STUDI KINERJA 802.11P PADA PROTOKOL DSR di LINGKUNGAN VANETs

Muhammad Yusuf S.
NRP. 05111250010030

DOSEN PEMBIMBING
Dr. Eng. Radityo Anggoro, S.Kom, M.Sc.
NIP. 198410162008121002

PROGRAM MAGISTER

BIDANG KEAHLIAN KOMPUTASI BERBASIS JARINGAN

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA

2017

[Halaman ini sengaja dikosongkan]

STUDI KINERJA 802.11P PADA PROTOKOL DSR di LINGKUNGAN VANETs

Muhammad Yusuf S.
NRP. 05111250010030

DOSEN PEMBIMBING
Dr. Eng. Radityo Anggoro, S.Kom, M.Sc.
NIP. 198410162008121002

PROGRAM STUDI MAGISTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

[Halaman ini sengaja dikosongkan]

PERFORMANCE EVALUATION 802.11P USING DSR PROTOCOL ON VANETs

Muhammad Yusuf S.
NRP. 05111250010030

SUPERVISOR
Dr. Eng. Radityo Anggoro, S.Kom, M.Sc.
NIP. 198410162008121002

PROGRAM STUDI MAGISTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

[Halaman ini sengaja dikosongkan]

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom)
di
Institut Teknologi Sepuluh Nopember Surabaya

Oleh:
MUHAMMAD YUSUF SYAIFUDDIN
NRP. 5112201030

Dengan judul:
STUDI KINERJA 802.11P PADA PROTOKOL DSR di LINGKUNGAN VANETs

Tanggal Ujian : 6 Januari 2018
Periode Wisuda : 2017 Gasal

Disetujui Oleh:

Dr.Eng. Radityo Anggoro, S.Kom.,M.Sc.
NIP. 19841016 200812 1 002 ..... (Pembimbing 1)

Waskitho Wibisono, S.Kom, M.Eng, Ph.D
NIP. 19741022 200003 1 001 ..... (Penguji 1)

Tohari Ahmad S.Kom., MIT, Ph.D
NIP. 19750525 200312 1 002 ..... (Penguji 2)

Royyana Muslim I, S.Kom.M.Kom, Ph.D.
NIP. 19770824 200604 1 001 ..... (Penguji 3)

Dekan Fakultas Teknologi Informasi,

Dr. H. Agus Zainal Arifin, S.Kom.,M.Kom
NIP. 19720809 199512 1 001

[Halaman ini sengaja dikosongkan]

Studi Kinerja 802.11p Pada Protokol DSR di Lingkungan VANETs

Nama Mahasiswa : Muhammad Yusuf S.
NRP : 5112201030
Pembimbing : Dr. Eng. Radityo Anggoro, S.Kom, M.Sc.

ABSTRAK

Vehicular Adhoc Network (VANET) adalah suatu kelompok jaringan nirkabel yang berjenis *infrastructure-less* yaitu tidak ada nya suatu *router* pusat atau *access-point* karena semuanya bertipe *adhoc* yang merupakan semua node tersebut merupakan *client* dan *server*-nya, sama seperti Mobile Adhoc Network (MANET). Perbedaan antara MANET dan VANET ini adalah dari penggunaan protokolnya, yaitu 802.11p pada VANET yang merupakan perkembangan dari 802.11 standar yang digunakan oleh MANET. Optimisasi ini dilakukan karena pada 802.11 standar tidak mampu menyediakan komunikasi yang dapat diandalkan karena karakteristik VANET yang memiliki kepadatan bervariasi dan cenderung berkecepatan tinggi, sehingga jendela komunikasinya sempit.

VANET pada dasarnya adalah salah satu DSRC (*Dedicated short-range communications*) yang berarti komunikasi jarak pendek. Penggunaan VANET sendiri lebih kearah keselamatan umum, contohnya adalah untuk mengirimkan data pengereman kendaraan ke kendaraan dibelakangnya sehingga tidak terjadi kecelakaan.

Penggunaan protokol *routing* dalam penelitian ini adalah protokol reaktif *Dynamic Source Routing* atau biasa disebut dengan DSR. Penggunaan DSR pada VANET ini digunakan untuk mengevaluasi seberapa bagus protokol DSR pada lingkungan VANET dengan menggunakan skenario *grid* dan *real world*. Protokol diujikan dengan tingkat variasi kecepatan dan kepadatan kendaraan yang berbeda-beda pada jalan bebas hambatan yang memiliki persimpangan. Parameter yang dievaluasi antara lain adalah *packet delivery ratio*, *end to end delay* dan *routing overhead*.

Kata kunci : MANET, VANET, DSR, persimpangan, 802.11p, WAVE

[Halaman ini sengaja dikosongkan]

Performance Evaluation of 802.11p in DSR Protocol at Vanet's Environment

Student Name : Muhammad Yusuf S.
NRP : 5112201030
Supervisor : Dr. Eng. Radityo Anggoro, S.Kom, M.Sc.

ABSTRACT

Vehicular Adhoc Network (VANET) is a group of wireless network that doesn't have any centralized infrastructure, there are no central router or access-point because all of nodes are adhoc. Differences between MANET and VANET are the protocol that they used, VANET Use 802.11p and MANET using standard 802.11. VANET use 802.11p because the standard 802.11 cannot survive againts VANET's that have dynamic density and high speed that makes small communication window between nodes.

VANET are DSRC (Dedicated short-range communications), that used by public safety, i.e. send information to the car behind when the car try to sudden-stop, so the other cars behind can avoid an accident.

This research using routing protocl called Dynamic Source Routing (DSR). The usage of DSR are to evaluate how good the DSR against VANET environment using grid and real world scenarios. This scenario contains dynamic speed and density and evaluating the result of packet delivery ratio, end-to-end delay and route overhead.

***Keywords** : MANET, VANET, DSR, junctions, 802.11p, WAVE*

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah memberikan rahmatnya sehingga penulis dapat menyelesaikan tesis yang berjudul "Studi Kinerja 802.11p Pada Protokol DSR di Lingkungan VANETs". Tesis ini disusun untuk memenuhi sebagian persyaratan guna memperoleh gelar sarjana Magister Komputer di Institut Teknologi Sepuluh Nopember.

Penulis mengucapkan terima kasih dan penghargaan yang sebesar-besarnya kepada berbagai pihak yang telah memberikan bantuan dan dukungan sehingga tesis ini dapat terselesaikan dengan baik, khususnya kepada:

1. Bapak Dr. Eng. Radityo Anggoro, S.Kom, M.Sc., sebagai pembimbing tesis yang telah membimbing penulis hingga tesis ini dapat terselesaikan dengan baik.
2. Bapak Waskitho Wibisono, S.Kom, M.Eng, Ph.D, selaku Ketua Program Studi Pascasarjana Teknik Informatika Institut Teknologi Sepuluh Nopember yang telah memberi saran dan kemudahan dalam pengerjaan tesis ini
3. Keluarga penulis, yang telah memberikan dukungan mental dan spiritual yang sangat berharga selama penulis menuntut ilmu di Institut Teknologi Sepuluh Nopember ini.
4. Bapak Ibu Dosen pengajar di Program Pascasarjana Teknik Informatika Institut Teknologi Sepuluh Nopember, yang telah mengajarkan banyak ilmunya.
5. Bapak Andrew Hadinyoto selaku CEO PT. Garasilabs Manivesta yang telah memudahkan penulis untuk melakukan ijin bimbingan sehingga menyelesaikan tesis ini.
6. Teman-teman Pasca FTIF angkatan 2012, atas bantuan, semangat, do'a, pengalaman, dan kenangan-kenangan berharga selama penulis menuntut ilmu di Institut Teknologi Sepuluh Nopember ini. Terutama keluarga besar Sekte NCC 2012, yang telah menyemangati untuk terus berjuang sampai akhir.

7. Nadila Cindi Wantari, atas penyemangat dan do'a yang membantu penulis menyelesaikan tesis ini.
8. Anak-anak grup Podmam yang telah menyemangati penulis selama menyelesaikan tesis ini.
9. Warkop Podomampir Ketintang yang sudah sudi ditempati penulis ketika mengerjakan tesis ini.
10. Grup Kngntrs yang membantu menyemangati dan menghibur selama pengerjaan tesis ini.
11. Segenap karyawan Tata Usaha Teknik Informatika yang telah membantu mengurus syarat-syarat administratif penyelesaian tesis.

Penulis juga menyampaikan terima kasih kepada berbagai pihak yang tidak dapat penulis tulis satu-persatu yang telah membantu dalam telah penyelesaian tesis ini.

Apa yang penulis kerjakan dalam tesis ini hanya menyelesaikan sebagian permasalahan kecil yang ada, maka dari itu penulis mengharapkan penelitian ini dapat dikembangkan dan diperbaiki pada penelitian-penelitian selanjutnya sehingga hasilnya dapat bermanfaat bagi masyarakat.

Surabaya, 20 Januari 2018

Muhammad Yusuf S.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Mobile Adhoc Network (MANET) adalah suatu kelompok jaringan yang berbasis nirkabel yang terdiri dari sekumpulan node-node yang bergerak tanpa adanya manajemen jaringan secara khusus yang mengatur tiap-tiap node tersebut. Dalam perkembangannya, MANET dibagi menjadi beberapa kategori. Salah satu kategori yang saat ni banyak diteliti adalah *Vehicular Adhoc Network* atau yang lebih dikenal dengan sebutan VANET. VANET ini lebih memiliki beberapa karakteristik tambahan dibandingkan dengan MANET, yaitu tingkat mobilitas yang berkecepatan tinggi, dan waktu komunikasi antar node jaringan yang lebih singkat, dan beragam skenario aplikasi yang disesuaikan dengan karakteristik pergerakan (Li & Wang, 2007). VANET sendiri adalah suatu kelompok jaringan yang sama seperti MANET, bedanya adalah VANET terdiri dari perangkat bergerak seperti truk, mobil, sepeda motor, bus, dan beberapa infrastruktur tetap seperti lampu lalu lintas, jalan bebas hambatan, dan lain-lain. hal ini menyebabkan beberapa perbedaan antara MANET dan VANET, yaitu pada pola pergerakannya. karena pada VANET, yang bergerak bukanlah suatu perangkat node yang pergerakannya terbatas. Hal ini menyebabkan perubahan topologi jaringan yang lebih cepat berubah-ubah, dan akan berpengaruh pada stabilitas koneksi yang terbentuk.

Protokol routing pada MANET dapat dibedakan menjadi banyak kategori dan kriteria. Misalnya berdasarkan mekanisme proses discovery, protokol routingnya dapat dibedakan menjadi protokol reaktif dan protokol proaktif. Pada routing proaktif, seperti pada OLSR (*Optimized Link State Routing*) (Clausen, 2003), setiap jalur terlebih dahulu dibentuk dan ditentukan oleh masing-masing node walupun sedang tidak ada data yang dikirimkan. Sedangkan, pada protokol routing reaktif, seperti *Dynamic Source Routing* (DSR) (Jhonson, 2007) dan *Adhoc on demand Distance Vector*

(AODV) (Perkin, 1999), pembentukan jalur dilakukan apabila terjadi permintaan routing atau pengiriman data. sehingga meminimalisir *routing overhead.*

Karena VANET adalah turunan dari MANET, maka pada VANET sangat memungkinkan untuk mengadaptasi protokol-protokol routing yang ada pada MANET. Akan tetapi, karena memiliki pergerakan yang lebih tinggi, maka VANET menggunakan IEEE 802.11p atau biasa disebut juga WAVE (*Wireless Access in Vehicular Environment*) yang merupakan peningkatan dari wireless standard 802.11.

Protokol 802.11p sendiri adalah salah satu jenis DSRC (*Dedicated Short-Range Communication*) yaitu komunikasi jarak pendek, yang dikhususkan untuk sistem transportasi cerdas, dan beberapa contohnya adalah untuk peringatan bahaya kecelakaan, misalnya ada mobil rem mendadak, maka akan melakukan *broadcast* ke mobil belakangnya sehingga tidak terjadi kecelakaan. (Jiang, D. dan Delgrossi, L., 2008,)

Peningkatan yang dilakukan pada WAVE ini adalah dengan optimisasi pada layer MAC dan PHY, sehingga komunikasi yang terjadi antar kendaraan semakin maksimal. Karena seperti yang diketahui bahwa didalam VANET, waktu komunikasi kendaraan sangat bervariasi dan sebentar (misalnya ketika sedang berpapasan). Beberapa optimisasi pada layer MAC ini antara lain seperti menghilangkan proses autentikasi di level MAC, sehingga mempercepat terbentuknya koneksi dan segera bisa dilakukan komunikasi. Sedangkan pada layer PHY, hanya sedikit optimisasi yang dilakukan, seperti penggunaan frekuensi pada 5.9Ghz.

Penelitian ini melakukan studi kinerja protokol routing *Dynamic Source Routing* (DSR) pada VANET dan diharapkan memiliki kinerja yang baik sehingga layak diterapkan pada lingkungan VANET. Tingkat kelayakan diukur dengan melakukan evaluasi performansi pada ruang lingkup simulasi berbentuk *grid* yang mendekati *real world* yang dilengkapi dengan persimpangan. Parameter yang dibandingkan adalah *routing overhead, end to end delay,* dan *packet delivery ratio*.

Untuk mengetahui seberapa baik protokol 802.11p ini dibandingkan dengan protokol 802.11 lain, dilakukan juga studi kinerja dengan memilih 802.11a yang

menjadi dasar dari optimisasi 802.11p ini, dengan skenario yang sama yaitu pada kecepatan dan kepadatan yang sama dengan 802.11p.

Untuk routing protokolnya, dipilih prokotol DSR karena memiliki kinerja yang baik pada kecepatan pergerakan sedang (15m/s) dan dalam kepadatan rendah (30 nodes) sampai sedang (90 nodes) dibanding dengan protokol AODV, maka sangat cocok untuk diimplementasikan kedalam VANET yang memiliki lingkungan *urban* (perkotaan) (Paul, Bijan 2012).

1.2 Perumusan Masalah

Rumusan masalah yang diangkat dalam penelitian ini adalah:

1. Bagaimana anlisa performa 802.11p dibandingkan dengan 802.11a pada kecepatan dan kepadatan kendaraan yang sama
2. Bagaimana performa protokol routing DSR pada lingkungan VANETs yang dinamis dengan kecepatan dan kepadatan kendaraan yang bervariasi pada skenario *real world* maupun *grid*.

1.3 Hipotesa

DSR merupakan protokol reaktif yang digunakan untuk MANETs. Berdasarkan penelitian sebelumnya, protokol routing yang didesain untuk MANETS, tidak bisa bertahan dengan baik jika diimplementasikan dalam lingkungan VANETs, termasuk DSR. Dalam penelitian ini, performansi DSR pada lingkungan VANETs dengan menambahkan konfigurasi WLAN 802.11p (WAVE) yang secara khusus merupakan standar WLAN untuk lingkungan VANETs apda protokol DSR. Performa DSR dengan konfigurasi WLAN 802.11p menghasilkan *packet delivery ratio, routing overhead* dan *end to end delay* yang bagus.

1.4 Tujuan dan Manfaat Penelitian

Tujuan dari penelitian ini adalah menganalisa performansi protokol Dynamic Source Routing dengan menggunakan parameter 802.11p pada lingkungan VANETs dengan berbagai macam variasi. Manfaat dari penelitian ini adalah untuk mengetahui

sejauh mana Dynamic Source Routing dapat mempertahankan performansinya pada lingkungan VANETs yang sangat dinamis.

1.5 Kontribusi Penelitian

Kontribusi yang dihasilkan dari penelitian ini adalah melakukan analisa pada protokol DSR dengan parameter 802.11p pada lingkungan grid dan realworld dengan berbagai macam skenario kepadatan dan kecepatan kendaraan. Hasil analisa ini akan menunjukkan protokol DSR tersebut dapat bekerja dengan layak pada lingkungan VANET ketika memakai parameter 802.11p.

1.6 Batasan Masalah

Batasan masalah dari penelitian ini sebagai berikut:

- Pengujian 802.11a hanya pada peta berbentuk grid
- Peta yang digunakan berjenis dua buah, grid dan real world
- Peta berjenis grid dan real world dihasilkan dengan network simulator sumo
- Peta real world digunakan untuk pembatasan jumlah persimpangan yang akan diujikan di peta grid
- Untuk analisa dan simulasi akan dilakukan pada lingkungan simulator NS-2
- Ukuran lingkungan simulasi adalah 2000m x 2000m
- Waktu simulasi 900 detik.

1.7 Sistematika Penulisan

Sistematika dalam penulisan buku tesis ini disusun sebagai berikut:

1. Bab I Pendahuluan. Bab ini berisi pendahuluan yang menjelaskan latar belakang permasalahan, perumusan masalah, batasan masalah, tujuan dan manfaat penelitian, kontribusi penelitian dan sistematika penulisan.
2. Bab II Teori dan Tinjauan Pustaka. Berisi tinjauan pustaka dan teori-teori pendukung lainnya. Teori yang dibahas meliputi teori dasar tentang MANET, VANET, macam-macam protokol pada MANET dan VANET, Routing pada MANET, Routing pada VANET, DSR, AODV, serta penelitian-penelitian yang berkaitan dengan metode yang dikembangkan dalam tesis ini.
3. Bab III Metodologi Penelitian. Bab ini berisi tentang langkah-langkah penelitian

yang dilakukan meliputi analisis dan desain routing dan jaringan yang dikembangkan, langkah-langkah kerja yang disusun berdasarkan teori-teori yang melandasi pada bab berikutnya, dan rancangan pengujian serta pengamatan yang dilakukan pada penelitian ini.

4. Bab IV Hasil dan Pembahasan. Bab ini berisi tentang pengujian dan analisis terhadap hasil implementasi sistem yang akan dikembangkan.
5. Bab V Penutup. Berisi kesimpulan dan saran untuk pengembangan pada penelitian selanjutnya.

[Halaman ini sengaja dikosongkan]

BAB II

KAJIAN PUSTAKA

2.1 Vehicular Ad Hoc Network (VANET)

Vehicular Adhoc Network (VANET) adalah bagian dari *Mobile Adhoc Network* (MANET). VANET memiliki pengelompokan tersendiri karena memiliki karakteristik yang berbeda dibanding dengan MANET. VANET adalah teknologi yang menggabungkan kemampuan komunikasi wireless pada kendaraan menjadi sebuah jaringan *infrastructure-less* (Najafabadi, 2011). *Infrastructure-less* yang dimaksud disini adalah suatu jaringan yang tidak memiliki jalur akses komunikasi tunggal, yang dimana tidak membutuhkan jalur akses pusat yang menghubungkan semua perangkat. Proses pengiriman pada *adhoc* ini sendiri memanfaatkan node-node atau perangkat disekitarnya untung mengirimkan data antara sumber ke tujuan. Perbedaan antara VANET dibanding dengan MANET adalah pada VANET memiliki pergerakan node dengan kecepatan tinggi, mobilitas yang telah ditentukan oleh jalur tertentu (seperti jalan raya, perempatan, dll), sumber daya dan media penyimpanan yang mencukupi dan lingkungan komunikasi yang sulit karena singkatnya jangka waktu satu dengan yang lain.

2.1.1 Mekanisme Komunikasi pada VANET

Setiap node didalam VANET dapat difenisikan sebagai kendaraan atau infrastruktur tetap disepanjang jalan (seperti misalnya tiang pemancar yang ada disepanjang jalan raya). Setiap node akan tersebar disepanjang jalan raya dengan memiliki pola tertentu. Setiap node di dalam VANET dilengkapi dengan perangkat komunikasi yang dikelola oleh node tersebut sendiri. Untuk dapat berkomunikasi, masing-masing node harus mengijinkan node lainnya agar dapat berkomunikasi dengan node tersebut. Mekanisme komunikasi antar tiap node yang berada pada sepanjang jalan raya terdiri atas *vehicle to vehicle communication* (V2V), *vehicle to*

road side communication infrastructure (V2I), dan *hybrid* (atau kombinasi dari V2V dengan V2I).

2.1.1.1 Vehicle to Vehicle Communication (V2V)

Vehicle to vehicle communication(V2V) adalah model komunikasi antar kendaraan yang bergerak. Mekanisme komunikasi ini mengijinkan antara satu atau lebih kendaraan yang saling berkomunkasi antara satu dengan yang lainnya. Ketika berkomunikasi secara V2V, maka setiap kendaraan harus dapat mendeteksi posisi dan pergerakan dari setiap node untuk dapat melakukan pengiriman data. Dalam implementasinya, mekanisme komunikasi V2V dapat dibedakan menjadi *single hop* dan *multi hop*, Mekanisme transmisi *single hop* dilakukan untuk melakukan transmisi data pada kumpulan node tetangga yang berada dalam radius. *Single hop* dilakukan untuk kebutuhan pengiriman data dengan jarak yang pendek, yaitu antara node-node tersebut yang berada dalam jangkauan transmisinya. Pada transmisi *multi hop* biasanya digunakan untuk transmisi data jarak jauh yang node nya berada diluar jalur transmisi. Agar data dapat mencapai node tujuan, maka diperlukan node-node antara sumber dan tujuan untuk melakukan *relay* (menyalurkan) data hingga sampai pada node tujuan.

2.1.1.2 Vehicle to Roadside Unit Communication

Vehicle to roadside unit communication biasa disebut juga *vehicle to infrastructure*(V2I) memanfaatkan penggunaan infrastruktur nirkabel yang ada disepanjang jalur yang dia lewati. Setiap kendaran yang melintas akan berkomunikasi dengan infrastruktur yang telah diletakkan di sepanjang jalan untuk mengirimkan data ke node tujuan. Dan yang menangani pengiriman data sepenuhnya berada pada infrastruktur yang tersedia.

2.1.1.3 Hybrid

Mekanisme komunikasi *hybrid* menggabungkan penggunaan infrastruktur V2I dengan V2V dalam pengiriman data untuk berkomunikasi. Pada mekanisme *hybrid* setiap node dapat melakukan komunikasi antara infrastruktur yang tersedia dan juga dapat melakukan komunikasi antar kendaraan (V2V)

2.1.2 Protokol Routing Pada VANET

Routing merupakan proses meneruskan data dari node sumber menuju ke node tujuan melalui node-node antara mereka secara *multi-hop*. Protokol routing adalah penanggung jawab supaya data sampai pada node tujuan, dengan melakukan pemilihan jalur dan melakukan perbaikan jalur ketika terjadi kegagalan pada proses pengiriman data sebagai akibat terputusnya sebuah jalur. Karena dalam VANET ataupun MANET sendiri, putusnya suatu jalur menjai hal yang lumrah, karena terjadi pergerakan antar node. Sebuah protkol routing yang baik harus mampu melakukan pengiriman data dengan waktu yang seingkat mungkin dengan konsumsi sumber daya dan *bandwitdh* seminimal mungkin (Yahya, 2011). Berbeda dengan MANET, pembentukan protokol routing pada VANET memiliki tantangan tersendiri karena perubahan topologi yang lebih dinamis, pengelompokan area jaringan dan waktu jendela transmisi yang singkat.

1. Topologi Dinamis : VANET terbentuk dari sekumpulan kendaraan yang berkomunikasi secara adhoc. Sebuah kendaraan dapat masuk dan keluar dalam area jaringan secara bebas sewaktu-waktu.
2. Pengelompokan Jaringan: kondisi dari kepadatan suatu area menyebabkan area jangkauan yang terpisah dengan yang memiliki kepadatan daerah yang sedikit.
3. Waktu jendela tranmisi : beberapa aplikasi atau data membutuhkan waktu pengiriman data secara cepat dengan tingkat prioritas tertentu. Sehingga ketika melewati waktu jendela transmisi tersebut, maka bisa mengakibatkan tertundanya data yang terkirim atau bahkan gagal.

Implementasi protokol routing pada MANET tidak dapat langsung dilakukan dalam lingkungan VANET karena pada MANET tidak memperhitungkan faktor dan karakteristik pada VANET. Berdasarkan hal tersebut, dibutuhkan beberapa

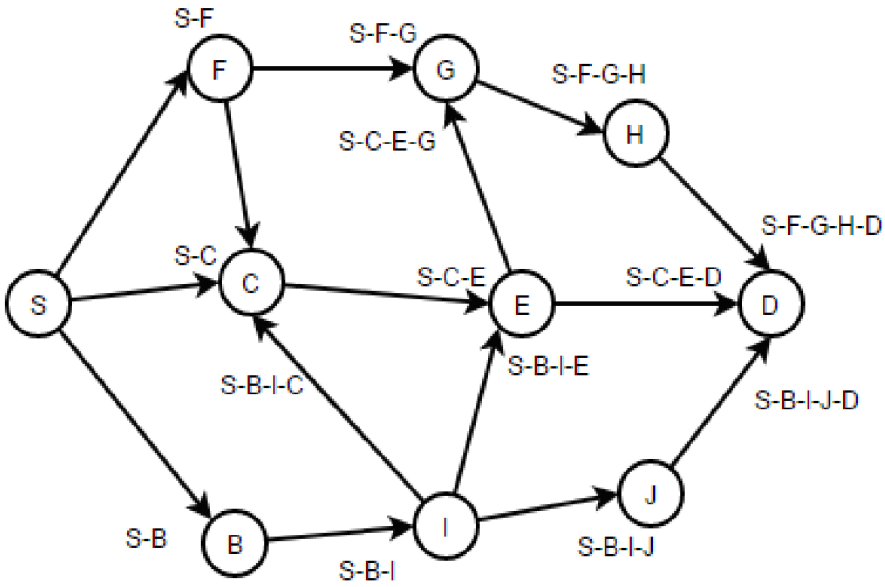
penyesuaian dengan memperhatikan faktor-faktor dan karakteristik untuk melakukan implementasi protokol MANET tersebut ke dalam lingkungan VANET.

2.2 Dynamic Source Routing (DSR)

Dynamic Source Routing adalah salah satu protokol routing reaktif pada MANET yang dapat diimplementasikan pada VANET. Routing pada VANET disusun dengan mengandalkan informasi topologi jaringan. Tujuan utama dari routing yang berbasis topologi ini adalah untuk menjaga keseimbangan antara proses pencarian jalur dan menjaga *overhead* seminimal mungkin. DSR menggunakan mekanisme *flooding* terhadap suatu jaringan dengan melakukan pencarian paket untuk mendapatkan jalur sampai ke node tujuan. Dan ketika pesan tersebut sampai ke tujuan, maka akan direspon balik ke node pengirim dengan informasi kandidat jalur yang digunakan. Setelah periode waktu tertentu, berdasarkan informasi jalur yang diterima, maka node pengirim akan memutuskan jalur yang digunakan. Pada DSR ini, jalur yang digunakan adalah jalur yang memiliki jumlah loncatan atar node terkecil yang berarti jalur tersebut adalah jalur terpendek.

2.2.1 Route Discovery

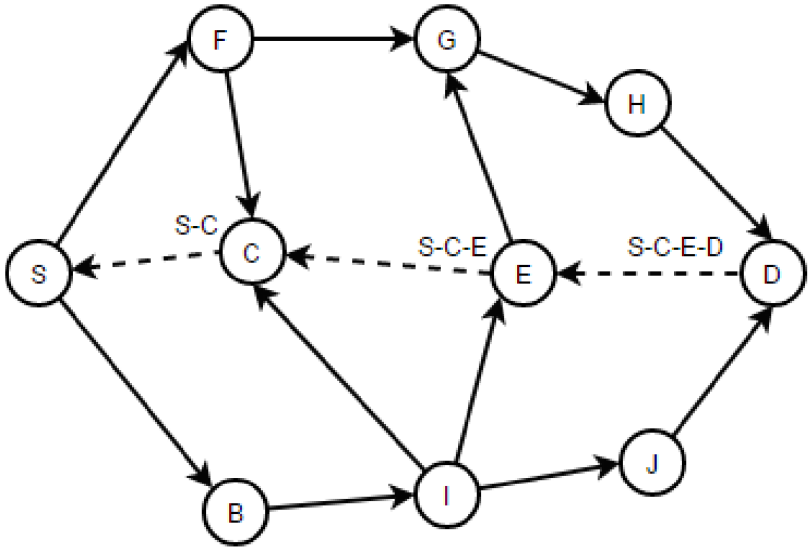
Route discovery adalah inisialisasi awal ketika suatu node ingin mengirimkan data. Node sumber akan melakukan pengecekan rute didalam *route cache* nya, *route cache* ini adalah daftar rute yang pernah disimpan. Ketika di dalam *cache* nya kosong, maka akan dilakukan *route request*(RREQ) secara broadcast keseluruh node.



Gambar 2.1 Proses Route Request

Pada gambar di atas, diasumsikan dari node S ingin mengirimkan data ke node D, sedangkan didalam *route cache* nya tidak terdapat daftar rute yang pernah dipakai, maka node S akan melakukan Route Request, yaitu melakukan *broadcast* paket dengan menambahkan id unik setiap request, dan menambahkan info node setiap melakukan penerusan paket. Misalnya pada gambar di atas, id unik untuk S ke C adalah 2, id unik S ke F adalah 3,dst. untuk info node, akan dilakukan penambahan (*append*) setiap node yang dilewati,seperti ketika S ke C akan menambahakn C untuk dibroadcastkan ke node selanjutnya, sampai menemukan node tujuan D.

Proses selanjutnya adalah *Route Reply* (RREP), RREP terjadi ketika node tujuan menerima RREQ pertama yang sampai pada dirinya. Maka dari informasi tersebut, dikirimlah RREP menuju node sumber melalui node-node sesuai dengan data yang dibawa oleh RREQ tersebut.

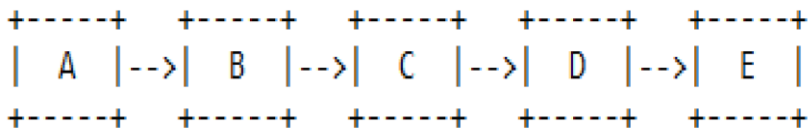


Gambar 2.2 Proses Route Reply

Pada gambar diatas, asumsikan jalur yang diterima adalah S-C-E-D, maka ketika node D menerima RREQ dengan membawa info S-C-E-D, maka node D akan mengirimkan paket RREP dengan membaca info tersebut, dan akan mengirimkan RREQ melalui E, C, dan akhirnya sampai ke node sumber S (digambarkan dengan garis putus-putus). Setelah RREP didapat oleh node sumber, maka rute telah terbentuk dan data siap dikirimkan.

2.2.2 Route Maintenance

Ketika pengiriman paket, atau penerusan paket menggunakan rute yang telah diperoleh dari proses Route Discovery maupun dari *route cache*, setiap node yang mengirimkan paket bertanggung jawab kepada data yang dikirimkan untuk sampai pada node selanjutnya.

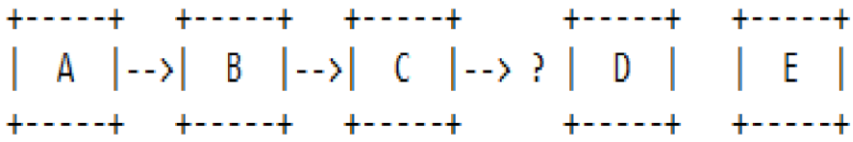


Gambar 2.3 Contoh Route Maintenance

Pada gambar di atas, semisal rute yang terbentuk adalah A menuju E, maka node A bertanggung jawab paket yang dikirimkan harus sampai pada B, dan node B harus memastikan paketnya sampai ke C, dan seterusnya. Dan untuk menjaga jalur antar node, dilakukan pengiriman suatu *acknowledgement request* secara berkala, request ini berada pada link-layer dan biasanya memiliki struktur paket yang sederhana.

2.2.3 Route Error

Route error (RERR) adalah situasi dimana suatu jalur terindikasi rusak oleh suatu node, sehingga node tersebut tidak akan bisa meneruskan paket yang akan dikirim ke node selanjutnya.



Gambar 2.4 Ketika Terjadi Route Error

Pada gambar diatas, Route Error (RERR) terjadi pada node C menuju ke D. Antara node mengirimkan paket *ack* secara berkala untuk menjaga jalur antar node. Akan tetapi setelah beberapa pengiriman ulang paket *ack* menuju node D, dan tidak ada *ack* balasan, yang berarti pada node D, tidak menerima *ack* dari node C, sehingga pada node C, akan memperlakukan jalur selanjutnya (yaitu ke node D) sebagai "*broken link*" atau jalur yang rusak, dan menghapus jalur tersebut dari daftar *route cache* nya. Kemudian node C akan mengirimkan Route Error (RERR) menuju node sumber (node A) melalui jalur yang ada (C - B - A). Dan ketika node sumber (node A) melakukan pengiriman ulang pada tujuan yang sama, node A akan melakukan pengecekan *route cache* nya, apakah ada rute alternatif lainnya yang menuju node yang sama seperti jalur yang mengalami error, yang pernah didapat dari route discovery sebelumnya atau hasil dari menguping (*overhear*) informasi routing dari paket lain. Ketika ternyata tidak ada rute alternatif yang didapat dari *route cache* nya, maka akan dilakukan Route Discovery ulang untuk mendapatkan jalur baru.

2.3 Wireless Access in Vehicular Environment (WAVE) - 802.11p

Didalam VANET, protokol 802.11 standard tidak cukup untuk menangani cepatnya pergerakan yang ada di dalam VANET. Maka diperlukan peningkatan dari 802.11 standard untuk menangani karakteristik dalam VANET, antara lain adalah peningkatan pada layer MAC dan peningkatan pada layer fisik.

2.3.1 IEEE 802.11p (WAVE) MAC Enhancements

Hampir semua peningkatan pada pada IEEE 802.11p standard berkaitan dengan layer MAC, karena pada layer MAC lebih mudah diubah-ubah secara perangkat lunak (*software*) dibanding dengan perubahan pada layer fisik. Karena pada IEEE 802.11 standard MAC membutuhkan waktu yang lebih lama untuk melakukan pemindaian kanal untuk *beacon* dari *Basic Service Set* (BSS) dan melakukan banyak *handshakes* untuk membangun suatu komunikasi. Sedangkan di dalam VANET memiliki lingkungan mobilitas yang sangat tinggi, dan ketika berkomunikasi antar node membutuhkan kemampuan pertukaran data yang sangat cepat, sehingga sangat penting pada semua perangkat IEEE 802.11p memiliki konfigurasi kanal yang memiliki BSSID yang sama untuk keamanan komunikasi tanpa jeda. Contohnya ketika dua node atau kendaraan yang saling berpapasan, waktu dari komunikasi antar mereka mungkin sangatlah pendek,. Di dalam mode WAVE, dapat dilakukan pengiriman dan penerimaan data tanpa perlu mengontak BSS (Maqsood, A. dan Khan, R, 2012).

2.3.2 IEEE 802.11p (WAVE) Physical Layer Enhancements

Pada IEEE 802.11p standard, layer fisik tidak berubah karena jaringan 802.11 telah beroperasi pada 5 GHz dan tidak terlalu sulit untuk merubah konfigurasi jaringan tersebut untuk bekerja pada 5.9 GHz (Maqsood, A. dan Khan, R, 2012). Tujuan merubah layer fisik pada IEEE 802.11 adalah agar perangkat WAVE dapat berkomunkasi secara efisien diantara kecepatan pergerakan kendaraan yang berbeda didalam lingkungan jalan dan minimnya gangguan dari frekuensi lain, misalnya kalau

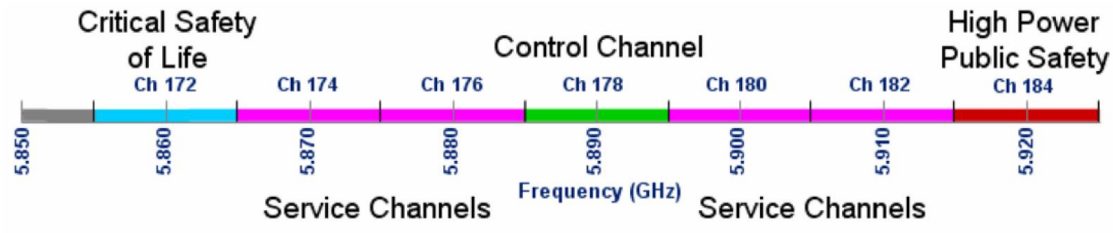
memakai 2.4Ghz, kemungkinan gangguan yang muncul lebih besar karena rata-rata perangkat nirkabel sekarang jalan di frekuensi 2.4Ghz. Disamping itu, peningkatan layer MAC merupakan peningkatan secara perangkat lunak, yang mana mudah untuk dibuat, sedangkan peningkatan layer fisik memerlukan desain keseluruhan dari arsitektur teknologi nirkabel.Oleh karena itu hanya terdapat beberapa peningkatan pada layer fisik seperti kanal frekuensi dan peningkatan kinerja perangkat penerima.

2.3.2.1 Peningkatan pada Kanal Frekuensi

Standar IEEE 802.11p memiliki dasar dari IEEE 802.11a yang menggunakan modulasi OFDM untuk berkomunikasi dengan kanal frekuensi 20 MHz, sedangkan didalam IEEE 802.11p menggunakan kanal 10 MHz. Implementasi dari kanal 10 MHz hanya memerlukan peningkatan OFDM parameter yang digunakan oleh transmisi 20 MHz 802.11a. Tujuan utama dari melakukan peningkatan dari 802.11a ini adalah untuk meningkatan waktu tunda penyebaran RMS didalam lingkungan VANET (Jiang,D dan Delgrossi,2008).

2.3.2.2 Peningkatan Kinerja Penerima

Mereka juga menyebutkan bahwa salah satu masalah yang umum diketahui adalah sifat dari komunikasi nirkabel yang memiliki interferensi lintas kanal, yang memiliki potensi yang saling mengganggu antar kendaraan yang saling bertetangga. IEEE 802.11p menunjukan peningkatan pada kinerja penerima, yang dapat melakukan penolakan kanal ketika terjadi dua kanal yang berdekatan.



Gambar 2.5 Spektrum dan kanal DSRC di U.S. (Sumber : Jiang, D. dan Delgrossi,2008)

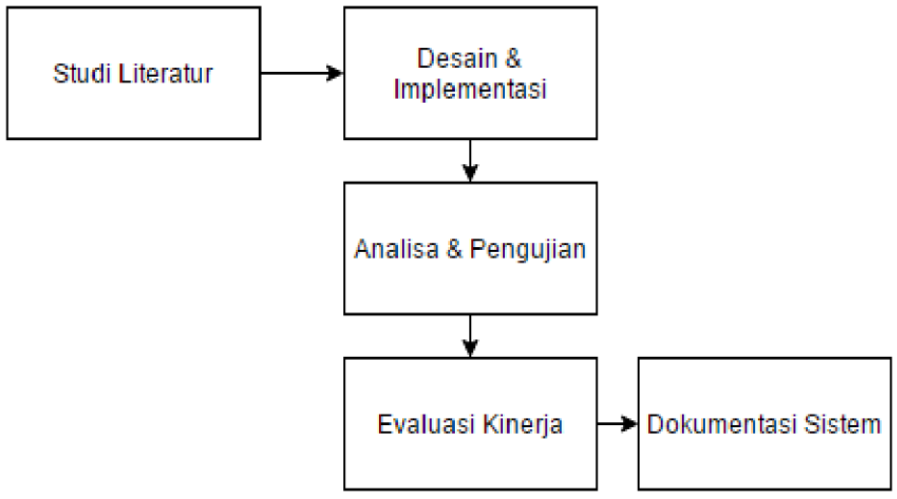
Gambar diatas merupakan spektrum dan kanal dari DSRC , contoh dari interferensi sesuai gambar di atas adalah misal antara kendaraan A menggunakan kanal 176 dapat mengganggu dan menghalangi kendaraan B pada jalur selanjutnya untuk menerima pesan dari kendaraan C yang menggunakan kanal 178.

BAB III

METODOLOGI PENELITIAN

3.1 Langkah-Langkah Penelitian

Tahapan-tahapan dalam penelitian ini dibagi menjadi 5 tahap yaitu studi literatur, desain dan implementasi, analisa dan pengujian, evaluasi kinerja dan dokumentasi sistem. Bagan alur kegiatan penelitian ini dituangkan dalam gambar 3.1 dan pada setiap tahap tersebut saling berkaitan antara satu dengan yang lain, sehingga tingkat keberhasilan pada tiap tahap sangat berpengaruh pada pencapaian tahap selanjutnya



Gambar 3.1 Alur Kegiatan Penelitian

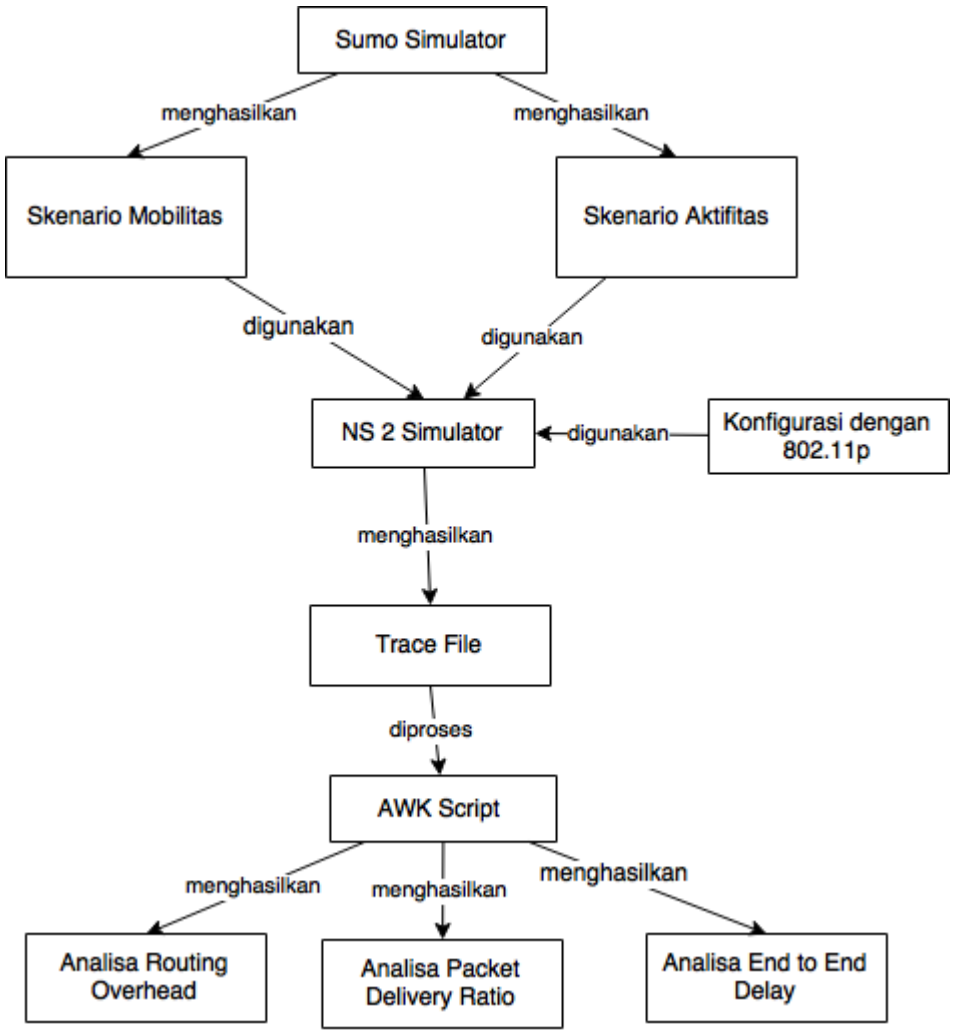
3.2 Studi Literatur

Proses studi literatur ini diawali dengan pengkajian terhadap penelitian-penelitian lain yang telah dilakukan oleh peneliti lain sebelumnya. Referensi yang digunakan adalah jurnal-jurnal yang berkaitan dengan VANET, DSR, dan WAVE. Implementasi DSR pada VANET dengan menggunakan simulator-simulator yang dapat digunakan dalam pengujian untuk melakukan analisa hasil penelitian. Selain itu *paper* pendek yang termuat dalam prosiding-prosiding, artikel ilmiah dan situs-situs penyedia informasi yang terkait juga akan digunakan.

Berdasarkan studi literatur yang telah dilakukan, dapat dijabarkan informasi berkaitan dengan penelitian tentang teknik pembentukan topologi dan model mobilitas pada VANET agar dapat diimplementasi dan disimulasikan untuk kemudian dianalisa tingkat performansinya.

3.3 Desain dan Implementasi

Pembangunan sistem dalam penelitian ini dimulai dengan pembentukan model mobilitas. Pembentukan model mobilitas dilakukan agar protokol dapat diimplementasi dalam bentuk simulasi untuk mengetahui tingkat performansi dari protokol yang diimplementasi.



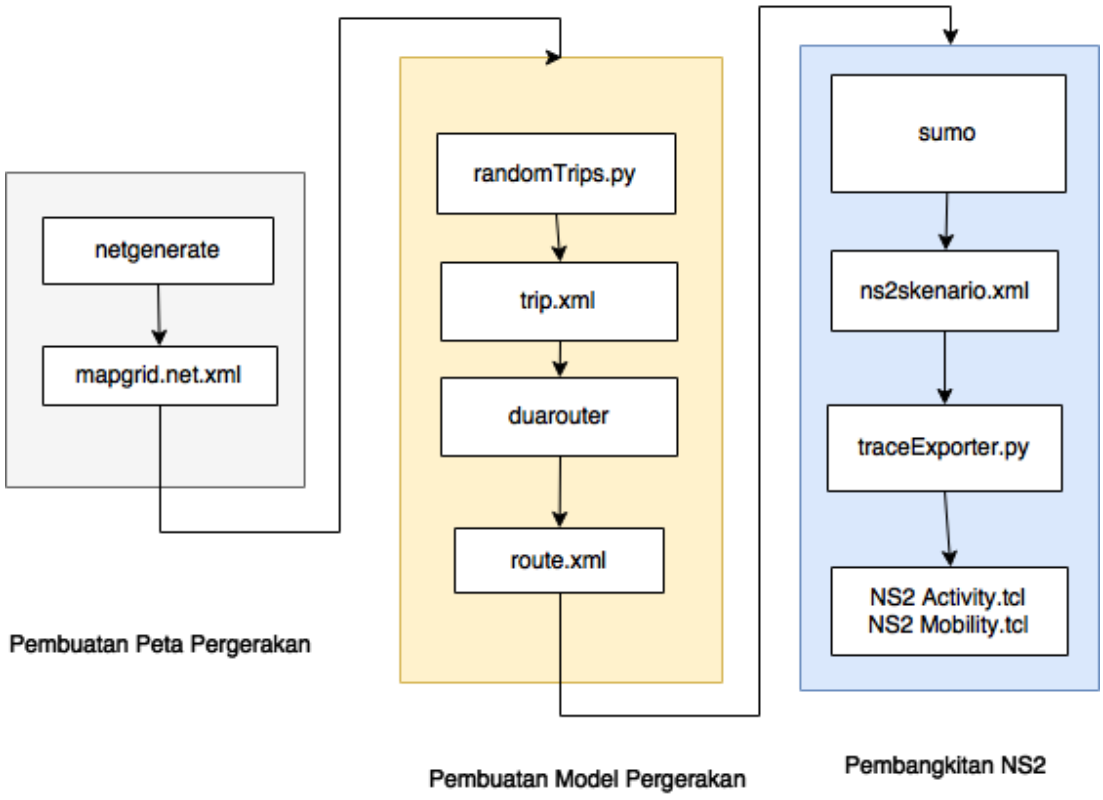
Gambar 3.2 Diagram Alur Rancangan Simulasi

Pada gambar di atas, alur dimulai dengan pembangunan lingkungan pada SUMO Simulator yang terdiri dari pembentukan lingkungan grid (peta simulasi berupa grid, dengan jumlah tikungan yang sudah ditentukan) dan lingkungan jalan rill (peta Surabaya). Setelah didapat skenario mobilitas dan aktifitas, maka dilakukan simulasi dengan NS2 dengan menggunakan protokol routing DSR dengan menggunakan tipe protokol 802.11p atau biasa disebut WAVE (*Wireless Access in Vehicular Environment*) yang akan menghasilkan suatu *trace file* (sebuah file yang memiliki detil aktifitas yang berasal dari NS 2). Di dalam *trace file* ini, terdapat informasi detail aktifitas pengiriman paket, paket yang *drop*, paket routing nya, dll.

Setelah dilakukan analisis dengan skrip awk (skrip yang sebagian besar sudah ada pada sistem operasi Unix, yang biasa digunakan untuk melakukan analisa text dan ekstraksi data) setelah itu didapat data yang dapat digunakan untuk menghitung *routing overhead, packet delivery ratio* (PDR) dan *end-to-end delay*. Dengan hasil dari skrip awk ini, akan dilakukan iterasi dengan setiap skenario dan dimodelkan dalam bentuk grafik.

3.3.1 Alur Skenario Grid

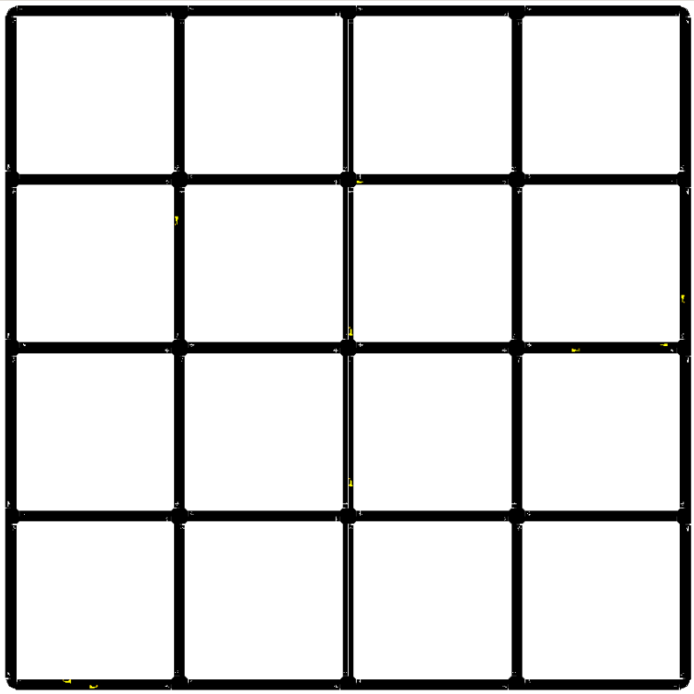
Pengembangan model pergerakan dibagi menjadi beberapa tahap yaitu pengambilan jalur pergerakan, perancangan pergerakan kendaraan dan konversi lalu lintas pergerakan. Langkah-langkah pengembangan model pergerakan dapat dilihat pada gambar 3.3



Gambar 3.3 Proses pengembangan model pergerakan pada skenario grid

Pada gambar diatas, proses pembuatan peta pergerakan dan pembuatan model pergerakan dilakukan pada simulator SUMO (*Simulation of Urban Mobility*) yaitu simulator yang digunakan untuk melakukan testing suatu protokol routing pada VANET, yang mendekati dunia nyata karena terdiri dari macam-macam kendaraan, jalanraya berdasarkan peta (*open street map*) , dll. Setelah itu dilanjutkan untuk konversi pembangkitan *script* NS2 yang akan menghasilkan *trace file* yang akan digunakan untuk analisa performansi. Pada proses pembuatan peta pergerakan, digunakan alat *netgenerate* yang akan melakukan pembuatan peta *mapgrid.net.xml*. Pada proses pembuatan model pergerakan, digunakan *tools* dari SUMo yaitu *randomTrips.py* untuk menghasilkan file pergerakan *route.xml* . Pada pembangkitan NS2, dilakukan pembangkitan skenario NS2 dengan program SUMo, sehingga dihasilkan file *activity.tcl* dan *mobility.tcl*. yang akan kemudian digunakan didalam NS2.

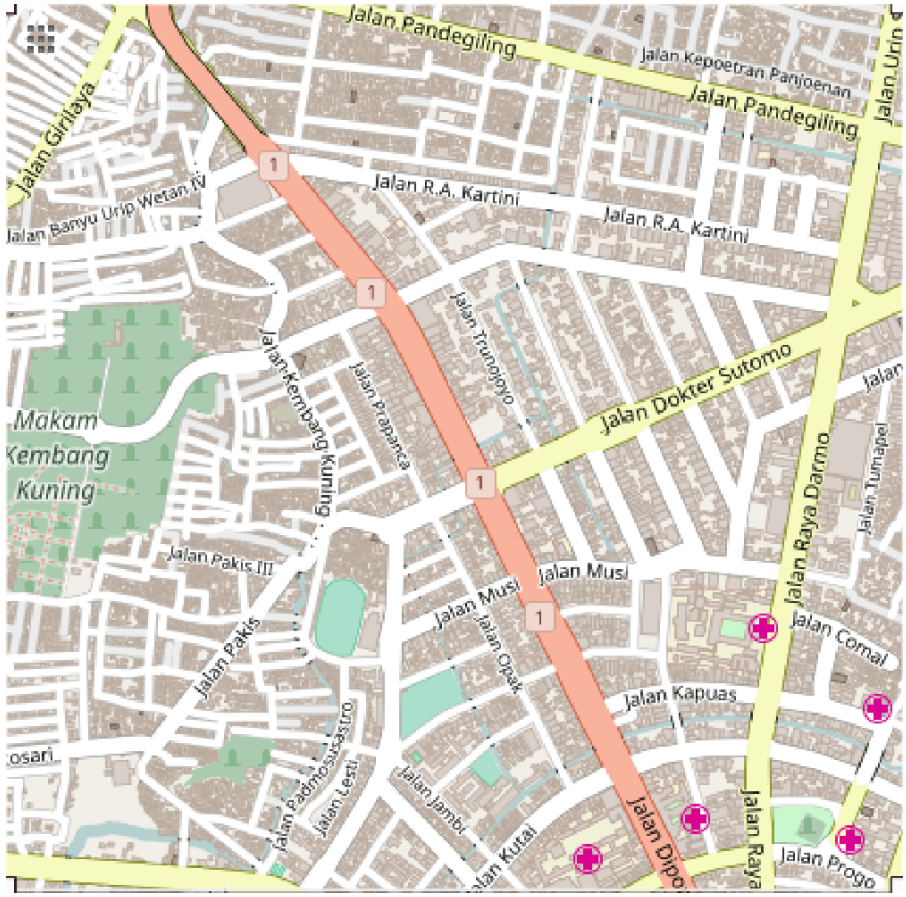
Kemudian dilakukan pengambilan jalur pergerakan dilakukan untuk menentukan jalur pergerakan yang akan dilalui oleh kendaraan pada simulasi sebagai *test bed*. Terdapat dua buah macam jalur yang digunakan, yaitu *grid* dan *real world*. Untuk grid adalah suatu peta pergerakan yang dibuat yang terdiri dari dua buah jalur jalan (*lane*). Dimana setiap lane akan digunakan sebagai jalur pergerakan kendaraan. Lajur kendaraan yang digunakan merupakan 2 jalur dua arah yang saling berlawanan seperti pada gambar 3.4



Gambar 3.4 Contoh Peta Pergerakan Grid berukuran 5x5

Gambar di atas adalah contoh peta grid berukuran 5x5, yang memiliki 25 titik persimpangan. Dilakukan pula pembentukan peta grid lainnya sampai mendekati jumlah titik persimpangan yang dimiliki oleh *real world.*

Skenario jalur pergerakan *real world* yang digunakan sebagai acuan adalah peta kota Surabaya yang merupakan salah satu jalan protokol utama dengan banyaknya persimpangan seperti tampak pada gambar 3.5



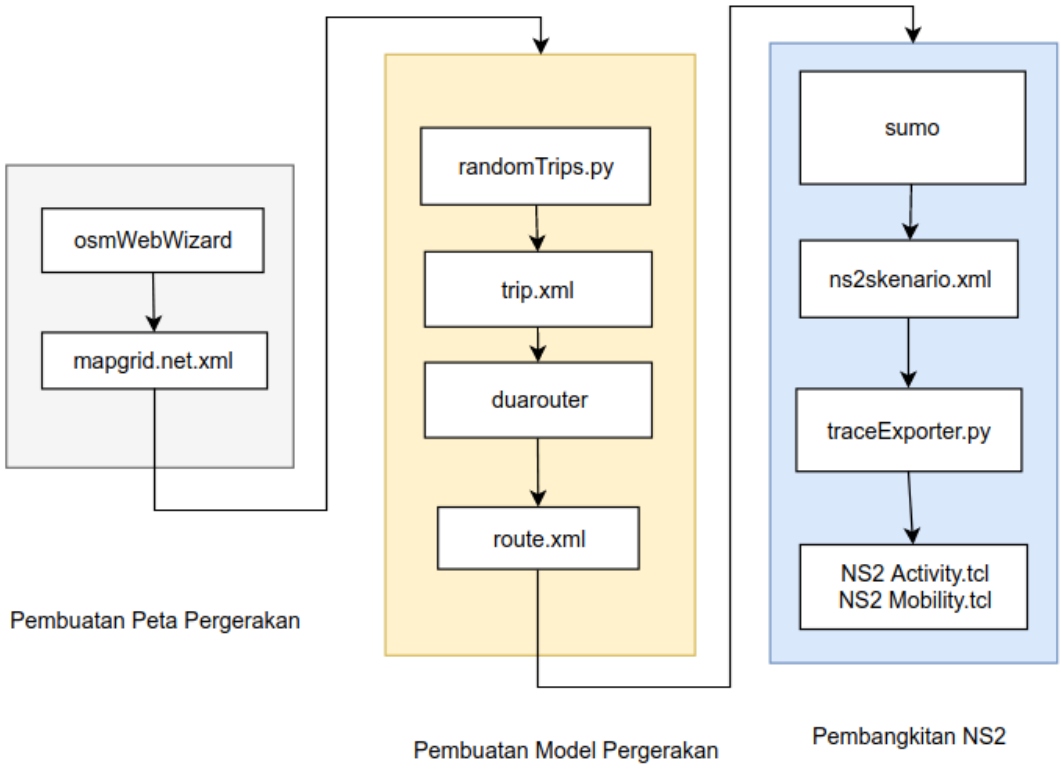
Gambar 3.5 Peta *real world*

Peta yang diambil untuk digunakan dalam skenario berukuran 2000x2000 meter atau 4km2. Dari peta *real world* diatas, diambil banyaknya jumlah persimpangan yang ada, dan kemudian dijadikan acuan simpangan tertinggi. Pada gambar 3.5 didapat jumlah persimpangan sebanyak 686 persimpangan, (ini didapat dengan cara membaca file *net* dari hasildari SUMo, dan ditotal semua inisalisasi persimpangannya yang diapit oleh elemen xml <*junction*></*junction*>).

Dari hasil jumlah persimpangan diatas, diambil kesimpulan bahwa untuk uji coba pada peta grid, akan dilakukan dengan 676 persimpangan (ini dianggap paling mendekati jumlah persimpangan pada *realworld* yang berjumlah 686 buah). Setiap kendaraan ditempatkan secara acak pada tiap-tiap posisi di peta tersebut. Setelah seluru node ditempatkan dan dirancang, dilakukan pembuatan script konfigurasi untuk peta yang akan diajalankan dalam simulator.

3.3.2 Alur Skenario Real World

Pengembangan model pergerakan pada skenario *real* world juga dibagi menjadi tiga tahap yaitu pengambilan jalur pergerakan, perancangan pergerakan kendaraan dan konversi lalu lintas pergerakan. Langkah-langkah pengembangan model pergerakan ini dapat dilihat pada gambar 3.6



Gambar 3.6 Proses pengembangan model pergerakan pada skenario *real world*

Pada gambar diatas, proses pembuatan peta pergerakan dan pembuatan model pergerakan dilakukan pada simulator SUMO (*Simulation of Urban Mobility*) yaitu simulator yang digunakan untuk melakukan testing suatu protokol routing pada VANET. Setelah itu dilanjutkan untuk konversi pembangkitan *script* NS2 yang akan menghasilkan *trace file* yang akan digunakan untuk analisa performansi. Pada proses pembuatan peta pergerakan, digunakan alat *osmWebWizard.py* yang akan melakukan pembuatan peta *mapgrid.net.xml* sesuai dengan area yang kita inginkan. Pada proses pembuatan model pergerakan, digunakan *tools* dari SUMo yaitu *randomTrips.py* untuk menghasilkan file pergerakan *route.xml*. Pada pembangkitan NS2, dilakukan

pembangkitan skenario NS2 dengan program SUMo, sehingga dihasilkan file *activity.tcl* dan *mobility.tcl*. yang akan kemudian digunakan didalam NS2.

3.3.3 Lalu Lintas Pergerakan

Setelah seluruh parameter pada peta kendaraaan didefinisikan, maka seluruh parameter tersebut dikonversi kedalam *script* agar dapat disimulasikan dalam simulator. Parameter simulasi secara lengkap dapat dilihat pada tabel 3.1

Tabel 3.1 Parameter Simulasi pada NS2

No	Parameter	Spesifikasi
1	Simulator	NS2, Versi 2.35
2	Waktu Simulasi	900 detik
3	Area Simulasi	2000m x 2000m
4	Banyaknya Kendaraan / node	30,90,150
5	Radius Transmisi	200m
6	Kecepatan Kendaraan / node	5 m/s - 15 m/s
7	Tipe Mac	MAC 802.11p (WAVE)
8	Tipe Data	Constant Bit Rate (CBR)
9	Protokol Routing	Dynamic Source Routing (DSR)
10	Tipe kanal	Wireless Channel
11	Model Antena	Omnidirectional

Simulasi dilakukan dengan menggunakan simulator NS2 versi 2.35, dengan waktu simulasi yang diambil selama 900 detik. Area simulasi sesuai dengan perancangan yang dibuat dengan luas 4 km^2. Banyaknya kendaraan yang akan diujicobakan dibuat dengan 3 buah kondisi, yaitu dengan banyaknya kendaraan 30,90,150 untuk mendapatkan tingkat kepadatan yang berbeda. Hal ini dilakukan untuk menguji tingkat performansi dari protokol routing yang diimplementasikan terhadap tingkat kepadatan.

Radius transmisi yang digunakan untuk tiap kendaraan adalah sejauh 200 meter. Kecepatan kendaraan bervariasi yaitu 5 m/s, 10 m/s dan 15 m/s. sedangkan tipe data yang digunakan untuk melakukan ujicoba pengiriman paket adalah *constant bit rate* dengan data sebesar 512 bytes. Tipe Mac 802.11a juga diujikan dengan parameter ini, dengan skenario *grid*, untuk menguji seberapa bagus 802.11p dibandingkan dengan 802.11a pada berbagai macam kepadatan dan kecepatan.

3.4 Analisa dan Pengujian

Analisa dilakukan berdasarkan variabel-variabel yang diamati yaitu *packet delivery ratio* (PDR), *routing overhead ratio, end to end delay*, dan *throughput* untuk kemudian dianalisa performansinya dengan protokol routing DSR. Analisa dan pengujian ini terdiri dari 2 skenario, yaitu skenario pada peta *grid* dan pada peta *real world*.

3.4.1 Perhitungan Routing Overhead

Yang dimaksud routing overhead terbanjirinya node-node yang ada dalam VANET / MANET dengan jumlah paket routing, baik itu banyaknya *Route Request*, *Route Reply*, dan Route Error. Untuk perhitungan yang dilakukan adalah dengan membagi jumlah paket routing dengan jumlah paket yang didapat seperti rumus pada persamaan 3.1 dibawah ini.

$$\boldsymbol{RoutingOverhead} = \frac{\sum \boldsymbol{RTRpckts}}{\sum \boldsymbol{CBRrcv}} \qquad (3.1)$$

Dengan :

$\sum RTRpckts$ = jumlah paket RTR (paket yang digunakan untuk mengontrol routing pada protokol MANET.

$\sum CBRrcv$ = jumlah paket yang diterima.

Rasio dari *routing overhead* ini memiliki nilai yang semakin kecil maka semakin bagus.

3.4.2 Perhitungan Packet Delivery Ratio (PDR)

Setiap paket dipengaruhi oleh banyak faktor, misalnya kecepatan node. Hal ini menyebabkan gagalnya suatu pengiriman paket. Dari sini dapat dihitung seberapa paket delivery ratio pada protokol routing tersebut. Dengan demikian, paket delivery ratio dapat dihitung dengan persamaan dibawah ini.

$$\boldsymbol{PacketDelivery} = \frac{\boldsymbol{Packetrecv}}{\boldsymbol{Packetsent}} \quad (3.2)$$

Dengan:

Packetrecv = paket yang diterima

Packetsent = paket yang dikirim

3.4.3 Perhitungan End to End Delay

End-to-end delay adalah waktu rata-rata dari setiap paket yang dihitung berawal dari paket dikirim sampai paket tersebut sampai ke tujuan. End to end ini dihitung dengan persamaan di bawah ini.

$$\boldsymbol{E2EAverageDelay} = \frac{\sum_{i=0}^{n} \boldsymbol{delaypacket}[i]}{\sum \boldsymbol{CBRrcv}} \quad (3.3)$$

Persamaan 3.3 Perhitungan nilai rata-rata *end-to-end* delay

delaypacket = waktu yang dibutuhkan mulai dari paket dikirim sampai ke tujuan

$\sum$*CBRrcv* = jumlah paket yang diterima

BAB IV

HASIL DAN PEMBAHASAN

Dalam Bab IV dipaparkan bagaimana dilakukan implementasi dan evaluasi dari penerapan data, kebutuhan, dan alur sistem berdasarkan desain dan perancangan seperti yang telah dibahas pada bab sebelumnya. Yaitu langkah-langkah uji coba yang dilakukan, hasil uji coba yang telah dilakukan, dan analisis terhadap hasil uji coba yang telah dilakukan.

4.1 Langkah-langkah Implementasi Metode

Pada langkah-langkah implementasi ini, terdapat dua buah lingkungan, yaitu skenario riil dan skenario *grid*. Dan tiap skenario akan dijelaskan bagaimana cara pembuatannya dan kegunaannya pada bab ini.

1. Tahap pertama adalah tahap perancangan VANET berdasarkan skenario riil dan skenario grid dengan menggunakan tools SUMO simulator versi 0.29
2. Tahap kedua adalah pengembangan model pergerakan dan implementasi dari skenario grid yang dihasilkan dari tahap pertama untuk selanjutnya menghasilkan file tcl yang akan digunakan pada NS-2.35.
3. Tahap ketiga adalah melakukan pengujian pada keseluruhan sistem melalui program NS-2. Pada tahap ini dilakukan pengujian pada protokol DSR dengan konfigurasi file tcl yang telah dihasilkan dari tahap pertama dengan menggunakan model pergerakan berdasarkan dari tahap kedua.

4.2 Langkah-langkah Uji Coba

Tujuan dari uji coba dalam penelitian ini adalah untuk mengamati kinerja dari protokol DSR untuk setiap skenario dan untuk setiap kepadatan kendaraan yang berbeda. Kinerja protokol ini dinilai berdasarkan parameter-parameter pengujiannya.

Langkah-langkah uji coba pada penelitian ini diawali dengan penentuan skenario-skenario pengujian, penentuan kepadatan kendaraan dalam setiap skenario, parameter-

parameter pengujian, dan pembuatan skrip simulasi berdasarkan skenario-skenario tersebut, serta diuji coba dalam program Network Simulator 2 versi 2.35

Langkah-langkah pengujian penelitian dijelaskan secara lebih terperinci dalam sub bab-sub bab berikut ini.

4.2.1 Skenario Pengujian

Pemilihan skenario pengujian didasarkan atas kondiisi dan lingkungan yang ada di perkotaan. Terdapat beberapa kondisi lalu lintas yang umum dijumpai dalam lingkungan perkotaan. Kondisi-kondisi tersebut antara lain terdapatnya lampu lalu lintas, kecepatan maksimum kendaraan, dan banyaknya tikungan atau jalan yang bisa dilewati.

Skenario pengujian yang digunakan dalam penelitian ini dibagi berdasarkan model pergerakannya yaitu:

1. Pergerakan node di jalan dengan lampu lalu lintas kepadatan yang bervariasi
2. Pergerakan node di jalan dengan kepadatan serta kecepatan yang bervariasi

Skenario-skenario di atas memiliki kondisi-kondisi yang sesuai dan mirip dengan kondisi lalu lintas yang umum dijumpai di daerah perkotaan. Jalur uji coba kendaraan merupakan jalur bebas hambatan berbentuk grid yang memiliki persimpangan dan tikungan sesuai dengan skenario yang sudah ditentukan.

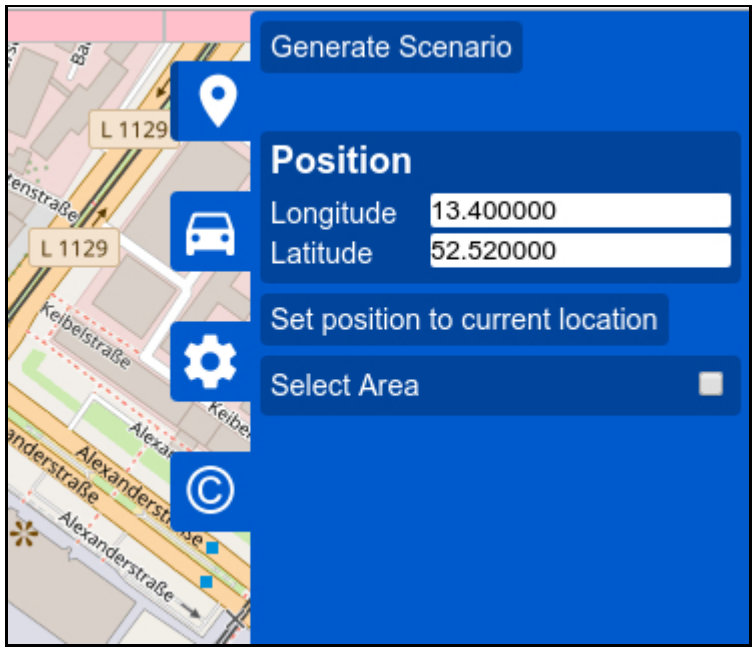
4.2.1 Skenario Riil

Pembuatan peta pada skenario riil ini dihasilkan langsung dari sebuah peta asli. pembuatannya menggunakan SUMo dengan memodifikasi beberapa sintak dalam file **osmWebWizard.py** pada baris 150 dengan menambahkan koordinat yang didapat dari Google Maps dengan koordinat *"-7.2926","112.7237","-7.2748", "112.7419"*, yaitu area persegi dengan luas 2000m x 2000m pada daerah Raya Darmo Surabaya. Penambahan koordinat dilakukan karena ketika dijalankan secara manual untuk mendapatkan peta *OSM* (Open Street Map) pada daerah Surabaya, tidak ada detil yang jelas berapakah luas dari area yang dipilih. Oleh karena itu, dilakukan pencarian koordinat dan luas wilayah secara manual menggunakan Google Maps, dan dimasukkan kedalam skrip **osmWebWizard.py** sehingga setiap dijalankan akan selalu mendapatkan *OSM* sesuai dengan peta yang diinginkan.

```
if 'osm' in self.data:
    # testing mode
    shutil.copy(data['osm'], self.files["osm"])
else:

    self.data["coords"]=["-7.2926","112.7237","-7.2748", "112.7419"]
    self.report("Downloading map data")
    osmGet.get(
        ["-b", ",".join(map(str, self.data["coords"])), "-p", self.prefix])
    print(self.data["coords"])

options = ["-f", self.files["osm"], "-p", self.prefix, "-d", self.tmp]
```

Gambar 4.1 Pengambilan Koordinat wilayah kota Surabaya

Kegunaan dari peta riil ini adalah sebagai acuan yang akan digunakan pada peta dalam bentuk grid karena peta yang dihasilkan memiliki kesamaan dengan situasi dunia nyata (jumlah banyak persimpangannya).



Gambar 4.2 Tampilan ketika menjalankan *osmWebwizard.py*

Ketika dijalankan, *osmWebwizard* akan membuka browser dan menjalankan skripnya, dan karena setelah diset koordinat manual, maka hanya perlu melakukan **Generate Scenario** maka akan langsung menghasilkan beberapa file hasil dari *export* koordinat tersebut (sudah bukan berupa file *osm* lagi). Untuk mendapatkan jumlah persimpangan dari file yang dihasilkan, maka digunakan perintah linux standar. Yaitu ***cat osm.net.xml | grep '/junction>' | wc -l***.

Perintah *cat* digunakan untuk melihat isi file *network* yang dihasilkan, kemudian dilakukan *piping* (|) yaitu menggunakan hasil output dari perintah sebelumnya ke perintah selanjutnya yaitu perintah *grep*, perintah ini digunakan untuk mencari sesuatu teks sesuai dengan pola tertentu (dalam hal ini akan mencari pola *'/junction>'* , tag ini adalah akhir atau penutup dari XML persimpangan) sehingga ketika dihitung, jumlah ini adalah jumlah persimpangan yang terdapat pada *OSM* tersebut. Penghitungannya dilakukan dengan perintah *wc*, yaitu perintah linux untuk menghitung jumlah suatu kata atau baris.

Setelah itu, hasil dari *osmWebWizard.py* ini juga diolah untuk dicek performansinya, dengan menggunakan peta yang didapat, ditambah dengan konfigurasi tambahan untuk menentukan kecepatan maksimal node yang dijalankan.

```
<additional>
    <vType id="myType" maxSpeed="5" vClass="passenger"/>
</additional>
```

Gambar 4.3 Konfigurasi Tambahan pada simulasi *real world*

Pada gambar 4.3 di atas, adalah konfigurasi tambahan untuk mendefinisikan suatu node, dengan kecepatan maksimal 5 m/s, ini akan dilakukan juga pada kecepatan 10 m/s dan 15 m/s sesuai skenario yang telah ditulis.

```
$SUMO_HOME/tools/randomTrips.py -n mymap.net.xml -e 450
--intermediate=10 --trip-attributes='departLane="best"
departPos="random_free" type="myType"' -a 5ms.xml
--edge-permission passenger -o mypassenger.30.trips.xml
```

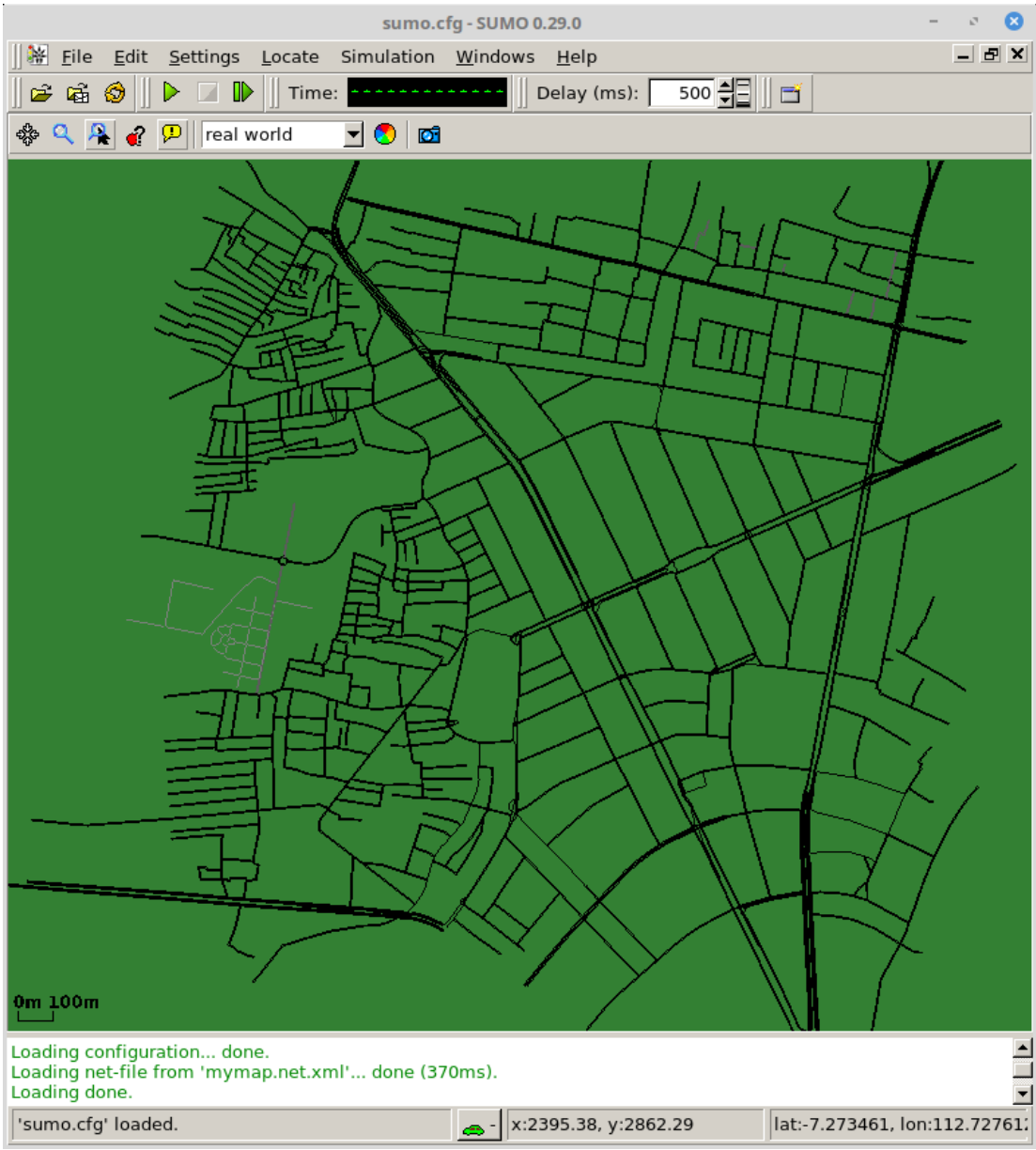
Gambar 4.4 Perintah Untuk Membangkitkan trip pada skenario *real world*

Pada gambar 4.4 di atas, dilakukan pembangkitan skenario rute yang akan dilewati oleh node yang berada pada skenario *real world* secara acak, yang akan menghasilkan file *mypassenger.30.trips.xml*. Setelah dilakukan pembangkitan rute, dibutuhkan pembuatan konfigurasi untuk menjalankan SUMo.

```
<?xml version="1.0" encoding="iso-8859-1"?>

<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="http://sumo.sf.net/xsd/
sumoConfiguration.xsd">
<gui-settings-file value="/home/auzty/Documents/sumodoc/gui-
settings.cfg" />

<input>
<net-file value="mymap.net.xml"/>
<route-files value="myroute.150.rou.xml"/>
</input>

<time>
<begin value="0"/>
<end value="900"/>
</time>

</configuration>
```

Gambar 4.5 Contoh File sumo.cfg pada real world

Pada Gambar 4.5 di atas, adalah contoh dari file konfigurasi sumo yang akan dijalankan pada skema *real world*. Untuk melihat hasil dari konfigurasi sumo pada *real world* tersebut, dapat dilakukan eksekusi perintah ***sumo-gui* -c sumo.cfg**.



Gambar 4.6 Hasil dari menjalankan perintah *sumo-gui*

Dalam *sumo-gui* ini kita dapat melihat apakah parameter-parameter seperti lampu lalu lintas, rute, dan tikungan benar-benar dilewati oleh node. Setelah memastikan bahwa semua parameter telah benar, maka dilakukan pembangkitan file yang akan dieksport untuk NS-2.

```
sumo -c sumo.cfg \
--fcd-output readytoexport.30.sumo.xml
```

Gambar 4.7 Perintah untuk membangkitkan file xml yang siap di export

Pada Gambar 4.7 di atas, akan menghasilkan file xml yang akan dikonversi menjadi file *mobility.tcl* yang akan digunakan pada NS2. File *mobility.tcl* ini akan digunakan untuk menentukan pergerakan node pada NS2 sesuai dengan skenario yang telah bangkitkan pada SUMo.

```
$SUMO_HOME/tools/traceExporter.py --fcd-input
readytoexport.30.sumo.xml -e 450 --ns2config-output 30.tcl
--ns2mobility-output mobility.30.tcl
```

Gambar 4.8 Pembangkitan *mobility.tcl*

Setelah pembangkitan *mobility.tcl* ini, akan dilakukan langkah selanjutnya yaitu pembangkitan skrip pada NS2 untuk diujikan dan dianalisa hasilnya dengan menggunakan parameter kecepatan dan kepadatan kendaraan yang bervariasi.

4.2.2 Skenario Grid

Pada skenario *grid* ini dijelaskan bagaimana skenario yang digunakan untuk membuat suatu rute yang berbentuk *grid* dengan bantuan *tool* netgenerate yang telah disediakan oleh SUMO. Skenario *grid* ini dibuat dengan luas 2000meter x 2000 meter. Jumlah ujung persimpangan antara jalan vertikal dan horizontal bervariasi dengan komposisi yang sama, mulai dari 5x5 (25 tikungan), 10x10 (100 tikungan), 15x15(225 tikungan), 20x20(400 tikungan), 25x25(625 tikungan) dan 26x26(676 tikungan), maksimal dari jumlah tikungan ini didapatkan dari perhitungan jumlah tikungan pada *real world* sebelumnya yang digunakan sebagai acuan jumlah banyaknya tikungan pada skenario *grid*.

```
$ netgenerate --grid true --grid.number 5 \
  --grid.length=400 --output mymap.net.xml
```

Gambar 4.9 Perintah untuk melakukan generate 5x5 dengan luas 2000m x 2000m

Perintah netgenerate akan menghasilkan suatu grid sejumlah 5x5, dengan tiap jalan 400m, sehingga panjangnya adalah 2000m (5x400m). Untuk ukuran 10x10 maka grid.length nya menjadi 200m (sehingga luasnya tetap 2000m x 2000m).

Selanjutnya dibuat file konfigurasi yang berisi detail node yang akan diletakkan pada peta grid.

```
<additional>
 <vType id="car" maxSpeed="15" vClass="passenger"/>
</additional>
```

Gambar 4.10 Konfigurasi tambahan untuk node yang akan ditempatkan pada grid

Konfigurasi file tambahan di atas untuk set batas maksimal kecepatan node yang akan diletakkan pada jalan raya hasil dari netgenerate sebelumnya. Kemudian dilakukan skrip ***randomTrips.py*** untuk membuat posisi dan rute dalam skenario.

```
/home/auzty/sumo-0.29.0/tools/randomTrips.py \
-n mymap.net.xml -e 50 --intermediate=5 \
--trip-attributes='departLane="best" departPos="random_free"' \
--additional-file myadd.add.xml -o mypassenger.trips.xml
```

Gambar 4.11 Perintah untuk pembangkitan posisi node dalam skenario

Skrip dari *randomTrips.py* ini akan melakukan pembangkitan posisi node berdasarkan konfigurasi dari file tambahan konfigurasi sebelumnya, kemudian akan menghasilkan file **mypassenger.trips.xml** yang akan digunakan dalam simulasi sumo.

```
$SUMO_HOME/bin/duarouter -n mymap.net.xml \
-t mypassenger.trips.xml -o myroute.rou.xml \
--repair --ignore-errors --end 900 --keep-all-routes
```

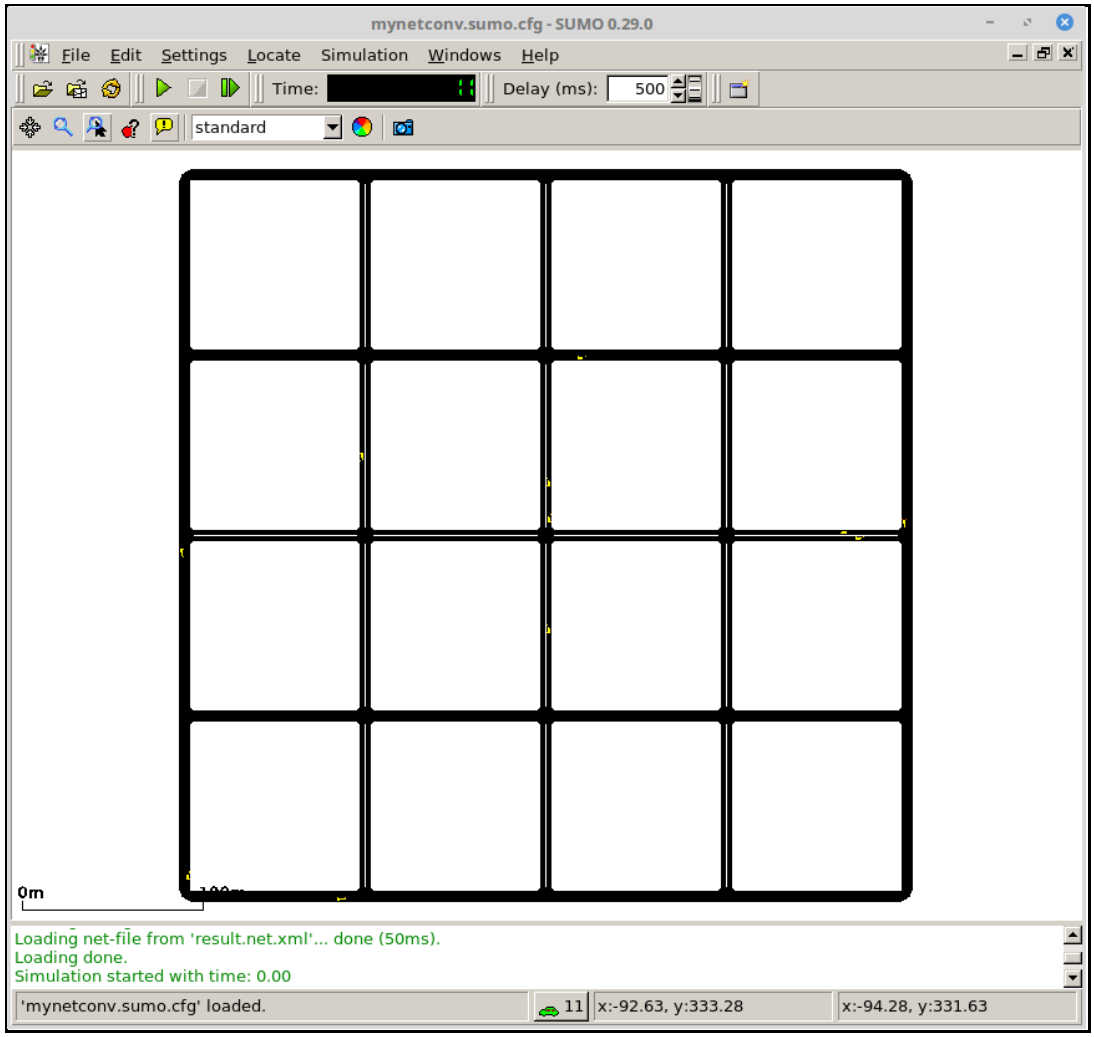
Gambar 4.12 Perintah Duarouter untuk pembangkitan rute node dalam skenario

Perintah duarouter diatas digunakan untuk melakukkan pembangkitan rute node berdasar dari pembangkitan posisi node sebelumnya. Perintah ini menghasilkan file **route.xml** dan **route.alt.xml**. Selanjutnya dilakukan pembuatan file sumo.cfg dimana file ini nantinya akan digunakan sebagai argumen dalam perintah SUMO. Adapun fungsi dibuatnya file ini adalah untuk mendefinisikan lokasi file-file yang dibutuhkan. isi dari sumo.cfg ini dapat dilihat pada Gambar 4.13

```
<?xml version="1.0" encoding="iso-8859-1"?>

<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://sumo.sf.net/xsd/sumoConfiguration.xsd">

<input>
<net-file value="result.net.xml"/>
<route-files value="route.xml"/>
</input>

<time>
<begin value="0"/>
<end value="900"/>
</time>

</configuration>
```

Gambar 4.13 Konfigurasi file sumo.cfg pada grid

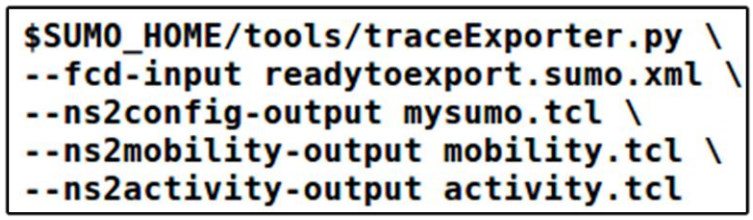
Setelah ini, dapat dilakukan simulasi pada SUMO, simulasi ini berbentuk visualisasi lalu lintas sesuai dengan definisi file konfigurasi diatas. Untuk menjalankan visualisasi ini digunakan program *sumo-gui*, yaitu program GUI (*Graphical User Interface*) yang sudah disediakan oleh SUMO dan dapat dijalankan dengan perintah seperti pada skenario riil. Untuk contoh tampilan ketika dijalankan dapat dilihat pada Gambar 4.14.



Gambar 4.14 Contoh tampilan *sumo-gui* ketika dijalankan

Kemudian dilakukan proses konversi agar dapat digunakan pada NS-2, yaitu menjadi format file (*.tcl*). Untuk melakukan proses konversi ini, digunakan skrip dari sumo yaitu *traceExporter.py*. Sebelumnya digunakan export dengan berdasarkan data-data yang disebutkan dalam file konfigurasi sumo dengan perintah seperti pada skenario riil yaitu ***sumo -c sumocfg –fcd-output readytoexport.sumo.xml***.

Perintah di atas akan menghasilkan file xml yang digunakan oleh skrip *traceExporter.py*. Skrip ini membutuhkan input file xml dan akan menghasilkan file tcl, yang nantinya akan digunakan dalam NS-2 untuk menjalankan simulasi yang digunakan.



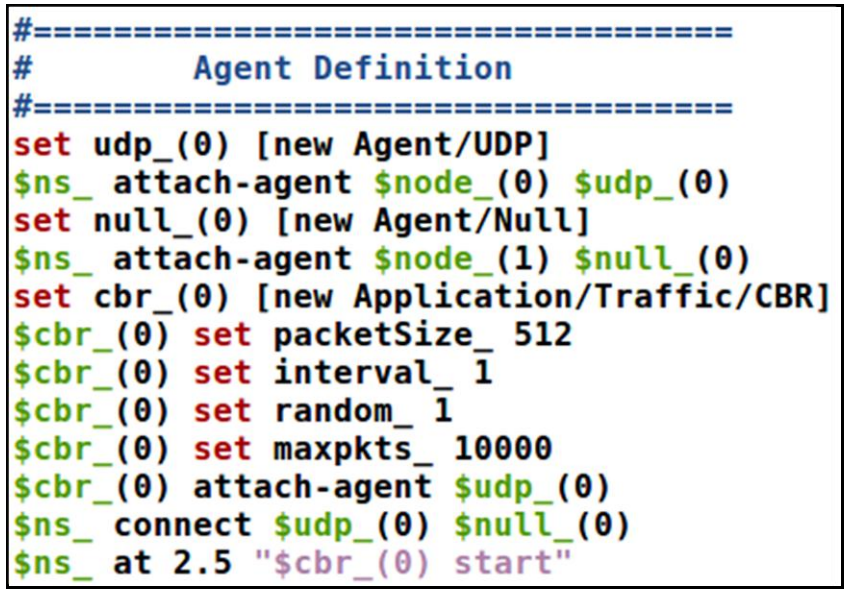
Gambar 4.15 Perintah konversi dari file xml ke tcl

Pada gambar 4.15 di atas merupakan perintah unguk konversi dari xml menjadi tcl. Hasil dari konversi file diatas menghasilkan *mobility.tcl* yang berisi data-data pergerakan pada tiap node dan file *activity.tcl* yang merupakan data-data aktifitas node pada NS-2.

4.3 Pembangunan Skrip untuk Pengujian NS-2

Untuk Menjalankan simulasi Wireless pada NS-2, dibutuhkan file konfigurasi yang berupa *tcl* file, untuk contoh dari wireless tcl dapat diambil atau mengacu pada lokasi folder NS-2 ns-2.35/tcl/ex/simple-wireless.tcl dan dibutuhkan beberapa modifikasi seperti yang diperlukan seperti koneksi CBR, protokol *routing* yang digunakan (DSR) dan konfigurasi sesua protokol *routing* tersebut.

Pada skrip tcl yang akan digunakan, diperlukan konfigurasi koneksi CBR dengan menggunakan inisialisasi *Agent* UDP. dimana koneksi UDP ini akan digunakan sebagai konfigurasi antar node kedua ke node ketiga yang memiliki interval pengiriman paket yang dilakukan tiap detik dengan besar paket 512 bytes dengan maksimal 10000 paket. Seperti yang terlihat pada gambar 4.16 dibawah ini.



Gambar 4.16 Potongan Konfigurasi CBR pada skrip tcl

Potongan konfigurasi CBR diatas akan dimulai pada detik ke 2.5 pada NS-2, dan akan mencoba mengirimkan paket berukuran 512bytes setiap 1detik dan akan dikirimkan dari *node_1* menuju *node_2*. Untuk menjalankan simulasi VANET pada lingkungan NS-2 dengan implementasi DSR, diperlukan beberapa konfigurasi pada skrip tcl nya seperti dibawah ini.

```
set val(chan)   Channel/WirelessChannel ;
set val(prop)   Propagation/TwoRayGround;
set val(ant)    Antenna/OmniAntenna ;
set val(ll)     LL ;
set val(ifq)    CMUPriQueue ;
set val(ifqlen) 50 ;
set val(netif)  Phy/WirelessPhyExt ;
set val(mac)    Mac/802_11Ext ;
set val(nn)     50 ;
set val(rp)     DSR ;
set val(x)      2000 ;
set val(y)      2000 ;
set val(stop)   900.0  ;

$ns_ node-config  -adhocRouting $val(rp) \
        -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -topoInstance $topo \
                -agentTrace OFF \
                -routerTrace ON \
                -macTrace ON \
                -movementTrace OFF \
                -channel $chan_1
```

Gambar 4.17 Potongan Konfigurasi node NS-2 dengan protokol DSR

Penjelasan dari potongan konfigurasi NS-2 diatas dapat dilihat pada tabel 4.1. Konfigurasi lainnya pada file *tcl* tersebut antara lain lokasi penyimpanan *trace file,* lokasi *file* mobilitas, dapat dilihat pada lampiran A.1. Untuk implementasi VANET menggunakan protokol DSR, diperlukan beberapa konfigurasi pada layer fisik dan layer mac, seperti pada Gambar 4.18 di bawah ini.

```
Phy/WirelessPhyExt set CSThresh_                7.5195e-12
Phy/WirelessPhyExt set RXThresh_                1.15361e-10
Phy/WirelessPhyExt set Pt_                      0.281838
Phy/WirelessPhyExt set CPThresh_                10
Phy/WirelessPhyExt set freq_                    5.9e+9
Phy/WirelessPhyExt set noise_floor_             1.26e-13
Phy/WirelessPhyExt set L_                       1.0
Phy/WirelessPhyExt set PowerMonitorThresh_      3.80675e-11
Phy/WirelessPhyExt set HeaderDuration_          0.000040
Phy/WirelessPhyExt set BasicModulationScheme_   0
Phy/WirelessPhyExt set PreambleCaptureSwitch_   1
Phy/WirelessPhyExt set DataCaptureSwitch_       0
Phy/WirelessPhyExt set SINR_PreambleCapture_    3.1623
Phy/WirelessPhyExt set SINR_DataCapture_        10.0;
Phy/WirelessPhyExt set trace_dist_              1e6
Phy/WirelessPhyExt set PHY_DBG_                 0

Mac/802_11Ext set CWMin_                        15
Mac/802_11Ext set CWMax_                        1023
Mac/802_11Ext set SlotTime_                     0.000013
Mac/802_11Ext set SIFS_                         0.000032
Mac/802_11Ext set ShortRetryLimit_              7
Mac/802_11Ext set LongRetryLimit_               4
Mac/802_11Ext set HeaderDuration_               0.000040
Mac/802_11Ext set SymbolDuration_               0.000008
Mac/802_11Ext set BasicModulationScheme_        0
Mac/802_11Ext set use_802_11a_flag_             true
Mac/802_11Ext set RTSThreshold_                 2346
Mac/802_11Ext set MAC_DBG                       0
```

Gambar 4.18 Konfigurasi *Mac Layer* dan *Phyisical Layer*

Konfigurasi pada layer fisik dan layer mac diperlukan sesuai dengan standar 802.11p (VANET) seperti pengaturan frekuensi yang digunakan (5.9Ghz), minimal dan maksimal *contention window*, semua mengacu pada standar 802.11p yang tidak ada dalam konfigurasi NS-2 default.

Tabel 4.1 Penjelasan dari Parameter pengaturan node pada file tcl.

Parameter	**Nilai**	**Deskripsi**
adhocRouting	DSR	Menggunakan protokol DSR
llType	LL	Menggunakan Link-Layer standar.

macType	Mac/802_11Ext	Menggunakan konfigurasi Mac/802_11 dengan konfigurasi untuk 802_11p
ifqType	CMUPriQueue	Interface Queue untuk protokol DSR (melakukan pengaturan prioritas pada paket)
ifqLen	50	maksimal panjang paket queue
antType	Antenna/OmniAntenna	tipe antena yang digunakan bertipe omni
propType	Propagation/TwoRayGround	propagasi menggunakan *tworayground* model
phyType	Phy/WirelessPhyExt	menggunakan interface *wireless* dengan beberapa konfigurasi yang sesuai dengan 802.11p
channel	Channel/WirelessChannel	tipe kanal yang digunakan adalah kanal wireless default

Tabel 4.2 Penjelasan Layer Fisik dan Layer Mac sesuai standar 802.11p

CSThresh_	7.5195e-12	ambang batas Carrier Sensing 440m (paket akan dapat di*decode* dibawah 440m)
RXThresh_	1.15361e-10	ambang batas dari penerimaan transmisi node (200m paket akan diterima)

CPThresh_	10	ambang batas Capture Packet (ketika terjadi tabrakan paket yang diterima, maka yang diambil adalah paket yang memiliki nilai kekuatan sinyal 10x lipat)
Pt_	0.281838	Kekuatan Transmisi dari node (max 250m)
freq_	5.9e+9	frekuensi standar 802.11p (5.9 Ghz)
noise_floor_	1.26e-13 (-99 dBm)	Noise floor dari 802.11p standar (10 Mhz)
L_	1.0	nilai default gain/loss untuk sirkuit radio
PowerMonitorThresh_	3.981071705534985e-18 (-174 dBm)	level dari noise yang digunakan
HeaderDuration	0.000040	40 micro seconds
BasicModulationScheme_	0	skema modulasi dasar
PHY_DBG_	0	tidak akan menulis debug layer fisik pada trace file
CWMin_	15	*Contention WIndow* minimal
CWMax_	1023	*Contention Window* maksimal

Untuk implementasi 802.11p dari simulator SUMo, maka hanya perlu menambahkan *file* mobilitas yang didapat dari langkah sebelumnya (mobility.tcl) yang berisi detil dari pergerakan sesuai dengan simulator SUMo pada NS-2 yaitu dengan menyisipkan skrip ***source mobility.tcl.***

Penambahan skrip mobilitas tersebut adalah cara menambahkan *file* mobilitas ke dalam skrip *tcl* milik ns, sehingga waktu menjalankan *ns* akan menghasilkan pergerakan sesuai dengan skenario yang didapat dari SUMo.

4.4 Hasil dan Analisis

Seluruh hasil dari pengujian didapatkan dengan mengelola file *trace* yang dihasilkan oleh simulator NS-2 ketika menjalankan skrip pengujian. Hasil dari simulator tersebut kemudian diolah dengan program AWK untuk mendapatkan nilai-nilai dari parameter-parameter uji coba. Data tersebut kemudian dibentuk menjadi tabel dan disajikan dalam bentuk grafik garis untuk memudahkan dalam pembacaan analisa.

Sebuah skrip awk dirancang dan dibangun untuk membaca hasil penelusuran file *trace* NS-2. Hasil dari menjalankan file AWK ini berupa sebuah file teks yang berisikan informasi tentang *routing overhead*, *packet delivery ratio* dan *end-to-end delay*. Gambar 4.19 mengilustrasikan hasil dari menjalankan skrip AWK tersebut.

```
$ awk -f ../analisa.awk dsr-out.tr
Total Packet Sent : 1825
Total Packet Received : 830
Packet Delivery Ratio : 45.4795%
Routing Overhead Ratio : 16.559
End-to-end Delay nya : 6.60359 ms
```

Gambar 4.19 Hasil dari menjalankan skrip AWK untuk simulasi skenario pada protokol DSR

Dari gambar diatas, terlihat hasil simulasi dari protokol DSR yang mengimplementasi VANET (802.11p) pada saat penelusuran hasil tracefile dilakukan. Pada output

menampilkan informasi dari banyaknya paket yang diterima, *routing overhead, packet delivery ratio* dan *end-to-end delay*. Nilai-nilai tersebut yang dimasukkan ke dalam tabel dan dibentuk menjadi grafik berbentuk garis. Untuk hasil yang lebih akurat, digunakan 10x percobaan untuk tiap-tiap skenario dan diambil nilai rata-rata dari masing-masing percobaan dan disajikan dalam tiap-tiap tabel yang bersangkutan.

4.4.1 Hasil dan Analisis pada Skenario *grid*

Pada analisa pertama ini, dilakukan pengujian pada skenario *grid*, meliputi kecepatan yang bervariasi, yaitu 5 m/s, 10 m/s dan 15 m/s, kepadatan node yang bervariasi (30 nodes, 90 nodes dan 150 nodes) sesuai dengan skenario yang telah dipersiapkan sebelumnya. Pada skenario grid ini, dilakukan juga perbandingan antara hasil dari 802.11p dan 802.11a dalam berbagai variasi kecepatan dan kepadatan.

4.4.1.1 Analisis Packet Delivery Ratio (pdr)

Pada parameter pertama yang dibandingkan adalah *packet delivery ratio*, hasil dari pengujian didapatkan dalam bentuk teks kemudian disajikan kedalam Tabel 4.3. Tabel tersebut terbagi menjadi 2 buah skenario, dan masing-masing skenario menunjukkan jumlah kendaraan yang berbeda-beda. Jumlah kendaraan terkecil adalah 30, kemudian meningkat menjadi 90 dan 150. Pengujian juga dilakukan berdasarkan kecepatan yang berbeda-beda, yaitu 5m/s, 10 m/s dan 15 m/s. Data yang tersaji dalam tabel dimasukkan ke dalam grafik garis untuk mempermudah dilakukan analisis. Grafik dibagi berdasarkan jumlah kendaraan, tipe protokol dan kecepatan. Pada gambar 4.17 digambarkan grafik untuk *packet delivery ratio* .

Untuk perhitungan yang digunakan, adalah dengan cara menghitung banyaknya paket yang diterima dibagi dengan banyaknya paket yang terkirim, kemudian dikali 100% untuk mendapatkan hasil dalam persen. contoh perhitungan PDR dilakukan sebagai berikut:

Untuk menghitung PDR, menggunakan formula sebagai berikut:

PDR (%) = (Total Packet Received / Total Packet Sent) 100%

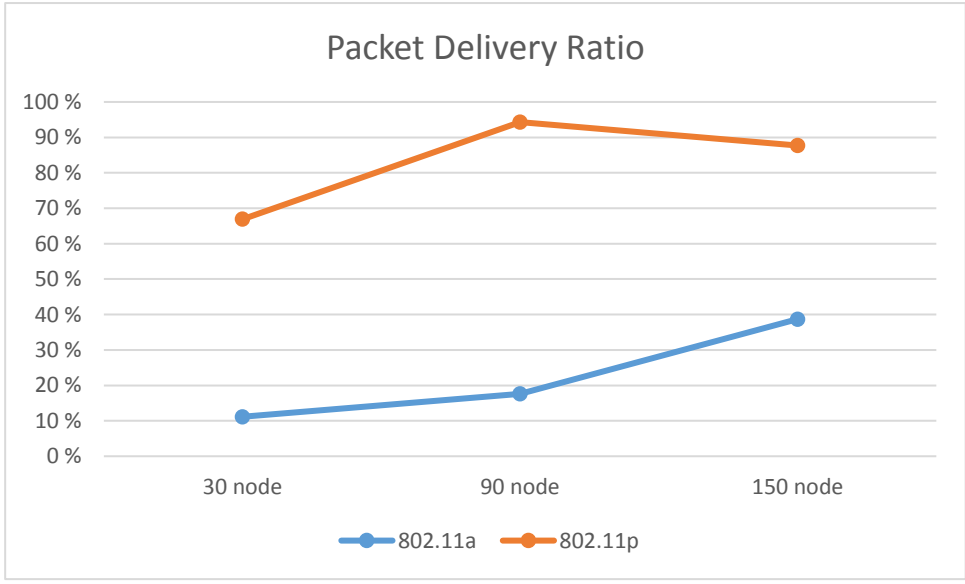
PDR (%) = (851/ 1800) x 100%

PDR (%) = 47.27798 %

Tabel 4.3 *Packet Delivery Ratio* Hasil Pengujian pada waktu kecepatan 5 m/s

	30 node	90 node	150 node
802.11a	11.14431	17.57479	38.699095
802.11p	66.90156	94.31216	87.71378

Dari Tabel 4.3. yang memuat hasil perhitungan PDR dari keseluruhan skenario pada kecepatan 5 m/s, dapat dilihat bahwa perubahan PDR untuk setiap skenario sesuai dengan banyaknya node. Faktor kepadatan kendaraan inilah yang mendorong bervariasinya PDR pada tiap skenario, yaitu semakin sedikit node, maka paket yang terkirim akan semakin susah untuk terkirim karena dibutuhkan node-node yang terhubung. Semakin banyak tikungan (dengan luas wilayah tetap) menyebabkan semakin banyaknya tikungan akan memperpendek jarak tikungan sehingga dengan node yang sedikit. Terlihat juga bahwa protokol 802.11a memiliki rata-rata *packet delivery ratio* yang buruk dibandingkan dengan 802.11p.



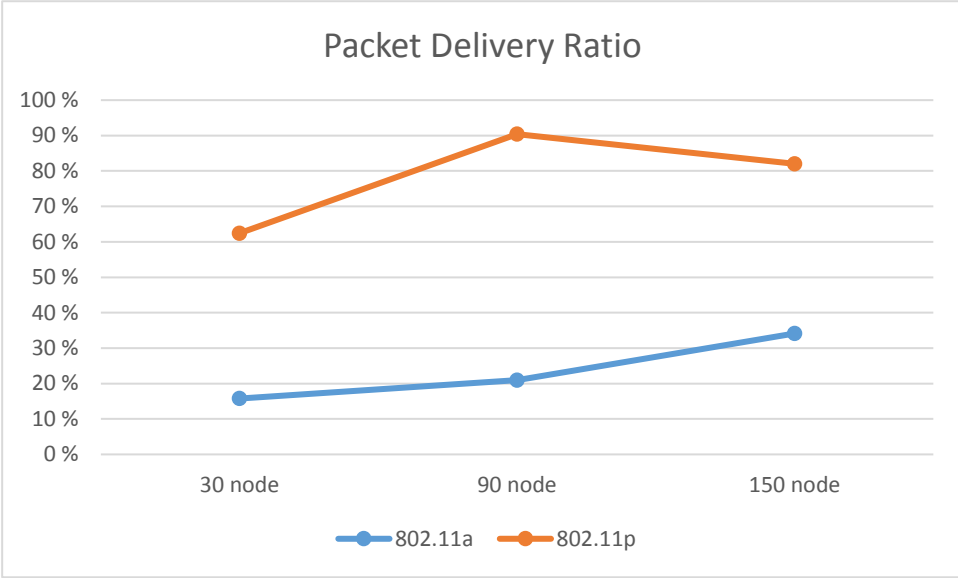
Gambar 4.20 Grafik *Packet Delivery Ratio* (5 m/s)

Gambar 4.20 adalah grafik yang mengilustrasikan nilai-nilai PDR dalam skenario 1. Terlihat bahwa pergerakan dari PDR memiliki nilai-nilai yang bervariasi untuk tiap kepadatan. Pada 802.11p waktu node berjumlah 30, didapat PDR sebesar 66.90156 %. Pada waktu 90 node, nilai PDR sebesar 94.31216 % dan ketika node nya berjumlah 150, didapat nilai PDR sebesar 87.71378 %. pada perhitungan *packet delivery ratio* ini, semakin tinggi rasio persentasinya, maka performansinya semakin baik, disini ditunjukkan pada waktu node berjumlah 90 buah, memiliki PDR yang paling baik pada protokol 802.11p. Sedangkan untuk 802.11a , memiliki performa yang buruk, bahkan pada kepadatan tinggi (150 node) , hanya memiliki keberhasilan pengiriman paket kurang dari 40%

Tabel 4.4 Packet Delivery Ratio (10 m/s)

	30 node	90 node	150 node
802.11a	15.76955	20.99421	34.1282
802.11p	62.45241	90.39031	82.02466

Dilakukan juga pengujian pada kecepatan 10m/s pada skenario 1. Pada tabel 4.4 di atas, menunjukkan hasil dari pengujian dampak dari kecepatan terhadap jumlah node yang bervariasi. Adapun grafiknya adalah seperti ditunjukkan pada gambar 4.21 dibawah ini.



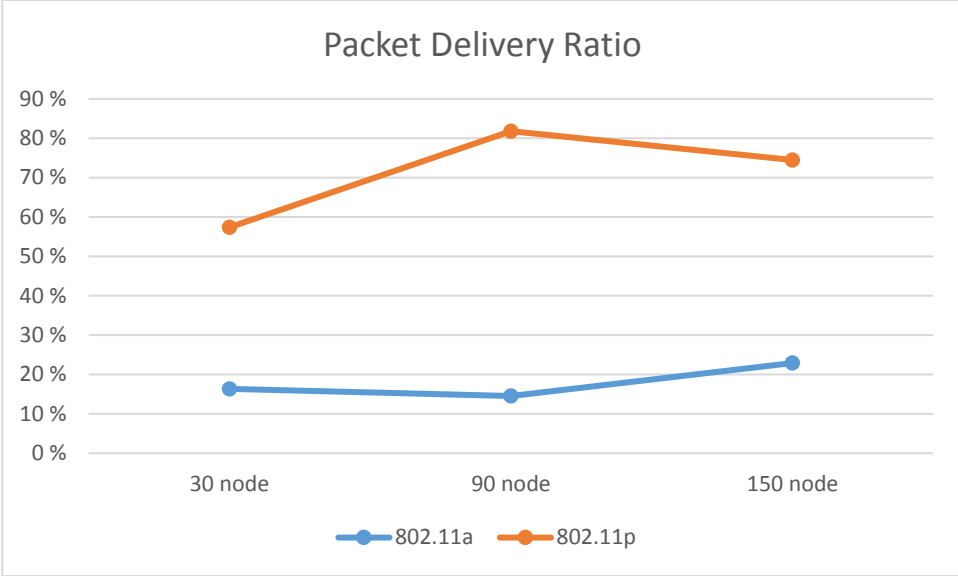
Gambar 4.21 Packet Delivery Ratio pada kecepatan 10 m/s

Pada grafik gambar 4.21 di atas, ditunjukkan bahwa *Packet Delivery Ratio* untuk 802.11p paling baik terjadi ketika kepadatan node berjumlah 90, sedangkan untuk 802.11a, memiliki performa jauh lebih buruk dibanding dengan 802.11p, bahkan pada waktu node sebanyak 150 buah, memiliki keberhasilan kurang dari 35% dalam pengiriman paket.

Tabel 4.5 Packet Delivery Ratio (15 m/s)

	30 node	90 node	150 node
802.11a	16.33934	14.5331	22.902359
802.11p	57.37797	81.80869	74.47056

Dilakukan juga pengujian pada kecepatan node sebesar 15 m/s, dan ditampilkan hasilnya pada Tabel 4.5 di atas.



Gambar 4.22 *Packet Delivery Ratio* ketika kecepatan node 15 m/s

Pada grafik Gambar 4.22 di atas, terlihat bahwa pada kecepatan yang lebih tinggi ini, 802.11p tetap jauh lebih unggul (paling baik ketika node berjumlah 90) dibandingkan dengan 802.11a yang walupun pada waktu node berjumlah 150 buah, keberhasilan pengiriman paketnya kurang dari 25%. Ini disebabkan karena semakin tinggi kecepatannya maka akan semakin rendah rasio pengiriman paketnya, karena jeda perpindahan paket / pengiriman paket antar node menjadi semakin kecil.

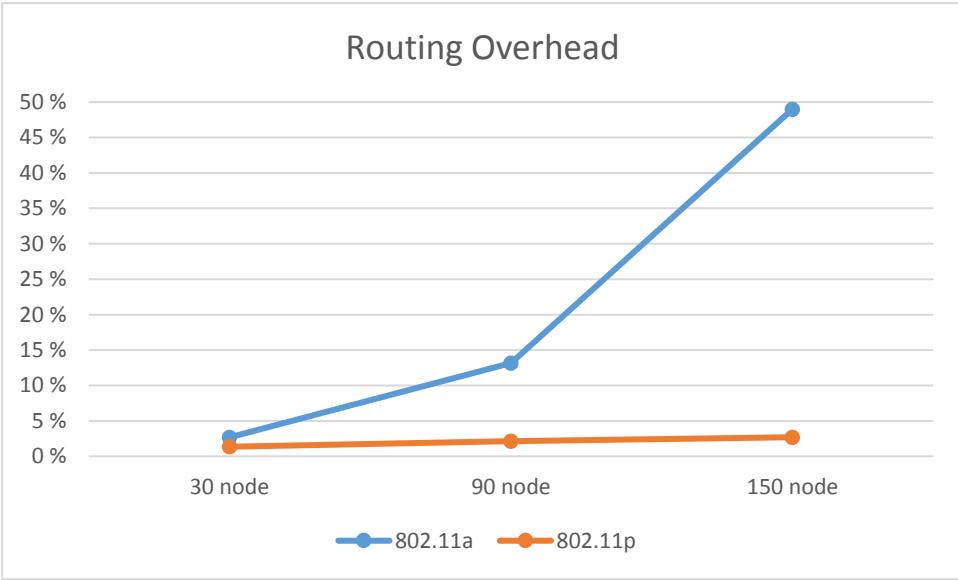
4.4.1.2 Analisis Routing Overhead (RO)

Routing Overhead atau biasa disebut jg beban jaringan atau *network load*, dapat dihitung dengan cara mencari banyaknya paket yang diterima dan dibandingkan dengan banyaknya paket yang melewati router (proses routing). Rasio dari banyaknya paket yang diterima dengan banyaknya paket yang di-*routing* disebut dengan beban jaringan yang dinormalisasi. Semakin rendah rasionya, maka semakin rendah paket yang dirouting pada jaringan tersebut, sedangkan, semakin banyak paket yang di*routing*, maka semakin tinggi beban jaringannya atau *overhead* nya.

Tabel 4.6 *Routing Overhead* Hasil Pengujian pada kecepatan 5 m/s

	30 node	90 node	150 node
802.11a	2.678054	13.16419	48.998746
802.11p	1.356476	2.154325	2.6934083

Tabel 4.6 Menunjukkan rasio dari beban jaringan yang telah diujicobakan. Untuk masing-masing skenario, mulai dari skenario 1 yaitu banyaknya jumlah node, *routing overhead* cenderung semakin tinggi dengan semakin banyaknya jumlah node. Terlihat juga bahwa pada 802.11a, memiliki *routing overhead* relatif jauh lebih besar dibandingkan dengan 802.11p,

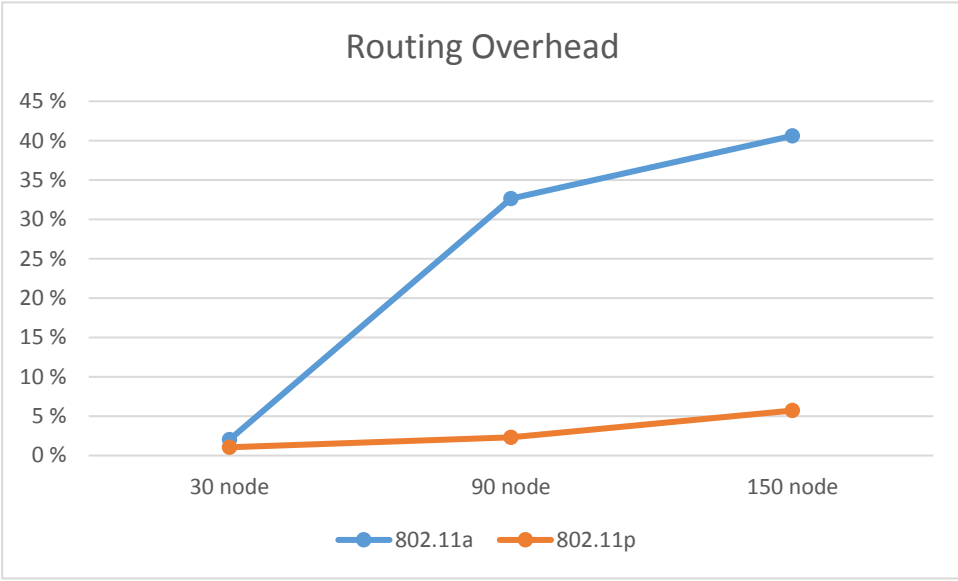


Gambar 4.23 Grafik *Routing Overhead* pada kecepatan 5 m/s

Gambar 4.23 diatas menunjukkan bahwa *routing overhead* paling kecil dimiliki oleh node yang berjumlah 30 karena semakin sedikit routing yang dilakukan. Pada 802.11a juga menunjukkan *routing overhead* paling sedikit pada node berjumlah 30, namun langsung melonjak naik ketika semakin padatnya dan banyaknya jumlah node.

Tabel 4.7 *Routing Overhead* pada kecepatan 10 m/s

	30 node	90 node	150 node
802.11a	2.017306	32.64885	40.61208
802.11p	1.051128	2.304273	5.715118

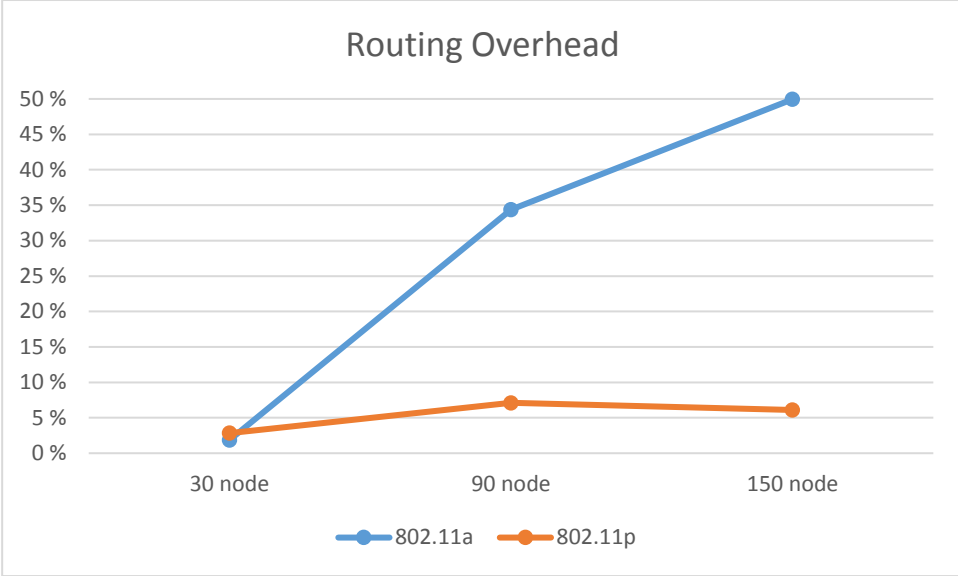


Gambar 4.24 *Routing Overhead* pada kecepatan 10 m/s

Terlihat juga pada Gambar 4.24 di atas, bahwa semakin banyaknya node menyebabkan semakin tingginya *routing ovherhead*. Pada Gambar 4.24 di atas juga menunjukkan bahwa 802.11a memiliki *routing overhead* yang lebih buruk dibandingkan dengan 802.11p.

Tabel 4.8 *Routing Overhead* pada kecepatan 15 m/s

	30 node	90 node	150 node
802.11a	1.850781	34.39307	49.95325
802.11p	2.833199	7.112559	6.100852



Gambar 4.25 *Routing Overhead* pada kecepatan 15 m/s

Pada Gambar 4.25 di atas, terlihat hal yang sama yaitu semakin sedikit node nya maka semakin kecil juga *routing overhead*nya dan berlaku untuk masing-masing protokol baik 802.11a maupun 802.11p, waulupun 802.11a memiliki *routing overhead* paling tinggi dibanding 802.11p.

4.4.1.3 Analisis End to End Delay (e2e delay)

Cara perhitungan *End-to-end Delay* (selanjutnya disebut *e2e delay*) didapatkan dengan cara menghitung lamanya sebuah paket mulai dikirimkan sampai dengan ACK untuk paket tersebut diterima di tujuan. *E2e delay* dihitung hanya untuk paket yang diterima (*received packet*). Keseluruhan waktu ini yang akan dijumlahkan dan dibagi dengan banyaknya paket yang diterima. Contoh perhitungan *e2e delay* dihitung dengan cara sebagai berikut:

E2E Delay = Jumlah waktu tempuh seluruh paket (*ms*) / banyak paket diterima

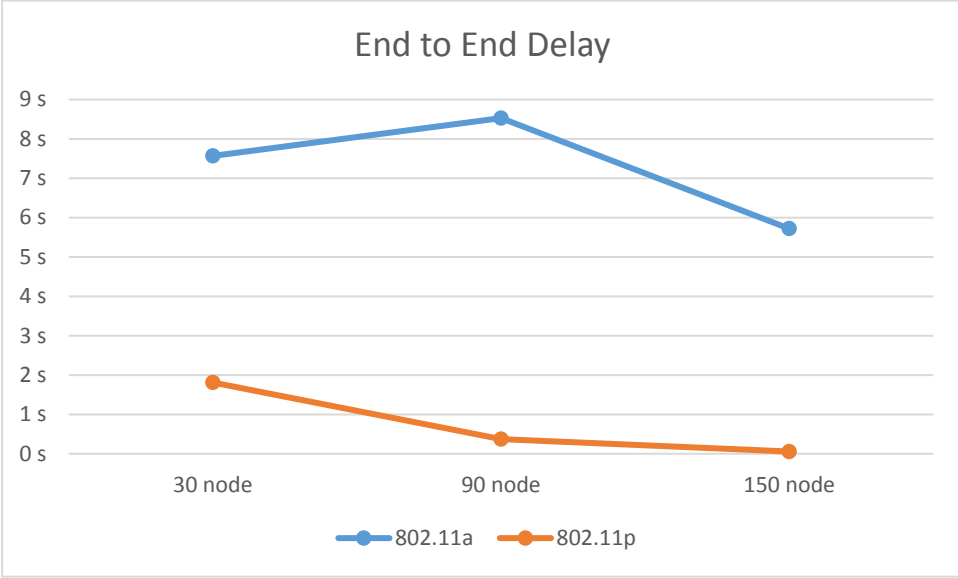
E2E Delay = 3391.08712/ 1132

E2E Delay = 2.99566 ms

Jadi total waktu tempuh paket adalah 3391.08712 ms, dan sebanyak 1132 paket yang diterima, sehingga *e2e delay*-nya adalah sebesar 2.99566 ms.

Tabel 4.9 *End-toEnd Delay* Hasil Pengujian pada kecepatan 5 m/s

	30 node	90 node	150 node
802.11a	7.570331	8.526797	5.7189136
802.11p	1.80983	0.374197	0.05736748

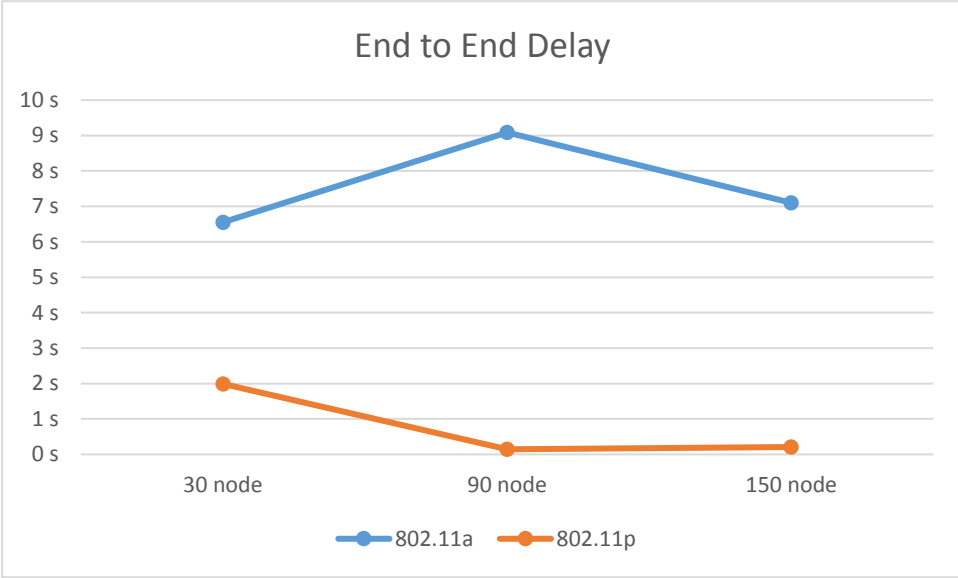


Gambar 4.26 Grafik *End-to-End Delay* pada kecepatan 5 m/s

Pada Gambar 4.26 diatas merupakan grafik dari *e2e delay*berdasarkan dari data Tabel 4.9. Grafik tersebut, terlihat bahwa *e2e delay* cenderung menurun berbanding lurus dengan jumlah banyaknya node, ini disebabkan karena semakin banyaknya node, maka kemungkinan paket tersebut akan sampai pada tujuan semakin besar (walupun mungkin akan memiliki *overhead* sedikit lebih besar). Pada Gambar 4.26 di atas juga terlihat bahwa 802.11a memiliki *delay* yang jauh lebih besar dibandingkan dengan 802.11p.

Tabel 4.10 *End-toEnd Delay* Hasil Pengujian pada kecepatan 10 m/s

	30 node	90 node	150 node
802.11a	6.553533	9.087879	7.099019
802.11p	1.989857	0.144389	0.21229158

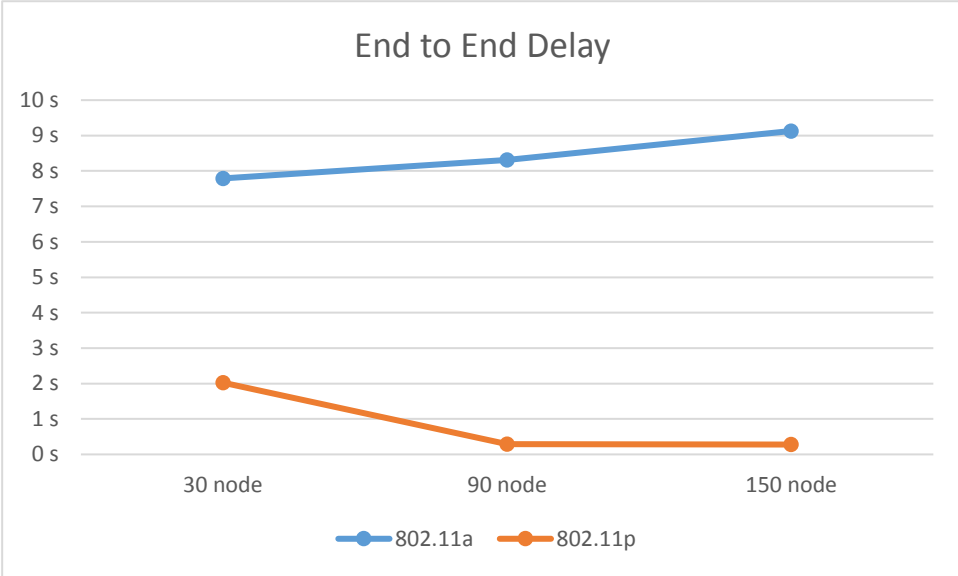


Gambar 4.27 Grafik *End-to-End Delay* pada kecepatan 10 m/s

Pada Gambar 4.27 di atas, terlihat juga ketika dilakukan pengujian pada kecepatan 10 m/s, *end-to-end delay* yang terjadi juga menunjukkan bahwa semakin sedikit node akan menyebabkan semakin tingginya delay seperti ditunjukkan ketika node berjumlah 30 buah baik pada protokol 802.11a maupun 802.11p, *end-to-end delay* yang didapat adalah yang paling besar dibandingkan dengan node berjumlah 150. Meskipun begitu, *end-to-end delay* pada 802.11p jauh lebih bagus dan dapat diandalkan dibandingkan dengan 802.11a.

Tabel 4.11 *End-toEnd Delay* Hasil Pengujian pada kecepatan 15 m/s

	30 node	90 node	150 node
802.11a	7.790268	8.310396	9.1286563
802.11p	2.025024	0.288788	0.2782122



Gambar 4.28 Grafik *End-to-End Delay* pada kecepatan 15 m/s

Pada Gambar 4.28 di atas, menunjukkan bahwa pada kecepatan 15 m/s pada skenario *grid* menghasilkan *end-to-end delay* yang relatif tinggi. Tetapi pola yang dihasilkan tetap, yaitu semakin sedikitnya node akan menghasilkan *end-to-end delay* yang besar (ditunjukkan ketika node berjumlah 30, menghasilkan *end-to-end delay* yang paling besar). Dimana untuk 802.11a tetap memiliki *end-to-end delay* yang paling buruk dibanding 802.11p.

4.4.2 Hasil dan Analisis pada Skenario *real world*

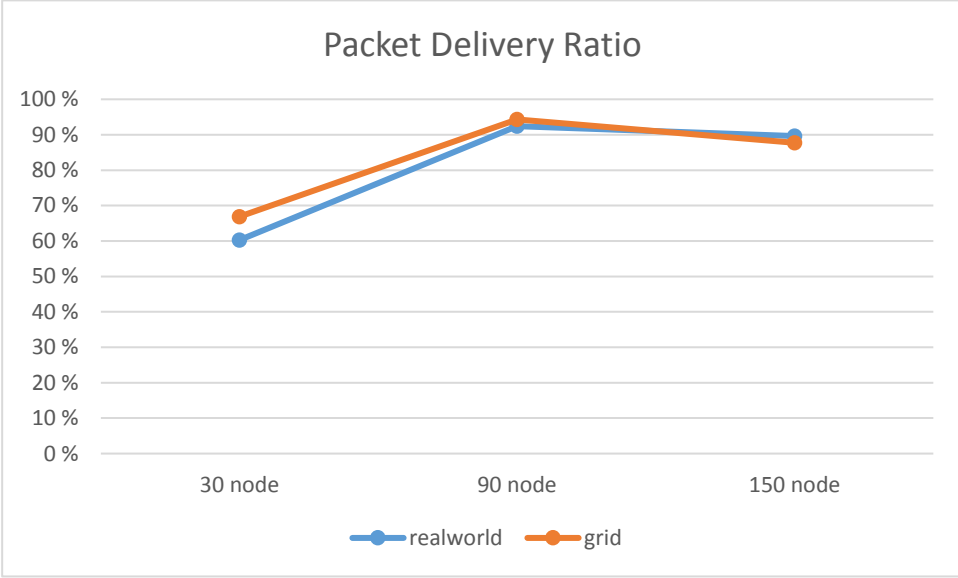
Setelah pengujian pada *grid* dilakukan, pengujian dan analisa *packet delivery ratio, routing overhead* dan *end to end delay* juga dilakukan pada *real world*. Dengan menggunakan parameter-parameter yang telah ditentukan, yaitu kecepatan yang bervariasi (5 m/s , 10 m/s, 15 m/s) dan kepadatan kendaraan (30, 90 dan 150 nodes). Pada pengujian *real world* ini dilakukan juga perbandingan antara 802.11p pada grid dan *real world*, untuk melihat seberapa besar perbedaan skenario *grid* dan skenario *real world*. Untuk 802.11a, tidak dilakukan pengujian pada *real world* karena pada saat pengujian *grid* sudah terlampau jauh performansinya dibandingkan dengan 802.11p.

4.4.2.1 Analisis Packet Delivery Ratio (pdr)

Pengujian *packet delivery ratio* yang dilakukan pada skema *real world* dimana peta didapat dari tahap sebelumnya dan dibandingkan dengan skema *grid*. Adapun hasilnya sebagai berikut.

Tabel 4.12 Packet Delivery Ratio pada skema real world (5m/s)

	30 node	90 node	150 node
realworld	60.28063	92.41208	89.58708
grid	66.90156	94.31216	87.71378

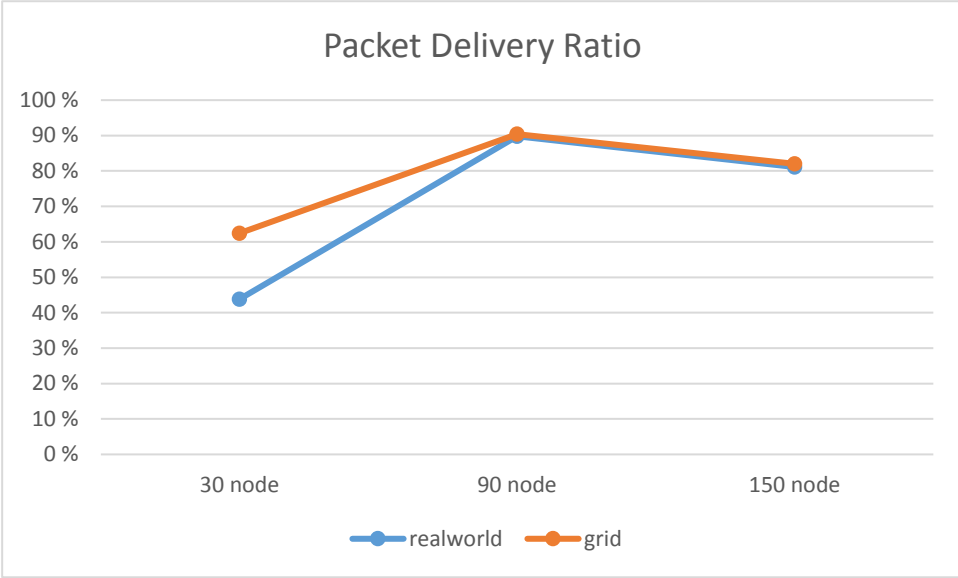


Gambar 4.29 Packet Delivery Ratio pada skema real world (5m/s)

Pada Gambar 4.29 di atas, terlihat bahwa pada perbandingan pengujian *real world* dengan menggunakan peta *grid* juga menunjukkan bahwa nilai paling maksimal adalah pada kepadatan node berjumlah 90.

Tabel 4.13 Packet Delivery Ratio pada skema real world (10m/s)

	30 node	90 node	150 node
realworld	43.84232	89.78876	81.16953
grid	62.45241	90.39031	82.02466

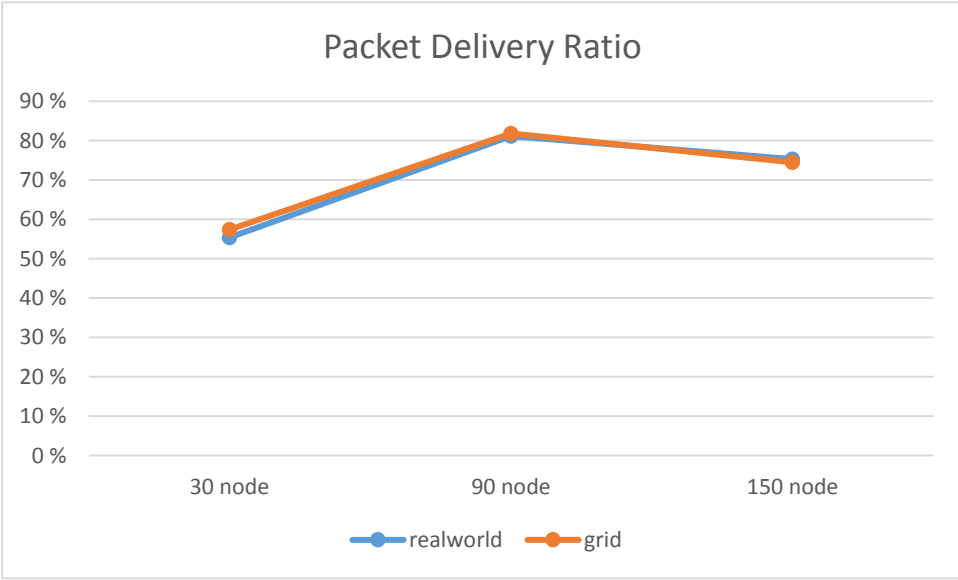


Gambar 4.30 Packet Delivery Ratio pada skema real world (10m/s)

Pada Gambar 4.30 di atas, terlihat bahwa hasil dari pengujian *real world* maupun *grid* menunjukkan pola yang sama, yaitu hasil paling bagus adalah ketika node berjumlah 90 buah.

Tabel 4.14 Packet Delivery Ratio pada skema real world (15m/s)

	30 node	90 node	150 node
realworld	55.36968	81.14132	75.33097
grid	57.37797	81.80869	74.47056



Gambar 4.31 Packet Delivery Ratio pada skema real world (15m/s)

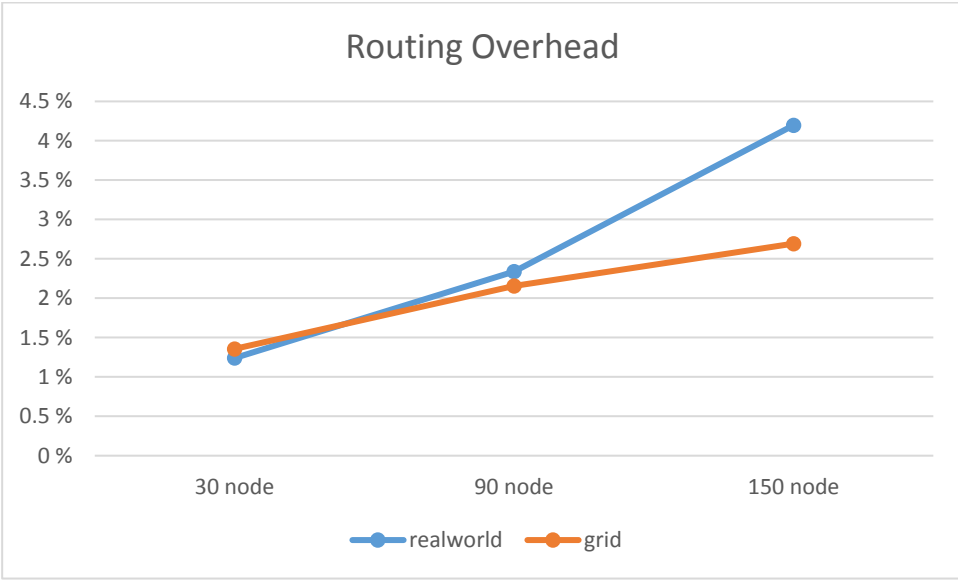
Pada Gambar 4.31 di atas, terlihat bahwa hasil dari pengujian *real world* maupun *grid* menunjukkan pola yang sama, yaitu hasil paling bagus adalah ketika node berjumlah 90 buah dan hampir memiliki nilai *packet delivery ratio* yang sama.

4.4.2.2 Analisis Routing Overhead (RO)

Pada percobaan *real world* juga dilakukan analisa *routing* overhead. Terlihat hasil yang hampir sama seperti pada analisis *routing overhead* pada model *grid*, seperti pada tabel dibawah ini.

Tabel 4.15 Routing Overhead pada skema real world (5m/s)

	30 node	90 node	150 node
realworld	1.239712	2.337624	4.196399
grid	1.356476	2.154325	2.693408



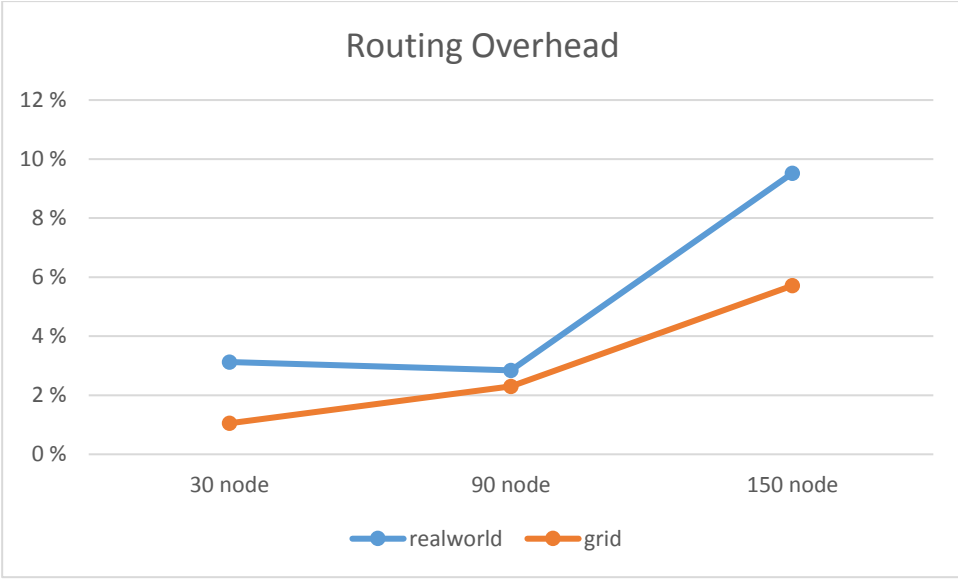
Gambar 4.32 Routing Overhead pada skema real world (5m/s)

Pada Gambar 4.32 di atas, terlihat juga bahwa semakin sedikit node nya maka semakin sedikit juga *routing ovherhead*nya, dan semua percobaan menunjukkan bahwa semakin banyak node tersebut maka semakin tinggi *route overhead*nya karena paket yang dikirimkan semakin banyak yang dirouting kan, sehingga menambah beban jaringan. Dan semakin sedikitnya node, maka semakin sedikitnya *routing overhead* yang terjadi, karena *routing*-nya sedikit, sedangkan semakin banyak node, akan menyebabkan

semakin banyaknya paket yang di*routing* oleh node-node penghubung, sehingga menyebabkan tingginya *routing overhead*. Dan masing-masing *real world* maupun *grid* memiliki pola yang sama.

Tabel 4.16 Routing Overhead pada skema real world (10m/s)

	30 node	90 node	150 node
realworld	3.121237	2.843739	9.525642
grid	1.051128	2.304273	5.715118

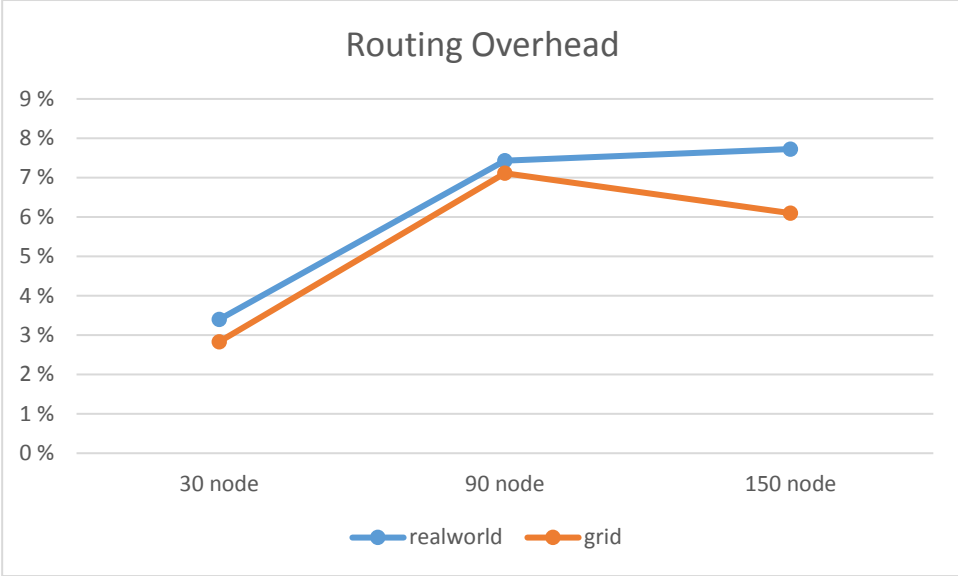


Gambar 4.33 Routing Overhead pada skema real world (10m/s)

Pada Gambar 4.33 di atas, terlihat juga hasil dari pengujian *real world* dan *grid*, dimana keduanya memiliki pola yang sama, yaitu ketika node semakin banyak, maka *routing overhead* yang dihasilkan juga semakin tinggi, dan hasil *routing overhead* dari *grid* maupun *real world* masih relatif kecil.

Tabel 4.17 Routing Overhead pada skema real world (15m/s)

	30 node	90 node	150 node
realworld	3.399848	7.429812	7.727023
grid	2.833199	7.112559	6.100852



Gambar 4.34 Routing Overhead pada skema real world (15m/s)

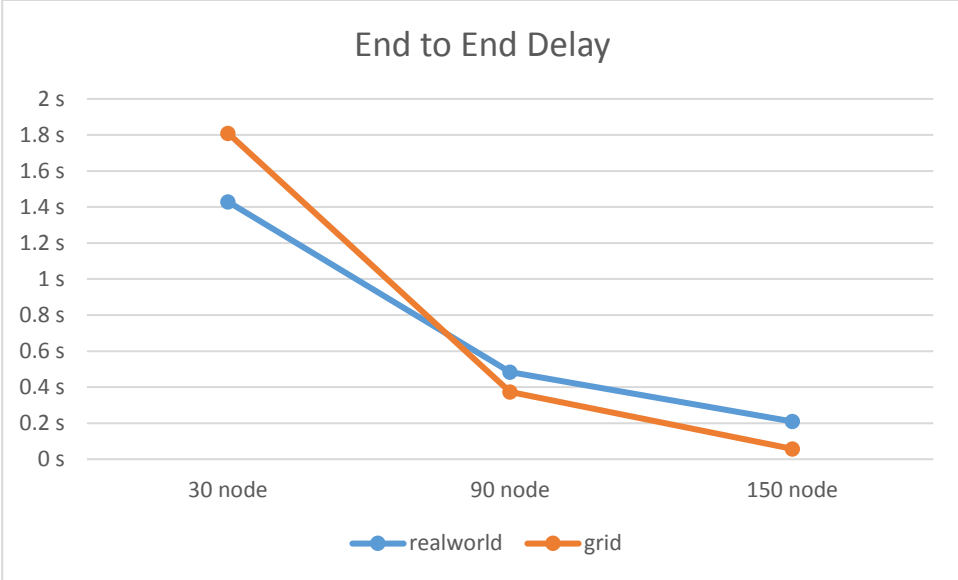
Pada Gambar 4.34 di atas, terlihat bahwa baik pengujian pada *real world* maupun *grid*, memiliki selisih yang relatif kecil pada saat node 30 sampai node 90. Dan memiliki nilai *routing overhead* yang kecil pada waktu node berjumlah 30.

4.4.2.3 Analisis End to End Delay (e2e delay)

Ketika analisis *end to end delay* dilakukan pada skenario *real world*, didapatkan hasil yang memiliki pola grafik yang hampir sama seperti skenario *grid*.

Tabel 4.18 *End-toEnd Delay* Hasil Pengujian pada skenario *real world* (5m/s)

	30 node	90 node	150 node
realworld	1.429219	0.484535	0.20941006
grid	1.80983	0.374197	0.05736748

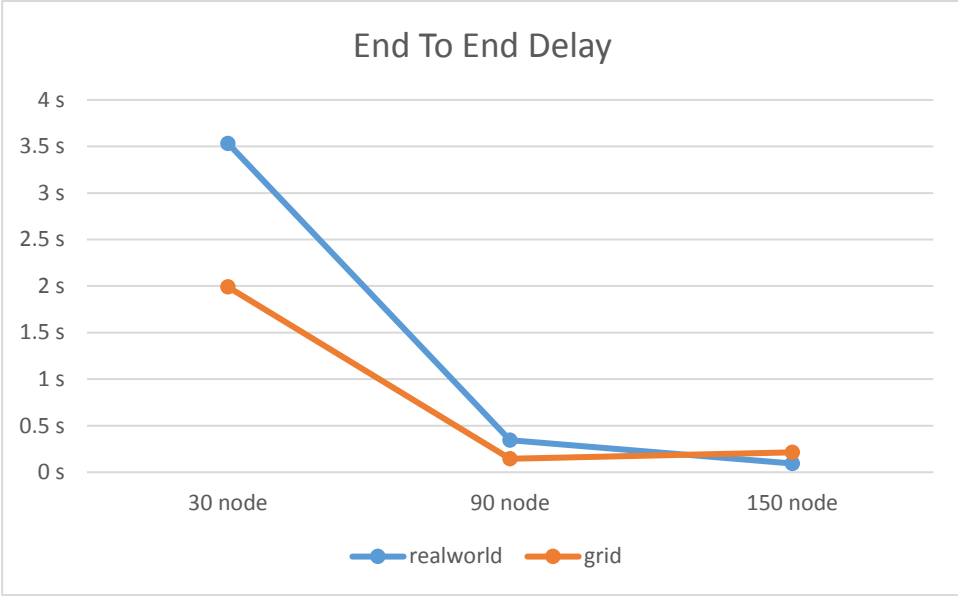


Gambar 4.35 Grafik *End-to-End Delay* pada skema *real world (5m/s)*

Pada Gambar 4.35 di atas, ditunjukkan juga bahwa ketika berjumlah node 30, pada *real world* memiliki delay yang paling besar yaitu sebesar 1.429219 ms dibandingkan dengan kepadatan lainnya. Ini juga berlaku pada skenario *grid* yaitu memiliki delay paling besar ketika node berjumlah 30

Tabel 4.19 *End-toEnd Delay* Hasil Pengujian pada skenario *real world* (10m/s)

	30 node	90 node	150 node
realworld	3.532971	0.343533	0.09352008
grid	1.989857	0.144389	0.21229158



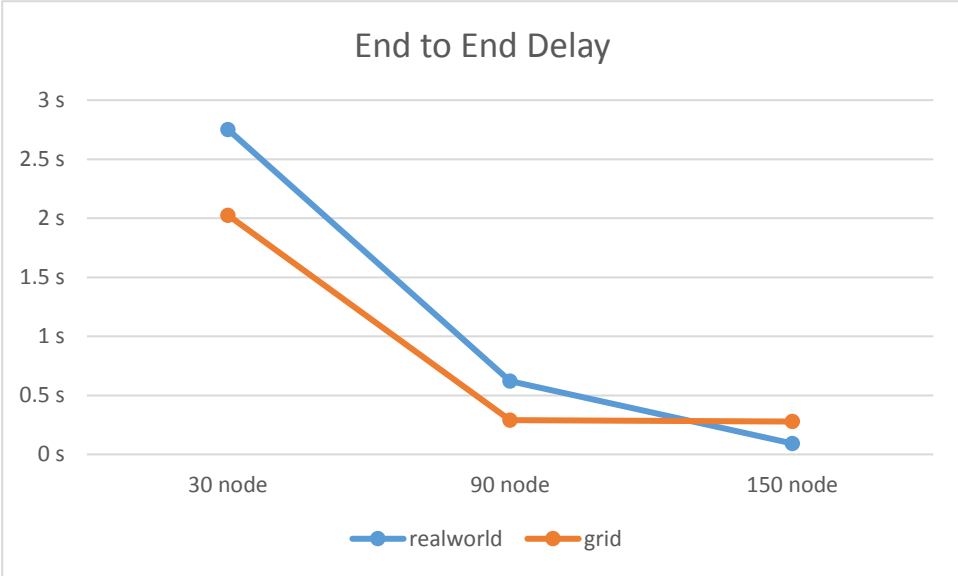
Gambar 4.36 Grafik *End-to-End Delay* pada skema *real world* (10m/s)

Pada Gambar 4.36 di atas, menunjukkan bahwa baik skenario *grid* maupun *real world*, memiliki pola yang hampir sama, yaitu semakin banyak node, maka dleay yang dihasilkan semakin kecil, berarti semakin bagus.

Tabel 4.20 *End-toEnd Delay* Hasil Pengujian pada skenario *real world* (15m/s)

	30 node	90 node	150 node
realworld	2.750882	0.621641	0.09259219
grid	2.025024	0.288788	0.2782122

Tabel 4.20 di atas adalah merupakan tabel hasil dari pengujian masing-masing skenario pada kecepatan 15 m/s, baik pada skenario *real world* maupun skenario *grid*, untuk mempermudah pembacaan tabel di atas, maka dimasukkan kedalam grafik seperti di bawah ini:



Gambar 4.37 Grafik *End-to-End Delay* pada skema *real world* (15m/s)

Pada Gambar 4.37 di atas, ternyata ketika kecepatan mencapai 15m/s , kedua pengujian baik *real world* maupun *grid* menunjukkan hasil *delay* paling bagus ketika node berjumlah 150 buah.

Setelah dilakukan semua pengujian pada skenario *grid* dan *real world*, menunjukkan bahwa semakin sedikit node akan mengakibatkan tingginya *end to end delay*. Ini disebabkan karena ketika semakin sedikitnya node, menyebabkan kurangnya *intermediate node*, yang berfungsi menyalurkan pengiriman data dari sumber ke tujuan, sehingga meningkatkan waktu tunggu (*delay*). Pada e2e delay ini, semakin sedikit delay adalah semakin bagus, karena delay adalah waktu tunggu atau waktu yang dibutuhkan suatu paket untuk sampai ke tujuan. Maka semakin cepat waktu tunggu yang dibutuhkan untuk melakukan pengiriman paket, maka semakin baik.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil pengujian dan analisis hasil yang telah dilakukan, maka dapat ditarik kesimpulan sebagai berikut:

1. Pada Penelitian tesis ini menyajikan analisa performa dari 802.11p pada skenario grid menunjukkan hasil yang jauh lebih bagus daripada 802.11a,
2. Pada penelitian tesis ini menyajikan analisa performa dari protokol routing dalam jaringan Vehicular Adhoc Network (VANET) dengan protokol DSR dan menggunakan simulasi SUMo dan NS2, dari hasil analisa peforma yang dilakukan, terlihat bahwa pengujian DSR pada protokol *grid* memiliki rata-rata *packet delivery ratio* yang relatif lebih baik (82.97583333%) dibanding dengan protokol *real world* (80.75993%).
3. Pada hasil pengujian *routing overhead* menunjukkan bahwa pada skenario *grid* (2.068069823) memiliki hasil yang relatif lebih baik dibanding dengan protokol *real world* (2.591244733).
4. Hasil Pengujian *end-to-end delay* juga menunjukkan bahwa skenario *grid* memiliki rata-rata delay yang lebih sedikit yaitu 0.782178949 detik sedangkan pada skenario *real world* memiliki rata-rata delay sebesar 1.323341193 detik.
5. Dari hasil pengujian pada skenario grid dan *real world*, didapatkan bahwa pola yang ditunjukkan kedua skenario tersebut hampir sama, yaitu ketika pada kepadatan rendah (30 node), memiliki *packet delivery ratio* yang kecil, dan pada kepadatan sedang (90 node) memiliki *packet delivery ratio* yang tinggi dibanding dengan kepadatan tinggi (150 node).
6. Perbedaan hasil dari skenario *grid* dan *real world* dihasilkan karena lalu lintas pada skenario *grid* lebih homogen dan memiliki jalan dengan lebar dan panjang yang sama, sedangkan pada skenario *real world* lebih bervariasi, walupun jumlah tikungannya sama, ini menyebabkan perbedaan yang tidak terlalu jauh.

5.2 Saran

Protokol routing pada VANET masih menjadi topik yang menarik untuk dikembangkan. Dari sekian banyak metode yang diusulkan oleh penlitian yang terdahulu, tentu masih banyak aspek yang bisa dikembangkan. Salah satunya adalah dengan mempertimbangkan proses pembagian beban dalam pengiriman suatu paket.

Daftar Pustaka

1. Jiang, D. and Delgrossi, L., 2008, May. IEEE 802.11 p: Towards an international standard for wireless access in vehicular environments. In Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE (pp. 2036-2040).
2. Paul, Bijan, et al. "Experimental analysis of AODV & DSR over TCP & CBR connections with varying speed and node density in VANET." arXiv preprint arXiv:1204.1206 (2012).
3. Mehmood, Z., Iqbal, M. and Wang, X., 2014. Comprehensive experimental performance analysis of DSR, AODV and DSDV routing protocol for different metrics values with predefined constraints. *International Journal of Information Technology and Computer Science (IJITCS)*, *6*(7), p.24.
4. Jagadevi, D. and Terdal, S., Performance Evaluation of IEEE 802.11 p MAC Protocol for VANETs.
5. Rahardjo, I.A., Anggoro, R. and Arunanto, F.X., 2017. Studi Kinerja 802.11 P pada Protokol Ad Hoc On-Demand Distance Vector (AODV) di Lingkungan Vehicular Ad Hoc Network (VANET) Menggunakan Network Simulator 2 (NS-2). *Jurnal Teknik ITS*, *6*(1), pp.163-167.
6. Maqsood, A. and Khan, R., 2012. Vehicular ad-hoc networks. *arXiv preprint arXiv:1204.1808*.
7. Singh, S., Kumari, P. and Agrawal, S., 2015, February. Comparative Analysis of Various Routing Protocols in VANET. In *Advanced Computing & Communication Technologies (ACCT), 2015 Fifth International Conference on* (pp. 315-319). IEEE.
8. Najafabadi, R. T. (2011). A Survey on Routing Technique for Vehicular Ad-hoc Networks
9. Johnson, D., Hu, Y.C. and Maltz, D., 2007. *The dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4* (No. RFC 4728).
10. Li, F. and Wang, Y., 2007. Routing in vehicular ad hoc networks: A survey. *IEEE Vehicular technology magazine*, *2*(2).

11. Behrisch, M., Bieker, L., Erdmann, J. and Krajzewicz, D., 2011. SUMO–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind.
12. Perkins, C., Belding-Royer, E. and Das, S., 2003. Ad hoc on-demand distance vector (AODV) routing (No. RFC 3561).

LAMPIRAN

Skrip Tcl NS-2

```
# ======================================================================
# Define options
# ======================================================================
  set val(chan)           Channel/WirelessChannel    ;# channel type
  set val(prop)           Propagation/TwoRayGround ;# radio-propagation model
  set val(ant)            Antenna/OmniAntenna          ;# Antenna type
  set val(ll)             LL                           ;# Link layer type
  set val(ifq)            CMUPriQueue    ;# Interface queue type
  set val(ifqlen)         50                           ;# max packet in ifq
  set val(netif)          Phy/WirelessPhyExt           ;# network interface type
  set val(mac)            Mac/802_11Ext                ;# MAC type
  set val(nn) 30 ;# number of mobilenodes
  set val(rp)           DSR                          ;# routing protocol
  set val(x)              2000
  set val(y)              2000
  set val(stop)     900.0    ;# end time


#802.11p default parameters



# Pt_ & freq_ based on paper "Measuring the Performance of IEEE 802.11p Using
ns2 simulator for vehicular networks

# for using 200m , you need propagation.cc
# run the propagation.cc with params
# ./calc_threshold -m TwoRayGround distance 200
# Above result are RXThresh_
# and calculate the carrier sense with distance 2.2 times from above
# ./calc_threshold -m TwoRayGround distance 440
# the result will placed to CSThresh_ and PowerMonitorThresh_

Phy/WirelessPhyExt set CSThresh_                 7.5195e-12      ;# Carrier
Sense threshold (440m)
Phy/WirelessPhyExt set RXThresh_                 1.15361e-10     ;#
```

```
Received transmission allowed (200m)
Phy/WirelessPhyExt set Pt_                      0.281838 ;# 250m default
power transmitter
Phy/WirelessPhyExt set CPThresh_                10

Phy/WirelessPhyExt set freq_                    5.9e+9
Phy/WirelessPhyExt set noise_floor_             1.26e-13        ;#-99 dBm for
10MHz bandwidth
Phy/WirelessPhyExt set L_                       1.0             ;#default
radio circuit gain/loss
Phy/WirelessPhyExt set PowerMonitorThresh_      3.80675e-11     ;#-174 dBm
power monitor    sensitivity
Phy/WirelessPhyExt set HeaderDuration_          0.000040        ;#40 us
Phy/WirelessPhyExt set BasicModulationScheme_   0
Phy/WirelessPhyExt set PreambleCaptureSwitch_   1
Phy/WirelessPhyExt set DataCaptureSwitch_       0
Phy/WirelessPhyExt set SINR_PreambleCapture_    3.1623;         ;# 5 dB
Phy/WirelessPhyExt set SINR_DataCapture_        10.0;           ;# 10 dB
Phy/WirelessPhyExt set trace_dist_              1e6             ;# PHY trace
until distance of 1 Mio. km ("infinty")
Phy/WirelessPhyExt set PHY_DBG_                 0

Mac/802_11Ext set CWMin_                        15
Mac/802_11Ext set CWMax_                        1023
Mac/802_11Ext set SlotTime_                     0.000013
Mac/802_11Ext set SIFS_                         0.000032
Mac/802_11Ext set ShortRetryLimit_              7
Mac/802_11Ext set LongRetryLimit_               4
Mac/802_11Ext set HeaderDuration_               0.000040
Mac/802_11Ext set SymbolDuration_               0.000008
Mac/802_11Ext set BasicModulationScheme_        0
Mac/802_11Ext set use_802_11a_flag_             true
Mac/802_11Ext set RTSThreshold_                 2346
Mac/802_11Ext set MAC_DBG                       0

set ns_ [new Simulator]
#ns-random 0

#Open the NS trace file
set tracefile [open dsr-out.$val(nn).tr w]
$ns_ trace-all $tracefile
$ns_ use-newtrace

set topo [new Topography]
```

```
$topo load_flatgrid $val(x) $val(y)

create-god $val(nn)
set chan_1 [new $val(chan)] ;



#Open the NAM trace file
set namfile [open dsr-out.$val(nn).nam w]
$ns_ namtrace-all $namfile
$ns_ namtrace-all-wireless $namfile $val(x) $val(y)

# CONFIGURE AND CREATE NODES

#===================================
#         Mobile node parameter setup
#===================================

$ns_ node-config    -adhocRouting $val(rp) \
             -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 \
                          -channel $chan_1


#===================================
#              Nodes Definition
#===================================
#Create 7 nodes
   for {set i 0} {$i < $val(nn) } {incr i} {
      set node_($i) [$ns_ node]
      $node_($i) random-motion 0        ;# disable random motion
      $ns_ initial_node_pos $node_($i) 20
   }
```

```
source mobility.$val(nn).tcl



#===================================
#               Agent Definition
#===================================
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(1) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.5
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)

$ns_ at 2 "$cbr_(0) start"

proc finish {} {
        global ns_ tracefile namfile
        $ns_ flush-trace
        close $tracefile
        close $namfile
      # exec nam dsr-out.nam &
        exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
        $ns_ at $val(stop) "\$node_($i) reset"
}


#$ns_ at $val(stop) "$ns_ nam-end-wireless $val(stop)"
$ns_ at $val(stop) "finish"
$ns_ at $val(stop) "puts \"done\" ; $ns_ halt"
$ns_ run
```

Skrip Awk

```
#!/usr/bin/awk -f
BEGIN {
	sendPacket = 0;
	recvPacket = 0;
	RTRsend = 0;
	cbrRecv = 0;

# e2e
	seqNo = -1;
	count = 0;
}
$0 ~/^s.*AGT.*/ {
sendPacket++;
}

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

$0 ~/^[sf].*RTR.*DSR/ {
	RTRsend++;
}

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

# e2e
{
	if($19 == "AGT" && $1 == "s" && seqno < $47){
		seqNo = $47;
	}

	if($19 == "AGT" && $1 == "s"){
		start_time[$47] = $3;
	}else if ($19 == "AGT" && $1 == "r"){
		end_time[$47] = $3;
	}else if($19 == "AGT" && $1 == "d"){
```

```
            end_time[$47] = -1;
        }

}
END {
        packetDelivery = 0.0;
        routingOverhead = 0.0;
        packetDelivery = (recvPacket/sendPacket) * 100.00
        routingOverhead = (RTRsend/cbrRecv)
        printf "Total Packet Sent : %d\n",sendPacket;
        printf "Total Packet Received : %d\n",recvPacket;
        printf "Packet Delivery Ratio : %s\n",packetDelivery;
        printf "Routing Overhead %s\n",routingOverhead;

        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<=count;i++){
            if(delay[i]>0){
                e2edelay = (e2edelay + delay[i]) # get total delay
            }
        }

        e2edelay = e2edelay / count ; # calculate avg end to end delay
        printf "End to End Delay : %s s\n",e2edelay;
}
```

Hasil Tabel Packet Delivery Ratio Grid 30 node 5 m/s

30 Node		1	2	3	4	5	6	7	8	9	10	Average
5x5	PDR	44.18	61.76	48.13	26.09	51.80	52.03	32.73	47.06	48.53	60.47	47.28
	RO	2.78	1.35	1.64	3.28	1.50	1.36	5.67	5.03	2.47	1.24	2.63
	E2E	3.04	4.45	1.93	4.15	3.96	3.35	1.63	2.74	0.84	3.87	3.00
10x10	PDR	28.35	42.59	82.26	47.85	63.52	47.68	79.54	80.65	79.84	14.91	56.72
	RO	3.24	2.53	1.27	1.16	1.16	1.81	0.92	0.29	1.03	2.23	1.57
	E2E	2.81	2.21	2.01	3.35	0.39	2.16	1.97	0.69	1.05	3.67	2.03
15x15	PDR	48.35	88.50	11.74	60.95	75.29	51.48	56.13	84.87	57.53	60.52	59.54
	RO	1.62	0.08	10.71	0.46	0.77	1.64	1.32	0.66	1.10	1.54	1.99
	E2E	2.20	0.64	4.24	1.48	2.18	1.57	1.00	1.43	0.83	3.42	1.90
20x20	PDR	59.60	44.23	58.99	46.05	51.06	46.97	72.15	45.55	64.36	74.61	56.36
	RO	1.35	1.46	0.48	4.01	0.99	1.73	0.78	1.83	1.37	1.10	1.51
	E2E	2.97	2.00	0.86	2.56	1.04	1.66	1.20	1.21	0.14	1.48	1.51
25x25	PDR	57.75	100.00	58.08	75.77	84.87	100.00	55.18	33.82	27.31	37.79	63.06
	RO	1.35	0.0033	0.71	0.65	0.39	0.01	1.63	3.71	2.89	1.23	1.26
	E2E	2.27	0.0029	1.91	2.80	0.61	0.0027	1.86	2.32	2.05	1.50	1.53
26x26	PDR	91.10	53.25	24.99	93.14	90.97	57.46	55.52	60.14	86.39	56.06	66.90
	RO	0.36	1.31	3.73	0.23	0.45	1.29	1.88	1.92	0.69	1.71	1.36
	E2E	1.16	2.13	2.10	0.61	1.23	0.94	2.72	3.02	1.11	3.08	1.81

Hasil Tabel Packet Delivery Ratio Grid 90 Node 5 m/s

90 Node		1	2	3	4	5	6	7	8	9	10	Average
5x5	PDR	92.06	87.65	88.55	75.91	82.31	93.59	78.96	89.19	83.75	90.53	86.25
	RO	8.86	5.44	2.86	7.87	6.48	34.96	7.07	7.46	6.64	1.98	8.96
	E2E	1.03	0.52	0.03	0.29	0.04	1.05	1.24	0.49	0.68	0.15	0.55
10x10	PDR	94.41	94.90	92.55	72.43	73.23	98.12	90.12	94.06	89.04	98.46	89.73
	RO	1.48	1.20	1.37	2.16	4.86	1.61	4.98	2.14	5.24	0.48	2.55
	E2E	0.04	0.01	0.01	0.02	0.11	0.50	0.09	0.03	0.13	0.01	0.10
15x15	PDR	87.53	97.64	82.03	96.09	99.28	95.02	94.28	97.11	98.48	99.56	94.70
	RO	1.05	1.68	3.64	3.06	0.86	1.02	1.85	1.61	1.13	0.53	1.64
	E2E	0.01	0.02	0.61	0.38	0.01	0.01	0.25	0.06	0.08	0.01	0.14
20x20	PDR	96.89	90.89	94.42	95.70	91.17	88.25	93.31	92.66	93.68	89.87	92.68
	RO	2.58	1.52	0.77	3.06	1.89	2.57	2.56	2.58	3.04	3.52	2.41
	E2E	0.01	0.01	0.01	0.07	0.01	0.20	0.08	0.03	1.22	1.00	0.26
25x25	PDR	91.54	83.21	99.00	97.66	100.00	99.23	97.24	99.67	94.57	97.66	95.98
	RO	1.85	2.90	0.45	0.94	0.06	0.31	0.80	0.77	1.67	0.22	1.00
	E2E	1.04	0.01	0.01	0.01	0.0019	0.01	0.01	0.24	0.06	0.0048	0.14
26x26	PDR	81.01	95.27	98.61	98.10	84.30	99.94	99.61	95.03	93.86	97.39	94.31
	RO	3.55	5.24	0.96	1.54	0.38	0.07	0.40	3.01	4.02	2.36	2.15
	E2E	2.51	0.01	0.01	0.53	0.0042	0.002	0.07	0.01	0.08	0.53	0.37

Hasil Tabel Packet Delivery Ratio Grid 150 node 5 m/s

150 node		1	2	3	4	5	6	7	8	9	10	Average
5x5	PDR	76.99	86.92	84.21	82.74	89.45	83.22	76.08	72.49	84.60	75.04	81.17
	RO	13.22	12.84	7.38	48.35	5.96	18.74	111.20	20.02	7.22	39.14	28.41
	E2E	0.16	0.77	0.16	0.31	0.24	0.35	0.34	0.20	0.30	0.22	0.31
10x10	PDR	74.18	91.33	87.25	92.80	94.48	99.55	89.22	94.56	88.31	90.83	90.25
	RO	4.44	2.77	1.26	1.60	4.29	0.57	2.74	19.14	3.49	1.35	4.16
	E2E	0.23	0.01	0.01	0.05	0.02	0.01	0.02	0.05	0.28	0.03	0.07
15x15	PDR	96.81	90.92	81.10	92.85	91.10	95.64	94.54	98.32	98.13	89.37	92.88
	RO	0.86	3.72	8.23	0.92	1.44	1.84	2.56	1.48	0.99	3.62	2.57
	E2E	0.03	0.05	0.10	0.22	0.01	0.04	0.02	0.25	0.17	0.02	0.09
20x20	PDR	83.92	88.10	85.90	93.97	63.29	81.04	82.45	87.78	93.49	86.64	84.66
	RO	1.66	2.56	1.68	14.68	5.18	3.35	11.88	0.40	6.70	9.30	5.74
	E2E	0.05	0.02	0.01	0.06	0.02	0.31	0.16	0.0046	0.03	0.21	0.09
25x25	PDR	96.02	99.45	93.83	92.29	96.50	99.72	99.56	98.44	95.43	98.44	96.97
	RO	1.05	1.40	1.75	4.01	1.97	0.34	0.17	0.69	1.09	1.08	1.36
	E2E	0.02	0.03	0.02	0.26	0.26	0.03	0.0037	0.01	0.01	0.04	0.07
26x26	PDR	100.00	94.06	99.34	92.96	63.73	97.68	91.34	90.45	91.58	56.00	87.71
	RO	0.25	2.88	1.19	2.48	3.96	0.36	2.81	4.56	0.85	7.58	2.69
	E2E	0.0041	0.03	0.05	0.29	0.02	0.01	0.09	0.02	0.04	0.02	0.06

Hasil Tabel Packet Delivery Ratio Grid 30 node 10 m/s

30 Node		1	2	3	4	5	6	7	8	9	10	Average
5x5	PDR	15.19	44.30	21.30	46.06	60.65	31.80	42.87	65.97	25.39	67.64	42.12
	RO	8.01	3.41	6.54	5.61	1.53	5.29	4.27	3.13	3.63	2.29	4.37
	E2E	3.74	3.32	5.08	4.09	3.81	5.05	3.31	2.81	2.06	6.37	3.96
10x10	PDR	67.82	51.27	63.85	52.98	53.48	50.06	44.70	49.50	48.94	51.87	53.45
	RO	2.01	3.48	2.80	2.73	3.93	3.91	3.25	2.62	1.53	3.52	2.98
	E2E	3.47	3.75	2.51	1.03	4.07	0.26	2.59	4.39	1.15	3.20	2.64
15x15	PDR	47.27	52.54	40.68	59.89	85.15	76.62	33.09	60.20	52.83	62.05	57.03
	RO	2.42	3.32	3.96	0.53	1.42	1.13	4.61	1.66	2.21	2.16	2.34
	E2E	2.81	1.92	1.94	1.84	0.09	3.04	2.64	4.04	3.38	3.35	2.51
20x20	PDR	38.27	65.47	72.88	78.42	82.75	70.27	63.10	56.51	75.04	81.81	68.45
	RO	1.86	1.54	1.21	1.41	1.53	0.98	1.37	1.22	0.88	0.93	1.29
	E2E	3.61	2.70	2.14	3.12	2.04	3.01	1.52	1.37	3.63	2.41	2.55
25x25	PDR	47.87	86.73	56.22	46.34	62.13	77.34	91.52	70.61	33.80	87.14	65.97
	RO	1.80	1.44	3.33	4.24	2.66	8.40	0.30	1.14	2.00	1.22	2.65
	E2E	3.43	3.09	2.14	2.43	3.90	3.98	1.48	1.71	1.75	2.33	2.62
26x26	PDR	32.64	38.15	76.96	67.18	46.43	66.01	88.86	67.57	84.90	55.82	62.45
	RO	1.49	1.06	0.45	0.99	1.90	1.50	0.31	0.57	0.91	1.33	1.05
	E2E	1.63	1.43	0.73	2.12	3.04	1.21	1.20	1.98	2.81	3.75	1.99

Hasil Tabel Packet Delivery Ratio Grid 90 node 10 m/s

90 Node		1	2	3	4	5	6	7	8	9	10	Average
5x5	PDR	76.57	71.73	72.36	76.86	77.98	75.61	85.32	82.87	68.11	69.40	75.68
	RO	11.53	36.77	16.21	16.49	4.99	13.44	6.47	32.32	22.32	33.71	19.43
	E2E	1.67	2.93	1.09	0.59	0.70	1.60	0.64	0.95	0.76	4.03	1.50
10x10	PDR	79.46	95.59	95.59	91.31	87.76	78.04	94.41	87.35	87.35	82.35	87.92
	RO	9.28	1.05	1.05	2.83	5.14	6.12	1.63	6.76	6.76	3.56	4.42
	E2E	0.08	0.05	0.05	0.03	0.07	0.70	0.01	0.07	0.07	0.17	0.13
15x15	PDR	85.59	78.77	78.35	82.62	81.60	86.46	89.80	82.99	86.18	85.26	83.76
	RO	8.45	5.53	9.57	5.24	2.89	2.06	2.76	3.71	4.78	6.48	5.15
	E2E	0.06	0.73	0.33	0.13	0.15	0.02	0.26	0.14	0.03	0.07	0.19
20x20	PDR	82.94	92.09	87.59	73.52	81.35	89.99	89.39	90.14	86.62	88.86	86.25
	RO	3.15	3.20	2.01	3.24	3.53	2.26	5.33	3.01	7.22	1.67	3.46
	E2E	0.06	0.42	0.03	0.02	0.42	0.23	0.03	0.06	0.47	0.05	0.18
25x25	PDR	93.93	93.42	92.97	91.24	90.48	85.31	88.10	82.79	95.32	69.33	88.29
	RO	1.41	1.54	9.32	7.41	10.80	8.15	5.50	15.54	5.43	4.81	6.99
	E2E	1.20	0.07	0.18	0.15	0.40	0.76	0.04	1.05	0.29	0.01	0.41
26x26	PDR	88.89	95.52	96.90	99.34	88.80	86.72	93.15	86.55	71.75	96.29	90.39
	RO	4.67	0.97	0.90	0.94	0.67	4.79	2.65	3.54	2.15	1.75	2.30
	E2E	0.20	0.01	0.01	0.01	0.01	0.06	0.01	0.38	0.62	0.15	0.14

Hasil Tabel Packet Delivery Ratio Grid 150 node 10 m/s

150 node		1	2	3	4	5	6	7	8	9	10	Average
5x5	PDR	79.372	69.961	69.659	78.918	70.950	71.006	68.480	67.307	63.158	67.679	70.649
	RO	4.781	137.354	18.591	88.094	10.475	41.661	127.703	79.722	48.537	29.266	58.618
	E2E	0.117	1.181	0.234	0.717	0.414	0.706	0.402	0.735	0.423	0.658	0.559
10x10	PDR	74.126	87.104	83.168	75.248	69.183	70.379	76.209	85.211	91.208	64.602	77.644
	RO	39.982	5.635	16.413	6.522	6.812	15.531	6.587	9.347	2.930	1047.890	115.765
	E2E	1.092	0.811	0.709	0.111	0.089	0.114	0.034	0.033	0.094	0.315	0.340
15x15	PDR	53.825	88.647	76.207	87.660	79.195	79.636	78.779	86.464	60.367	89.625	78.040
	RO	16.323	2.374	8.113	5.987	11.410	6.604	10.473	6.659	22.769	1.424	9.213
	E2E	0.203	0.208	0.028	0.024	0.336	0.032	0.240	0.256	0.098	0.034	0.146
20x20	PDR	80.132	74.874	85.399	88.586	70.323	78.319	65.888	72.518	74.474	80.603	77.112
	RO	8.089	6.586	11.790	2.793	9.791	4.917	7.444	3.152	9.121	18.037	8.172
	E2E	0.234	0.208	0.198	0.080	0.630	0.007	0.447	0.006	0.020	0.492	0.232
25x25	PDR	76.804	99.661	85.992	87.714	88.777	72.874	78.249	98.893	92.036	76.320	85.732
	RO	13.849	0.264	9.383	2.779	2.158	2.127	6.801	0.414	1.271	11.666	5.071
	E2E	0.050	0.003	0.214	0.064	0.012	0.008	0.292	0.003	0.056	0.039	0.074
26x26	PDR	77.114	98.816	71.089	67.875	78.626	84.144	74.486	90.328	89.984	87.785	82.025
	RO	11.388	0.498	5.645	7.823	3.165	10.751	11.433	3.332	0.946	2.170	5.715
	E2E	0.221	0.006	0.043	0.138	0.264	0.967	0.168	0.297	0.005	0.014	0.212

Hasil Tabel Packet Delivery Ratio Grid 30 node 15 m/s

30 Node		1	2	3	4	5	6	7	8	9	10	Average
5x5	PDR	39.40	60.12	24.14	27.93	30.72	48.37	39.62	29.08	44.94	47.13	39.15
	RO	5.40	1.85	4.90	4.39	5.45	3.26	3.16	3.51	3.76	2.60	3.83
	E2E	4.88	4.84	2.70	3.49	2.49	3.66	4.09	3.70	5.42	3.4	3.86
10x10	PDR	72.53	54.70	48.75	83.47	62.53	59.20	55.97	37.44	39.81	74.25	58.86
	RO	1.20	2.14	2.72	1.15	3.33	1.90	2.03	2.94	4.96	2.08	2.44
	E2E	2.47	3.47	1.70	1.71	0.13	1.83	2.17	1.42	3.57	3.17	2.16
15x15	PDR	75.32	62.70	55.54	41.57	48.25	59.15	68.13	81.91	64.40	80.34	63.73
	RO	0.67	3.05	2.19	4.19	2.08	2.21	2.27	1.27	5.45	1.66	2.50
	E2E	1.14	1.65	1.72	3.62	1.22	6.37	2.33	2.86	4.49	3.44	2.88
20x20	PDR	75.36	58.59	56.70	76.93	80.75	39.54	45.65	49.14	62.11	44.82	58.96
	RO	1.89	2.92	1.85	1.28	1.10	2.90	2.32	1.49	1.55	1.70	1.90
	E2E	1.95	3.94	2.36	1.91	2.20	2.58	3.90	0.98	0.39	1.13	2.13
25x25	PDR	63.65	81.38	53.19	37.92	60.92	33.82	85.15	74.55	54.82	82.57	62.80
	RO	2.36	1.24	11.28	4.80	1.95	4.01	1.27	0.98	2.41	2.25	3.26
	E2E	2.54	1.84	2.56	3.61	0.97	4.96	2.68	1.52	2.12	4.93	2.77
26x26	PDR	87.50	51.96	52.34	75.08	52.30	48.47	56.40	46.81	45.30	57.62	57.38
	RO	1.12	4.61	3.85	0.98	2.24	3.31	1.85	2.00	3.09	5.27	2.83
	E2E	0.78	3.06	0.85	1.73	2.02	2.08	2.27	2.71	1.28	3.48	2.03

Hasil Tabel Packet Delivery Ratio Grid 90 node 15 m/s

90 Node		1	2	3	4	5	6	7	8	9	10	Average
5x5	PDR	63.38	76.39	68.93	77.27	54.61	70.12	80.81	76.31	62.60	71.87	70.23
	RO	24.82	15.55	17.00	5.78	36.43	30.88	12.28	11.86	29.90	8.72	19.32
	E2E	1.57	1.78	2.04	1.03	1.15	1.70	1.01	1.33	4.21	1.83	1.77
10x10	PDR	74.82	54.70	48.75	83.47	62.53	59.20	55.97	37.44	39.81	74.25	59.09
	RO	49.86	2.14	2.72	1.15	3.33	1.90	2.03	2.94	4.96	2.08	7.31
	E2E	1.32	3.47	1.70	1.71	0.13	1.83	2.17	1.42	3.57	3.17	2.05
15x15	PDR	85.52	80.59	86.22	87.94	91.26	85.35	82.74	90.42	64.48	84.69	83.92
	RO	4.85	5.92	4.74	2.28	2.40	2.58	46.57	2.12	10.90	6.83	8.92
	E2E	0.20	0.16	0.65	0.02	0.14	0.17	0.73	0.01	0.67	0.95	0.37
20x20	PDR	84.70	78.41	77.73	93.68	84.79	85.89	83.84	74.62	84.59	93.34	84.16
	RO	3.91	4.71	7.37	4.21	7.91	2.33	17.06	4.20	6.64	5.00	6.34
	E2E	0.08	0.72	0.52	0.65	0.34	0.13	0.34	0.20	0.78	0.63	0.44
25x25	PDR	88.94	93.09	81.38	68.04	90.47	78.88	91.83	87.56	85.01	84.14	84.93
	RO	12.66	2.32	5.13	5.82	1.77	15.53	11.89	3.95	5.18	7.61	7.19
	E2E	0.19	0.01	0.14	0.24	0.04	0.13	1.00	0.16	0.60	0.36	0.29
26x26	PDR	86.63	82.44	89.96	69.33	79.55	81.38	85.71	84.01	71.90	87.19	81.81
	RO	4.34	5.85	14.97	20.04	6.99	4.41	4.79	3.04	5.75	0.97	7.11
	E2E	1.00	0.15	0.15	0.56	0.49	0.11	0.03	0.02	0.02	0.38	0.29

Hasil Tabel Packet Delivery Ratio Grid 150 node 15 m/s

150 node		1	2	3	4	5	6	7	8	9	10	Average
5x5	PDR	56.975	76.454	50.056	55.351	63.866	64.529	71.075	59.834	61.292	65.550	62.498
	RO	181.220	42.063	60.301	417.485	198.206	607.672	104.051	237.257	39.251	650.693	253.820
	E2E	0.536	0.552	1.087	0.799	1.200	0.719	0.947	0.388	1.079	1.489	0.880
10x10	PDR	69.278	94.658	85.762	76.087	64.892	78.091	55.835	75.944	93.229	65.280	75.905
	RO	13.959	1.262	4.347	12.180	22.236	14.560	15.091	39.281	1.790	10.909	13.562
	E2E	0.625	0.015	0.045	1.282	0.599	0.331	0.367	0.774	0.086	0.490	0.461
15x15	PDR	90.990	71.270	79.697	59.088	50.222	54.942	85.461	60.311	58.768	66.077	67.683
	RO	8.209	5.192	20.453	15.478	22.922	11.088	7.828	18.049	21.697	9.904	14.082
	E2E	0.670	0.059	0.146	0.298	0.743	0.012	0.566	0.072	0.102	0.526	0.319
20x20	PDR	75.820	86.384	75.722	77.013	56.354	95.533	71.789	82.522	60.134	98.256	77.953
	RO	16.285	5.232	8.241	17.204	23.696	1.555	28.562	16.962	35.608	0.582	15.393
	E2E	0.974	0.297	0.108	1.177	0.450	0.062	0.555	0.616	0.472	0.006	0.472
25x25	PDR	71.928	65.602	91.066	88.051	79.339	77.437	76.041	75.896	84.452	77.568	78.738
	RO	7.473	14.072	0.973	8.567	11.542	2.032	6.203	6.778	34.466	6.679	9.879
	E2E	0.125	0.235	0.007	0.399	0.109	0.004	0.155	0.066	0.154	0.527	0.178
26x26	PDR	58.343	54.828	79.755	71.842	76.454	93.134	77.778	84.956	76.124	71.492	74.471
	RO	11.183	5.985	5.968	3.558	3.999	3.112	5.937	5.293	11.246	4.729	6.101
	E2E	0.773	0.314	0.391	0.015	0.005	0.118	0.415	0.141	0.137	0.472	0.278

Hasil Tabel Packet Delivery Ratio Real World 30 node 5 m/s

30 NODE	Packet Delivery	Routing Overhead	End to End Delay
1	31.509	2.068	2.428
2	76.244	0.760	1.214
3	82.350	0.517	1.001
4	42.641	1.573	0.005
5	94.937	0.212	0.572
6	85.754	1.368	1.644
7	95.437	0.451	0.615
8	39.899	2.569	4.855
9	34.975	0.965	1.611
10	19.061	1.912	0.349
Average	60.281	1.240	1.429

Hasil Tabel Packet Delivery Ratio Real World 90 node 5 m/s

90 NODE	Packet Delivery	Routing Overhead	End to End Delay
1	94.299	0.452	0.010
2	97.263	4.187	0.328
3	85.156	3.824	0.905
4	96.365	0.834	0.553
5	97.567	1.265	0.026
6	80.067	3.738	1.698
7	85.538	2.531	0.194
8	93.289	2.345	0.022
9	97.777	2.395	0.586
10	96.801	1.805	0.521
Average	92.412	2.338	0.485

Hasil Tabel Packet Delivery Ratio Real World 150 node 5 m/s

150 NODE	Packet Delivery	Routing Overhead	End to End Delay
1	92.510	1.071	1.393
2	98.935	0.632	0.026
3	98.953	2.406	0.013
4	94.072	1.173	0.190
5	93.117	3.534	0.112
6	38.870	2.279	0.016
7	98.284	1.180	0.009
8	93.664	24.486	0.037
9	96.933	1.539	0.011
10	90.535	3.664	0.288
Average	89.587	4.196	0.209

Hasil Tabel Packet Delivery Ratio Real World 30 node 10 m/s

30 NODE	Packet Delivery	Routing Overhead	End to End Delay
1	21.135	3.716	4.246
2	31.195	4.388	2.781
3	32.836	3.296	1.923
4	72.266	0.808	2.284
5	30.526	2.933	4.574
6	37.458	2.827	2.482
7	28.222	6.687	7.321
8	32.431	3.537	3.834
9	80.579	1.831	2.716
10	71.776	1.189	3.168
Average	43.842	3.121	3.533

Hasil Tabel Packet Delivery Ratio Real World 90 node 10 m/s

90 NODE	Packet Delivery	Routing Overhead	End to End Delay
1	89.491	1.668	0.039
2	84.517	7.396	0.314
3	88.597	3.645	0.911
4	89.825	2.249	0.018
5	80.782	2.275	0.010
6	90.208	2.843	0.013
7	91.880	3.682	0.983
8	94.687	1.997	0.589
9	92.095	1.738	0.016
10	95.805	0.944	0.541
Average	89.789	2.844	0.344

Hasil Tabel Packet Delivery Ratio Real World 150 node 10 m/s

150 NODE	Packet Delivery	Routing Overhead	End to End Delay
1	81.869	6.322	0.092
2	79.634	36.291	0.157
3	78.737	7.395	0.185
4	93.556	2.445	0.021
5	39.731	7.006	0.032
6	93.919	1.707	0.010
7	85.944	6.392	0.125
8	89.940	10.026	0.058
9	85.006	7.129	0.162
10	83.361	10.543	0.093
Average	81.170	9.526	0.094

Hasil Tabel Packet Delivery Ratio Real World 30 node 15 m/s

30 NODE	Packet Delivery	Routing Overhead	End to End Delay
1	20.564	4.223	2.563
2	81.044	1.125	1.858
3	29.445	4.480	4.750
4	93.771	0.655	0.029
5	88.511	0.958	2.666
6	22.793	9.172	3.758
7	73.075	0.995	2.025
8	92.427	1.203	2.663
9	29.309	3.500	5.186
10	22.758	7.687	2.010
Average	55.370	3.400	2.751

Hasil Tabel Packet Delivery Ratio Real World 90 node 15 m/s

90 NODE	Packet Delivery	Routing Overhead	End to End Delay
1	83.223	7.355	0.083
2	90.117	11.126	0.943
3	96.998	1.949	0.004
4	81.315	3.255	0.774
5	82.036	8.113	0.441
6	88.722	1.052	0.233
7	85.643	10.791	0.190
8	36.760	18.350	1.599
9	83.444	9.224	0.578
10	83.157	3.085	1.370
Average	81.141	7.430	0.622

Hasil Tabel Packet Delivery Ratio Real World 150 node 15 m/s

150 NODE	Packet Delivery	Routing Overhead	End to End Delay
1	73.060	4.155	0.192
2	90.429	11.823	0.026
3	67.680	10.007	0.338
4	84.739	3.897	0.050
5	90.975	2.492	0.016
6	90.190	2.240	0.013
7	70.287	18.841	0.163
8	22.123	16.033	0.013
9	84.228	2.896	0.070
10	79.600	4.888	0.046
Average	75.331	7.727	0.093

BIOGRAFI PENULIS

Penulis adalah anak pertama dari enam bersaudara yang lahir di Jombang pada tanggal 1 Oktober 1988. Pendidikan sekolah dasarnya ditempuh di Sekolah Dasar Negeri Krian III, kemudian dilanjutkan ke SLTPN 1 Balongbendo dan SMAN 1 Taman. Pada tahun 2011 penulis berhasil mendapatkan gelar sarjana (S.Kom) dari Universitas Pembangunan Nasional "Veteran" Jawa Timur (UPN JATIM) dengan predikat lulus dengan memuaskan. Pendidikan pascasarja ditempuh penulis pada tahun 2012 di jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember (ITS) Suarabaya dan berhasil mendapatkan gelar magister di bidang komputer (M.Kom) pada tahun 2018 walupun telat, karena pekerjaan penulis yang non pendidikan dan harus mengatur waktu dengan ketat.

Korespondensi dengan penulis dapat dilakukan melalui email pada alamat yusuf.aja403@gmail.com.