Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - KI141502

IMPLEMENTASI PARTICLE SWARM OPTIMIZATION DALAM GREEDY PERIMETER STATELESS ROUTING (GPSR) PADA VANETs

BAGUS PUTRA MAYANI
NRP 5113 100 125

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

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

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

IMPLEMENTASI PARTICLE SWARM OPTIMIZATION DALAM GREEDY PERIMETER STATELESS ROUTING (GPSR) PADA VANETs

BAGUS PUTRA MAYANI
NRP 5113 100 125

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

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

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI141502

IMPLEMENTATION OF PARTICLE SWARM OPTIMIZATION IN GREEDY PERIMETER STATELESS ROUTING (GPSR) IN VANETs

BAGUS PUTRA MAYANI
NRP 5113 100 125

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

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

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

IMPLEMENTASI PARTICLE SWARM OPTIMIZATION DALAM GREEDY PERIMETER STATELESS ROUTING (GPSR) PADA VANETs

TUGAS AKHIR

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

Oleh :
BAGUS PUTRA MAYANI
NRP : 5113 100 125

Disetujui oleh Dosen Pembimbing Tugas Akhir :

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

Ir. F.X. Arunanto, M.Sc. ..... (pembimbing 2)
NIP: 195701011983031004

SURABAYA
JUNI, 2017

[Halaman ini sengaja dikosongkan]

IMPLEMENTASI PARTICLE SWARM OPTIMIZATION DALAM GREEDY PERIMETER STATELESS ROUTING (GPSR) PADA VANETs

Nama Mahasiswa	**: BAGUS PUTRA MAYANI**
NRP	**: 5113100125**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Dr. Eng. RADITYO ANGGORO, S. Kom., M. Sc.**
Dosen Pembimbing 2	**: Ir. F.X. ARUNANTO, M.Sc.**

Abstrak

Dewasa ini informasi merupakan hal yang sangat dibutuhkan. Setiap detiknya seseorang membutuhkan sebuah informasi baru, baik yang terjadi pada sekitarnya maupun yang terjadi pada belahan dunia lainnya. Dengan berkembangnya dunia internet yang sangat pesat, informasi yang dibutuhkan akan sangat mudah untuk didapatkan. Teknologi internet saat ini pun banyak digunakan sebagai pemecah masalah. Seperti permasalahan penentuan rute tercepat untuk sampai kepada tujuan lebih cepat, Proses penentuan rute ini disebut dengan *routing*. Dalam menentukan rute yang cepat diperlukan jalan alternatif yang banyak pula. Jalan alternatif disini berhubungan dengan infrastruktur dimana memerlukan biaya yang cukup besar. Untuk menanggulangi hal tersebut dapat memanfaatkan teknologi jaringan Ad-Hoc yang mana mendasari pembuatan *Vehicular Ad-Hoc Network* (VANET).

Protokol GPSR merupakah salah satu bagian dari *routing protocol* pada VANET. Dimana GPSR merupakan *geographic based routing*. Dimana GPSR memanfaatkan informasi posisi sebuah *node* dalam menentukan *forwarding node*. Dengan informasi tersebut pula GPSO dijalankan dengan tambahan nilai arah serta kecepatan sebuah *node*, GPSO menggunkan algoritama *particle swarm optimization* dalam menentukan *forwarding node*.

Dari uji coba yang dilakukan, GPSO memberikan rata-rata nilai *packet delivery ratio* yang lebih baik bila dibandingkan dengan protokol GPSR.

Kata kunci : VANET, GPSR, GPSO, PSO

IMPLEMENTATION OF PARTICLE SWARM OPTIMIZATION IN GREEDY PERIMETER STATELESS ROUTING (GPSR) IN VANETs

Student's Name	**: BAGUS PUTRA MAYANI**
Student's ID	**: 5113100125**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Dr. Eng. Radityo Anggoro, S. Kom., M. Sc**
Second Advisor	**: Ir. F.X. Arunanto, M.Sc.**

Abstract

Today information is a much needed thing. Every second a person needs a new information, whether that happens to the surrounding or that occur in other parts of the world. With the rapid development of the Internet world, the information needed will be very easy to obtain. Internet technology today is also widely used as a problem solver. As the fastest route determines to get to the destination faster, the process of determining this route is called routing. In determining the fast route required many alternative roads as well. Alternative roads here relate to infrastructure where the costs are considerable. To overcome this can take advantage of Ad-Hoc network technology which underlies the manufacture of Vehicular Ad-Hoc Network (VANET).

GPSR protocol is part of the routing protocol in VANET. Where GPSR is a geographic based routing. GPSR utilizes the position information of a node in determining the forwarding node. With this information also GPSO run with additional direction value and the speed of a node, GPSO using algorithm particle swarm optimization in determining the forwarding node.

From the experiments performed, GPSO provides a better packet delivery ratio value when compared to the GPSR protocol.

Keyword : VANET, GPSR, GPSO, PSO

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur setinggi-tingginya kepada Allah SWT, yang telah memberikan berkah dan kelancaran sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "Implementasi Particle Swarm Optimization dalam Greedy Perimeter Stateles Routing (GPSR) pada VANETs" dengan tepat waktu.

Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat berharga bagi penulis, karena dengan mengerjakan Tugas Akhir ini penulis dapat memperdalam, meningkatkan serta mengimplementasikan ilmu yang didapat selama penulis menempuh perkuliahan di jurusan Teknik Informatika ITS.

Terselesaikannya buku Tugas Akhir ini, tidak lepas dari bantuan dan dukungan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih kepada:

1. Allah SWT sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik.
2. Mama, Papa, dan Adik penulis yang telah memberikan dukungan moral dan material serta doa yang tak terhingga untuk penulis. Serta selalu memberikan semangat dan motivasi pada penulis dalam mengerjakan Tugas Akhir ini.
3. Bapak Dr. Eng. Radityo Anggoro, S. Kom., M. Sc selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
4. Bapak Ir. F.X. Arunanto, M.Sc. selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi penulis dalam mengerjakan Tugas Akhir ini.
5. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
6. Teman-teman TC angkatan 2013 yang sudah bersama-sama jatuh bangun menjalani kuliah di kampus TC sejak maba hingga akhir kuliah.

7. Teman-teman *Diary of Silver* Arvi, Kevin, Sumitra, Asbun, Bawanta, Nyoman yang selalu menemani penulis dan membantu penulis.
8. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Sebagai manusia biasa, penulis menyadari Tugas Akhir ini masih jauh dari kesempurnaan dan memiliki banyak kekurangan. Sehingga dengan segala kerendahan hari penulis mengharapkan saran dan kritik yang membangun dari pembaca.

Surabaya, Juni 2016

Bagus Putra Mayani

DAFTAR ISI

LEMBAR PENGESAHAN v
Abstrak vii
Abstract ix
KATA PENGANTAR x
DAFTAR ISI xiii
DAFTAR GAMBAR xvii
DAFTAR TABEL xix
DAFTAR PERSAMAAN xx
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 3
1.4 Tujuan 3
1.5 Manfaat 3
1.6 Metodologi 4
1.7 Sistematika Penulisan Laporan Tugas Akhir 5
BAB II TINJAUAN PUSTAKA 7
2.1 *Vehicular Ad Hoc Network* 7
2.2 *Greedy Perimeter Stateless Routing* (GPSR) 8
2.2.1 Greedy Forwarding 8
2.2.2 Perimeter Forwarding 9
2.2.3 Planarized Graph 10
2.3 *Particle Swarm Optimization* (PSO) 11
2.4 GPSO 11
2.5 *OpenStreetMap* 14
2.6 JOSM 14
2.7 *Simulation of Urban Mobility* (SUMO) 14
2.8 AWK 15
2.9 NS-2 16
BAB III DESAIN DAN PERANCANGAN 19
3.1 Deskripsi Umum Sistem 19
3.2 Perancangan Skenario Mobilitas 20
3.2.1 Perancangan Skenario Mobilitas *Grid* 21

3.2.2 Perancangan Skenario Mobilitas *Real* 21
3.3 Perancangan Simulasi pada NS-2 24
3.4 Perancangan Metriks Analisis.....24
3.4.1 *Packet Delivery Ratio* 24
3.4.2 Rata-rata *End-to-end Delay* 25
3.4.3 *Routing Overhead* 25
3.5 Protokol GPSO.....26
BAB IV IMPLEMENTASI.....29
4.1 Lingkungan Implementasi Protokol.....29
4.2 Implementasi Skenario Mobilitas 29
4.2.1 Implementasi Skenario Peta *Grid* 29
4.2.2 Implementasi Skenario Peta *Real* 34
4.3 Implementasi Protokol GPSO.....38
4.3.1 Modifikasi Struktur *Header* pada GPSR 38
4.3.2 Modifikasi Struktur *Header Hello Message* 39
4.3.3 Modifikasi Struktur Penyimpanan Data *Hello Message* 39
4.3.4 Implementasi Fungsi Perhitungan Kecepatan dan Nilai Arah 40
4.3.5 Modifikasi *Beacon Processing*..... 41
4.3.6 Modifikasi Struktur Penyimpanan Variabel Kecepatan, Arah, dan Waktu..... 42
4.3.7 Modifikasi Pemilihan *Forwading Node* 43
4.4 Implementasi Simulasi pada NS-2 43
4.5 Implementasi Metriks Analisis 46
4.5.1 Implementasi *Packet Delivery Ratio* 46
4.5.2 Implementasi *End-to-end Delay* 46
4.5.3 Implementasi *Routing Overhead* 47
BAB V PENGUJIAN DAN EVALUASI 48
5.1 Lingkungan Uji Coba.....49
5.2 Hasil Uji Coba.....49
5.2.1 Hasil Uji Coba Skenario *Grid*..... 50
5.2.1.1 Hasil *Packet Delivery Ratio* pada Skenario *Grid* 50

5.2.1.2 Hasil *End to End Delay* pada Skenario *Grid* 55
5.2.1.3 Hasil *Routing Overhead* pada Skenario *Grid* 57
5.2.2 Hasil Uji Coba Skenario *Real* 60
5.2.2.1 Peta Daerah Surabaya Wilayah Perumahan 60
5.2.2.2 Peta Daerah Surabaya Wilayah Perkotaan 63
BAB VI KESIMPULAN DAN SARAN 67
6.1 Kesimpulan 67
6.2 Saran 68
DAFTAR PUSTAKA 69

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

Gambar 2.1 Ilustrasi VANET....7
Gambar 2.2 *Ilustrasi Greedy Forwarding*....9
Gambar 2.3 Ilustrasi Pemilihan *Neighbour Greedy Forwarding*..9
Gambar 2.4 *Right Hand Rule* 10
Gambar 2.5 *Relative Neighbour Graph*....10
Gambar 2.6 *Gabriel Graph* 10
Gambar 2.7 Skema Pemilihan *Node Forwarding* 12
Gambar 2.8 Ilustrasi Pemilihan *Forwading Node*....13
Gambar 2.9 Contoh Baris Pengiriman *Hello Message*....17
Gambar 2.10 Contoh Baris Penerimaan Hello Message 17
Gambar 2.11 Contoh Paket Komunikasi Data CBR 17
Gambar 3.1 Diagram Rancangan Simulasi 20
Gambar 3.2 Diagram Alur Pembuatan Skenario Grid....21
Gambar 3.3 Diagram Alur Pembuatan Skenario Mobilitas Real 23
Gambar 3.4 *Pseudocode* Pemilihan *Forwarding Node*....26
Gambar 4.1 Peta Hasil netgenerate 30
Gambar 4.2 Hasil *Capture* sumo-gui 32
Gambar 4.3 Cuplikan Isi File scenario.tcl Pada Skenario Grid...33
Gambar 4.4 Menu Pemilihan Peta pada *OpenStreetMap* 34
Gambar 4.5 Hasil Peta *Real* yang Telah Dirubah....35
Gambar 4.6 Hasil *Capture* sumo-gui 37
Gambar 4.7 Cuplikan Isi File scenario.tcl Pada Skenario *Real*...37
Gambar 4.9 Cuplikan Pengaturan NS-2 45
Gambar 4.10 Hasil Perhitungan *Packet Delivery Ratio* 46
Gambar 4.11 Hasil Perhitungan *End to End Delay*....47
Gambar 4.12 Hasil Perhitungan *Routing Overhead*....47
Gambar 5.1 Grafik *Packet Delivery Ratio* Terhadap Banyak *Node* pada Kecepatan 10m/s....50
Gambar 5.2 Grafik *Packet Delivery Ratio* Terhadap Banyak *Node* pada Kecepatan 15m/s....51
Gambar 5.3 Grafik *Packet Delivery Ratio* Terhadap Banyak *Node* pada Kecepatan 20m/s....51

Gambar 5.4 Hasil Pemilihan *Forwarding Node* Pada Protokol GPSR 54
Gambar 5.5 Hasil Jumlah Nilai Arah Perhitungan Protokol GPSO 55
Gambar 5.6 Grafik *End to End Delay* Terhadap Banyak *Node* pada Kecepatan 10m/s 55
Gambar 5.7 Grafik *End to End Delay* Terhadap Banyak *Node* pada kecepeatan 15m/s.....56
Gambar 5.8 Grafik *End to End Delay* Terhadap Banyak *Node* pada kecepeatan 20m/s.....56
Gambar 5.9 Grafik *Routing Overhead* Terhadap banyak node pada Kecepatan 10m/s 57
Gambar 5.10 Grafik *Routing Overhead* Terhadap banyak node pada Kecepatan 15m/s 58
Gambar 5.11 Grafik *Routing Overhead* Terhadap banyak node pada Kecepatan 20m/s 58
Gambar 5.12 Grafik *Packet Delivery Ratio* dengan Skenario *Real* Tanpa *Traffic Light*.....61
Gambar 5.13 Grafik *End to End Delay* dengan Skenario *Real* Tanpa *Traffic Light*.....62
Gambar 5.14 Grafik RO dengan Skenario *Real* Tanpa *Traffic Light* 63
Gambar 5.15 Grafik *Packet Delivery Ratio* dengan Skenario *Real* Dengan *Traffic Light* 64
Gambar 5.16 Grafik *End to End Delay* dengan Skenario Real Tanpa *Traffic Light*.....64
Gambar 5.17 Grafik RO dengan Skenario *Real* Dengan *Traffic Light*.....65

DAFTAR TABEL

Tabel 2.1 Tabel Detail Penjelasan Data *Trace File*.....17
Tabel 3.1 Parameter Lingkungan Simulasi dengan Skenario.....24
Tabel 4.1 Spesifikasi Perangkat yang Digunakan untuk.....29
Tabel 5.1 Spesifikasi Perangkat yang Digunakan.....49
Tabel 5.2 Data Uji Skenario Grid pada node sebanyak 50 dan kecepatan 20m/s.....52
Tabel 5.3 Data Uji Skenario Grid Nilai *Packet Delivery Ratio* pada node sebanyak 25 dan kecepatan 20m/s.....59
Tabel 5.4 Data Uji Skenario Grid Nilai *Routing Overhead* pada node sebanyak 25 dan kecepatan 20m/s.....59

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini akan dijelaskan mengenai beberapa hal dasar dalam Tugas Akhir ini yang meliputi latar belakang, perumusan masalah, batasan, tujuan dan manfaat pembuatan Tugas Akhir serta metodologi dan sistematika pembuatan buku Tugas Akhir ini. Dari uraian di bawah ini diharapkan gambaran Tugas Akhir secara umum dapat dipahami dengan baik.

1.1 Latar Belakang

Dewasa ini informasi merupakah hal yang sangat dibutuhkan. Setiap detiknya seseorang membutuhkan sebuah informasi baru, baik yang terjadi pada sekitarnya maupun yang terjadi pada belahan dunia lainnya. Dengan berkembangnya dunia internet yang sangat pesat, informasi yang dibutuhkan akan sangat mudah untuk didapatkan. Teknologi internet saat ini pun banyak digunakan sebagai pemecah masalah. Seperti permasalahan penentuan rute tercepat untuk sampai kepada tujuan lebih cepat, Proses penentuan rute ini disebut dengan *routing*. Dalam menentukan rute yang cepat diperlukan jalan alternatif yang banyak pula. Jalan alternatif disini berhubungan dengan infrastruktur dimana memerlukan biaya yang cukup besar. Untuk menanggulangi hal tersebut dapat memanfaatkan teknologi jaringan *Ad-Hoc* yang mana mendasari pembuatan *Vehicular Ad-Hoc Network* (VANET).

VANET merupakan pengembangan dari *Mobile Ad-Hoc Network* (MANET) yang diaplikasikan dalam kendaraan. Jaringan VANET ini memungkinkan kendaraan saling berkomunikasi antara satu dan lainnya. Pada jaringan VANET, kendaraan dapat berkomunikasi tanpa membutuhkan pengaturan infrastruktur tersentral ataupun server yang digunakan untuk mengontrol. Implementasi dari VANET ini telah menciptakan beberapa *routing protocol*. *Routing protocol* VANET terbagi menjadi lima kategori,

diantaranya *Position based routing protocol,Topology based routing protocol, Geo cast routing protocol, Cluster based routing protocol* dan *Broadcast routing protocol*. Pada *geographic routing, node forwarding* ditentukan oleh posisi tujuan paket serta *neighbour node*. Optimisasi dibutuhkan oleh routing protocol dalam menentukan pilihan *node* selanjutnya agar mendapatkan performa yang terbaik.

Salah satu contoh *routing protocol* yang ada pada VANET dengan memanfaatkan informasi mengenai posisi dari sebuah *node* adalah *Greedy Perimeter Stateless Protocol* (GPSR). Dimana pada GPSR memanfaatkan informasi posisi dari sebuah *node* untuk menjadi *hop* pengirim untuk mencapai *node* penerima. Dalam pemilihan *node* digunakan metode *greedy algorithm* dimana pada metode tersebut *node* yang dipilih berdasarkan posisi terdekat dengan *node* penerima. Kelemahan dari metode tersebut adalah *node* selanjutnya belum tentu merupakan *node* yang optimal dikarenakan hanya menggunakan satu informasi sebagai penentu pemilihan *forwarding node*.

Maka untuk membuat pemilihan *node* selanjutnya agar lebih optimal diperlukan informasi tambahan diantaranya adalah nilai arah, kecepatan, serta jarak antara *node* dengan penerima. Dengan menggunakan metode *Particle Swarm Optimazation* (PSO) yang membutuhkan informasi yang telah disebutkan akan dapat lebih akurat dalam menentukan *node* selanjutnya. Sehingga pengiriman paket data dapat lebih optimal dan *ratio* dalam mengirimkan paket dapat meningkat.

Pada Tugas Akhir ini akan menggunakan *Greedy Perimeter Stateless Routing* (GPSR) yang dimodifikasi untuk meningkatkan kinerjanya. Adapun bagian yang akan di modifikasi dalam pemilihan *forwarding node* yang terbaik dengan menggunakan *Particle Swarm Optimization*.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana menentukan *forwarding node* terbaik pada dalam protokol GPSR
2. Seberapa besarkah pengaruh perubahan end to *end delay*, *routing overhead* serta *packet delivery ratio* dengan perubahan penentuan *forwarding node* terhadap performa *routing protocol* GPSR?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. *Routing protocol* yang diuji coba adalah *Greedy Perimeter Stateless Routing* (GPSR).
2. Optimasi pemilihan *forwarding node* pada protokol GPSR.
3. Algoritma optimasi pemilihan *forwarding node* menggunakan *Particle Swarm Optimization Algorithm*.
4. Uji coba menggunakan *Network Simulator* 2 (NS-2).
5. Pembuatan scenario uji coba menggunakan *Simulation of urban Mobility*(SUMO).

1.4 Tujuan

Tujuan pengerjaan Tugas Akhir ini untuk mengetahui performa protokol *routing* GPSR terhadap pengaruh penambahan *Particle Swarm Optimization Algorithm* dalam metode penentuan *forwarding node* yang terbaik.

1.5 Manfaat

Manfaat dari pengerjaan Tugas Akhir ini antara lain :

1. Menjadi acuan untuk topik penelitian *geographic protocol* yang menggunakan NS2.
2. Menjadi acuan untuk optimasi pemilihan *forwarding node* pada VANET.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.
   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Proposal Tugas Akhir yang diajukan memiliki gagasan yang sama dengan Tugas Akhir ini, yaitu mengimplementasikan algoritma *particle swarm optimization* pada protokol GPSR.

2. Studi literatur
   Pada tahap ini dilakukan pemahaman informasi dan literatur yang diperlukan untuk pembuatan implementasi program. Tahap ini diperlukan untuk membantu memahami pengunaan komponen-komponen terkait dengan sistem yang akan dibangun, antara lain : GPSR, PSO, NS2, dan SUMO.

3. Analisis dan desain perangkat lunak
   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain suatu sistem dan desain proses-proses yang ada.

4. Implementasi perangkat lunak
   Implementasi merupakan tahap membangun rancangan program yang telah dibuat. Pada tahapan ini merealisasikan apa yang terdapat pada tahapan sebelumnya, sehingga menjadi sebuah program yang sesuai dengan apa yang telah direncanakan.

5. Pengujian dan evaluasi
   Pada tahapan ini dilakukan uji coba pada data yang telah dikumpulkan. Tahapan ini dimaksudkan untuk mengevaluasi kesesuaian data dan program serta mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan pada program.

6. Penyusunan buku Tugas Akhir.
   Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan Tugas Akhir

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

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini. Dasar teori yang digunakan adalah GPSR, PSO, NS2, serta SUMO.

Bab III Desain dan Perancangan

Bab ini berisi tentang rancangan sistem yang akan dibangun dan disajikan dalam bentuk diagram alir. Fungsi utama yang akan dibuat pada Tugas Akhir ini

meliputi fungsi penghintungan jarak, penghitungan kecepatan, penghitungan arah, serta algoritma *particle swarm optimization*.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode program yang digunakan untuk proses implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan Dan Saran

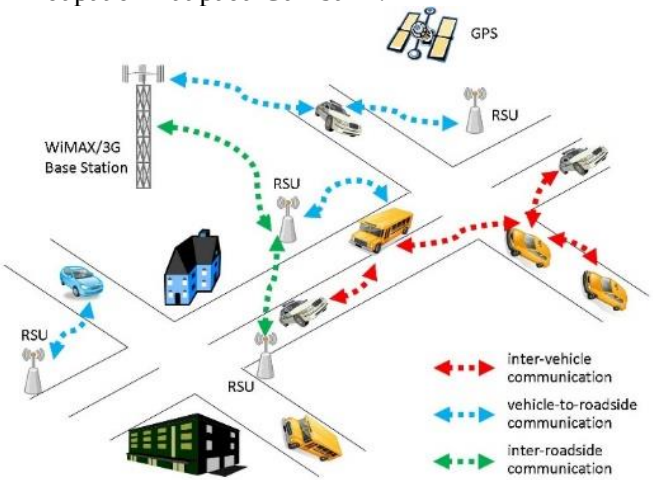
Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai teori yang menjadi dasar dari pembuatan Tugas Akhir ini. Teori yang dibahas mencakup elemen-elemen yang terkait dalam topik Tugas Akhir mulai dari sumber dari permasalahan, pendekatan yang digunakan, serta metode dan teknologi yang digunakan untuk pengerjaan Tugas Akhir ini.

2.1 *Vehicular Ad Hoc Network*

Vehicular Ad Hoc Network (VANET) adalah teknologi yang digunakan pada pengiriman paket dengan menggunakan kendaraan yang bergerak sebagai *node*nya [1]. VANET membuat berbagai kendaraan yang berpartisipasi menjadi sebuah *node*, memungkinkan kendaraan dapat berkomunikasi dengan jarak antara 100m sampai dengan 300m. Setiap kendaraan dapat mengirimkan paket kepada kendaraan selanjutnya apabila kendaraan tersebut sudah jauh dari jangkauan penerima. Ilustrasi VANET dapat dilihat pada Gambar 2.1



Gambar 2.1 Ilustrasi VANET [2]

VANET memiliki karakteristik yang unik bila dibandingkan dengan MANET dalam mendesain aplikasinya. Beberapa karakteristik unik yang dimiliki oleh VANET diantaranya adalah *high dynamic topology, frequent disconnect network, mobility modelling, battery power and storage capacity*.

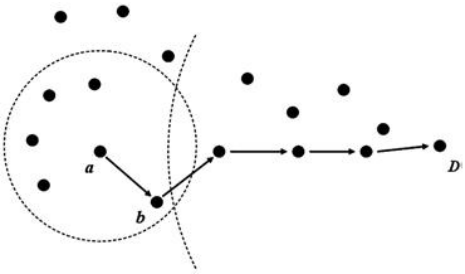
2.2 *Greedy Perimeter Stateless Routing* (GPSR)

Greedy Perimeter Stateless Routing adalah salah satu varian dari *Geographic Routing Protocol* yang menggunakan *position-based routing*. Tidak seperti *topology based routing protocol* yang membuat daftar neighbour, GPSR menggunakan *beaconing* sebagai informasi seperti *hello message* yang dikirimkan secara berkala untuk mengetahui keberadaan *neighbour* sekitar. GPSR melakukan pengiriman paket kepada *node* yang memiliki posisi paling dekat dengan destinasi [2]. GPSR menggunakan dua metode dalam mengirimkan paket kepada *node* selanjutnya, yaitu dengan menggunakan *greedy forwarding* serta *perimeter forwarding*.

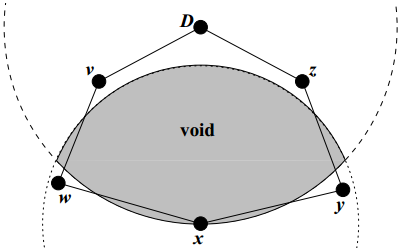
2.2.1 Greedy Forwarding

Greedy Forwarding adalah tipe pengiriman paket yang digunakan pada GPSR. *Greedy Forwarding* menggunakan informasi dari posisi destinasi paket, dimana *node* yang akan dipilih adalah *node* yang memiliki posisi terdekat dengan destinasi paket. Seperti yang ditunjukkan pada Gambar 2.2. *Greedy forwarding* terus dilakukan sampai paket diterima oleh *node receiver*.

Greedy Forwarding memiliki kelemahan dimana apabila tidak ada *neighbour* yang paling dekat dengan destinasi atau *node* yang optimum tidak terjangkau oleh pengirim paket [3] seperti yang ditunjukkan pada Gambar 2.3, pengirim harus memilih rute yang lebih jauh lagi.



Gambar 2.2 *Ilustrasi Greedy Forwarding* [3]



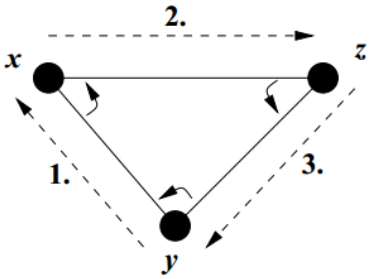
Gambar 2.3 Ilustrasi Pemilihan *Neighbour Greedy Forwarding* [3]

2.2.2 Perimeter Forwarding

Perimeter Forwarding dilakukan saat *greedy forwarding* tidak menemukan *node* yang lebih dekat dengan tujuan seperti yang ditunjukkan pada Gambar 2.3. *Perimeter Forwarding* menggunakan aturan tangan kanan (*right hand rule*) seperti pada Gambar 2.4 untuk mengitari daerah *perimeter* yang *void* dengan berlawanan arah jarum jam untuk mencari *node* yang sesuai untuk meneruskan paket.

Metode *perimeter forwarding* memerlukan *no-crossing heuristic* yang membuat aturan tangan kanan untuk memilih *perimeter* yang menutup *void* di daerah dimana *node* berada di tepi jangkauan transmisi yang bersilang dengan *node* lainnya. *No-crossing heuristic* membuat *node* tepi yang saling silang dihapus,

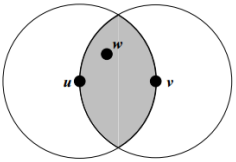
untuk mencegah itu digunakan metode lain yaitu *planarized graph* [3].



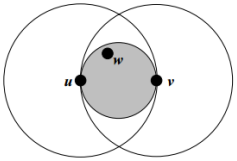
Gambar 2.4 *Right Hand Rule* [3]

2.2.3 Planarized Graph

Planarized graph terjadi ketika dua atau lebih *node* yang berada di tepi jangkauan transmisi bertemu satu dengan lainnya dalam sebuah grafik [3]. Metode dalam menghapus dua tepi yang saling silang pada *planarized graph* memiliki dua tipe, yaitu *Relative Neighborhood Graph* (RNG) yang ditunjukkan pada Gambar 2.5 dan *Gabriel Graph* (GG) yang ditunjukkan pada Gambar 2.6.



Gambar 2.5 *Relative Neighbour Graph* [3]



Gambar 2.6 *Gabriel Graph* [3]

2.3 *Particle Swarm Optimization* (PSO)

Particle Swarm Optimization (PSO) merupakan suatu algoritma optimasi yang didasarkan pada interaksi sosial dan komunikasi makhluk hidup. Dalam PSO, setiap anggotanya disebut dengan *particle*. *Particle* ini memiliki tiga karakteristik yaitu posisi, arah dan kecepatan. Setiap *particle* akan bergerak dalam ruang tertentu dan akan menyampaikan informasi mengenai posisi terbaik yang didapatkannya kepada *particle* yang lain dan akan menyesuaikan posisi dan kecepatan masing masing berdasarkan informasi yang diterima mengenai posisi terbaik yang didapat [4].

PSO memanfaatkan informasi kecepatan yang didapatkan dari persamaan (2.1). Serta memnfaatkan informasi dari arah yang didapatkan dari persamaan (2.2).

$$\boldsymbol{Speed} = \sqrt{\frac{(\boldsymbol{y2}-\boldsymbol{y1})^2 + (\boldsymbol{x2}-\boldsymbol{x1})^2}{\boldsymbol{t2}-\boldsymbol{t1}}} \tag{2.1}$$

$$\boldsymbol{\theta} = \boldsymbol{tan}^{(-1)}\frac{\boldsymbol{y2}-\boldsymbol{y1}}{\boldsymbol{x2}-\boldsymbol{x1}} \tag{2.2}$$

Keuntungan dalam menggunakan PSO diantaranya adalah [5]

- PSO mudah untuk diimplementasikan karena memiliki parameter yang sedikit.
- Pada PSO, semua partikel mengingat nilai terbaik sebelumnya.
- PSO lebih efisien dalam mengatasi perbedaan partikel.

2.4 GPSO

Dalam proses *greedy forwarding*, GPSR melakukan penerusan data hanya berdasarkan informasi mengenai tetangga-tetangganya pada saat itu juga. Ini dapat menimbulkan suatu masalah karena bisa jadi, tetangga yang akan menjadi *forwarding*

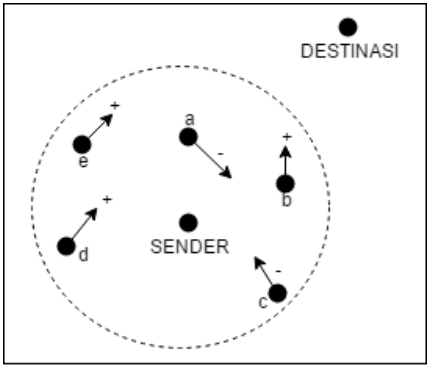
node bergerak menjauhi destinasi, serta *node* yang optimum pada *greedy forwarding* hanya memanfaatkan informasi posisi.

Dengan menggunakan algoritma *particle swarm optimization* maka diperlukan informasi arah serta kecepatan dari sebuah *node* sehingga *node* yang dipilih menjadi lebih optimum dibandingkan hanya memilih *node* berdasarkan posisi paling dekat dengan destinasi.

Menggunakan informasi yang didapatkan dari persamaan 2.1 serta persamaan 2.2 digunakan pemilihan *node* selanjutnya menggunakan algoritma *particle swarm optimization* dimana akan dihitung *node* yang optimum dengan menggunakan informasi kecepatan, arah serta jarak dengan destinasi.

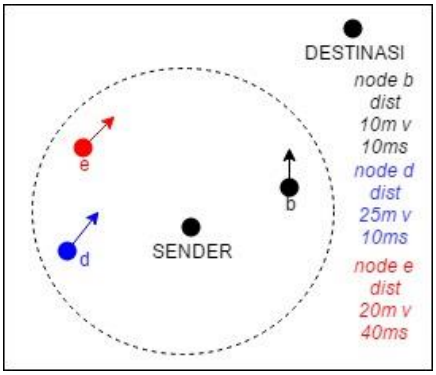
Berdasarkan Gambar 2.7, aturan memilih *node forwarding* pada GPSO adalah sebagai berikut:

- Pertama, *node sender* mendapatkan *hello message* dari *node* sekitar yang berada pada jangkauan *node sender*, ketika mengirimkan *hello message*, *node* sekitarnya akan mengirimkan informasi mengenai posisi terakhir dari *node* tersebut, sehingga ketika menerima *hello message*, *node sender* akan menghitung nilai arah serta kecepatan dari *node* yang mengirimkan *hello message*.



Gambar 2.7 Skema Pemilihan *Node Forwarding*

- Kedua, setelah mendapatkan informasi nilai arah serta kecepatan dari *node* tetangga, sender akan memilih *node* yang memiliki nilai arah yang sama dengan nilai arah *sender* terhadap destinasi, berdasarkan Gambar 2.7 *node* b,e dan d memiliki nilai arah yang sama dengan nilai arah *sender* terhadap destinasi, sehingga *node* a dan c tidak akan dipilih untuk menjadi *forwarding node*.



Gambar 2.8 Ilustrasi Pemilihan *Forwading Node*

- Terakhir untuk memilih *node* yang optimum berdasarkan informasi dan perhitungan menggunakan persamaan 2.3

$$\boldsymbol{time} = \frac{(\boldsymbol{y2} - \boldsymbol{y1}) + (\boldsymbol{x2} - \boldsymbol{x1})}{\sqrt{\frac{(\boldsymbol{y2}-\boldsymbol{y1})^2+(\boldsymbol{x2}-\boldsymbol{x1})^2}{\boldsymbol{t2}-\boldsymbol{t1}}}} \qquad (2.3)$$

didapatkan bahwa *node* **e** yang paling optimum untuk menjadi *forwarding* node dikarenakan memiliki kecepetan 40m/s serta jarak 20m sehingga dibutuhkan waktu 0.2s untuk mencapai destinasi.

2.5 *OpenStreetMap*

OpenStreetMap adalah *website* yang berguna untuk memetakan seluruh dunia secara bebas dan dan dapat diubah dengan mudah, dimana *openstreetmap* dibangun oleh komunitas. Mereka bertujuan agar seluruh dunia dapat dipetakan dan dirubah datanya sesuai yang kita inginkan. *OpenStreetMap* dapat digunakan oleh siapa saja secara gratis asalkan memberikan kredit dalam penggunaan *openstreetmap* [6].

Pada Tugas Akhir ini penulis menggunakan *openstreetmap* untuk memetakan kota Surabaya.

2.6 JOSM

JOSM adalah editor tambahan untuk merubah data *openstreetmap* untuk Java 8. JOSM mendukung kemampuan dalam memuat jalur GPX, citra latar belakang dan data OSM dari data lokal maupun data *online* dan memungkinkan pengguna untuk merubah data OSM seperti *node*, jalan, dan relasi antar jalam serta metadata yang dimiliki oleh OSM [7]. Pada Tugas Akhir ini penulis menggunakan JOSM untuk merubah data peta Surabaya yang telah didapat agar tidak ada jalan yang buntu .

2.7 *Simulation of Urban Mobility* (SUMO)

SUMO merupakan sebuah aplikasi simulasi lalu lintas jalan yang bersifat *open source*, portabel, mikroskopik, dan kontinu yang di desain untuk menangani jaringan jalan yang besar [8].

SUMO terdiri dari beberapa *tools* yang dapat membantu simulasi lalu lintas. *Tools* yang digunakan dalam pembuatan Tugas Akhir ini adalah:

- *netgenerate* merupakan *tools* yang berfungsi untuk membuat peta jalan yang seperti *grid, spider* atau bahkan bersifat abstrak. *Netgenerate* juga berfungsi untuk menentukan kecepatan maksimum jalan dan membuat *traffic light* pada peta. Hasil dari *netgenerate* adalah file

dengan ekstensi .net.xml. Dalam Tugas Akhir ini, *netgenerate* digunakan untuk membuat peta skenario dengan bentuk *grid*.

- *netconvert* merupakan program CLI yang berfungsi untuk melakukan konversi dari peta seperti *OpenStreetMap* menjadi format *native* SUMO. Pada Tugas Akhir ini penulis menggunakan netconvert untuk mengkonversi peta dari *OpenStreetMap*.
- *randomTrips.py merupakan tool* dalam SUMO untuk membuat rute acak yang akan dilalui oleh kendaraan dalam simulasi.
- *duarouter merupakan tool untuk m*embuat detail perjalanan setiap kendaraan berdasarkan output dari *randomTrips.py*.
- *sumo adalah p*rogram yang melakukan simulasi lalu lintas berdasarkan data - data yang didapatkan dari map.net.xml dan route.rou.xml. Menghasilkan *file* scenario.xml.
- *sumo-gui* untuk melihat simulasi yang dilakukan oleh SUMO secara grafis.
- *traceExporter.py* yang bertujuan untuk mengkonversi output dari sumo menjadi format yang dapat digunakan pada *simulator* NS-2. Pada Tugas Akhir ini penulis menggunakan *traceExporter.py* untuk mengonversi scenario.xml menjadi scenario.tcl yang dapat digunakan pada NS-2.

2.8 AWK

AWK adalah sebuah program yang digunakan untuk mengolah sebuah text (*text processing*) yang digunakan sebagai ekstraksi dari sebuah *dataset*. AWK ditulis dengan menggunakan bahasa pemrogramannya sendiri yaitu awk *programming language*. AWK berisi kumpulan perintah yang dijalankan pada *dataset* yang tesedia [9]. Pada Tugas Akhir ini penulis menggunakan AWK untuk memproses data yang dihasilkan dari

simulasi dengan menggunakan NS-2 untu mendapatkan anilisis mengenai *packet delivery ratio* serta *end-to-end delay* dan *routing overhead* dari simulasi yang telah dijalankan.

2.9 NS-2

NS2 adalah alat simulasi jaringan *open source* yang banyak digunakan dalam mempelajari struktur dinamik dari jaringan komunikasi. Simulasi dari jaringan nirkabel dan protokol (seperti algoritma routing, TCP, dan UDP) dapat diselesaikan dengan baik dengan simulator ini. Karena kefleksibelannya, NS2 menjadi popular dikalangan komunitas peneliti sejak awal kemunculannya pada tahun 1989.

NS2 terdiri dari dua bahasa pemrograman yaitu C++ dan OTcl(Objek-oriented Tool Command Language). C++ mendefinisikan mekanisme internal dari simulasi objek dan OTcl berfungsi untuk menset simulasi dengan assembly dan mengkonfigurasi objek sebagai penjadwalan diskrit. C++ dan OTcl saling berhubungan menggunakan TclCL. Setelah simulasi, output NS2 dapat berupa basis teks atau animasi berdasarkan simulasi. Untuk menginterprestasikan output ini secara grafik dan interaktif maka dibutuhkan NAM (Network Animator) dan XGraph. Untuk menganalisa tingkah laku dari jaringan user dapat mengekstrak subset dari data teks dan mentransformasikannya agar menjadi lebih atraktif.

Pada prakteknya, NS2 merupakan simulasi yang berjalan pada sistem UNIX. Oleh sebab itu NS2 dapat berjalan dengan baik di sistem operasi Linux, OpenBSD, FreeBSD, dan sistem operasi berbasis unix lainnya. Walaupun demikian NS2 dapat juga berjalan pada Windows dengan menggunakan tool tambahan yaitu Cygwin. Cygwin adalah port dari tool pengembangan GNU (*GNU's Not UNIX*) untuk Microsoft Windows [10].

Hasill dari simulasi NS-2 yang dijalankan berbentuk *trace file* dengan menggunakan ekstensi *file* .tr. *Trace file* berisi data pengiriman serta penerimaan paket yang dilakukan oleh setiap *node* ketika melakukan simulasi. Setiap jenis paket memiliki pola

yang berbeda sehingga dapat dilakukan analisis terhadap simulasi yang telah dijalankan [11]. Secara umum format penulisan pada *trace file* dapat dilihat pada Gambar 2.9, 2.10 dan 2.13

```
s 0.009388887 _8_ RTR  --- 0 gpsr 29 [0 0 0 0] ------- [8:255 -1:255 32 0]
```

Gambar 2.9 Contoh Baris Pengiriman *Hello Message*

```
r 0.010644982 _2_ RTR  --- 0 gpsr 29 [0 ffffffff 8 800] ------- [8:255 -
1:255 32 0]
```

Gambar 2.10 Contoh Baris Penerimaan Hello Message

```
s 10.000000000 _26_ AGT  --- 297 cbr 32 [0 0 0 0] ------- [26:0 25:0 32 0]
[0] 0 0
```

Gambar 2.11 Contoh Paket Komunikasi Data CBR

Pada Gambar 2.9 dan 2.10 menunjukkan pengiriman *hello message* yang dilakukan oleh *node* dengan ID-8 sehingga paket *hello message* tersebut diterima oleh *node* dengan ID-2, dengan melihat data pada *trace file* tersebut didapatkan informasi bahwa *node* dengan ID-2 memiliki *neighbour* dengan ID-8. Sedangkan Gambar 2.11 menunjukkan *node* dengan ID-26 mengirimkan paket *constant bit rate* CBR. Untuk melihat keterangan lebih lengkap dari data yang dihasilkan oleh *trace file* dapat dilihat pada Tabel 2.1.

Tabel 2.1 Tabel Detail Penjelasan Data *Trace File*

Kolom ke-	Penjelasan	Isi
1	*Event*	*r : Received* *s : Send* *f : Forwarded* *D : Dropped*
2	Time	*Timestap* pada *event* paket
3	ID *node*	_xx_ , dari 0 hingga banyak *node*
4	Layer	AGT : *Application* RTR: *Routing* LL : *Link Layer* IFQ : *Queue* Paket

Kolom ke-	Penjelasan	Isi
		MAC : MAC PHY : *Physical*
5	*Flags*	- - - : Tidak ada COL : MAC *Collision* NRTE : RTR *No Route Entry*
6	Nomor paket	Angka bulat dari 0 - banyak paket
7	Tipe paket	CBR : Berkas paket CBR (*Constant Bit Rate*) GPSR : Paket *routing* GPSR (*control* paket) RTS : *Request To Send* yang dihasilkan oleh MAC 802.11 CTS : *Clear To Send* yang dihasilkan oleh MAC 802.11 ACK : MAC ACK ARP : Paket *link layer address resolution protocol*
8	Ukuran	Ukuran *header*
9	Detail MAC	[a b c d] a : Perkiraan waktu transmisi b : Alamat penerima c : Alamat asal d : IP *header*
10	Mode GPSR	0 : *Greedy Forwarding* 1 : *Perimeter Forwarding* 2 : *Perimeter Probes* 3 : Paket *Beacon*
11	Banyak penerusan paket	Angka bulat dari 0 hingga total jumlah paket diteruskan
12	Panjang jalur	[a b] a : Panjang jalur *greedy* (-1 = *unreachable*) b : Panjang jalur terpendek (-1 = *unreachable*)

BAB III
DESAIN DAN PERANCANGAN

Pada bab ini akan dijelaskan mengenai hal-hal berkaitan dengan perancangan sistem yang akan dibuat. Perancangan tersebut mencakup deskripsi umum aplikasi, arsitektur sistem, model fungsional, diagram alir aplikasi serta antarmuka aplikasi.

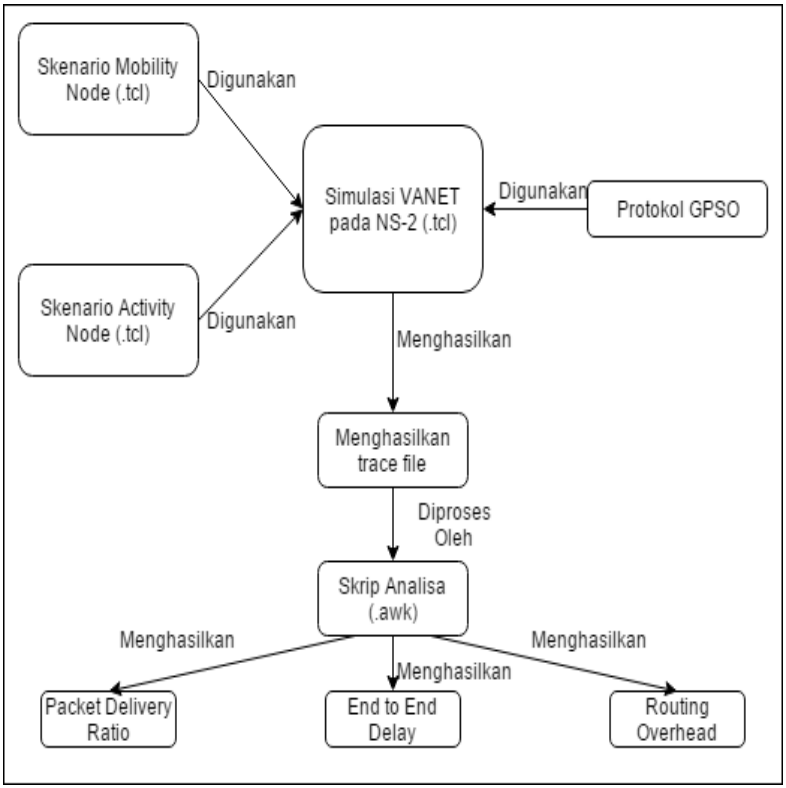
3.1 Deskripsi Umum Sistem

Pada Tugas Akhir ini penulis akan mengimplementasikan protokol GPSR dengan menambahkan algoritma PSO (GPSO) pada NS-2. Diagram dari rancangan simulasi dapat dilihat pada Gambar 3.1.

Dalam tugas akhir ini peta skenario yang digunakan didasarkan pada peta *grid* dan peta *real* lingkungan lalu lintas kota Surabaya. Peta *grid* dibuat menggunakan *tools* dari SUMO, peta tersebut digunakan untuk melakukan simulasi lalu lintas menggunakan SUMO. Kemudian peta lalu lintas kota Surabaya diambil datanya dengan menggunakan OpenStreetMap, dan dirubah datanya untuk dirapihkan terlebih dahulu menggunakan aplikasi JOSM.

Hasil simulasi dari SUMO tersebut kemudian digunakan model lalu lintas pada NS-2 menggunakan protokol GPSR dan GPSO sebagai protokol pengiriman data untuk melakukan simulasi VANET. Simulasi yang telah dijalankan pada NS-2 dengan menggunakan protokol GPSO serta menggunakan protokol GPSR tersebut kemudian dianalisis menggunakan skrip AWK untuk mendapatkan nilai dari *packet delivery rate*, *end-to-end delay* dan *routing overhead* dari simulasi tersebut.

Hasil analisis tersebut dapat menjadi acuan dalam mengukur performa protokol GPSO dan protokol GPSR pada peningkatan realibilitas pengirimin paket dengan menganilisis nilai *packet delivery rate*, *end-to-end delay* dan *routing overhead*.



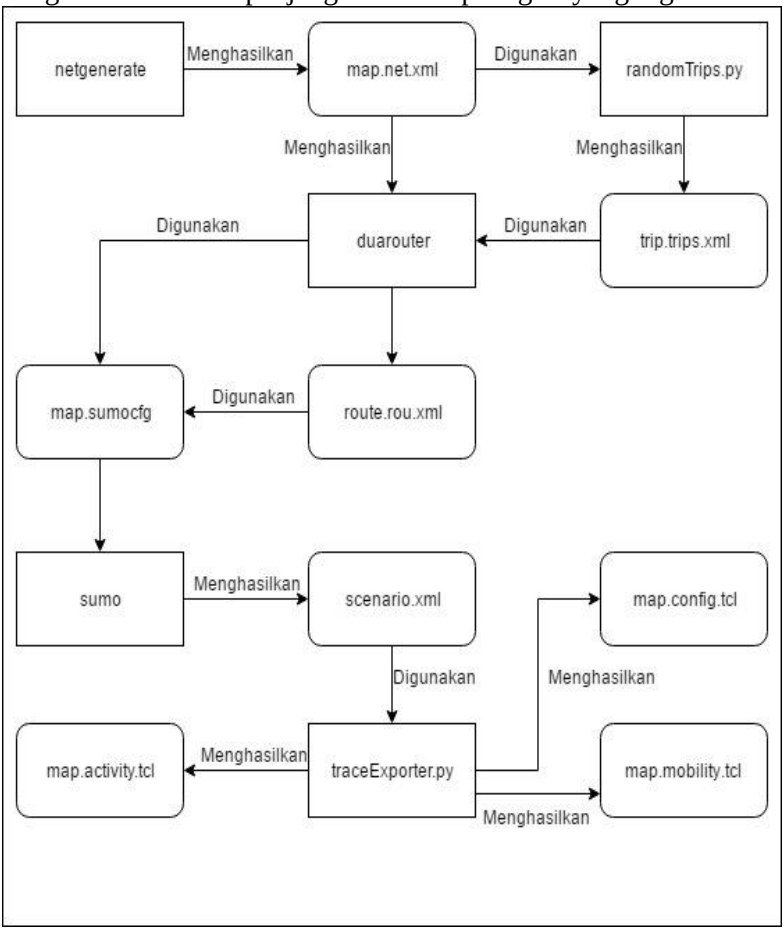
Gambar 3.1 Diagram Rancangan Simulasi

3.2 Perancangan Skenario Mobilitas

Perancangan skenario mobilitas uji coba VANET dimulai dari perancangan peta, pergerakan *node*, pembuatan aturan lalu lintas dan implementasi pergerakan. Dalam Tugas Akhir ini, bagian perencanaan peta pergerakan *node* menggunakan skenario peta *grid*. Peta *grid* yang dimaksud berupa jalan-jalan yang saling berpotongan dan membentuk petak yang menggambarkan contoh lingkungan di dunia nyata dalam bentuk yang sangat sederhana. Peta *grid* digunakan sebagai tes awal implementasi VANET karena peta *grid* lebih seimbang dan stabil.

3.2.1 Perancangan Skenario Mobilitas *Grid*

Perancangan skenario mobilitas dengan peta *grid* diawali dengan menentukan panjang dan lebar peta grid yang digunakan.



Gambar 3.2 Diagram Alur Pembuatan Skenario Grid

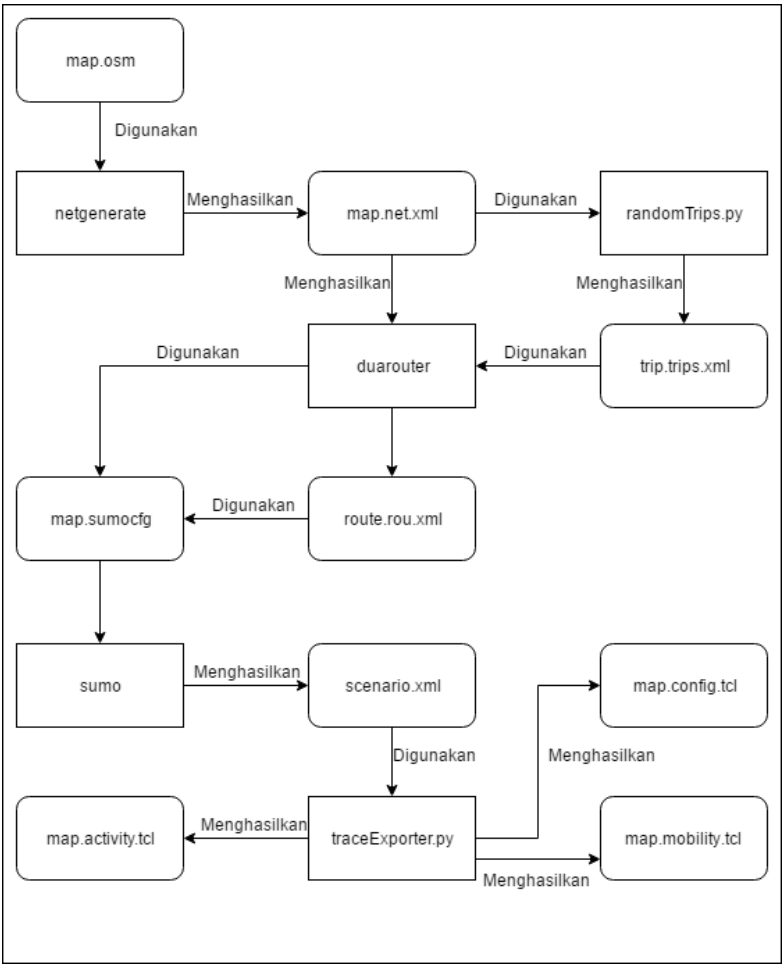
Setelah ukuran peta ditentukan, tentukan berapa banyak *grid* yang diinginkan. Kemudian, tentukan berapa banyak jumlah persimpangan pada peta. Misalkan ingin membuat peta dengan

grid sebanyak 7 x 7, maka persimpangan pada peta ada 8 buah. Peta *grid* kemudian akan dibuat dengan menggunakan *tools* SUMO yaitu netgenerate. Selain pengaturan mengenai banyak *grid* dan panjang *grid*, pengaturan mengenai kecepatan juga dilakukan saat pembuatan peta. Peta *grid* yang dihasilkan oleh netgenerate akan memiliki ekstensi .net.xml. Peta ini kemudian digunakan untuk membuat *file* deskripsi pergerakan kendaraan menggunakan *tool* randomTrips dan duarouter. Ilustrasi alur pembuatan skenario *grid* dapat dilihat pada Gambar 3.2

Simulasi lalu lintas dilakukan dengan menggunakan file peta yang telah digenerate dan file pergerakan yang telah dibuat. Hasil dari simulasi SUMO adalah sebuat file yang berekstensi .xml. Untuk dapat digunakan pada NS-2, file keluaran SUMO tadi harus dikonversi menjadi file dengan ekstensi .tcl yang dapat dilakukan dengan traceExporter.

3.2.2 Perancangan Skenario Mobilitas *Real*

Perangcangan skenario *real* dimulai dengan pemilihan tempat yang akan digunakan sebagai model pada simulasi NS-2. Skenario *real* menggunakan daerah Surabaya sebagai contoh peta yang diambil dengan menggunakan aplikasi *OpenStreetMap*, dengan menggunakan fitur *export* pada aplikasi *OpenStreetMap* penulis mendapatkan peta Surabaya. Setelah mendapatkan data peta Surabaya dengan ekstensi file .osm penulis diharuskan untuk merubah datanya terlebih dahulu agar tidak ada jalan yang terputus maupun daerah yang terisolasi. Setelah file OSM selesai dirubah sesuai dengan kebutuhan selanjutnya adalah merubah file .osm tersebut menjadi ekstensi .xml agar dapat dibaca oleh sumo. *File* .osm dikonversi menjadi .xml menggunakan fitur netconvert sehingga file yang dihasilkan adalah map.xml. Simulasi lalu lintas kemudian dilakukan dengan menggunakan *file* peta yang telah dikonversi dan *file* pergerakan yang dihasilkan sebelumnya.



Gambar 3.3 Diagram Alur Pembuatan Skenario Mobilitas Real

Hasil dari simulasi dari SUMO adalah sebuah *file* dengan ekstensi .xml. Lalu hasil simulasi tersebut dikonversi menjadi *file mobility*, *activity* dan *configuration* yang dapat kompatibel dengan NS-2 menggunakan traceExporter. Sehingga *file* terakhir yang dihasilkan oleh SUMO adalah file dengan ekstensi .tcl agar dapat

digunakan oleh NS-2 dalam menjalankan simulasi Ilustrasi alur pembuatan skenario mobilitas *real* dapat dilihat pada Gambar 3.3

3.3 Perancangan Simulasi pada NS-2

Simulasi VANET pada NS-2 dilakukan dengan menggunakan skenario mobilitas dan digabungkan dengan skrip Tcl yang berisikan konfigurasi mengenai lingkungan simulasi. Konfigurasi lingkungan simulasi VANET pada NS-2 dapat dilihat pada Tabel 3.1.

Tabel 3.1 Parameter Lingkungan Simulasi dengan Skenario

No	Parameter	Spesifikasi
1	*Network simulator*	NS-2, 2.35
2	*Routing Protocol*	GPSR dan GPSO
3	Waktu simulasi	200 detik
4	Area simulasi	700m x 700m
5	Banyak Kendaraan	25,50,75 dan 100
6	Radius transmisi	250m
7	Agen pengirim	*Constant Bit Rate* (CBR)
8	*Source / Destination*	Statis
9	*Packet Rate*	2kB
10	Ukuran paket	32
11	Protokol MAC	IEEE 802.11p
12	Propagasi sinyal	*Two-ray ground*
13	Tipe kanal	*Wireless Channel*

3.4 Perancangan Metriks Analisis

Berikut ini merupakan beberapa parameter yang dianalisis dalam Tugas Akhir ini:

3.4.1 *Packet Delivery Ratio*

Packet delivery ratio merupakan perbandingan dari jumlah paket komunikasi yang dikirimkan dengan paket komunikasi yang

diterima. *Packet delivery ratio* dihitung menggunakan persamaan 3.1, dimana *received* adalah jumlah paket komunikasi yang diterima dan *sent* adalah jumlah paket komunikasi yang dikirimkan [12]. Semakin tinggi *packet delivery ratio* semakin berhasil pengiriman paket yang dilakukan.

$$\boldsymbol{Packet\ Delivery\ Ratio} = \frac{\boldsymbol{received}}{\boldsymbol{sent}} \tag{3.1}$$

3.4.2 Rata-rata *End-to-end Delay*

Rata-rata *end-to-end delay* mengindikasikan interupsi transmisi paket dari *node* asal ke tujuan. Total interupsi didapatkan dari akumulasi *delay-delay* kecil yang ada dalam jaringan. Total interupsi terdiri dari *delay* yang mungkin karena *buffer* pada *route discovery latency*, *delay* pada antrian *interface*, retransmisi [12]. Rata-rata *end to end delay* pada paket yang diterima bisa dihitung berdasarkan selisih waktu antara transmisi dan respon paket pada *Constant Bit Rate* (CBR) dan membaginya dengan jumlah total transimi CBR menggunakan persamaan 3.2.

$$\boldsymbol{End\ to\ End\ Delay} = \sum_{i \leq sent}^{i=0} \frac{\boldsymbol{t_{received[i]} - t_{sent[i]}}}{\boldsymbol{sent}} \tag{3.2}$$

3.4.3 *Routing Overhead*

Routing overhead adalah jumlah paket routing yang ada didalam sebuah jaringan dibagi dengan jumlah keseluruhan paket data yang diterima [12], perhitungan menggunakan persamaan 3.3.

$$\boldsymbol{Routing\ Overhead} = \sum_{i \leq sent}^{i=0} \frac{\boldsymbol{packet\_routing}_{\boldsymbol{i}}}{\boldsymbol{received_i}} \tag{3.3}$$

3.5 Protokol GPSO

Protokol GPSO merupakan turunan dari protokol GPSR. Protokol GPSR memiliki dua metode dalam mengirimkan paket data yaitu menggunakan *greedy forwarding* dan *perimeter forwarding*. Implementasi *particle swarm optimazation* dilakukan pada saat mode *greedy forwarding* khusunya pada saat pemilihan *forwarding node*.

```
Input: ForwardingNodeF; DestinationD; NeighborList(F)
Output: Next − Hop − Node if Particle Swarm Optimization is
        successfull
TimesToDestination ← Distance_D/Velo_D;
for every neighbor node_i ∈ NeighborList_(F) do
  if Direction_(i−>D)value in accordance with Direction_(F−>D)velue
then
    if TimesToDestination_(i) < TimesToDestination then
      TimesToDestination ←TimesToDestination(i);
      Next − Hop – Node ← i;
    end
    if TimesToDestination(i) = TimesToDestination then
      if Distance(i) < Distance(D) then
        TimesToDestination ←TimesToDestination_(i);
        Next − Hop – Node ← i;
      end
    end
  end
end
if Next − Hop − Node > 0 then
  return Next − Hop − Node
end
else
  return NULL
end
```

Gambar 3.4 *Pseudocode* Pemilihan *Forwarding Node*

Langkah penilihan *forwarding node* pada GPSO pada bagian 2.4 diwujudkan dalam *pseudocode* yang dapat dilihat pada Gambar 3.4. Pada *pseudocode* dalam Gambar 3.4 dijelaskan *node* harus memiliki arah yang sama kepada *node* tujuan, karena biasanya pada GPSR paket akan dikirimkan kepada *node* dengan posisi yang lebih dekat dengan destinasi tetapi tidak melihat apakah mengarah kepada destinasi atau tidak sehingga memungkinkan paket menjauhi destinasi dan gagal untuk terkirim. Maka dengan memilih *node* yang memiliki nilai arah yang sama terhadap destinasi bisa mengurangi kegagalan dalam mengirimkan paket, dan setelah terkumpul list *neighbour*, dipilih berdasarkan waktu tempuh kepada destinasi yang lebih kecil. Ketika tidak ditemukan *forwarding node* yang searah, maka *forwarding* akan gagal. Kemudian, sesuai dengan algoritma protokol GPSR yang sudah dijelaskan pada bagian 2.2, paket akan diteruskan menggunakan *perimeter mode*.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi perancangan perangkat lunak dari aplikasi yang merupakan penerapan data, kebutuhan dan alur sistem yang mengacu pada desain dan perancangan yang telah dibahas sebelumnya. Selain itu, bab ini juga membahas lingkungan pembangunan perangkat lunak yang menjelaskan spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam pembangunan sistem.

4.1 Lingkungan Implementasi Protokol

Pengimplementasian protokol dilakukan pada lingkungan pengembangan yang rinciannya dapat dilihat pada Tabel 4.1

Tabel 4.1 Spesifikasi Perangkat yang Digunakan untuk

Komponen	Spesifikasi
CPU	*Intel® Core™ i7-3630QM CPU @ 2.40GHz × 8*
Memori	4GB
Penyimpanan	40GB
Sistem Operasi	Ubuntu 14.04 64 Bit
Network Simulator	2.35
SUMO	0.30
JOSM	12275

4.2 Implementasi Skenario Mobilitas

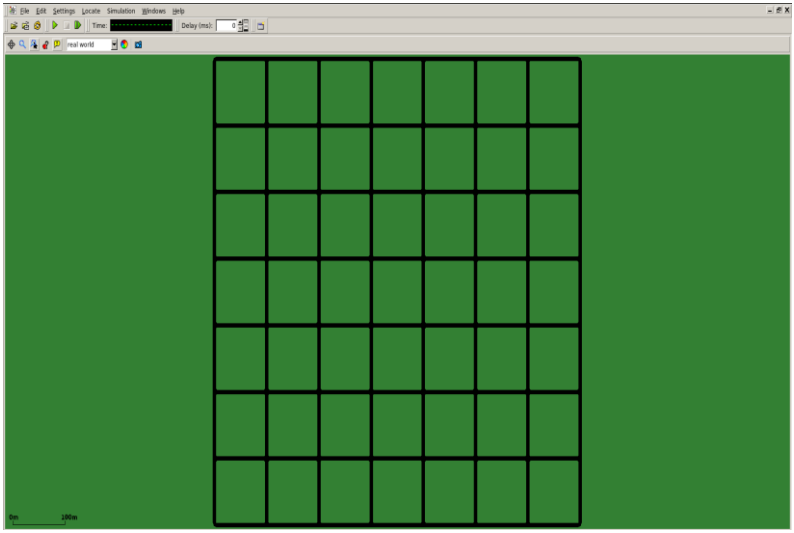
4.2.1 Implementasi Skenario Peta *Grid*

Dalam mengimplementasikan skenario *grid*, SUMO menyediakan *tools* untuk membangun peta *grid* yaitu netgenerate. Untuk menggunakan netgenerate dalam membangun skenario dengan peta seluas 700 x 700 dengan detail jumlah petak dari kiri ke kanan sebanyak 7 petak, sehingga diperlukan jumlah titik

persimpangan antara jalan vertikal dan horizontal sebanyak 8 titik x 8 titik dengan masing-masing luasan per petak 100m x 100m) dan dengan kecepatan standar 15 m/s digunakan perintah sesuai dengan perintah di bawah ini,

```
netgenerate -g --grid.number=8 --grid.length=100
default.speed=15 --tls.guess=1 output-file=map.net.xml
```

Screenshot hasil peta yang telah dibuat dengan netgenerate dapat dilihat pada Gambar 4.1.



Gambar 4.1 Peta Hasil netgenerate

Lalu dilakukan pembuatan model kendaraan dan pergerakan kendaraan dengan menentukan titik awal dan titik akhir setiap kendaraan secara acak menggunakan modul randomTrips.py Untuk memperpanjang perjalanan kendaraan agar kendaraan tetap stabil sejumlah *node* yang diinginkan, ditambahkan parameter --intermediate yang digunakan untuk memperlama *trips* di dalam jaringan dengan parameter *integer* sebesar kira-kira setengah

ukuran peta. Jikapun, *trips* yang dibuat melebihi batas waktu simulasi, tidak masalah karena dapat dipangkas menggunakan pengaturan *end time*, pembuatan *random trip* dapat dilihat pada perintah randomTrips.py di bawah ini,

```
/usr/local/share/sumo-0.30.0/tools/randomTrips.py -n
map.net.xml -e 25 --intermediate=500 -l --seed $RANDOM
--trip-attributes="departLane=\"best\"
departSpeed=\"max\" departPos=\"random_free\"" --o
trip.trips.xml
```

File route.rou.xml yang merupakan *output* dari randomTrips.py kemudian diproses menggunakan duarouter untuk membuat jalur yang digunakan kendaraan untuk mencapai tujuannya dengan menggunakan perintah duarouter di bawah ini,

```
duarouter -n map.net.xml -t trip.trip.xml -o route.rou.xml
```

Setelah didapatkan jalur kendaraan selanjutnya adalah membuat skenario dari jalannya kendaraan dengan menggunakan modul **sumo**, didalamnya dilakukan pengaturan lama jalannya simulasi dengan waktu 200 detik, perintah selengkapnya dapat dilihat di bawah,

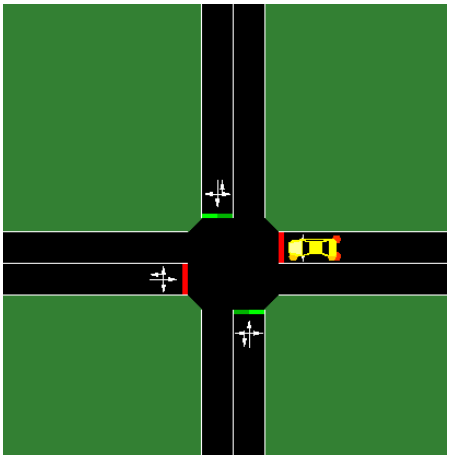
```
sumo -b 0 -e 200 -n map.net.xml -r route.rou.xml --fcd-
output=scenario.xml
```

Selanjutnya agar dapat melihat skenario yang terbentuk dari scenario.xml diharuskan membuat *file* tambahan sehingga dapat melihat hasil dari *random trip* yang telah dibuat sebelumnya, dengan menggunakan kode sumber di bawah ini *file* untuk melihat hasil skenario yang dibuat dinamai sumo.sumocfg.

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

Untuk menjalankan hasil dari skenario digunakan modul sumo-gui dimana perintah sumo-gui dapat dilihat di bawah.

```
sumo-gui sumo.sumocfg
```



Gambar 4.2 Hasil *Capture* sumo-gui

Hasil *capture* dari sumo-gui dapat dilihat pada Gambar 4.2. Terrakhir, setelah skenario terbentuk dibuat menjadi *output* scenario.tcl agar bisa dibaca oleh simulasi, perintah selengkapnya dapat dilihat pada perintah traceExporter.py di bawah ini.

```
/usr/local/share/sumo-0.30.0/tools/traceExporter.py --fcd-
input=scenario.xml --ns2mobility-output=scenario.tcl
```

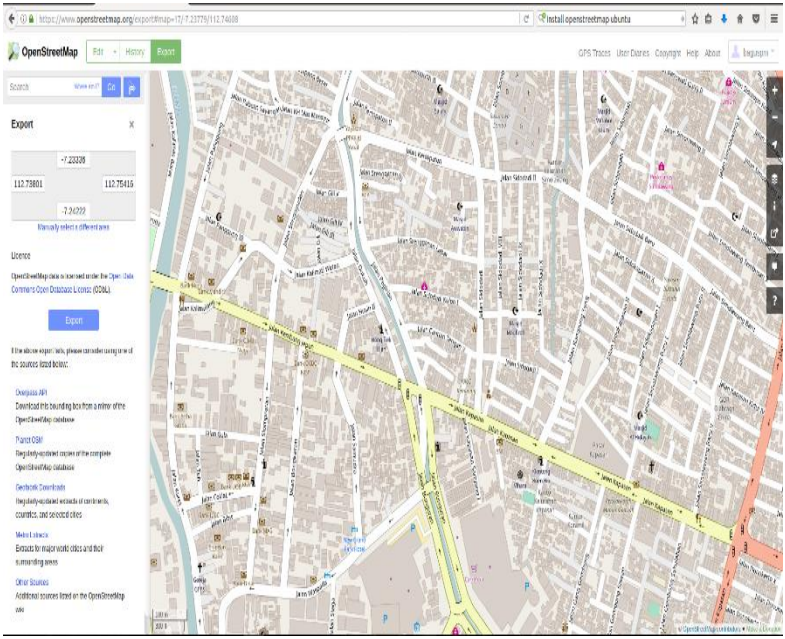
Hasil dari skenario *grid* yang telah dibuat bisa dilihat pada Gambar 4.10.

```
$node_(19) set X_ 591.3
$node_(19) set Y_ 1.65
$node_(19) set Z_ 0
$ns_ at 0.0 "$node_(19) setdest 41.86 501.65 15.00"
$node_(20) set X_ 591.3
$node_(20) set Y_ 1.65
$node_(20) set Z_ 0
$ns_ at 0.0 "$node_(20) setdest 591.3 1.65 15.00"
$node_(21) set X_ 427.99
$node_(21) set Y_ 101.65
$node_(21) set Z_ 0
$ns_ at 0.0 "$node_(21) setdest 427.99 101.65 12.24"
$node_(22) set X_ 98.35
$node_(22) set Y_ 72.85
$node_(22) set Z_ 0
$ns_ at 0.0 "$node_(22) setdest 98.35 72.85 15.00"
$node_(23) set X_ 454.1
$node_(23) set Y_ 98.35
$node_(23) set Z_ 0
$ns_ at 0.0 "$node_(23) setdest 454.1 98.35 15.00"
$node_(24) set X_ 398.35
$node_(24) set Y_ 427.82
$node_(24) set Z_ 0
```

Gambar 4.3 Cuplikan Isi File scenario.tcl Pada Skenario Grid

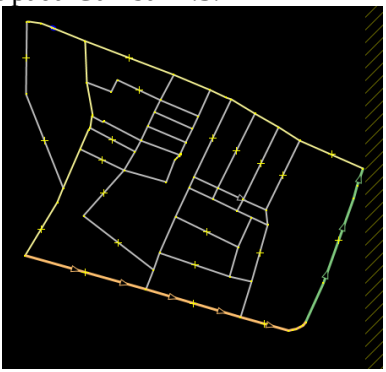
4.2.2 Implementasi Skenario Peta *Real*

Implementasi skenario pada peta *real* hampir mirip dengan cara membuat skenario peta *grid*, yang membedakan adalah kita diharuskan untuk mengekspor peta yang akan digunakan sebagai peta simulasi melalui website *OpenStreetMap*. Dengan memilih posisi yang diinginkan pada *OpenStreetMap* dimana telah menampilkan peta Surabaya, setelah menandai wilayah yang akan digunakan untuk simulasi gunakan menu "Manually select a different area" yang ada pada halaman *OpenStreetMap* setelah itu pilih menu "Export" maka secara otomatis akan terunduh file map.osm. Menu pada *OpenStreetMap* dapat dilihat pada Gambar 4.4.



Gambar 4.4 Menu Pemilihan Peta pada *OpenStreetMap*

Hasil ekspor peta dari *OpenStreetMap* masih mentah, dimana masih banyak jalan yang terpotong sehingga diperlukan perubahan sehingga tidak ada jalan yang terpotong maupun jalan yang terisolasi. *Screenshoot* dari hasil proses perubahan data peta *real* dapat dilihat pada Gambar 4.5.



Gambar 4.5 Hasil Peta *Real* yang Telah Dirubah

Setelah proses perubahan peta selesai selanjutnya adalah merubah *file* map.osm menjadi *file* yang dapat diketahui oleh SUMO. Dengan menggunakan modul netconvert dimana modul tersebut merubah file dengan ekstensi .osm menjadi .net.xml agar dapat dibaca oleh SUMO dalam membuat *random trip* pada peta yang telah dibuat. Perintah untuk merubah *file* map.osm menjadi map.net.xml dapat dilihat pada perintah netconvert di bawah ini.

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

Setelah peta selesai dibuat maka selanjutnya adalah membuat *random trip* yang akan dijalankan pada peta yang telah dibuat. Perintah untuk membuat *random trip* dapat dilihat pada perintah randomTrips.py di bawah ini.

```
/usr/local/share/sumo-0.30.0/tools/randomTrips.py -n
map.net.xml -e 25 -l --trip-
attributes="departLane=\"best\" departSpeed=\"max\"
departPos=\"random_free\"" --o trip.trips.xml
```

Sama dengan sebelumnya diperlukan *file* sumo.sumocfg yang ditunjukkan pada kode sumber di bawah ini untuk menjalankan hasil skenario yang telah dibuat.

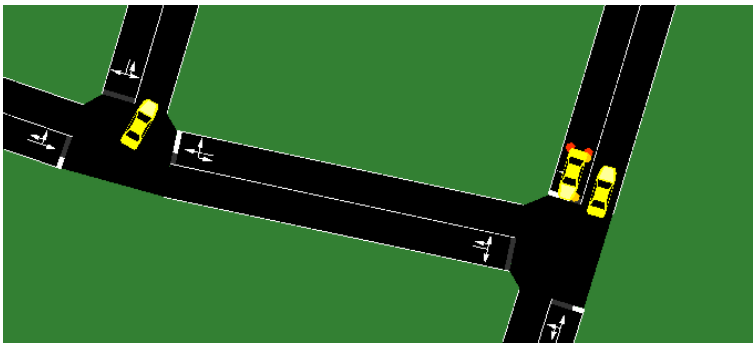
```
<?xml version="1.0" encoding="UTF-8"?>
<configuration
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/
sumoConfiguration.xsd">
        <input>
                <net-file value="map.net.xml"/>
                <route-files value="route.rou.xml"/>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

Untuk menjalankan hasil dari skenario digunakan modul sumo-gui dimana perintah selengkapnya dapat dilihat pada perintah sumo-gui di bawah ini.

```
sumo-gui sumo.sumocfg
```

Perintah selanjutnya sampai dengan pembuatan scenario.tcl sama dalam membuat implementasi pada peta *grid*. Hasil dari *capture* sumo-gui dapat dilihat pada Gambar 4.6. Hasil

dari pembuatan scenario.tcl pada skenario *real* dapat dilihat pada Gambar 4.7.



Gambar 4.6 Hasil *Capture* sumo-gui

```
$node_(21) set X_ 360.17
$node_(21) set Y_ 350.19
$node_(21) set Z_ 0
$ns_ at 0.0 "$node_(21) setdest 360.17 350.19 13.89"
$node_(22) set X_ 536.37
$node_(22) set Y_ 345.27
$node_(22) set Z_ 0
$ns_ at 0.0 "$node_(22) setdest 536.37 345.27 13.89"
$node_(23) set X_ 436.07
$node_(23) set Y_ 101.84
$node_(23) set Z_ 0
$ns_ at 0.0 "$node_(23) setdest 436.07 101.84 9.71"
$node_(24) set X_ 296.86
$node_(24) set Y_ 208.08
$node_(24) set Z_ 0
$ns_ at 0.0 "$node_(24) setdest 296.86 208.08 13.89"
```

Gambar 4.7 Cuplikan Isi File scenario.tcl Pada Skenario *Real*

4.3 Implementasi Protokol GPSO

Kode protokol GPSR pada NS-2 versi 2.35 berada pada direktori ns-2.35/gpsr. Di dalam direktori tersebut, terdapat beberapa *file*. File yang akan digunakan untuk mengimplementasikan GPSO adalah gpsr_neighbour.cc, gpsr_neighbour.h, gpsr.cc, gpsr.h dan gpsr_packet.h. Implementasi protokol GPSO dibuat sesuai dengan perancangan implementasi yang sudah dijabarkan pada subseksi 3.5. Perubahan yang dilakukan pada protokol GPSR agar menjadi GPSO antara lain sebagai berikut:

4.3.1 Modifikasi Struktur *Header* pada GPSR

Struktur *header* yang dimiliki oleh GPSR tidak menyertakan variabel kecepatan serta arah yang diperlukan oleh GPSO untuk menentukan *forwarding node*, oleh karena itu diperlukan tambahan variabel kecepatan serta arah pada *header* ketika *node* mengirimkan *hello message* ke sekitarnya. *Header* GPSR terletak pada gpsr_neighbour.h, pada *header* GPSR tambahkan tiga variabel yaitu double dir, double velo, serta double ts_. Variabel ts_ digunakan untuk menyimpan detik ke-berapa node tersebut mengirimkan *hello message* sehingga bisa digunakan dalam perhitungan kecepatan. Kode implementasi modifikasi struktur header pada GPSR dapat dilihat selengkapnya pada kode fungsi struct gpsr_neighbor.

```
1.  struct gpsr_neighbor {
2.      .
3.      double dir;
4.      double velo;
5.      double ts_;     //the last time stamp of the hello msg
    from it
6.  };
```

4.3.2 Modifikasi Struktur *Header Hello Message*

Pada struktur header GPSR ditambahkan variable waktu sehingga diperlukan pula tambahan variabel waktu pada *header hello message* yang terdapat pada gpsr_packet.h. Kode implementasi perubahan *header hello message* selengkapnya dapat dilihat pada funsgi struct hdr_gpsr_hello.

```
1.  struct hdr_gpsr_hello {
2.          .
3.      float ts_;
4.  };
```

4.3.3 Modifikasi Struktur Penyimpanan Data *Hello Message*

Setelah membuat struktur baru dalam *header* pada GPSR maka diperlukan perubahan pada data-data yang ada pada *hello message* yang dikirimkan oleh sebuah *node* untuk melakukan pembaharuan data supaya data yang diolah untuk menjadi arah dan kecepatan lebih akurat, data tersebut terdapat pada gpsr.cc. Perubahan tersebut terjadi dengan mengambil posisi paling terbaru dari sebuah node ketika melakukan pengiriman paket. Kode implementasi perubahan *hello message* selengkapnya dapat dilihat pada fungsi hellomsg.

```
1.  void
1.  GPSRAgent::hellomsg(){
2.      .
3.      .
4.      .
5.      MobileNode *thisNode = (MobileNode
    *)(Node::get_node_by_address(cmh->last_hop_));
6.      double newx = thisNode->X();
7.      double newy = thisNode->Y();
8.
9.      ghh->type_ = GPSRTYPE_HELLO;
```

```
10.     ghh->x_ = newx;
11.     ghh->y_ = newy;
12.     ghh->ts_ = (float)GPSR_CURRENT;
13.
14.     send(p, 0);
15. }
```

4.3.4 Implementasi Fungsi Perhitungan Kecepatan dan Nilai Arah

Informasi penting yang digunakan untuk membangun protokol *routing* GPSO adalah kecepatan dan nilai arah. Dengan memanfaatkan data yang disimpan pada *hello message*, kecepatan dan nilai arah dihitung sesuai dengan persamaan 2.1 dan 2.2 untuk kemudian dimasukkan dalam *header* GPSR yang akan disimpan pada gpsr_neigbour.cc. Perhitungan arah kecepatan di implementasikan dalam fungsi find_dir seperti pada cuplikan fungsi GPSRneighbors::find_dir dan perhitungan kecepatan pada fungsi count_velo seperti pada cuplikan kode sumber pada fungsi GPSRNeighbors::count_velo.

```
1.  double GPSRNeighbors::find_dir(double x, double y,
    double nx, double ny){
2.      if (y == ny)
3.              return 0; // 0 degree
4.      else if (x == nx)
5.              return 1.5708; //90 degree
6.      else
7.              //count angle
8.              return atan((ny - y)/(nx - x));
9.  }
```

```
1.  double GPSRNeighbors::count_velo(double y, double
    ny, double x, double nx, double t, double nt){
2.      if (t == nt)
3.              //time is same, so the vehile is not in motion
```

```
4.                  return 0;
5.          else{
6.                  return fabs(sqrt(pow((nx-x),2)+pow((ny-
    y),2))/nt-t);
7.          }
8.  }
```

4.3.5 Modifikasi *Beacon Processing*

Setelah setiap *node* mengirimkan *hello message* maka diperlukan adanya perubahan pada *beacon processing* dikarenakan menambahkan informasi mengenai kecepatan serta nilai arah dari *node* yang bersangkutanm perubahan tersebut dilakukan pada gpsr.cc. Pada fungsi *beacon processing* ditambahkan proses perhitungan kecepatan serta nilai arah, sehingga setiap *node* memiliki tambahan informasi tersebut. Kode implementasi modifikasi *beacon processing* dapat dilihat selengkapnya pada fungsi GPSRAgent::recvHello.

```
1.  void
2.  GPSRAgent::recvHello(Packet *p){
3.          struct hdr_cmn *cmh = HDR_CMN(p);
4.          struct hdr_gpsr_hello *ghh =
    HDR_GPSR_HELLO(p);
5.
6.          MobileNode *thisNode = (MobileNode
    *)(Node::get_node_by_address(cmh->last_hop_));
7.          double newx = thisNode->X();
8.          double newy = thisNode->Y();
9.
10.         double dir =    nblist_->find_dir((double)ghh->x_,
    (double)ghh->y_, newx, newy);
11.         double column =  nblist_-
    >find_column((double)ghh->x_, (double)ghh->y_,
    newx, newy);
```

```
12.     double velo = nblist_->count_velo((double)ghh->y_,
    newy, (double)ghh->x_, newx, (double)ghh->ts_,
    (double)GPSR_CURRENT);
13.
14.     nblist_->newNB(cmh->last_hop_, newx, newy, velo,
    dir, column);
15. }
```

4.3.6 Modifikasi Struktur Penyimpanan Variabel Kecepatan, Arah, dan Waktu

Setelah menyimpan informasi arah, kecepatan dan waktu pada struktur *header* GPSR, informasi tersebut perlu disimpan ketika insialisasi data *neighbour*, karena menambahkan beberapa variabel diperlukan untuk menambahkan insialisasi penyimpanan arah, kecepatan serta waktu. Kode implementasi dapat dilihat selengkapnya pada fungsi GPSRNeighbors::newNB.

```
1.  void
2.  GPSRNeighbors::newNB(nsaddr_t nid, double nx,
    double ny, double velo, double dir, double column){
3.      struct gpsr_neighbor *temp = getnb(nid);
4.      if(temp==NULL){ //it is a new neighbor
5.              temp=(struct
    gpsr_neighbor*)malloc(sizeof(struct gpsr_neighbor));
6.              temp->id_ = nid;
7.              temp->x_ = nx;
8.              temp->y_ = ny;
9.              temp->dir = dir;
10.             temp->velo = velo;
11.             temp->ts_ = GPSR_CURRENT;
12.             temp->next_ = temp->prev_ = NULL;
13.     if(tail_ == NULL){ //the list now is empty
14.             head_ = tail_  = temp;
15.             nbSize_ = 1;
16.     }
```

```
17.     else { //now the neighbors list is not empty
18.             tail_->next_ = temp;
19.             temp->prev_ = tail_;
20.             tail_ = temp;
21.             nbSize_++;
22.     }
23.     }
24.     else { //it is a already known neighbor
25.     temp->ts_ = GPSR_CURRENT;
26.     temp->x_ = nx; //the updating of location is allowed
27.     temp->y_ = ny;
28.             temp->dir = dir;
29.             temp->velo = velo;
30.     }
31. }
```

4.3.7 Modifikasi Pemilihan *Forwading Node*

Proses pemilihan *forwarding node* diimplementasikan dalam fungsi forwardData(). Dalam fungsi forwardData(), disediakan metode untuk memilih *forwarding node* berdasarkan mode *forwarding*. Pada mode *greedy forwarding*, proses pemilihan *forwarding node* terbaik terdapat pada fungsi gf_nextnop(). Langkah pemilihan *forwarding node* GPSO yang telah dirancang sebelumnya pada bagian 3.5, kemudian diimplementasikan dan *pseudocode*nya dapat dilihat pada gambar 3.4. Kode implementasi fungsi pemilihan *forwarding node* dapat dilihat pada lampiran A.6.

4.4 Implementasi Simulasi pada NS-2

Implementasi simulasi NS-2 dilakukan dengan cara pendeskripsian lingkungan simulasi pada sebuah file dengan eksentsi .tcl, file-file ini berisi konfigurasi setiap node dan proses yang dilakukan selama simulasi berjalan.

Pada awal instalasi dan proses *patching* protol GPSR Ke Liu terdapat *source code* original yang bernama wireless-gpsr.tcl serta cbr100.tcl dan simulasi *grid* berukuran 10x10. Penulis menggunakan file tersebut dengan merubah pengaturan sesuai dengan simulasi yang telah dibuat sebelumnya. Cuplikan pengaturan NS-2 yang telah dibuat dapat dilihat pada Gambar 4.9. Pengaturan yang ada pada wireless-gpsr.tcl akan dijelaskan pada Tabel 4.2.

Tabel 4.2 Tabel Konfigurasi TCL

Parameter	**Nilai**	**Penjelasan**
Chan	Wireless	Komunikasi menggunakan media wireless
Prop	TwoRayGroun d	Tipe propagasi sinyal wireless adalah two ray ground
Mac	Mac/802_11	Menggunakan tipe MAC 802.11 karena komunikasi data bersifat wireless
Ifq	Queue/Droptail /Priqueue	Menggunakan priority queue sebagai antrian paket dan paket yang dihapus saat antrian penuh adalah paket yang paing baru
Ll	LL	Menggunakan link layer standar
Ant	Antenna/Omni Antenna	Jenis antena yang digunakan adalah omni antenna
Opt(x)	800	Ukuran simulasi adalah 800m
Opt(y)	800	Ukuran Simulasi adalah 800m
Opt(cp)	gpsoCBRtest.tc l	Menjalankan simulasi mengirimkan paket CBR

Parameter	Nilai	Penjelasan
		berada pada file gpsoCBRtest.tcl
Opt(sc)	GPSOnodePos.tcl	Menjalankan simulasi posisi setiap *node* berada pada file gpsoCBRtest.tcl
Opt(ifqlen)	50	Jumlah maksimal paket pada antrian
Opt(nn)	102	Jumlah maksimal node
Opt(stop)	200.0	Waktu simulasi berjalan
Hello_period_	1	Periode pengiriman *hello message*
Phy/Wirelessphy	0.2818	Setiap node memiliki jangkauan 250m

Konfigurasi selengkapnya dari TCL yang dijalankan ada pada lampiran A.1.

```
set opt(chan)       Channel/WirelessChannel
set opt(prop)       Propagation/TwoRayGround
set opt(netif)      Phy/WirelessPhy
set opt(mac)        Mac/802_11
set opt(ifq)        Queue/DropTail/PriQueue
set opt(ll)         LL
set opt(ant)        Antenna/OmniAntenna
set opt(x)          800
set opt(y)          800
set opt(cp)         "./gpsoCBRtest.tcl"
set opt(sc)         "./GPSOnodePos.tcl"
set opt(ifqlen)     50;
set opt(nn)         52;
```

Gambar 4.8 Cuplikan Pengaturan NS-2

4.5 Implementasi Metriks Analisis

Simulasi yang sudah dijalankan oleh NS-2 menghasilkan keluaran *trace file* bernama trace.tr yang berisikan data mengenai apa saja yang terjadi selama simulasi dalam bentuk *plain text*. Dari data *trace file*, dapat dianalisa performa dari *routing protocol* yang dijalankan. Dua buah metrik yang akan dianalisa adalah *packet delivery ratio* dan *end-to-end delay*.

4.5.1 Implementasi *Packet Delivery Ratio*

Packet Delivery Ratio didapatkan dengan membandingkan pengiriman dan penerimaan data yang dikirimkan oleh agen. Pada Tugas Akhir ini, penulis menggunakan agen dengan pengiriman data CBR. Kemudian, pada Persamaan 3.1 telah dijelaskan rumus untuk menghitung *packet delivery ratio*. Skrip awk untuk menghitung *packet delivery ratio* berdasarkan kedua informasi tersebut dapat dilihat pada lampiran A.9. Cara menjalankan skrip awk dapat dilihat pada perintah awk di bawah ini.

```
awk –f PDR.awk trace.tr
```

Hasil dari perintah yang dijalankan adalah *packet delivery ratio* dari simulasi yang telah dijalankan dapat dilihat pada gambar 4.10.

Gambar 4.9 Hasil Perhitungan *Packet Delivery Ratio*

4.5.2 Implementasi *End-to-end Delay*

Perhitungan *end to end delay* didasarkan pada Persamaan 3.2 dan sudah dijelaskan pada bagiann 3.4.2 Skrip awk untuk menghitung *end to end delay* berdasarkan kedua informasi tersebut dapat dilihat pada lampiran A.10. Contoh perintah untuk memanggil skrip tcl

untuk menganilisis *trace file* dan *outputnya* dapat dilihat pada perintah awk di bawah ini.

```
awk –f E2E.awk trace.tr
```

Hasil dari perintah yang dijalankan adalah *end-to-end delay* dari simulasi yang telah dijalankan dapat dilihat pada gambar 4.11.

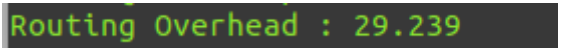
Gambar 4.10 Hasil Perhitungan *End to End Delay*

4.5.3 Implementasi *Routing Overhead*

Perhitungan RO didasarkan pada Persamaan 3.3 dan sudah dijelaskan pada bagian 3.4.3 Skrip awk untuk menghitung RO berdasarkan kedua informasi tersebut dapat dilihat pada lampiran A.11. Contoh perintah untuk memanggil skrip tcl untuk menganilisis *trace file* dan *outputnya* dapat dilihat pada perintah awk di bawah ini.

```
awk –f E2E.awk trace.tr
```

Hasil dari perintah yang dijalankan adalah *end-to-end delay* dari simulasi yang telah dijalankan dapat dilihat pada Gambar 4.12.



Gambar 4.11 Hasil Perhitungan *Routing Overhead*

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan membahas uji coba dan evaluasi dari skenario yang telah dijalankan pada NS-2. Uji coba dilakukan untuk mengetahui hasil dari protokol GPSO dalam menjawab rumusan masalah pada tugas akhir ini.

5.1 Lingkungan Uji Coba

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

Tabel 5.1 Spesifikasi Perangkat yang Digunakan

Komponen	Spesifikasi
CPU	*Intel® Core™ i7-3630QM CPU @ 2.40GHz × 8*
Sistem Operasi	Ubuntu 14.04 64 Bit
Memori	4GB
Penyimpanan	40GB

Pengujian dilakukan dengan menjalankan skenario yang disimulasikan pada NS-2. Keluaran dari simulasi skenario pada NS2 adalah sebuah *trace file* dengan nama file trace.tr yang akan dianalisis dengan menggunakan *script* AWK untuk menghitung *packet delivery ratio* dan menghitung *end-to-end delay.*

5.2 Hasil Uji Coba

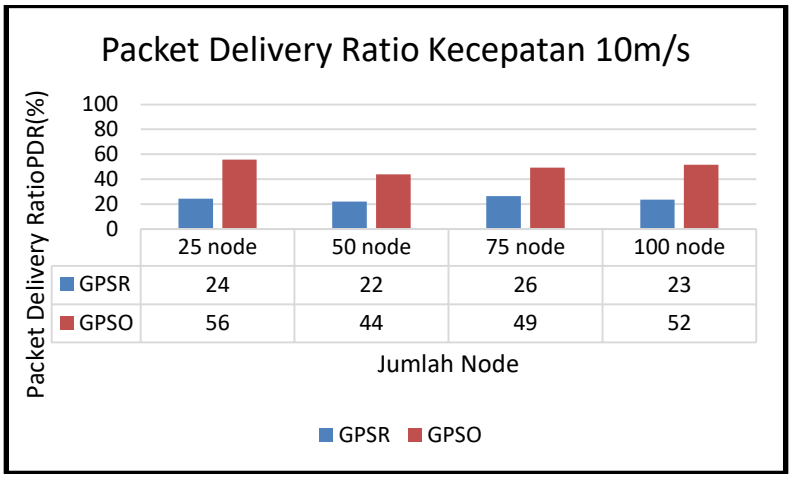
Uji coba dari skenario yang telah dijalankan dari skenario grid yang telah dibuat berdasarkan implementasi yang telah dijelaskan pada bagian 4.2.1 dapat dilihat hasil nilai *packet delivery ratio, end to end delay* dan *routing overhead* sebagai berikut :

5.2.1 Hasil Uji Coba Skenario *Grid*

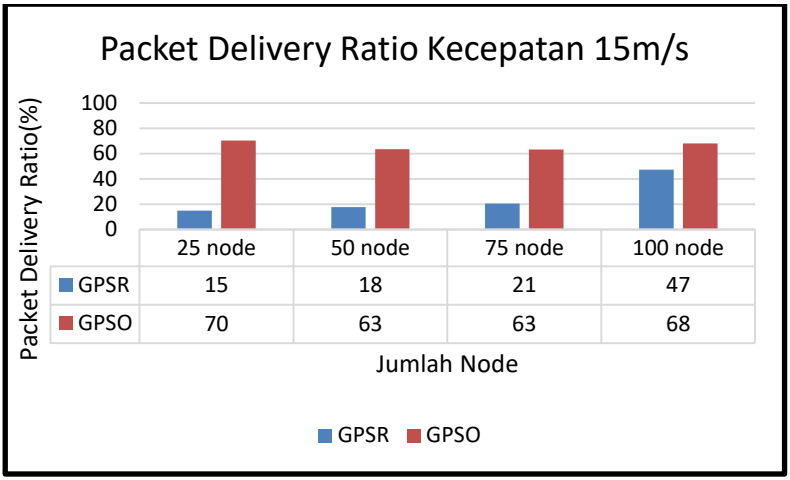
Hasil pengujian data *packet delivery ratio* dan *end-to-end delay* pada skenario grid dilakukan pada area dengan luasan 700x700 meter dengan node sebanyak 25, 50, 75 dan 100 pada 3 kecepetan yang berbeda, yaitu pada kecepatan maksimal 10m/s, 15m/s dan 20m/s. Pada setiap banyak node, uji coba dilakukan pada 6 data, kemudian di rata-rata. Hasil pengambilan data *packet delivery ratio* kecepatan maksimal 10m/s, 15m/s dan 20m/s dapat dilihat pada Gambar 5.1, 5.2 dan 5.3.

5.2.1.1 Hasil *Packet Delivery Ratio* pada Skenario *Grid*

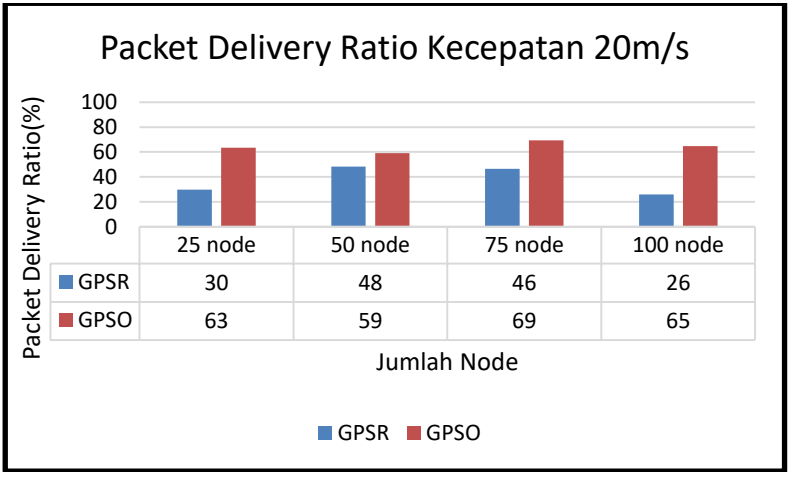
Berdasarkan grafik yang ditunjukkan pada Gambar 5.1, 5.2 dan 5.3 dapat dilihat bahwa GPSO memiliki rata-rata *packet delivery ratio* yang lebih tinggi dibandingkan dengan GPSR.



Gambar 5.1 Grafik *Packet Delivery Ratio* Terhadap Banyak *Node* pada Kecepatan 10m/s



Gambar 5.2 Grafik *Packet Delivery Ratio* Terhadap Banyak *Node* pada Kecepatan 15m/s



Gambar 5.3 Grafik *Packet Delivery Ratio* Terhadap Banyak *Node* pada Kecepatan 20m/s

Pada grafik dapat terlihat dengan kecepatan 15m/s sampai dengan 20m/s protokol GPSO memiliki nilai *packet delivery ratio*

yang lebih tinggi dibandingkan dengan *packet delivery ratio* dengan kecepatan 10m/s sedangkan pada GPSR tidak tentu namun dengan kecepatan 20m/s memiliki nilai *packet delivery ratio* yang lebih baik dibandingkan dengan kecepatan 10m/s dan 15m/s

Namun, jika melihat satu-satu dari 6 data yang diambil untuk hasil pengujian seperti yang dapat dilihat pada Tabel 5.2 yang merupakan cuplikan hasil pengujian pada kecepatan standar 20 m/s dan jumlah *node* 75, *packet delivery ratio* GPSO tidak selalu lebih baik daripada GPSR.

Tabel 5.2 Data Uji Skenario Grid pada node sebanyak 50 dan kecepatan 20m/s

Data uji Ke-	***Packet Delivery Ratio* GPSR (%)**	***Packet Delivery Ratio* GPSO (%)**
1	52	82
2	61	48
3	26	46
4	51	70
5	55	57
6	45	51

Selama data uji ke-1 hingga ke-6 pada protokol GPSR pada uji coba melakukan *drop paket* dikarenakan memilih *forwarding node* berdasarkan posisi saja tidak berdasarkan nilai arah serta waktu tempuh terhadap destinasi sehingga nilai tesebut sangat membantu pada protokol GPSO.

Nilai GPSO bisa lebih kecil bisa disebabkan oleh rentang nilai arah yang minus serta nilai arah yang positif tidak terlalu besar serta *node* yang optimum untuk dipilih memiliki rentang nilai arah yang lebih kecil sehingga memiliki nilai *packet delivery ratio* yang lebih kecil dibandingkan dengan protokol GPSR. Mengapa protokol GPSO memilih nilai arah yang positif untuk menjadi acuan *node* yang optimum, hal itu disebabkan oleh nilai arah *sender* terhadap *receiver* adalah positif sehingga GPSO mencari

node yang memiliki nilai arah positif tanpa menghiraukan nilai arah yang negatif.

Seperti pada data uji ke-2 untuk *node* berjumlah 50 dengan kecepatan 20m/s, *node* yang dipilih oleh protokol GPSR adalah *node* 9,14 dan 50 seperti yang bisa dilihat pada Gambar 5.4, dimana pada Gambar tersebut ditunjukkan bahwa protokol GPSR memilih *node* 9,14 dan 50 sebagai *forwarding node* sedangkan pada protokol GPSR *node* 9,14 dan 50 tidak akan dipilih dikarenakan *node* 9,14 dan 50 memiliki nilai arah yang negatif. Sedangkan nilai arah dari *node* pengirim terhadap *node* penerima pada skenario bernilai positif. Seperti yang ditunjukkan dengan analisis menggunakan AWK untuk melihat nilai arah pada *node* 9, 14 dan 50 yang kodenya dapat dilihat pada kode sumber di bawah ini.

```
1.  BEGIN{
2.      minus = 0;
3.      plus = 0;
4.      nol = 0;
5.  }
6.  {
7.      if (($5=9 ||$5=50 ||$5=14) && ( $8<0 ))
8.          minus++;
9.      if (($5=9 ||$5=50 ||$5=14) && ( $8>0 ))
10.         plus++;
11.     if (($5=9 ||$5=50 ||$5=14) && ( $8=0 ))
12.         nol++;
13. }
14. END{
15.     printf("minus %d", minus);
16.     printf("\nplus %d", plus);
17.     printf("\nnol %d\n", nol);
18. }
```

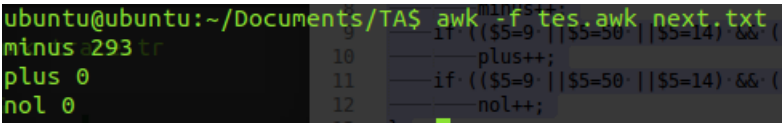
gf_nexthop 9
gf_nexthop 9
gf_nexthop 9
gf_nexthop 9
gf_nexthop 9
gf_nexthop 9
gf_nexthop 50
gf_nexthop 9
gf_nexthop 50
gf_nexthop 9
gf_nexthop 50
gf_nexthop 9
gf_nexthop 50
gf_nexthop 14
gf_nexthop 50
gf_nexthop 14
gf_nexthop 50
gf_nexthop 14
gf_nexthop 50
gf_nexthop 14

Gambar 5.4 Hasil Pemilihan *Forwarding Node* Pada Protokol GPSR

Pada kolom ke-5 dipilih nilai 9, 50 dan14 dikarenakan *node* yang dipilih oleh protokol GPSR untuk menjadi *forwarding node* adalah *node* 9,14 dan 50 sehingga pada kolom ke-5 dimana kolom ke-5 berisi id *node* dipilih untuk analisis hasil nilai arah pada protokol GPSO. Hasil dari nilai arah yang didapatkan oleh protokol GPSO dapat dilihat pada Gambar 5.5

Dari hasil yang didapatkan *node* 9,14 dan 50 memiliki nilai arah negatif. Hal itu yang menjadi penyebab nilai *packet delivery ratio* GPSO bisa lebih kecil dibandingkan dengan protokl GPSR, *node* optimum yang dimiliki oleh protokol GPSR memiliki nilai arah negatif sehingga protokol GPSO tidak bisa memiliki *node* tersebut menjadi *node* optimum. Maka dari itu pada uji skenario ke-2 bisa dilihat pada Tabel 5.2 protokol GPSO memiliki nilai *packet delivery ratio* yang lebih besar dan jauh lebih banyak bila

dibandingkan dengan protokol GPSO yang hanya memiliki nilai *packet delivery ratio* sebersar 48% berbanding dengan 61% milik protokol GPSR.

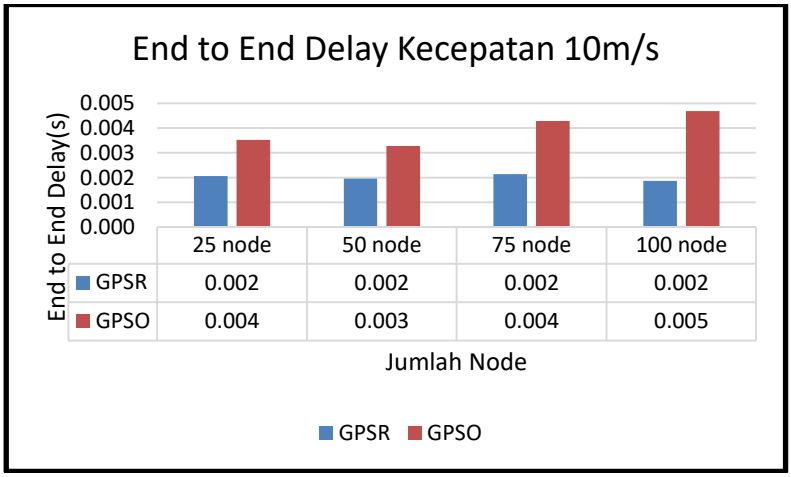


Gambar 5.5 Hasil Jumlah Nilai Arah Perhitungan Protokol GPSO

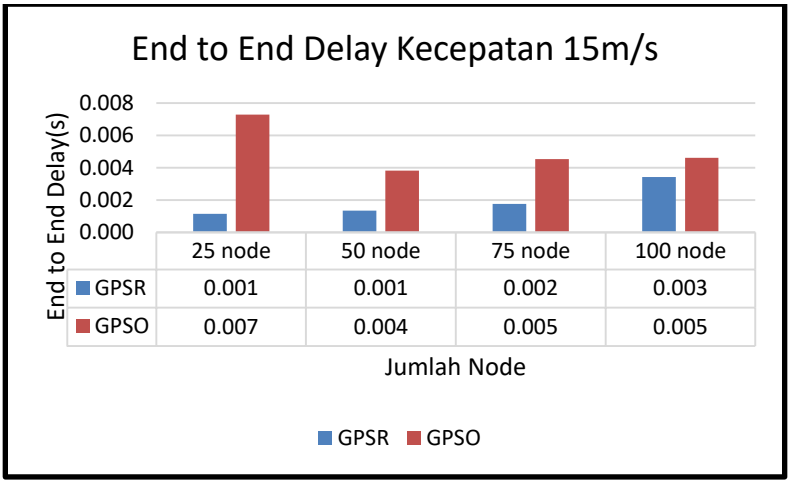
Rata-rata *packet delivery ratio* berdasarkan Gambar 5.1, 5.2 dan 5.3 untuk GPSR adalah 29% sedangkan untuk protokol GPSO mendapatkan rata-rata 60%

5.2.1.2 Hasil *End to End Delay* pada Skenario *Grid*

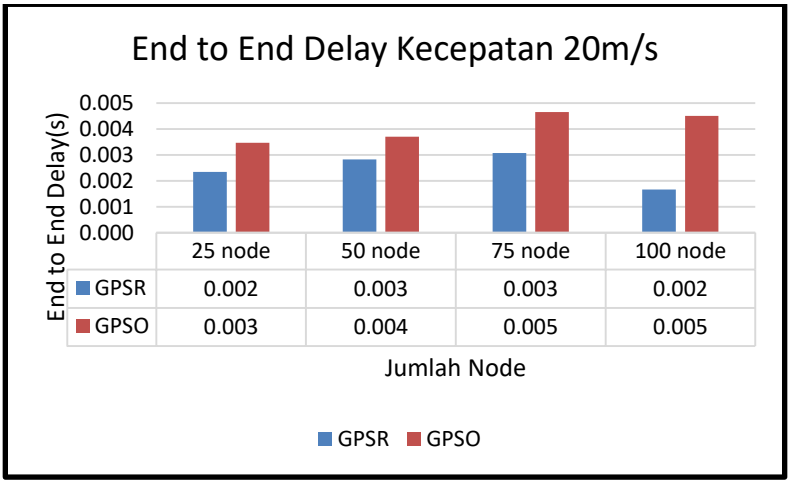
Pada Gambar 5.6, 5.7 dan 5.8 menjelaskan grafik hubungan antara *end to end delay* dengan kecepatan 10m/s, 15m/s dan 20m/s pada uji coba dengan *node* sebanyak 25, 50, 75 dan 100 *node*.



Gambar 5.6 Grafik *End to End Delay* Terhadap Banyak *Node* pada Kecepatan 10m/s



Gambar 5.7 Grafik *End to End Delay* Terhadap Banyak *Node* pada kecepeatan 15m/s



Gambar 5.8 Grafik *End to End Delay* Terhadap Banyak *Node* pada kecepeatan 20m/s

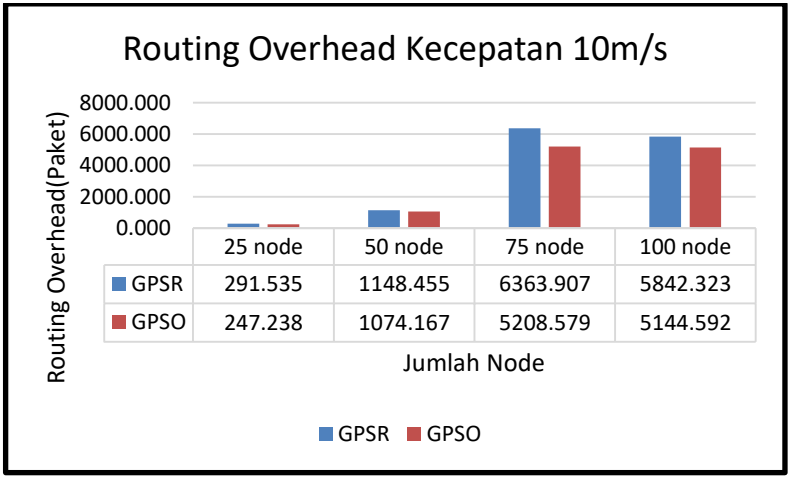
Pada hasil *end to end delay* dengan kecepatan 10m/s, 15m/s dan 20m/s GPSO selalu kalah dibandingkan dengan GPSR

hal ini dikarenakan GPSO memerlukan waktu untuk mengolah data pada pemilihan *forwarding node* untuk mendapatkan *node* yang optimum, hal itu dibarengi dengan peningkatan *packet delivery ratio* dari GPSO yang meningkat hampir dua kali lipat dibandingkan dengan GPSR.

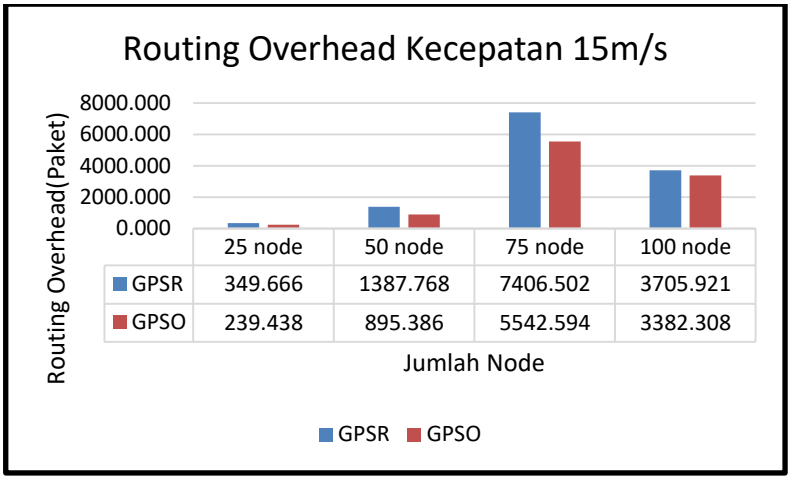
Rata-rata *end to end delay* berdasarkan Gambar 5.6, 5.7 dan 5.8 untuk GPSR adalah 0.002s sedangkan untuk protokol GPSO adalah 0.004s.

5.2.1.3 Hasil *Routing Overhead* pada Skenario *Grid*

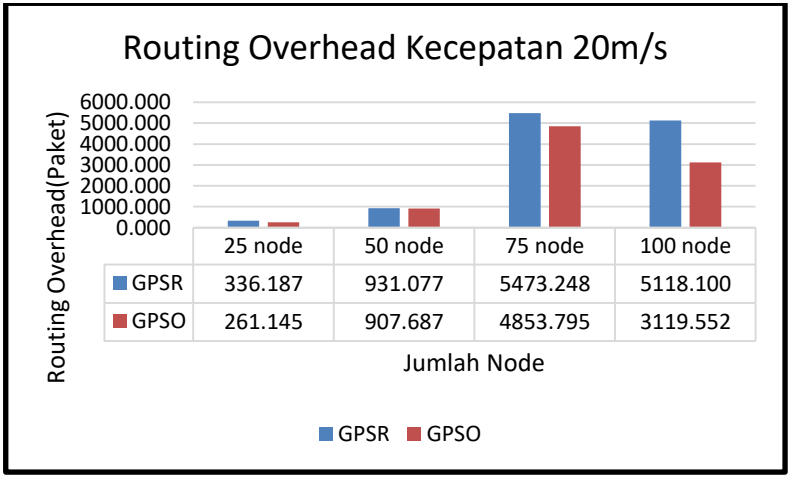
Pada Gambar 5.90, 5.10 dan 5.11 menjelaskan grafik hubungan antara *routing overhead* dengan kecepatan 10m/s, 15m/s dan 20m/s pada uji coba dengan *node* sebanyak 25, 50, 75 dan 100 *node*.



Gambar 5.9 Grafik *Routing Overhead* Terhadap banyak node pada Kecepatan 10m/s



Gambar 5.10 Grafik *Routing Overhead* Terhadap banyak node pada Kecepatan 15m/s



Gambar 5.11 Grafik *Routing Overhead* Terhadap banyak node pada Kecepatan 20m/s

Pada hasil *routing overhead* dengan kecepatan 10m/s, 15m/s dan 20m/s GPSO terbukti lebih baik dibandingkan dengan

GPSR hal ini dikarenakan GPSO mendapatkan *forwarding node* lebih cepat dibandingkan GPSR sehingga tidak perlu mencari *neighbour* lain yang memakan waktu lebih lama dibandingkan dengan protokol GPSR. Perbedaan nilai *routing overhead* berjalan sesuai dengan kenaikan banyaknya *node* yang digunakan pada simulasi. Semakin banyak *node* maka nilai dari *routing overhead* semakin besar.

Seperti yang ditunjuk pada Gambar 5.9, 5.10 dan 5.11 hasil dari *routing overhead* yang dimiliki oleh protokol GPSO memiliki nilai yang lebih baik dibandingkan dengan protokol GPSR. Namun apabila melihat hasil dari simulasi *node 25* dengan kecepatan 20m/s yang ditunjukkan pada Tabel 5.3 dan Table 5.4.

Tabel 5.3 Data Uji Skenario Grid Nilai *Packet Delivery Ratio* pada node sebanyak 25 dan kecepatan 20m/s

Data uji Ke-	*Packet Delivery Ratio* GPSR (%)	*Packet Delivery Ratio* GPSR (%)
1	30	72
2	75	86
3	23	51
4	2	74
5	30	40
6	18	57

Tabel 5.4 Data Uji Skenario Grid Nilai *Routing Overhead* pada node sebanyak 25 dan kecepatan 20m/s

Data uji Ke-	*Routing Overhead* GPSR	*Routing Overhead* GPSO
1	395.788	235.752
2	232.004	232.004
3	410.023	282.177
4	410.609	254.156
5	281.463	298.754
6	287.236	264.025

Nilai dari GPSR pada data uji ke-5 memiliki nilai *routing overhead* yang lebih baik dibandingkan dengan nilai *routing overhead* yang dimiliki protokol GPSO dimana protokol GPSR memiliki nilai 281 berbanding dengan 298 yang dimiliki oleh protokol GPSO sehingga nilai *routing overhead* tidak hanya bergantung pada hasil *packet delivery ratio*, meskipun nilai *packet delivery ratio* dari protokol GPSO memiliki nilai yang lebih baik dibandingkan dengan protokol GPSR. Namun semakin baik nilai *packet delivery ratio* cenderung memiliki nilai *routing overhead* yang dimiliki, hal ini disebabkan perbedaan masing-masing protokol dalam memilih *forwarding node* sehingga hasil dari *routing overhead* pun berbeda.

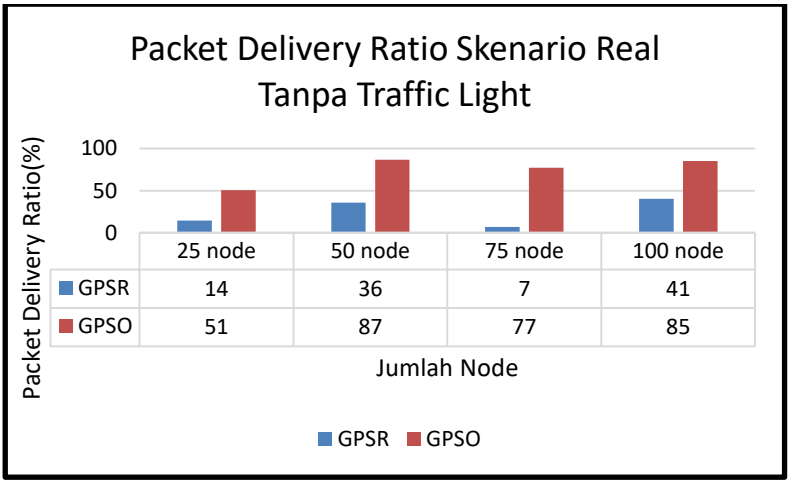
Rata-rata *routing overhead* berdasarkan Gambar 5.10, 5.11 dan 5.12 untuk GPSR adalah 3196.224 sedangkan untuk protokol GPSO adalah 2573.039.

5.2.2 Hasil Uji Coba Skenario *Real*

Pengujian pada skenario *real* dilakukan untuk melihat bagaimana performa algoritman GPSO jika diimplementasikan menggunakan peta dunia nyata dimana persebarannya kendaraan yang ada pada dunia nyata tidak selalu merata. Pengambilan data uji untuk penilaian *packet delivery ratio* dan *end to end delay* berdasarkan pada peta kota Surabaya dimana memiliki luasan area 700x700 dan node sebanyak 25, 50, 75 dan 100 dengan kecepatan standar sesuai dengan tipe jalan pada wilayah perumahan (tanpa *traffic light*) dan wilayah perkotaan (menggunakan *traffic light*).

5.2.2.1 Peta Daerah Surabaya Wilayah Perumahan

Hasil dari uji coba pengambilan nilai *packet delivery ratio* untuk peta daerah Surabaya dengan wilayah perumahan (tanpa *traffic light*) dapat dilihat pada Gambar 5.12

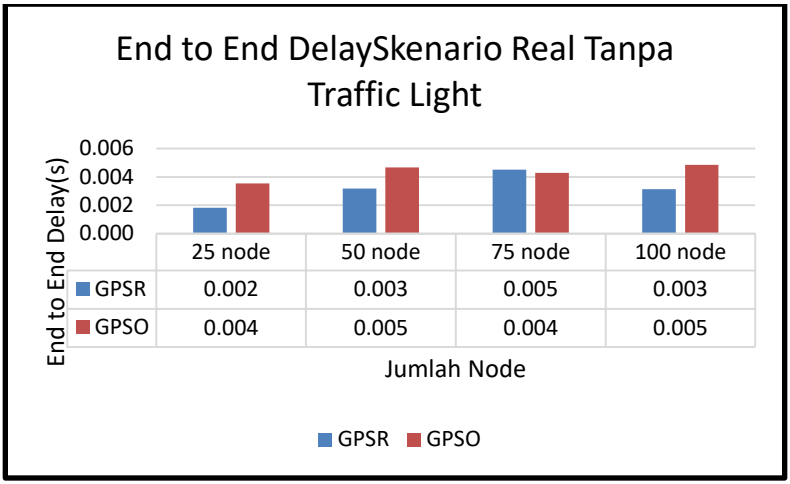


Gambar 5.12 Grafik *Packet Delivery Ratio* dengan Skenario *Real* Tanpa *Traffic Light*

Berdasarkan hasil uji coba pada skenario *real* dapat terlihat protokol GPSO masih lebih unggul dibandingkan dengan protokol GPSR hal ini disebabkan pemilihan *forwarding node* yang telah ditambahkan dengan algoritma *particle swarm optimization*. Hal ini disebabkan karena kecepetan yang dijalankan pada simulasi skenario *real* memiliki variasi yang sangat berbeda dikarenakan mengikuti aturan yang telah dibuat oleh *OpenStreetMap* ketika melakukan ekspor peta untuk kemudian digunakan pada uji coba ini.

Rata-rata *packet delivery ratio* pada uji skenario *real* ini adalah 24% bagi protokol GPSR sedangkan untuk protokol GPSO memiliki rata-rata sebesar 75%.

Sedangkan untuk hasil *end to end delay* protokol GPSR kembali unggul dibandingkan dengan protokol GPSO, namun perbedaan nilai *end to end delay* tersebut masih dibarengi dengan lebih tingginya nilai *packet delivery ratio* dari GPSR. Seperti yang ditunjukkan pada Gambar 5.13.



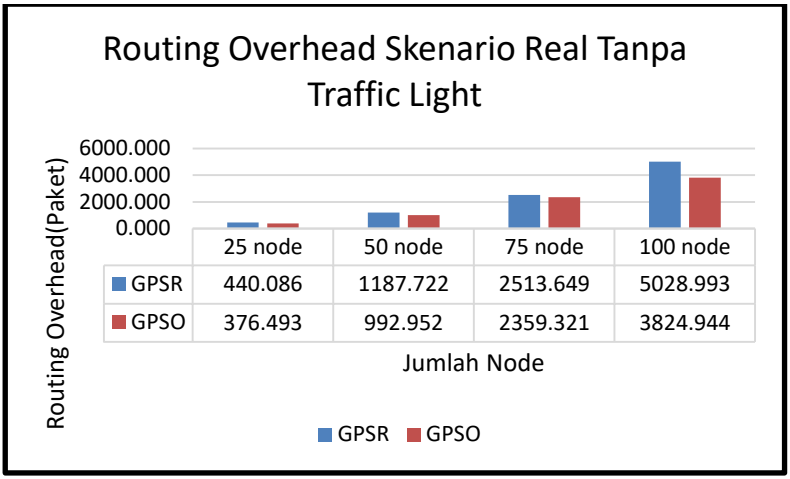
Gambar 5.13 Grafik *End to End Delay* dengan Skenario *Real* Tanpa *Traffic Light*

Dengan perbedaan nilai 0.001s antara protokol GPSR dengan protokol GPSO pada uji coba skenario *real* tanpa *traffic light* mengalami peningkatan nilai *packet delivery ratio* sebesar tiga kali lipat.

Nilai dari *end to end delay* GPSO tidak selalu lebih buruk dibandingkan dengan protokol GPSO, pada *node* 75 protokol GPSO memiliki nilai *end to end delay* yang lebih baik dibandingkan dengan protokol GPSR dan tetap memiliki nilai *packet delivery ratio* yang lebih baik dibandingkan dengan protokol GPSR.

Rata-rata *end to end delay* pada uji skenario *real* ini adalah 0.003s bagi protokol GPSR sedangkan untuk protokol GPSO memiliki rata-rata sebesar 0.004s.

Terakir untuk hasil *routing overhead* protokol GPSO kembali unggul dibandingkan dengan protokol GPSR, dan perbedaan nilai *routing overhead* tersebut masih dibarengi dengan lebih tingginya nilai *packet delivery ratio* dari GPSO. Seperti yang ditunjukkan pada Gambar 5.14.



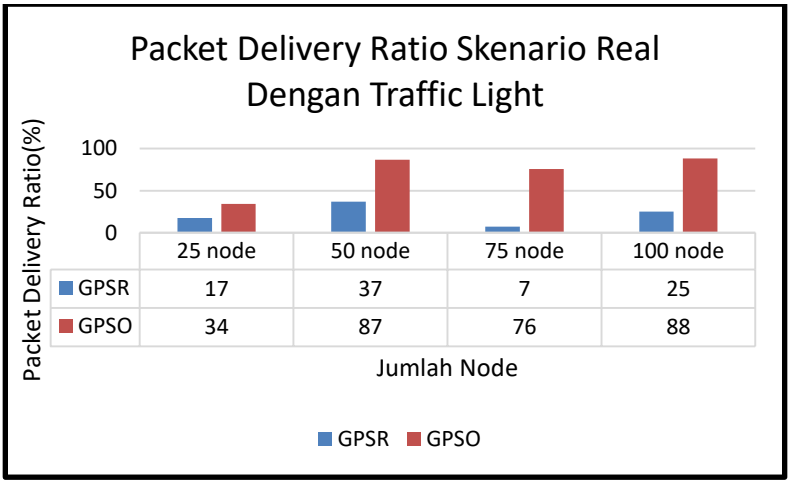
Gambar 5.14 Grafik RO dengan Skenario *Real* Tanpa *Traffic Light*

Rata-rata RO pada uji skenario *real* ini adalah 2292.613 paket bagi protokol GPSR sedangkan untuk protokol GPSO memiliki rata-rata sebesar 1888.427 paket.

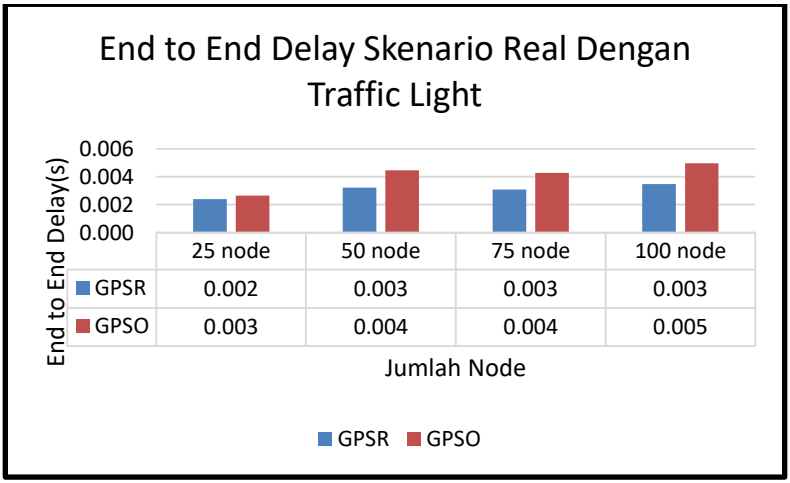
5.2.2.2 Peta Daerah Surabaya Wilayah Perkotaan

Hasil dari uji coba pengambilan nilai *packet delivery ratio* untuk peta daerah Surabaya dengan wilayah perkotaan (dengan *traffic light*) dapat dilihat pada Gambar 5.15.

Berdasarkan hasil uji coba skenario *real* pada peta kota Surabaya dengan wilayah perkotaan (dengan *traffic light*) hasil dari nilai *packet delivery ratio* protokol GPSO masih mengungguli protokol GPSO, dimana rata-rata nilai *packet delivery ratio* pada protokol GPSO adalah 71% berbanding dengan 22% yang dimiliki oleh protokol GPSO. Kembali protokol GPSO memiliki nilai *packet delivery ratio* yang hampir tiga kali lipat lebih besar dibandingkan dengan protokol GPSO.



Gambar 5.15 Grafik *Packet Delivery Ratio* dengan Skenario *Real* Dengan *Traffic Light*



Gambar 5.16 Grafik *End to End Delay* dengan Skenario Real Tanpa *Traffic Light*

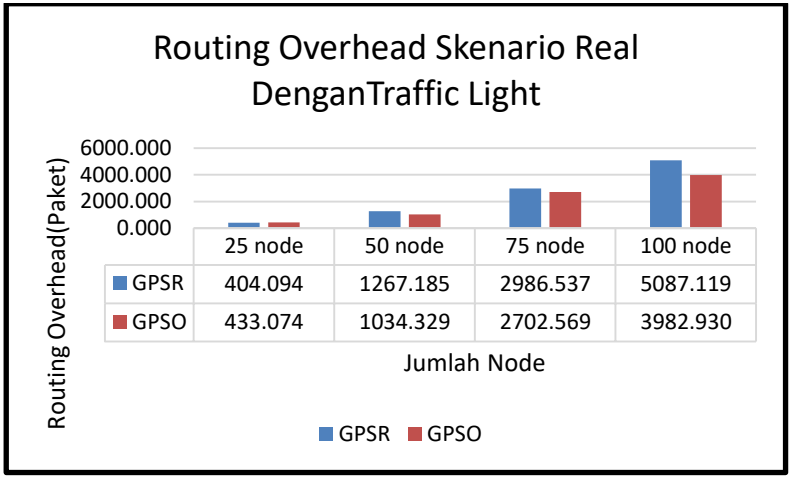
Sedangkan untuk hasil *end to end delay* protokol GPSO kembali memiliki nilai yang lebih buruk yang ditunjukkan pada

Gambar 5.16 dibandingkan dengan protokol GPSR dimana kali ini protokol GPSO selalu memiliki nilai yang lebih buruk bila dibandingkan dengan protokol GPSO. Pada skenario wilayah perkotaan protokol GPSR memiliki rata-rata nilai 0.003s sedangkan GPSO memiliki nilai 0.004s.

Rata-rata *end to end delay* pada uji skenario *real* ini adalah 0.003s bagi protokol GPSR sedangkan untuk protokol GPSO memiliki rata-rata sebesar 0.004s.

Terakir untuk hasil *routing overhead* protokol GPSO kembali unggul dibandingkan dengan protokol GPSR, dan perbedaan nilai *routing overhead* tersebut masih dibarengi dengan lebih tingginya nilai *packet delivery ratio* dari GPSO. Seperti yang ditunjukkan pada Gambar 5.17.

Rata-rata *routing overhead* pada uji skenario *real* ini adalah 2436.233 paket bagi protokol GPSR sedangkan untuk protokol GPSO memiliki rata-rata sebesar 2038.225 paket.



Gambar 5.17 Grafik RO dengan Skenario *Real* Dengan *Traffic Light*

Hasil dari nilai *packet delivery ratio* untuk uji coba skenario *real* mengalami peningkatan yang cukup signifikan bagi

protokol GPSO dimana mengalami peningkatan sebesar 13% bila dibandingkan dengan uji skenario *grid* sebaliknya protokol GPSR mengalamin penurunan meskipun tidak terlalu banyak yaitu 6% hal ini bisa jadi disebabkan oleh kecepatan yang ada pada skenario *real* ini memiliki variasi yang sangat berbeda sehingga mempengaruhi hasil dari *packet delivery ratio*.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan Tugas Akhir ini serta saran-sarang tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dalam proses pengerjaan Tugas Akhir yang melalui tahap perancangan, implementasi, serta uji coba, didapatkan kesimpulan sebagai berikut :

1. Hasil *packet delivery ratio* pada protokol GPSO lebih baik dibandingkan dengan GPSR yang telah diuji coba pada skenario *grid*, dimana menghasilkan *packet delivery ratio* yang hampir dua kali lipat yaitu pada GPSR memiliki rata-rata *packet delivery ratio* 29% berbanding 60% yang dimiliki oleh GPSO.
2. Hasil *end to end delay* protokol GPSO memiliki nilai yang lebih besar dibandingkan dengan protokol GPSR hal ini dikarenakan protokol GPSO memerlukan waktu dalam memilih *forwarding node* yang lebih optimum dibandingkan dengan protokol GPSR, dimana hasil rata-rata *end to end delay* untuk protokol GPSO adalah 0.003s berbanding dengan 0.002s yang dimiliki oleh protokol GPSR.
3. Hasil *routing overhead* protokol GPSO memiliki nilai yang lebih baik dibandingkan dengan protokol GPSR sesuai dengan peningkatan nilai *packet delivery ratio*. Dimana pada skenario grid protokol GPSO memiliki rata-rata nilai 2573.04 dan protokol GPSR memiliki nilai rata-rata 3196.224.
4. Kecepatan 15m/s cenderung memiliki *packet delivery ratio* yang lebih baik dibandingkan dengan kecepatan 10m/s dan 20m/s dimana kecepatan 15m/s memiliki *packet delivery ratio* rata-rata 66% berbanding dengan 50% yang dimiliki oleh kecepatan 10m/s dan 64% yang dimiliki oleh 20m/s.

5. Protokol GPSO pun masih lebih unggul bila dibandingkan dengan protokol GPRS pada uji skenario *real* dimana protokol GPSO memiliki rata-rata *packet delivery ratio* sebesar 73% dibandingkan dengan protokol GPSR yang memiliki nilai 23%.
6. Implementasi PSO pada protokol GPSR dapat meningkatkan reliabilitas pengiriman data pada VANET.

6.2 Saran

Adapun saran-saran yang diberikan untuk pengembangan sistem ini selanjutnya adalah dikarenakan menggunakan perhitungan tambahan pada pemilihan *forwading node* pada protokol GPSO hal tersebut meningkatkan nilai dari *end to end delay*. Perlu adanya batasan lain yang membuat proses perhitungan *forwarding node* yang optimum menjadi lebih cepat sehingga dapat menurunkan nilai *end to end delay* dari protokol GPSO.

DAFTAR PUSTAKA

[1] N. Ayub, A. Saeed dan L. Khan, "**Anycast based Routing in VANETs Using Mobisim**," *IDOSI Publication,* 2009.

[2] H. s. N. Navroop Kaur, "**Pros and Cons: Various Routing Protocols based on**," *International Journal of Computer Applications (0975 – 8887),* vol. 106, no. 8, p. 42, 2014.

[3] B. Karp dan H. T. Kung, "**GPSR: Greedy Perimeter Stateless Routing for Wireless**," *MobiCom,* 2000.

[4] A. C. Oliviera, E. Alba dan J. Garc, "**Optimal Cycle Program of Traffic Lights**," *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.*

[5] A. TAMIZHSELVI dan R. S. D. WAHIDA BANU, "**HYBRID OPTIMIZATION ALGORITHM FOR GEOGRAPHIC**," *Journal of Theoretical and Applied Information Technology,* vol. 65, no. 2, p. 323, 2014.

[6] "**OpenStreetMap**," [Online]. Available: http://www.openstreetmap.org. [Accessed 31 May 2017].

[7] "**JOSM**," [Online]. Available: https://josm.openstreetmap.de/. [Diakses 31 May 2017].

[8] "**SUMO - Simulation of Urban MObility**," [Online]. Available: http://www.sumo.dlr.de/wiki/Simulation_of_Urban_MObility_-_Wiki. [Accessed 10 May 2017].

[9] D. B. Close, P. H. Rubn, A. D. Robbins, R. Stallman dan P. V. Oostrum, **The AWK Manual**, Massachusetts: Free Software Foundation, Inc., 1995.

[10] "**NS-2 Simulator**," [Online]. Available: http://www.isi.edu/nsnam/ns/. [Accessed 5 April 2017].

[11] M. Claypool, "**Trace Analysis Example**," Worchester Polytechnic Institute, [Online]. Available: http://nile.wpi.edu/NS/analysis.html. [Diakses 30 May 2017].

[12] F. Dwi, A. Ghany, P. Adi, R. H dan E. N, "**Analisis Performansi Protokol Routing AODV dan DSR pada MANET**," Fakultas Tenik Informatika, Universitas Telkom, Bandung, Indonesia, Bandung.

LAMPIRAN

Bagian ini merupakan lapiran sebagai dokumen pelengkap dari buku Tugas Akhir, pada bagian ini diberikan informasi mengenai kode sumber dari sistem yang dibuat.

A. Kode Sumber

A.1. Kode Skenario NS-2

```
1.  # Copyright (c) 1997 Regents of the University
    of California.
2.  # All rights reserved.
3.  #
4.  # Redistribution and use in source and binary
    forms, with or without
5.  # modification, are permitted provided that the
    following conditions
6.  # are met:
7.  # 1. Redistributions of source code must retain
    the above copyright
8.  #    notice, this list of conditions and the
    following disclaimer.
9.  # 2. Redistributions in binary form must
    reproduce the above copyright
10. #    notice, this list of conditions and the
    following disclaimer in the
11. #    documentation and/or other materials
    provided with the distribution.
12. # 3. All advertising materials mentioning
    features or use of this software
13. #    must display the following acknowledgement:
14. #      This product includes software developed
    by the Computer Systems
15. #      Engineering Group at Lawrence Berkeley
    Laboratory.
16. # 4. Neither the name of the University nor of
    the Laboratory may be used
17. #    to endorse or promote products derived from
    this software without
18. #    specific prior written permission.
```

```
19. #
20. # THIS SOFTWARE IS PROVIDED BY THE REGENTS AND
    CONTRIBUTORS ``AS IS'' AND
21. # ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,
    BUT NOT LIMITED TO, THE
22. # IMPLIED WARRANTIES OF MERCHANTABILITY AND
    FITNESS FOR A PARTICULAR PURPOSE
23. # ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS
    OR CONTRIBUTORS BE LIABLE
24. # FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
    EXEMPLARY, OR CONSEQUENTIAL
25. # DAMAGES (INCLUDING, BUT NOT LIMITED TO,
    PROCUREMENT OF SUBSTITUTE GOODS
26. # OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
    BUSINESS INTERRUPTION)
27. # HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
    WHETHER IN CONTRACT, STRICT
28. # LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
    OTHERWISE) ARISING IN ANY WAY
29. # OUT OF THE USE OF THIS SOFTWARE, EVEN IF
    ADVISED OF THE POSSIBILITY OF
30. # SUCH DAMAGE.
31. #
32. # $Header:
    /home/cvs/repository/kliu/ns2/gpsr/wireless-
    gpsr.tcl,v 1.8 2005/12/01 00:03:17 kliu Exp $
33. #
34. # Ported from CMU/Monarch's code, nov'98 -Padma.
35.
36.
37.
38.
39. #
    ================================================
    ======================
40. # Default Script Options
41. #
    ================================================
    ======================
42. set opt(chan)           Channel/WirelessChannel
43. set opt(prop)           Propagation/TwoRayGround
44. set opt(netif)          Phy/WirelessPhy
45. set opt(mac)            Mac/802_11
```

```
46. set opt(ifq)            Queue/DropTail/PriQueue
       ;# for dsdv
47. set opt(ll)             LL
48. set opt(ant)             Antenna/OmniAntenna
49.
50. set opt(x)              800          ;# X
    dimension of the topography
51. set opt(y)              800          ;# Y
    dimension of the topography
52. set opt(cp)             "./gpsoCBRtest.tcl"
53. set opt(sc)             "./GPSOnodePos.tcl"
54.
55. set opt(ifqlen)         50           ;# max
    packet in ifq
56. set opt(nn)                   77           ;#
    number of nodes
57. set opt(seed)           0.0
58. set opt(stop)           200.0        ;#
    simulation time
59. set opt(tr)                   trace.tr
       ;# trace file
60. set opt(nam)             nam.out.tr
61. set opt(rp)              gpsr         ;# routing
    protocol script (dsr or dsdv)
62. set opt(lm)              "off"        ;# log
    movement
63.
64. Antenna/OmniAntenna set X_ 0
65. Antenna/OmniAntenna set Y_ 0
66. Antenna/OmniAntenna set Z_ 1.5
67. Antenna/OmniAntenna set Gt_ 1.0
68. Antenna/OmniAntenna set Gr_ 1.0
69.
70. Phy/WirelessPhy set Pt_ 0.2818    ;
71.
72. # Agent/GPSR setting
73. Agent/GPSR set planar_type_  1   ;#1=GG
    planarize, 0=RNG planarize
74. Agent/GPSR set hello_period_   1.0 ;#Hello
    message period
75.
76. #
    ================================================
    ======================
```

```
77.
78. proc usage { argv0 }  {
79.     puts "Usage: $argv0"
80.     puts "\tmandatory arguments:"
81.     puts "\t\t\[-x MAXX\] \[-y MAXY\]"
82.     puts "\toptional arguments:"
83.     puts "\t\t\[-cp conn pattern\] \[-sc
    scenario\] \[-nn nodes\]"
84.     puts "\t\t\[-seed seed\] \[-stop sec\] \[-tr
    tracefile\]\n"
85. }
86.
87.
88. proc getopt {argc argv} {
89.     global opt
90.     lappend optlist cp nn seed sc stop tr x y
91.
92.     for {set i 0} {$i < $argc} {incr i} {
93.           set arg [lindex $argv $i]
94.           if {[string range $arg 0 0] != "-"}
    continue
95.
96.           set name [string range $arg 1 end]
97.           set opt($name) [lindex $argv [expr
    $i+1]]
98.     }
99. }
100.
101.
102.    #proc cmu-trace { ttype atype node } {
103.    #      global ns_ tracefd
104.    #
105.    #        puts ABC
106.    #      if { $tracefd == "" } {
107.    #            return ""
108.    #      }
109.    #      puts BCD
110.    #      set T [new CMUTrace/$ttype $atype]
111.    #      $T target [$ns_ set nullAgent_]
112.    #      $T attach $tracefd
113.    #        $T set src_ [$node id]
114.    #
115.    #        $T node $node
116.    #
```

```
117.   #      return $T
118.   #}
119.
120.
121.
122.
123.   proc log-movement {} {
124.       global logtimer ns_ ns
125.
126.       set ns $ns_
127.       source ../tcl/mobility/timer.tcl
128.       Class LogTimer -superclass Timer
129.       LogTimer instproc timeout {} {
130.          global opt node_;
131.          for {set i 0} {$i < $opt(nn)} {incr i}
   {
132.              $node_($i) log-movement
133.          }
134.          $self sched 0.1
135.       }
136.
137.       set logtimer [new LogTimer]
138.       $logtimer sched 0.1
139.   }
140.
141.   #
   ==============================================
   ======================
142.   # Main Program
143.   #
   ==============================================
   ======================
144.   #
145.   # Source External TCL Scripts
146.   #
147.   #source ../lib/ns-mobilenode.tcl
148.
149.   #if { $opt(rp) != "" } {
150.          #source ../mobility/$opt(rp).tcl
151.          #} elseif { [catch { set
   env(NS_PROTO_SCRIPT) } ] == 1 } {
152.          #puts "\nenvironment variable
   NS_PROTO_SCRIPT not set!\n"
153.          #exit
```

```
154.   #} else {
155.          #puts "\n*** using script
   $env(NS_PROTO_SCRIPT)\n\n";
156.           #source $env(NS_PROTO_SCRIPT)
157.   #}
158.   #source ../tcl/lib/ns-cmutrace.tcl
159.   source ../tcl/lib/ns-bsnode.tcl
160.   source ../tcl/mobility/com.tcl
161.
162.   # do the get opt again incase the routing
   protocol file added some more
163.   # options to look for
164.   getopt $argc $argv
165.
166.   if { $opt(x) == 0 || $opt(y) == 0 } {
167.          usage $argv0
168.          exit 1
169.   }
170.
171.   if {$opt(seed) > 0} {
172.          puts "Seeding Random number generator
   with $opt(seed)\n"
173.          ns-random $opt(seed)
174.   }
175.
176.   #
177.   # Initialize Global Variables
178.   #
179.   set ns_              [new Simulator]
180.   set chan      [new $opt(chan)]
181.   set prop      [new $opt(prop)]
182.   set topo      [new Topography]
183.
184.   set tracefd   [open $opt(tr) w]
185.   $ns_ trace-all  $tracefd
186.
187.   set namfile [open $opt(nam) w]
188.   $ns_ namtrace-all $namfile
189.
190.   $topo load_flatgrid $opt(x) $opt(y)
191.
192.   $prop topography $topo
193.
194.   #
```

```
195.   # Create God
196.   #
197.   set god_ [create-god $opt(nn)]
198.
199.
200.   #
201.   #  Create the specified number of nodes
   $opt(nn) and "attach" them
202.   #  the channel.
203.   #  Each routing protocol script is expected
   to have defined a proc
204.   #  create-mobile-node that builds a mobile
   node and inserts it into the
205.   #  array global $node_($i)
206.   #
207.
208.   $ns_ node-config -adhocRouting gpsr \
209.                    -llType $opt(ll) \
210.                    -macType $opt(mac) \
211.                    -ifqType $opt(ifq) \
212.                    -ifqLen $opt(ifqlen) \
213.                    -antType $opt(ant) \
214.                    -propType $opt(prop) \
215.                    -phyType $opt(netif) \
216.                    -channelType $opt(chan) \
217.                    -topoInstance $topo \
218.                    -agentTrace ON \
219.                    -routerTrace ON \
220.                    -macTrace OFF \
221.                    -movementTrace OFF
222.
223.   source ./gpsr.tcl
224.
225.   for {set i 0} {$i < $opt(nn) } {incr i} {
226.       gpsr-create-mobile-node $i
227.   }
228.
229.
230.   #
231.   # Source the Connection and Movement scripts
232.   #
233.   if { $opt(cp) == "" } {
234.         puts "*** NOTE: no connection pattern
   specified."
```

```
235.            set opt(cp) "none"
236.    } else {
237.           puts "Loading connection pattern..."
238.           source $opt(cp)
239.    }
240.
241.
242.
243.
244.    #
245.    # Tell all the nodes when the simulation ends
246.    #
247.    for {set i 0} {$i < $opt(nn) } {incr i} {
248.        $ns_ at $opt(stop).000000001 "$node_($i)
    reset";
249.    }
250.    $ns_ at $opt(stop).00000001 "puts \"NS
    EXITING...\" ; $ns_ halt"
251.
252.
253.    if { $opt(sc) == "" } {
254.           puts "*** NOTE: no scenario file
    specified."
255.            set opt(sc) "none"
256.    } else {
257.           puts "Loading scenario file..."
258.           source $opt(sc)
259.           puts "Load complete..."
260.    }
261.
262.    puts $tracefd "M 0.0 nn $opt(nn) x $opt(x) y
    $opt(y) rp $opt(rp)"
263.    puts $tracefd "M 0.0 sc $opt(sc) cp $opt(cp)
    seed $opt(seed)"
264.    puts $tracefd "M 0.0 prop $opt(prop) ant
    $opt(ant)"
265.
266.    puts "Starting Simulation..."
267.
268.    proc finish {} {
269.        global ns_ tracefd namfile
270.        $ns_ flush-trace
271.        close $tracefd
272.        close $namfile
```

```
273.        exit 0
274.    }
275.
276.    $ns_ at $opt(stop) "finish"
277.
278.    $ns_ run
```

A.2. Kode Skenario CBR

```
1.
2.  # GPSR routing agent settings
3.  for {set i 0} {$i < $opt(nn)} {incr i} {
4.       $ns_ at 0.00002 "$ragent_($i) turnon"
5.       $ns_ at 3.0 "$ragent_($i) neighborlist"
6.  #     $ns_ at 30.0 "$ragent_($i) turnoff"
7.  }
8.
9.  $ns_ at 5.0 "$ragent_(75) startSink 10.0"
10. #$ns_ at 10.5 "$ragent_(99) startSink 10.0"
11.
12.
13. # GPSR routing agent dumps
14. #$ns_ at 25.0 "$ragent_(24) sinklist"
15.
16.
17. # Upper layer agents/applications behavior
18. set null_(1) [new Agent/Null]
19. $ns_ attach-agent $node_(75) $null_(1)
20.
21. set udp_(1) [new Agent/UDP]
22. $ns_ attach-agent $node_(76) $udp_(1)
23.
24. set cbr_(1) [new Application/Traffic/CBR]
25. $cbr_(1) set packetSize_ 32
26. $cbr_(1) set interval_ 2.0
27. $cbr_(1) set random_ 1
28. #     $cbr_(1) set maxpkts_ 100
29. $cbr_(1) attach-agent $udp_(1)
30. $ns_ connect $udp_(1) $null_(1)
31. $ns_ at 10.0 "$cbr_(1) start"
32. $ns_ at 195.0 "$cbr_(1) stop"
```

A.3. Kode Impelementasi Perubahan Struktur *Header* pada GPSR

```
1.  struct gpsr_neighbor {
2.    nsaddr_t id_;
3.    double x_;    //the geo info
4.    double y_;
5.     double dir;
6.     double velo;
7.    double column;
8.
9.    double ts_;     //the last time stamp of the
   hello msg from it
10.   struct gpsr_neighbor *next_;
11.   struct gpsr_neighbor *prev_;
12. };
```

A.4. Kode Implementasi Perubahan *Header Hello Message*

```
1.  struct hdr_gpsr_hello {
2.    u_int8_t type_;
3.    float x_;     //My geo info
4.    float y_;
5.    float ts_;
6.    inline int size(){
7.      int sz =
8.        sizeof(u_int8_t) +
9.        2*sizeof(float);
10.     return sz;
11.   }
12. };
```

A.5. Kode Implementasi Penyimpanan Data Hello Message

```
1.  void
2.  GPSRAgent::hellomsg(){
3.
4.    if(my_id_ < 0) return;
5.
6.    Packet *p = allocpkt();
7.    struct hdr_cmn *cmh = HDR_CMN(p);
8.    struct hdr_ip *iph = HDR_IP(p);
9.    struct hdr_gpsr_hello *ghh =
    HDR_GPSR_HELLO(p);
10.
11.   cmh->next_hop_ = IP_BROADCAST;
12.   cmh->last_hop_ = my_id_;
13.   cmh->addr_type_ = NS_AF_INET;
14.   cmh->ptype() = PT_GPSR;
15.   cmh->size() = IP_HDR_LEN + ghh->size();
16.
17.   iph->daddr() = IP_BROADCAST;
18.   iph->saddr() = my_id_;
19.   iph->sport() = RT_PORT;
20.   iph->dport() = RT_PORT;
21.   iph->ttl_ = IP_DEF_TTL;
22.
23.   MobileNode *thisNode = (MobileNode
    *)(Node::get_node_by_address(cmh->last_hop_));
24.   double newx = thisNode->X();
25.   double newy = thisNode->Y();
26.
27.   ghh->type_ = GPSRTYPE_HELLO;
28.   ghh->x_ = newx;
29.   ghh->y_ = newy;
30.   ghh->ts_ = (float)GPSR_CURRENT;
31.
32.   send(p, 0);
33. }
```

A.6. Kode Implementasi Fungsi Penghitungan Nilai Arah dan Kecepatan

```
1. void
2. GPSRNeighbors::newNB(nsaddr_t nid, double nx,
   double ny, double velo, double dir, double
   column){
3.   struct gpsr_neighbor *temp = getnb(nid);
4.   if(temp==NULL){ //it is a new neighbor
5.     temp=(struct
   gpsr_neighbor*)malloc(sizeof(struct
   gpsr_neighbor));
6.     temp->id_ = nid;
7.     temp->x_ = nx;
8.     temp->y_ = ny;
9.           temp->dir = dir;
10.          temp->velo = velo;
11.    temp->column = column;
12.    temp->ts_ = GPSR_CURRENT;
13.    temp->next_ = temp->prev_ = NULL;
14.
15.    if(tail_ == NULL){ //the list now is empty
16.      head_ = tail_  = temp;
17.      nbSize_ = 1;
18.    }
19.    else { //now the neighbors list is not empty
20.      tail_->next_ = temp;
21.      temp->prev_ = tail_;
22.      tail_ = temp;
23.      nbSize_++;
24.    }
25.  }
26.  else { //it is a already known neighbor
27.    temp->ts_ = GPSR_CURRENT;
28.    temp->x_ = nx; //the updating of location is
   allowed
29.    temp->y_ = ny;
30.          temp->dir = dir;
31.          temp->velo = velo;
32.    temp->column = column;
33.  }
34.}
```

A.7. Kode Sumber Pemilihan *Forwarding Node*

```
1.  nsaddr_t
2.  GPSRNeighbors::gf_nexthop(double dx, double dy,
    double dirtodest, double column){
3.    struct gpsr_neighbor *temp = head_;
4.    //initializing the minimal distance as my
    distance to sink
5.    double mindis =getdis(my_x_, my_y_, dx, dy);
6.     double minvelo = 0.1;
7.     double timesmin = mindis/minvelo;
8.    nsaddr_t nexthop = -1; //the nexthop result
9.
10.   while(temp){
11.     double tempdis = getdis(temp->x_, temp->y_,
    dx, dy);
12.      double timestemp = tempdis/temp->velo;
13.
14.     if(dirtodest >= 0){
15.                   if(temp->dir >= 0 /*&& temp-
    >column == column*/){
16.     if(timestemp < timesmin){
17.           mindis = tempdis;
              timesmin = timestemp;
              nexthop = temp->id_;
18.     }
19.     else if(timestemp == timesmin){
20.     if(tempdis < mindis){
21.           mindis = tempdis;
              timesmin = timestemp;
              nexthop = temp->id_;
22.           }
23.     }
24.     }
25.     }
26.     else if(dirtodest < 0){
27.     if(temp->dir < 0){
28.     if(timestemp < timesmin){
29.           mindis = tempdis;
30.           timesmin = timestemp;
31.           nexthop = temp->id_;
32.     }
33.     else if(timestemp == timesmin){
```

```
34.    if(tempdis < mindis){
35.           mindis = tempdis;
36.           timesmin = timestemp;
37.           nexthop = temp->id_;
38.           }
39.     }
40.     }
41.    }
42.    temp = temp->next_;
43.   }
44.   return nexthop;
45.   struct gpsr_neighbor *temp = head_;
46.   MobileNode *thisNode = (MobileNode
   *)(Node::get_node_by_address(my_id_));
47.   double mindis =getdis(thisNode->X(), thisNode-
   >Y(), dx, dy);
48.   nsaddr_t nexthop = -1; //the nexthop result
49.
50.   while(temp){
51.     double tempdis = getdis(temp->x_, temp->y_,
   dx, dy);;
52.     if(tempdis < mindis){
53.       mindis = tempdis;
54.       nexthop = temp->id_;
55.     }
56.     temp = temp->next_;
57.   }
58.
59.   return nexthop;*/
60. }
```

A.8. Kode AWK Untuk Analisis Hasil Perhitungan Arah Protokol GPSO

```
1.  BEGIN{
2.      minus = 0;
3.      plus = 0;
4.      nol = 0;
5.  }
6.  {
7.      if (($5=9 ||$5=50 ||$5=14) && ( $8<0 ))
8.             minus++;
9.      if (($5=9 ||$5=50 ||$5=14) && ( $8>0 ))
10.            plus++;
11.     if (($5=9 ||$5=50 ||$5=14) && ( $8=0 ))
12.            nol++;
13. }
14. END{
15.     printf("minus %d", minus);
16.     printf("\nplus %d", plus);
17.     printf("\nnol %d\n", nol);
18. }
```

A.9. Kode awk untuk Menghitung Packet Delivery Ratio

```
1. BEGIN {
2.     sendLine = 0;
3.     recvLine = 0;
4.     forwardLine = 0;
5. }
6.
7. $0~/^s.*AGT/{
8.     sendLine++
9. }
10.
11. $0~/^r.*AGT/{
12.     recvLine++
13. }
14.
15. $0~/^f.*RTR/{
16.     forwardLine++
17. }
18.
19. END{
20.     printf"Ratio:%.4f\n",(recvLine/sendLine)*100;
21. }
```

A.10. Kode AWK untuk menghitung End to End Delay

```
1. BEGIN {
2. seqno = -1;
3. # droppedPackets = 0;
4. # receivedPackets = 0;
5. count = 0;
6. }
7. {
8. if($4 == "AGT" && $1 == "s" && seqno < $6) {
9. seqno = $6;
10.}
11.if($4 == "AGT" && $1 == "s") {
12.start_time[$6] = $2;
13.} else if(($7 == "cbr") && ($1 == "r")) {
14.end_time[$6] = $2;
15.} else if($1 == "D" && $7 == "cbr") {
16.end_time[$6] = -1;
17.}
18.}
19.END {
20.for(i=0; i<=seqno; i++) {
21.if(end_time[i] > 0) {
22.delay[i] = end_time[i] - start_time[i];
23.count++;
24.}
25.else
26.{
27.delay[i] = -1;
28.}
29.}
30.for(i=0; i<=seqno; i++) {
31.if(delay[i] > 0) {
32.n_to_n_delay = n_to_n_delay + delay[i];
33.}
34.}
35.n_to_n_delay = n_to_n_delay/count;
36.print n_to_n_delay * 1000 ;
37.}
```

A.11. Kode AWK untuk Menghitng *Routing Overhead*

```
1. BEGIN{
2.     recvs = 0;
3.     routing_packets = 0;
4. }
5. {
6.     if (( $1 == "r") && ( $7 == "cbr" ) &&
   ( $19=="4" ))
7.             recvs++;
8.     if (($1 == "s" || $1 == "r") && $4 == "RTR"
   && $7 =="GPSR")
9.             routing_packets++;
10.}
11.END{
12.    printf("Routing Overhead : %.3f\n",
   routing_packets/recvs);
13.}
```

B. Instalasi

B.1 Instalasi NS-2

Instalasi NS-2 dilakukan pada sistem operasi Ubuntu 14.04. Yang diperlukan untuk menggunakan NS-2 adalah melakukan instalasi dependensi dan *source code* ns-2.35. Sebelum melakukan instalasi NS-2 diperlukan beberapa dependensi agar NS-2 dapat dijalankan. Cara instalasi dependensi tersebut ditunjukkan pada perintah di bawah.

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

Setelah semua dependensi terpasang selanjutnya adalah mengunduh *source code* ns-2.35 dengan cara yang ditunjukkan pada perintah di bawah.

```
wget
http://jaist.dl.sourceforge.net/project/nsnam/allinone/nsallinone-2.35/ns-allinone-2.35.tar.gz
```

Ekstark source code tersebut dengan perintah pada kode sumber 2.3 :

```
tar -xvf ns-allinone-2.35.tar.gz
```

B.2 Patching GPSR

Pada *source code* yang baru diunduh tidak terdapat protokol GPSR, oleh karena itu diperlukan untuk melakukan *patching* sebelum instalasi NS-2. Pada Tugas Akhir ini, penulis menggunakan patch GPSR yang dibuat Ke Liu. Patch dapat diunduh dengan tautan pada perintah di bawah.

```
https://drive.google.com/file/d/0B7S255p3kFXNV2ctNFctd3JsZGs/view
```

File yang terunduh bernama **gpsr-KeLiu_ns235.patch**. Letakkan file tersebut didalam direktori ns-allinone-2.35. Kemudian lakukan patch dengan perintah sesuai pada perintah di bawah.

```
patch –p0 < gpsr-KeLiu_ns235.patch
```

Setelah tahap diatas selesai hal terakhir yang perlu dilakukan adalah menjalankan skrip instalasi NS-2, pada direktori ns-allinone-2.35 jalankan perintah sesuai pada perintah di bawah.

```
./install
```

Aplikasi NS-2 yang telah dilakukan instalasi merupakan aplikasi yang berdiri sendiri sehingga diperlukan pengaturan agar dapat dijalankan pada terminal dengan menggunakan perintah **ns**, caranya adalah lakukan *copy* pada direktori ns-allinone-2.35 sesuai pada perintah di bawah.

```
cp ns-2.35/ns /usr/local/bin
```

Untuk mengecek apakah berhasil melakukan instalasi, ketikan perintah **ns** pada terminal.

BIODATA PENULIS

Bagus Putra Mayani, lahir Bandung, 29 April 1996. Penulis adalah anak pertama dari dua bersaudara. Menempuh pendidikan di SD Negeri Angkasa VIII, SMP Negeri 1 Margahayu, SMA Negeri 6 Bandung, dan terakhir melanjutkan kuliah di jurusan Teknik Informatika – ITS.

Selain menjalankan tugas mahasiswa, penulis juga aktif menjadi asisten dosen mata kuliah jaringan komputer. Ketertarikan penulis di bidang informatika berada pada bidang sistem teknologi informasi, sekuritas jaringan, perancangan keamanan sistem dan jaringan, teknologi antar jaringan, dan teknologi tepat guna.

Penulis dapat dihubungi dengan mengirimkan pesan elektronik ke alamat bagus.pm29@gmail.com.