2E mengalami penurunan 23,25%. Pada kecepatan maksimal *node* 50m/s nilai E2E mengalami penurunan 1,60%. Penurunan nilai E2E keenam variasi kecepatan *node* tersebut apabila di rata-rata adalah 6,9%.

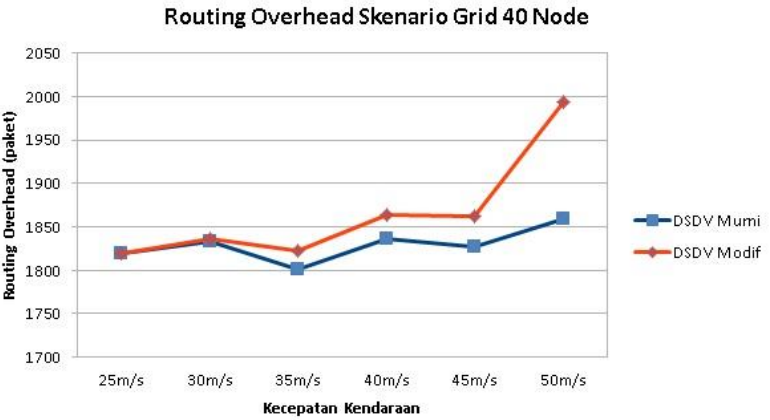


Gambar 5.2 Grafik E2E Skenario *Grid* 40 Node

Tabel 5.3 Data E2E Skenario *Grid* 40 *Node*

Kecepatan	DSDV Murni	DSDV Modif
25m/s	30,191	30,191
30m/s	30,496	29,816
35m/s	31,457	29,375
40m/s	31,274	31,020
45m/s	39,977	30,679
50m/s	33,562	33,024

Untuk hasil pengambilan data *routing overhead* (RO) pada skenario *grid* 40 *node* dapat dilihat pada Gambar 5.3 dan Tabel 5.4.



Gambar 5.3 Grafik RO Skenario *Grid* 40 *Node*

Tabel 5.4 Data RO Skenario *Grid* 40 *Node*

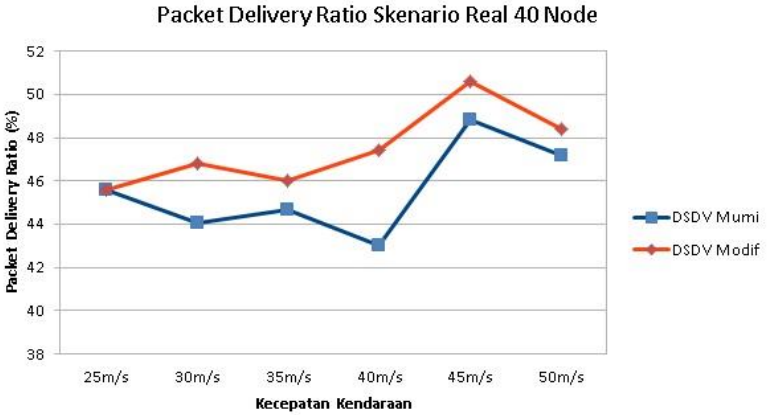
Kecepatan	DSDV Murni	DSDV Modif
25m/s	1.819,5	1.819,5
30m/s	1.833,2	1.835,9
35m/s	1.800,9	1.822,5
40m/s	1.836,3	1.863,3
45m/s	1.827,7	1.861,9
50m/s	1.859,1	1.993,8

5.2.2 Hasil Uji Coba Skenario *Real*

Pengujian pada skenario *real* digunakan untuk melihat perbandingan *Packet Delivery Ratio* (PDR), *Average End to End Delay* (E2E), dan *Routing Overhead* (RO) antara *routing protocol* DSDV murni dan *routing protocol* DSDV hasil dimodifikasi. Pengujian dilakukan pada peta sekitar Jalan Dr.Soetomo Surabaya yang telah dikonversi menjadi peta .xml.

Pengambilan data uji PDR, rata-rata *end to end delay*, dan *routing overhead* dilakukan pada peta *real* dengan variasi jumlah *node* 30, 40 dan 50. Dalam hal ini, variasi jumlah *node* digunakan untuk melihat pengaruh jumlah *node* terhadap performa. Setiap variasi skenario tersebut di*generate* 10 kali. Hasil scenario yang didapatkan, disimulasikan menggunakan *routing protocol* DSDV murni dan DSDV yang telah dimodifikasi. Setelah selesai melakukan simulasi, setiap skenario dievaluasi menggunakan metrik analisis. Selanjutnya dilakukan perhitungan rata-rata, sehingga menghasilkan nilai metrik analisis setiap variasi skenario. Nilai tersebut yang akan dibandingkan antara DSDV murni dan DSDV yang telah dimodifikasi.

Hasil pengambilan data nilai *Packet Delivery Ratio* (PDR) pada skenario *real* 40 *node* dapat dilihat pada Gambar 5.4 dan Tabel 5.5. Pada kecepatan maksimal *node* 25m/s nilai PDR tidak mengalami perubahan. Hal ini terjadi karena modifikasi periode perbaruan *node* hanya akan dilakukan jika kecepatan *node* lebih dari 25 m/s. Pada kecepatan maksimal *node* 30m/s nilai PDR mengalami kenaikan 6,24%. Pada kecepatan maksimal *node* 35m/s nilai PDR mengalami kenaikan 3,00%. Pada kecepatan maksimal *node* 40m/s nilai PDR mengalami kenaikan 10, 24%. Pada kecepatan maksimal *node* 45m/s nilai PDR mengalami kenaikan 3,56%. Pada kecepatan maksimal *node* 50m/s nilai PDR mengalami kenaikan 2,56%. Kenaikan nilai PDR keenam variasi kecepatan *node* tersebut apabila di rata-rata mencapai angka 5,12%.



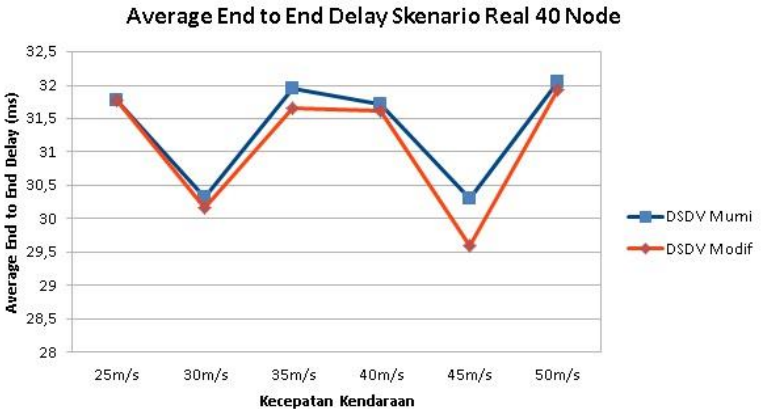
Gambar 5.4 Grafik PDR Skenario *Real* 40 *Node*

Tabel 5.5 Data PDR Skenario *Real* 40 *Node*

Kecepatan	DSDV Murni	DSDV Modif
25m/s	45,610	45,610
30m/s	44,033	46,782
35m/s	44,685	46,029
40m/s	43,022	47,431
45m/s	48,834	50,574
50m/s	47,205	48,415

Hasil pengambilan data nilai *average end-to-end delay* (E2E) pada skenario *real* 40 *node* dapat dilihat pada Gambar 5.5 dan Tabel 5.6. Pada kecepatan maksimal *node* 25m/s tidak terjadi perubahan nilai E2E. Hal ini terjadi karena modifikasi periode perbaruan *node* hanya akan dilakukan jika kecepatan *node* lebih dari 25 m/s. Pada kecepatan maksimal *node* 30m/s nilai E2E mengalami penurunan 0,53%. Pada kecepatan maksimal *node* 35m/s nilai E2E mengalami penurunan 0,89%. Pada kecepatan maksimal *node* 40m/s nilai E2E mengalami penurunan 0,29%. Pada kecepatan maksimal *node* 45m/s nilai E2E mengalami penurunan 2,29%. Pada kecepatan maksimal *node* 50m/s nilai

E2E mengalami penurunan 0,41%. Penurunan nilai E2E keenam variasi kecepatan *node* tersebut apabila di rata-rata adalah 0,88%.



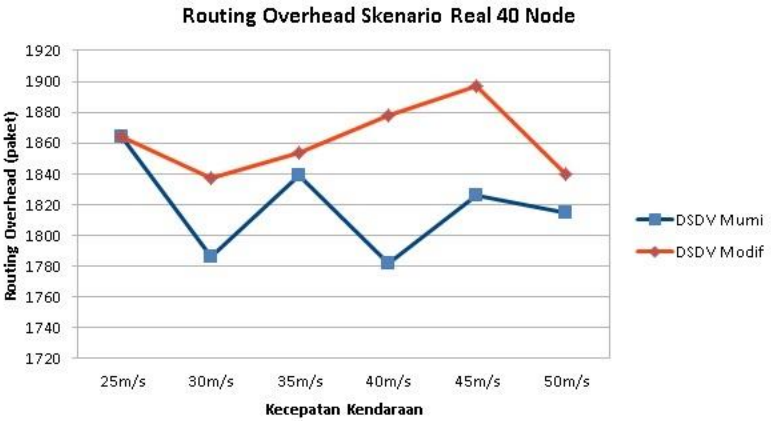
Gambar 5.5 Grafik E2E Skenario *Real* 40 *Node*

Tabel 5.6 Data E2E Skenario *Real* 40 *Node*

Kecepatan	DSDV Murni	DSDV Modif
25m/s	31,771	31,771
30m/s	30,318	30,155
35m/s	31,949	31,662
40m/s	31,707	31,612
45m/s	30,295	29,600
50m/s	32,058	31,925

Untuk hasil pengambilan data *routing overhead* (RO) pada skenario *real* 40 *node* dapat dilihat pada Gambar 5.6 dan Tabel 5.7. Pada kecepatan maksimal *node* 25m/s tidak terjadi perubahan nilai RO. Hal ini terjadi karena modifikasi periode perbaruan *node* hanya akan dilakukan jika kecepatan *node* lebih dari 25 m/s. Pada kecepatan maksimal *node* 30m/s nilai RO mengalami kenaikan 2,89%. Pada kecepatan maksimal *node* 35m/s nilai RO mengalami kenaikan 0,78%. Pada kecepatan

maksimal *node* 40m/s nilai RO mengalami kenaikan 5,39%. Pada kecepata maksimal *node* 45m/s nilai RO mengalami kenaikan 3,87%. Pada kecepatan maksimal *node* 50m/s nilai RO mengalami kenaikan 1,34%. Kenaikan nilai RO keenam variasi kecepatan *node* tersebut apabila di rata-rata mencapai angka 2,86%.



Gambar 5.6 Grafik RO Skenario *Real* 40 *Node*

Tabel 5.7 Data RO Skenario *Real* 40 *Node*

Kecepatan	DSDV Murni	DSDV Modif
25m/s	1.864,3	1.864,3
30m/s	1.785,7	1.837,4
35m/s	1.839,0	1.853,4
40m/s	1.782,0	1.878,2
45m/s	1.826,1	1.896,9
50m/s	1.815,0	1.839,5

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

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

6.1 Kesimpulan

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

1. Modifikasi fungsi periode perbaruan *node* dapat diterapkan berdasarkan faktor kecepatan kendaraan pada protokol perutean DSDV di dalam lingkup jaringan VANETs. Jika kecepatan kendaraan lebih besar dari *threshold* maka periode perbaruan *node* akan berubah.
2. Dampak dari modifikasi pada protokol perutean DSDV ini adalah peningkatan nilai PDR dan E2E dengan detail: Pada skenario *grid* 40 *node*, kenaikan PDR terbesar mencapai 17,68% dan penurunan E2E terbesar mencapai 23,25%. Pada skenario *real* 40 *node*, kenaikan PDR terbesar mencapai 6,24% dan penurunan E2E terbesar mencapai 2,29%. Jadi, dengan modifikasi ini nilai PDR dan E2E DSDV modifikasi lebih baik daripada nilai PDR dan E2E DSDV murni.
3. Dampak lain dari modifikasi ini adalah nilai RO DSDV modifikasi selalu lebih besar. Pada skenario *grid* 40 *node*, kenaikan RO terbesar mencapai 7,24%. Pada skenario *real* 40 *node*, kenaikan RO terbesar mencapai 5,39%. Hal ini karena semakin besar kecepatan kendaraan maka semakin sering *routing update* dilakukan. Jadi, dengan modifikasi ini nilai RO DSDV murni lebih baik daripada nilai RO DSDV modifikasi.

6.2 Saran

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

1. Untuk mendapatkan hasil uji coba yang lebih akurat, dapat dilakukan uji coba dengan jumlah skenario yang lebih banyak pada setiap variasi kecepatan, misalnya 20 skenario dalam satu variasi. Namun karena keterbatasan waktu dan memori sehingga dalam Tugas Akhir ini hanya dibuat 10 skenario.
2. Ide modifikasi periode perbaruan *node* untuk meningkatkan performa pengiriman paket sudah bagus, kedepannya bisa ditingkatkan lagi penelitian lebih lanjut.
3. Mencoba menerapkan modifikasi periode perbaruan *node* yang adaptif pada lingkungan simulasi lain.

DAFTAR PUSTAKA

[1] X. Yang, "Performance Optimisation for DSDV in VANETs," *UKSIM-AMSS International Conference on Modelling and Simulation*, 2015.

[2] Mahajan A. N. and Dadhich Reena, "Comparative Analysis of VANET Routing Protocols Using VANET RBC and IEEE 802.11p," *International Journal of Engineering Research and Applications (IJERA)*, Vol.3, Issue 4, Jul-Aug 2013, pp.531-538.

[3] Bodhy Krishna S, "Study of Ad hoc Networks with Reference to MANET, VANET, FANET," *International Journals of Advanced Research in Computer Science and Software Engineering (IJARCSEE)*, ISSN: 2277-128X (Volume-7, Issue-7), July 2017.

[4] Issariyakul Teerawat and Hossain Ekram, "Introduction to Network Simulator 2," DOI: 10.1007/ 978-0-387-7-71760-9_2, Springer Science+Business Media, 2009.

[5] A B M Moniruzzaman and Md Sadekur Rahmann, "Analysis of Topology Based Routing Protocols for Vehicular Ad-Hoc Network (VANET)," *International Journal of Computer Applications* (0975-8887), 2014.

[6] Samantha Badugu, Dr, K. Raja Kumar, and Nagarjuna Karyemsetty, "Design and Simulation of Vehicular Adhoc Network using SUMO and NS2," *Advances in Wireless and Mobile Communications*, ISSN 0973-6972 Volume 10, Number 5, Research India Publications 2017.

[7] Bouabdellah, M, Faissal, E B, Ben-Azza, H, "A Secure Cooperative Transmission Model in VANET Using Attribute Based Encryption," *International Conference on Advanced Communication Systems and Information Security (ACOSIS)*, October 2016

[8] “SUMO,” SUMO, [Online]. Available: http://sumo.dlr.de/userdoc/Installing/Linux_Build.html. [Accessed 15 November 2018].

[9] “OpenStreetMap,” OpenStreetMap, [Online]. Available: https://www.openstreetmap.org/about. [Accessed 20 December 2018].

[10] “JOSM,” JOSM, [Online]. Available: http://wiki.openstreetmap.org/wiki/JOSM. [Accessed 20 December 2018].

[11] "AWK," LinuxCommand.org: AWK, [Online]. Available: http://linuxcommand.org/lc3_adv_awk.php. [Accessed 27 December 2018].

[12] Dony Ariyus and Rum Andri K.R., "Komunikasi Data," ISBN: 978-979-29-0615-8, Yogyakarta: Andi Publisher, 2008.

[13] Sabih ur Rehman, M. Arif Khan, Tanveer A. Zia, and Lihong Zheng, "Vehicular Ad-Hoc Networks - An Overview and Challenges," *EURASIP Journal on Wireless Communications and Networking*, January 2013.

[14] Hemanth Narra, Yufei Cheng, "Destination-Sequenced Distance Vector (DSDV) Routing Protocol Implementation in ns-3," *Information and Telecommunication Technology Center*, USA, 2011.

[15] Guoyou He, "Destination-Sequenced Distance Vector (DSDV) Protocol," *Networking Laboratory*, HUT, 2002.

[16] "Documentation for ns & nam," Tutorial for the Network Simulator ns, [Online]. Available: http://www.isi.edu/nsnam/ns/tutorial/. [Accesed 31 December 2018].

LAMPIRAN

A.1 Kode Skenario NS2

```
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)       1000;
set opt(y)       1000;
set val(ifqlen)  1000;
set val(nn)            30;
set val(seed)          1.0;
set val(adhocRouting) DSDV
set val(stop)          200;
set val(cp)            "traffic";
set val(sc)            "scenario.tcl";
set ns_                [new Simulator]
set topo  [new Topography]
set tracefd [open scenario1.tr w]
set namtrace    [open scenario1.nam w]
set topo        [new Topography]
$topo load_flatgrid $opt(x) $opt(y)
set god_ [create-god $val(nn)]
$ns_ node-config -adhocRouting
$val(adhocRouting) \
                 -llType $val(ll) \
                 -macType $val(mac) \
                 -ifqType $val(ifq) \
                 -ifqLen $val(ifqlen) \
                 -antType $val(ant) \
```

```
                  -propType $val(prop) \
                  -phyType $val(netif) \
     -channelType $val(chan) \
     -topoInstance $topo \
     -agentTrace ON \
     -routerTrace ON \
     -macTrace ON \
     -movementTrace ON \
Phy/WirelessPhy set RXThresh_ 5.57189e-11
; #400m
Phy/WirelessPhy set CSThresh_ 5.57189e-11
; #400m
for {set i 0} {$i < $val(nn)} {incr i} {
    set node_($i) [$ns_ node]
    $node_($i) random-motion 0 ;# disable
random motion
}
puts "Loading connection pattern..."
source $val(cp)
puts "Loading scenario file..."
source $val(sc)
for {set i 0} {$i < $val(nn)} {incr i} {
    $ns_ initial_node_pos $node_($i) 20
}
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $val(stop).0 "$node_($i)
reset";
}
$ns_ at  $val(stop).0002 "puts \"NS
EXITING...\" ; $ns_ halt"

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

```
puts $tracefd "M 0.0 prop $val(prop) ant
$val(ant)"

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

A.2 Kode Skrip AWK *Packet Deliveri Ratio*

```
BEGIN {
sent=0;
received=0;
}

{

# count packet send
  if($1=="s" && $3=="_28_" && $4=="AGT" &&
$7=="cbr")
   {
    sent++;
   }

  else if($1=="r" && $3=="_29_" && $4=="AGT"
&& $7=="cbr")
   {
     received++;
   }

}
END {
 printf "%.2f|",(received/sent)*100;

}
```

A.3 Kode Skrip AWK *End-to-End Delay*

```
BEGIN {
        seqno = -1;
        #droppedPackets = 0;
        #receivedPackets = 0;
        count = 0;
}

{
        if($4 == "AGT" && $1 == "s" && seqno <
$6) {
                seqno = $6;
        }
        #else if(($4 == "AGT") && ($1 == "r"))
{
        #receivedPackets++;
        #} else if ($1 == "D" && $7 == "cbr"
&& $8 > 512){
        #droppedPackets++;
        #}
        #end-to-end delay
        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 "%.2f|", n_to_n_delay * 1000;
}
```

A.4 Kode Skrip AWK *Routing Overhead*

```
BEGIN {
      rt_pkts = 0;
}

{
      if (($1 == "s" || $1 == "f") && ($4 ==
"RTR") )
      rt_pkts++;
}

END {
      #printf ("Total number of routing
packets\t%d\n",rt_pkts);
      printf ("%d|",rt_pkts);
}
```

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Monica Indah Habsari, lahir pada tanggal 24 Juni 1996 di Kediri. Penulis menempuh pendidikan formal yaitu di RA Baiturrahman Jakut (2000-2002), SDN 04 Sukapura Jakut (2002-2004), SDN 1 Kandat Kediri (2004-2008), SMPN 1 Ngadiluwih (2008-2011), SMAN 2 Kediri (2011-2014) dan merupakan seorang mahasiswi angkatan 2014 bidang studi Arsitektur dan Jaringan Komputer (AJK) di Departemen Informatika Institut Teknologi Sepuluh Nopembe (ITS) Surabaya. Selama menempuh pendidikan di kampus, penulis aktif dalam beberapa organisasi, antara lain sebagai Asisten Manajer Departemen Information and Media di UKM TDC ITS, Staff PSDM Divisi *Culture* di IFLS ITS, Staff Internal di BIMITS, Staff Al-Afifah di Keluarga Muslim Informatika ITS, dan Staff Ahli di Departemen KWU HMTC ITS. Penulis juga aktif sebagai *Volunteer* di komunitas peduli lingkungan Surabaya Osoji Club. Penulis pernah kerja praktik di Unit JKTID PT. Garuda Indonesia (Persero) Tbk, periode Juli – Agustus 2017. Penulis dapat dihubungi melalui nomor ponsel: 085708024625 atau email: monicaindahh@gmail.com.